SoftTree
SQL Assistant™ 13.0

User's Guide

Supported database systems:

Oracle 11g - 23c

Microsoft SQL Server 2008 - 2022

Windows Azure SQL Database 11.x, 12.x (formerly SQL Azure)

Apache Cassandra 3.x - 4.x

DataStax Enterprise 6.x

Apache Hive 2.x - 3.x

Apache SparkSQL 2.x, 3.x

MySQL 5.x, 8.x

MariaDB 5.x, 10.x, 11.x

DB2 UDB for LUW 7.x - 11.x

DB2 UDB for iSeries 5.x - 7.x

SAP Sybase Adaptive Server Enterprise 12.x - 16.x

SAP Sybase SQL Anywhere 9.x - 17.x (formerly Adaptive Server Anywhere)

PostgreSQL 8.4 - 16.x

Amazon Redshift 1.x

Teradata 14.x - 17.x

IBM Netezza 7.x

Pivotal Greenplum 4.3.x - 6.x

Microsoft Access 2003 to Office 365 (11.x - 16.x)

SQLite 2.x - 3.x

MongoDB via Connector for BI 2.x

Snowflake 6.x, 7.x

Contents

About This Guide

This manual describes the features of the SoftTree SQL Assistant product, including how to:

- Use the graphical user interface

- Register target editors for SQL assistance

- Use code completion features, SQL Intellisense SQL reference functions, and database integration options

- How to execute SQL code, check SQL syntax, bulk-generate SQL procedures, and generate test data

- How to use many other built-in features.

Unless otherwise noted, the features and how-to instructions described in this manual apply to all supported database management systems running on any supported platform.

Intended Audience

This document is for Database Developers and Database Administrators.

Conventions Used in This Document

This section describes the style conventions used in this document.

Italic

An *italic* font is used for filenames, URLs, emphasized text, and the first usage of technical terms.

`Monospace`

A `monospaced` font is used for code fragments and data elements.

Bold

A **bold** font is used for important messages, names of options, names of controls and menu items, and keys.

User Input

Keys are rendered in **bold** to stand out from other text. Key combinations that are meant to be typed simultaneously are rendered with "+" sign between the keys, such as:

Ctrl+F

Keys that are meant to be typed in sequence will be separated with commas, for example:

Alt+S, H

This would mean that the user is expected to type the Alt and S keys simultaneously and then to type the H key.

Graphical symbols

 - This symbol is used to indicate DBMS specific options and issues and to mark useful auditing tips.

 - This symbol is used to indicate important notes.

Abbreviations and Product Reference Terms

DBMS – Database Management System

Oracle – This refers to all supported Oracle® database servers

SQL Server – This refers to all versions of Microsoft® SQL Server™ database servers.

DB2 for LUW– This refers to all versions of IBM® DB2® UDB database servers on all supported Linux Unix and Windows platforms, unless platform and version are specially mentioned in the text.

DB2 for iSeries – This refers to all versions of IBM® DB2® UDB database servers on i5/OS platform (formerly AS/400 midrange mainframe servers), unless platform and version are specially mentioned in the text.

MySQL – This refers to all versions of MySQL™ database servers starting with version 5.0.

PostgreSQL – This refers to all versions of PostgreSQL™ database servers starting with version 8.0.

Amazon Redshift – This refers to all versions of Amazon Redshift database servers.

Sybase ASE, and ASE – This refers to all versions of Sybase® Adaptive Server Enterprise™ database servers.

Sybase ASA, and ASA – This refers to all versions of Sybase® Adaptive Server Anywhere™ and Sybase® SQL Anywhere™ database servers.

Access – This refers to Microsoft® Access™ databases versions 2003, 2007, and 2010.

Greenplum – This refers to Pivotal Greenplum ® database servers.

Netezza – This refers to IBM Netezza ® database servers starting with version 7.0.

Teradata – This refers to all supported Teradata® database servers.

SQLite – This refers to SQLite database servers starting with version 3,0 and later.

Trademarks

SoftTree SQL Assistant, DB Audit, DB Mail, 24x7 Automation Suite, 24x7 Scheduler, 24x7 Event Server, DB Tools for Oracle are trademarks of SoftTree Technologies, Inc.

Windows 2000, Windows 2003, Windows 2008, Windows XP, Windows 7, Visual Studio, IntelliSense are registered trademarks of Microsoft Corporation.

Microsoft SQL Server and SQL Azure are registered trademarks of Microsoft Corporation.

Oracle and MySQL are registered trademark of Oracle Corporation.

IBM and DB2 are registered trademarks of IBM Corporation.

Teradata is registered trademark of Teradata Corporation

PostgreSQL is registered trademark of The PostgreSQL Global Development Group.

Sybase Adaptive Server Anywhere, Adaptive Server Enterprise, SQL Anywhere are registered trademarks of Sybase Inc an SAP company or its subsidiaries.

Toad for Oracle, Toad for SQL Server, Toad for DB2 and Toad for MySQL are trademarks of Dell Software.

All other trademarks appearing in this document are trademarks of their respective owners. All rights reserved.

(this page is intentionally left blank)

CHAPTER 1, Overview of the SoftTree SQL Assistant

Introduction

SoftTree SQL Assistant enables database developers and DBAs to realize amazing improvements in code quality and accuracy. It integrates with many widely used database editors, database management and development environments, as well as DBMS native utilities, transforming them to RAD database development tools. SQL Assistant delivers unparalleled support for code typing, automatic code completion, syntax references and validations, and database object and attributes browsing. It is quite simply the ultimate SQL code assistance solution available.

SQL Assistant is fast and can be used with both small and very large database systems. It is very flexible and can easily be customized by users to match individual coding habits and project requirements.

Key Benefits

- Dramatically increases database coding productivity, providing superior SQL Intellisense and type-ahead functionality.

- Improves code quality and accuracy. Supports best in class advanced SQL code formatting and layout functions.

- Provides fast code navigation and real-time code syntax checking functions.

- Provides real-time code performance analysis and suggestions.

- Supports 9 of the most popular database systems.

- Integrates with many SQL and non-SQL editors.

- Provides interactive SQL help and code assistance system.

- Provides a full set of functions for SQL editing, code execution, data viewing, and SQL scheduling. These functions can be used with all supported environments including all registered non-SQL editors.

- Provides data import and export facilities available for all registered SQL and non-SQL editors.

- Provides a full-featured, database source code control interface.

- Provides an advanced database schema and database data comparison tools with fully customizable comparison rules and schema change synchronization templates.

- Provides an advanced set of tools for code generation and deployment to multiple database servers.

- Is fast and has small disk and memory footprints.

- Can be installed easily and quickly without interrupting existing processes and settings; ready for immediate use.

32-bit and 64-bit Versions

SQL Assistant provides dual interface for 32-bit and 64-bit Windows versions. It supports both 32-bit and 64-bit target editors and also 32-bit and 64-bit database drivers. The following table describes support by the environment type

	Windows 32-bit	Windows 64-bit
32-bit target editors	Yes	Yes
64-bit target editors	*N/A*	Yes
32-bit native database drivers (OCI, LibMySQL, LibPQ, SQLite) editors	Yes	Yes
64-bit native database drivers (OCI, LibMySQL, LibPQ, SQLite) editors	*N/A*	Yes
ADO.NET drivers (environment agnostic)	Yes	Yes
32-bit ODBC drivers (Unicode)	Yes	Yes
64-bit ODBC drivers (Unicode)	*N/A*	Yes

Important Notes:

- Unless you are using ADO.NET database connectivity, for your database connections you must choose database driver having the same 32-bit or 64-bit architecture as the target editor. For example, if you run SQL Assistant within 32-bit Visual Studio on 64-bit Windows platform, you can use 32-bit database drivers only.

- The SQL Assistant Standard Edition includes Simple SQL Editor which is a 32-bit application. You can use 32-bit database drivers and ADO.NET with that target editor. You can also use SQL Assistant Standard Edition with 64-bit target editors like Dell's Toad ® which you can use with 64-bit database drivers.

Licensing and Editions

The SQL Assistant software is available in two editions: Standard and Professional. The binary code is the same in both editions but their feature coverage is not. The Standard Edition provides only the basic features required for database development. The Professional Edition provides all the features available in the SQL Assistant software. Both Editions support common set of databases and target editors with the exception of SQL Editor Professional Edition.

For a detailed list of the differences between Standard and Professional Editions see
http://www.softtreetech.com/sqlassist/compare_editions.htm

For the End-user License Agreement (EULA) see APPENDIX B at the end of this manual.

CHAPTER 2, Connecting to Your Database

Overview

SoftTree SQL Assistant connects to database servers using any of the following connectivity interfaces:

1. **ODBC** interface – Can be used with any supported database system. The appropriate ODBC database driver must be preinstalled.

2. **ADO.NET** interface – Can be used with any supported database system. The appropriate ADO.NET database driver must be preinstalled.

3. Native database interfaces such as
    Oracle OCI – Oracle Call Interface (OCI) can be used with any Oracle database server version 8.1 or later. Oracle OCI is part of Oracle Instant Client software. It can be licensed for free from Oracle Corporation. Visit https://www.oracle.com/database/technologies/instant-client.html for more details.
    LibPQ - can be used with any PostgreSQL databases version 8.0 and later as well as will Pivotal Greenplum databases version 4.0 or later, and Amazon Redshift databases. LibPQ is preinstalled with SQL Assistant and available for both 32-bit and 64-bit environments.
    LibMysqDB - can be used with any MySQL and MariaDB database server version 5.0 or later. LibMySQLDB is part of MySQL Connector and it can be licensed for free from Oracle Corporation. Visit https://www.mysql.com/products/connector/ for more details.
    LibMariaDB - can be used with any MariaDB database server version 5.0 or later. It can be also used with most MySQL versions. LibMariaDB is preinstalled with SQL Assistant software and available for both 32-bit and 64-bit environments..
    SQLite Wire Protocol Driver – can be used with SQLite databases versions 3.0 and later, supports extended set of functions. The Wire Driver is preinstalled with SQL Assistant and available for both 32-bit and 64-bit environments.
    Cassandra Wire Protocol Driver – can be used with Apache Cassandra databases versions 3.0 and later as well as DataStax Enterprise and DataStax Astra cloud based databases. It supports extended set of functions for Cassandra based database servers. The Wire Driver is preinstalled with SQL Assistant and available for both 32-bit and 64-bit environments.
    SMO – SQL Server Management Objects interface can be used SQL Server databases version 2005 and later. SMO is preinstalled with SQL Assistant.

When used with pre-configured editors maintaining persistent connections to the database, SQL Assistant automatically intercepts and shares the existing connection. If a connection cannot be shared, SQL Assistant displays the **SQL Assistant - Connect** dialog, which you can use to enter the appropriate database connection properties. Connection properties can be saved for future use so the next time you need that database connection, you can quickly pull it by name from the saved connections list.

When choosing connectivity interface and database driver for the connection, make sure to select 32-bit driver if your SQL editor is 32-bit application, and choose 64-bit driver if your SQL Editor is 64-bit application.

 Important Notes:

* The appearance of the **SQL Assistant - Connect** dialog differs for different types of databases and connectivity interfaces. Different options are provided depending on selected database type and connectivity interface.

* The **SQL Assistant - Connect** dialog will not appear if the **Auto-connect** option is set for the target. In this case, the SQL Assistant will automatically connect to the database server you selected last time to **Auto-connect**. However, If the automatic connection fails, the SQL Assistant will display the

Connect dialog and prompt you for the correct connection parameters.

- For ODBC-based connections, SQL Assistant allows using both DSN and DSN-less connections. "DSN" is a common acronym for "data source name."

  For DSN-less connections, the **DSN / Driver** property specifies the driver name; for example, "Driver={SQL Server}" (without double-quotes). The driver name must be entered exactly as it appears on the Drivers tab in the Windows ODBC Administrator program.

  For preconfigured DSN connections, the **DSN / Driver** property specifies the ODBC DSN name as it appears in the ODBC Administrator program on either the File DSN or System DSN tab; for example, "DSN={MyDSN}" (without double-quotes).

  When using a DSN-less connection, you must specify all required connection parameters, including the server name, port, user, and other parameters required for the specific database server type and connection method.

  When using a DSN connection, you do not need to enter any additional parameters unless database side user authentication is required. In this case, you must specify a valid user name and password.

The following options and parameters may appear on the **SQL Assistant - Connect** dialog:

DB Type – The type of database management system (DBMS). This parameter is defined in SQL Assistant options for each target and cannot be changed directly on the connection dialog.

Connection Type – The type of client connection that SQL Assistant uses to communicate with the database server. The available choices depend on the chosen **DB Type** parameter and could be different for different database types.

TNS Name – This is an Oracle specific parameter used with OCI and ADO.NET-based connections. Enter the Oracle server connection name as specified in your **TNSNAMES.ORA** file on the local computer.

Server Name – For SQL Server connections using DSN-less ODBC connections or ADO.NET connections, this is the name or IP address of the database server computer or, in case of connection to a named SQL Server instance name, the instance name in standard server/instance format.

For MySQL connections using DSN-less ODBC connections, ADO.NET or MySQL Native, this is the name or IP address of the database server computer.

For Sybase ASE connections using DSN-less ODBC connections or ADO.NET, this is the name of the Sybase server profile defined in the Sybase client settings **SQL.INI** file on the local computer.

For DB2 ADO.NET-based connections, this is the DB2 database alias defined in the DB2 client settings on the local computer.

This parameter is ignored for all other databases and connection types.

Hostname – For DB2 connections using DSN-less ODBC connections or ADO.NET, this is the name or IP address of the database server computer.

This parameter is ignored for all other databases and connection types.

Connect As – This is an Oracle specific parameter used with OCI and ADO.NET-based connections. Enter one of the following: Normal, SYSDBA or SYSOPER.

Server Port – Specify this parameter only if establishing a TCP/IP-based connection when your database server is using a non-default port number to listen for new network connections.

This parameter is ignored with ODBC DSN-based connections. It is also not used with Microsoft Access connections.

Database – For SQL Server, Sybase ASE, Sybase ASA, MySQL and DB2, enter default database used for the initial connection.

For Microsoft Access and for SQLite connections enter the full name of the database file to connect to.

This parameter is not used with ODBC DSN-based connections.

Data Provider – Specifies the database connection provider used with ADO.NET-based connections. Enter the provider name, for example, "**System.Data.SqlClient**" (without double-quotes). Note that some data providers are pre-installed with the .NET framework, while others are installed with your database client software. Typically, multiple types of database providers can be used to establish a connection to the database server, yet they all require that the database client software be installed on the local computer. For example, both Microsoft's data provider for Oracle, System.Data.OracleClient, and Oracle's data provider, Oracle.DataAccess.Client, can be used to establish an ADO.NET-based connection to an Oracle database, yet both software products internally use the Oracle client software to connect to an Oracle database. The same Oracle client software can be used by SQL Assistant directly with the OCI interface, bypassing the middleware ADO.NET database interface.

DSN / Driver– Specifies either the name of the ODBC driver used with ODBC-based connections or an ODBC Connection Profile name. You can enter the driver name in the **Driver={Driver}** format for DSN-less connections, if such connections are supported by the specified driver, or you can use name of an existing DSN profile in the **DSN=Profile** format.

The screenshots below show the difference between a DSN less and a DSN-based ODBC connection, and demonstrates how to enter a driver or DSN profile name.

ODBC DSN-less connection
(DB2 for LUW specific example, won't work with DB2 iSeries)

ODBC DSN-based connection
(DB2 specific example, should work for DB2 LUW and for DB2 iSeries)
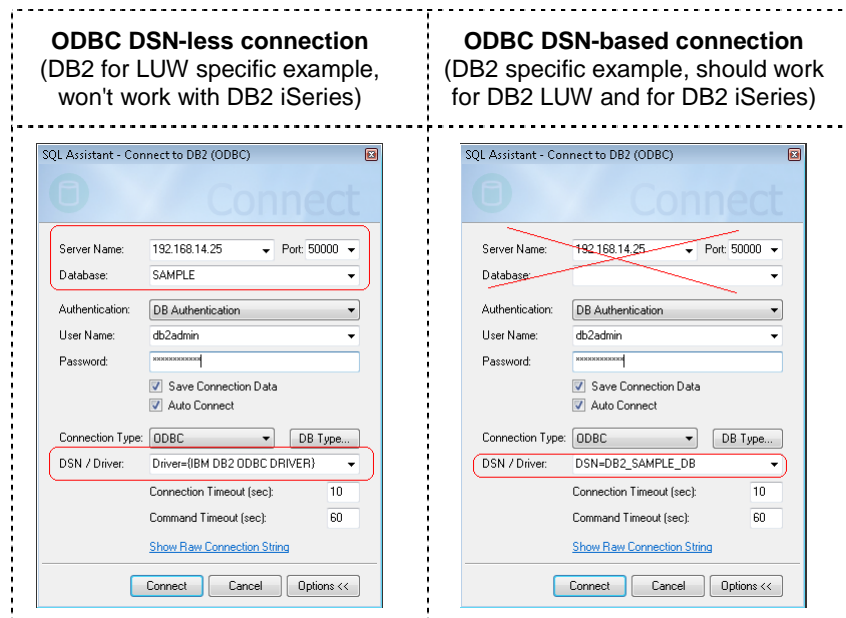
OCI DLL – This is Oracle specific parameter used with OCI-based connections. Enter the full path for the **OCI.DLL** file.

MySQL DLL – This is MySQL and also MariaDB specific parameter used with MySQL native library connections. Enter the full path for the **LIBMYSQL.DLL** file. You can also use the preinstalled **LIBMARIADB.DLL** that can be found in [SQL Assistant installation folder]\bin\MariaDB or [SQL Assistant installation folder]\bin64\MariaDB.

LIBPQ.DLL – This parameter is specific to PostgreSQL family of database servers including PostgreSQL, Amazon Redshift, Pivotal Greenplum. The LibPQ library is provided with SQL Assistant software. Normally no changes are required for customizing option. In case you want to use a different specific version of LIBPQ.DLL, enter the full path for the **LIBPQ.DLL** file. The preinstalled DLL file can be found in [SQL Assistant installation folder]\bin\PG or [SQL Assistant installation folder]\bin64\PG.

Authentication – This parameter can be used with ODBC DSN-less, ADO.NET and Cassandra wire driver connections only if supported by your database system.

For SQL Server, enter "Windows Authentication" or "SQL Server Authentication" (without double quotes). If you choose "Windows Authentication," you do not need to specify the **User Name** and **User Password** parameters.

For DB2, Sybase ASE or Sybase ASA connections, enter either "DB Authentication" or "OS Authentication" (without double quotes). Note that "OS Authentication" can only be used if your database server is configured to rely on client-side user authentication.

For DataStax Astra database select Astra if you Astra specific authentication method enabled. For Apache Cassandra select Server Authentication or No Authentication depending on how your Cassandra authentication is configured.

This parameter is not used with Microsoft Access connections.

User Name – Enter the user name for the database connection. This parameter is required only for database-side user authentications.

User Password – Enter the password for the database connection. This parameter is required only for database-side user authentications.

Connection Timeout – This parameter controls how many seconds to wait before canceling a connection attempt and reporting that the server as unreachable. This value can only be used if it is supported by the chosen database connection method and database driver. If the driver or the database does not support this parameter, the parameter value is ignored.

Command Timeout – This parameter controls how many seconds to wait before canceling an internal SQL command execute method call and generating an error. This value is supported only if it is supported by the chosen database connection method and database driver. If the driver or the database does not support this parameter, the parameter value is ignored.

Encrypt Connection – This parameter controls whether or not to use an encrypted communication protocol for a SQL Server database connection. This value can be used with SQL Server connections only if it is supported by the selected database connection method.

Use Compression – This parameter controls whether or not to use an encrypted and compressed communication protocol for a MySQL database connection. This value can be used only with MySQL connections if it is supported by the selected database connection method.

Save Connection Data – If checked, this option causes SQL Assistant to save the specified connection parameters in its configuration file. The connection can be reused later by selecting its name from the **Server Name** or **TNS Name** drop-down lists.

Show/Hide Raw Connection String – Click on this hyperlink to show or hide connection string. The connection string contains relevant parameters specified in other options. The format of the string differs for different connection types and methods.

> **Important Notes:** If necessary, you can specify additional custom connection parameters in the raw connection string. If **Save Connection Data** option is checked, these additional parameters will be saved along with the standard parameters and will be reused for future connections. When entering additional parameters using raw connection string, you must ensure that the additional parameters are supported by the database driver and client software. Consult your database driver and client software documentation for a list of supported parameters and their formats.

Ad-hoc and Remembered Connections

By default, SQL Assistant remembers entered connection parameters and saves them for future use. This behavior is controlled by **Save Connection Data** option on the **Connect to Database** dialog. If this option is selected SQL Assistant saves the specified connection parameters in its configuration file. The connection can be reused later by selecting its name from the **Server Name** or **TNS Name** drop-down lists.

Connection Settings for new connections can be also added to the DB Options tab in SQL Assistant's **Options** dialog. See the "Managing Database Connections" topic in CHAPTER 48 for more information. Just as all other settings they can imported from previously saved SQL Assistant's configuration files and shared between team members, using the built-in Settings Export/Import feature. See the Sharing SQL Assistant Settings Between Team Members topic in CHAPTER 48 for more details.

Reordering, and Filtering Connections by Name

The **Filter** combo box and **Database Type** drop-down list at the top of the Connections dialog can be used to search and filter connections by their name and type. The **Filter** list serves both as a search and filter for the connections shown in the list below it. It is important to note that only connections with the type selected in the **Database Type** drop-down list are shown by default. To show all database connections available ignoring their database type, select the **"All database types"** checkbox in the right-bottom corner of the dialog. The filter strings you enter into the **Filter** box will be remembered so that you can quickly reuse them if you need them again.
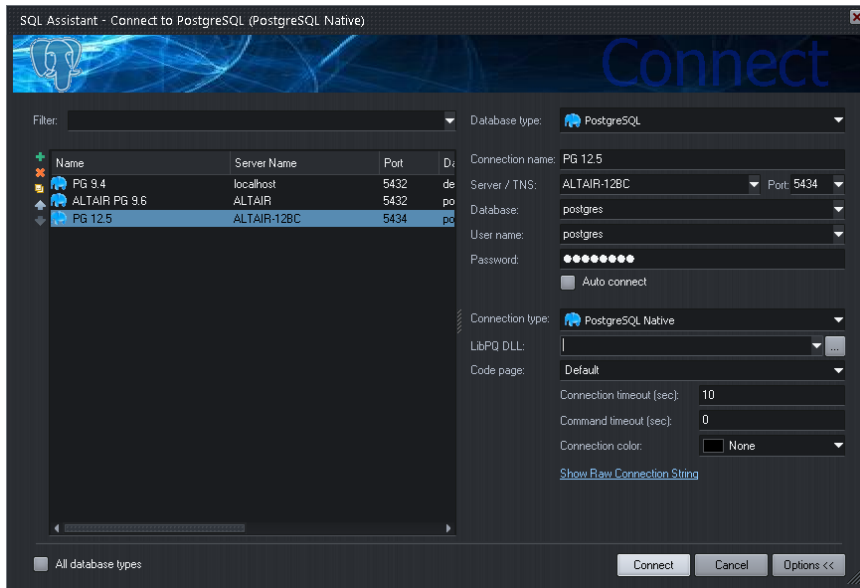
By default, most recently added connections appear at the top of the **Server Name** list. You can optimize the connection order for your environment using SQL Assistant's **Options** dialog. Activate **DB Options** tab then expand **DB Connections** section on the left. Use the Arrow Up ⬆ and Down ⬇ icons at the top of the connections list to rearrange connection order. You can also use drag-and-drop to quickly move connections in the list.

Organizing Connections

The **Connections Organizer** utility is provided to you to help you organize your connections in folders and subfolders and simplify working with many databases. The **Connection Organizer** can help to organize database connections by environments, by use cases, customers, data centers, or any other folder based organization of your choice. It also supports fast single double-click opening of an editor tab connected to a selected database which saves a few clicks and seconds required by other methods.

The **Connection Organizer** is available in SQL Assistant SQL Editor Professional Edition. To open the organizer, select File -> Connections … menu.
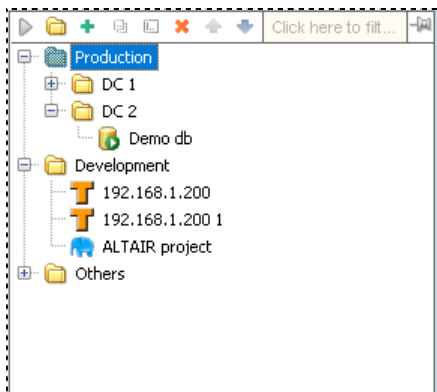
Connection Organizer Controls:

- You can use either right-click menu or toolbar buttons at the top of the organizer pane to add and

delete folders and connections.

- Existing connections not yet associated with a folder or subfolder initially appear in the special **Others** folder. The **Others** folder cannot be deleted or renamed. It is used as a catch-all folder for all connections not yet associated with a user-defined folder.

- Use drag-and-drop to move connections between folders. If the target folder is noti visible on the screen, you can use the right menu and select **Move to…** command. Then select the required folder in the next level menu.

- Double-click a database connection item in the organizer pane to open a new editor tab connected to that database server.

- To make the **Connection Organizer** pane sticky and automatically reopen after IDE restarts, click the *pushpin* icon in the right-top corner of the **Connection Organizer**. To hide the pane in the current session, close the pane. To remove stickiness, unpin the pane using pushpin icon.

Common Connection Properties and Database Driver Specific Properties

The following settings are supported with all database types and interfaces:

Connection Name – this property is provided for you to enter a descriptive identifier for the connection. If not entered, SQL Assistant will construct default name including several parts based on the database interface selected, typically database server or alias name followed by the selected database driver type, like ADO, ODBC, etc…

Connection Type – using this property you can choose one of the supported connection methods and drivers, for Oracle LibMySQL, OCI, ADO, ODBC, and so on. Different connectivity methods are supported for different database types. The connection methods and types are database type specific. First of all the Connection Type property enables you to choose the driver that you may already have installed on your system and so no additional software installations are required for connecting to your databases. But that's not all. If you have multiple types of drivers installed, you should choose carefully which one to use. All database drivers have their own functional differences and constraints, as well as defects (no software is bug free). You may need to choose different driver for different types of processing.

Connection Timeout (sec) – this property defines the maximum number of seconds that SQL Assistant will wait while attempting to open a connection to a database server. The default value for this property is 10 seconds. The property is used with most database interfaces and drivers.

Command Timeout (sec) - The number of seconds to wait while attempting to execute a SQL query, before cancelling the attempt and generating an error. The default is zero, which means the default timeout used by the selected database driver. This is typically "forever". However you shouldn't make any assumptions. Consult with the documentation supplied with the driver by the driver vendor for specific details.

Code Page – The Code Page property is used for selecting which character encoding to use when sending and retrieving data from the database server. For some types of database servers such as Oracle, MySQL, MariaDB, SQLite, it is important to choose the code page that matches your database server settings in order to render correctly all multinational characters.

Connection Color – This property is specific to SQL Editor Professional Edition installed with the SQL Assistant software Professional Edition license. It makes it easier to visually see which server an editor tab in the SQL Editor is connected to. The top of the connected editor tab and the background of the status bar section with the server name section are colorized according to your selection. For example, you can use that feature to make connections for the development or test servers shown in green, while important connections to production servers shown in orange or red.

Encrypted Connections and At Rest Data Encryption

The encryption settings are specific to the database type and the connectivity driver type selected for the database connection. Consult your database and driver documentation for additional details.

You can specify desired encryption settings on the **Encryption** tab in the **SQL Assistant - Connect** dialog.

SSH Tunnels

The **SSH Tunnel** connection options can be used to connect SQL Assistant through an intermediary proxy host to a database server that resides on a network to which you may not be able to connect directly from your computer. SQL Assistant supports both embedded SSH client as well as two types of external command line based SSH clients: Putty and OpenSSH. An OpenedSSH based client is available as Windows ssh command available on most Windows systems. Using the embedded client is recommended. It's more efficient and can better support multiple concurrent connections. SQL Assistant is a multi-threaded multi-connection application often using multiple concurrent connections for running background tasks.

To enable SSH tunnel connection using embded client:

1. Activate the **SSH Tunnel** tab in the Connect to Database dialog.

2. Select **Use SSH Tunnel** checkbox.

3. Enter name of the new **SSH profile** or if you already have one created, selected it fro mthe drop-down list. SSH Profiels are names groups of settings that can be reused with different database connections.

4. Specify the name or IP address of the SSH host through which the database connections will be forwarded in the **SSH host** field.

5. Specify the port of the SSH host (SSH server port) in the **SSH port** field.

6. Specify the name of a user with login privileges for the SSH host in the **SSH user** field. (Do not confuse this with the database user).

7. Specify the type of authentication to used for connecting to the SSH host in the Authentication field:

8. In case of password based autontication, enter the SSH user password in the **SSH password** field. In case of using public/private key based autontication, in the **Private Key** field select private key file for authentication, pleaseu se the Browse icon to select file to ensure the entered file is correct.

To enable SSH tunnel connection using external command line client:

1. Follow the same steps as described above for configuring SSH tunneling using an embded client.

2. Select the **Advanced settings** checkbox

3. Select the **User command line SSH client** checkbox.

4. In the **Program path** field enter the complete program path to either **ssh.exe** or **putty.exe** commands.

Bootstrap Queries

The **Bootstrap** settings in the **SQL Assistant - Connect** dialog can be used to specify bootstrap SQL commands executed at the start of every database session. For example, if you want to set your session timezone settings on each connection overriding local system defaults, set database schema search path, or other settings, you can add relevant SQL commands to the **Boostrap** tab. If you enter multiple commands, make sure each command is properly terminated, typically by a semicolon, so that they can be correctly executed. As with any other supported scripts SQL commands in the Bootsrap options can be entered on multiple lines.

Here are some examples:

Oracle
```
ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YYYY';
ALTER SESSION SET CURRENT_SCHEMA = SCOTT;
```

PostgreSQL
```
SET search_path TO "$user", public, 'some_other_chema';
SET timezone = 'UTC';
```

MySQL
```
SET time_zone = 'Europe/Helsinki';
SET SESSION sql_mode = 'TRADITIONAL';
USE some_database;
```

Snowflake
```
ALTER SESSION SET TIMEZONE = 'UTC';
ALTER SESSION SET LOCK_TIMEOUT = 3600;
```

Oracle Database Connections and Settings

Before you can connect to an Oracle database server, you must have Oracle client software installed on the computer running SQL Assistant. The client software must be properly installed and configured. Its network configuration files must include connection settings for any Oracle servers that SQL Assistant will connect to. These files must be located in the ORACLE_HOME directory whose name is referenced by the ORACLE_HOME variable in the system registry. This variable is stored under the registry key HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE.

If you have several versions of Oracle clients installed on your computer, you can choose which one you want to use with SQL Assistant. In the **SQL Assistant - Connect** dialog, click the **Options >>** button and then select the appropriate ORACLE_HOME from the Oracle homes drop-down list. Make sure the configuration files for the client you select contain all required Oracle server connection settings. You can verify connection settings in TNSNAMES.ORA, which is usually located in the [ORACLE_HOME]/NETWORK/ADMIN directory.

SQL Assistant supports both regular user connections and connections for users with SYSDBA privileges. You must choose SYSDBA type when connecting as a SYS user. The **Connect As** option on the **SQL Assistant -**

Connect dialog allows you to choose a database connection type. Please note that for SYSDBA connections, the assigned schema is SYS. For SYSOPER, the assigned schema is PUBLIC, regardless of the actual Oracle username supplied.

Important Note: When working with pre-configured targets for Oracle such as SQL*Plus, SQL Assistant silently intercepts the existing connection and uses that connection to access Oracle catalog tables and views. When working with user-defined targets, SQL Assistant opens a new connection to the database server that is not shared with the target application. This secondary connection can be opened using the same or different user credentials. The secondary connection is closed automatically in the event the target editor or SQL Assistant is closed.

SQL Server Database Connections and Settings

When connecting to a SQL Server database server, SQL Assistant uses either the Microsoft SQL Server ODBC driver that is by default pre-installed on all Windows systems or the SQL Server ADO.NET SQL Server provider which is typically installed with SQL Server Management Tools such as Microsoft SQL Server Management Studio or Microsoft Visual Studio. The connectivity method depends on the method used by the target editor.

SQL Assistant supports 2 types of user authentication for SQL Server connections:

- Windows Authentication - in this case, SQL Assistant does not prompt for user credentials. Instead, it uses an access token assigned at the time the user logged on using a Windows account. SQL Assistant only prompts for the name of the target SQL Server instance at the time the connection is attempted.

- Database Authentication -- in this case, SQL Assistant requires a user to specify login name, password and the name of the target SQL Server instance at the time the connection is attempted.

Important Notes:

When working with certain pre-configured targets for SQL Server that use ODBC connections, such as SQL Query Analyzer, SQL Assistant silently intercepts the existing connection and uses that connection to access SQL Server catalog tables and views.

When working with SQL Server Management Studio or SQL Server Management Studio Express that use ADO.NET connections, SQL Assistant silently intercepts the existing connection and uses that connection to access SQL Server catalog tables and views.

When working with user-defined targets and targets that don't use known connection types, SQL Assistant opens a new connection to the database server that is not shared with the target application. This secondary connection can be opened using the same or different user credentials. The secondary connection is closed automatically in the event the target editor or SQL Assistant is closed.

Important Notes for SQL Azure users:

For certain internal operations involving reverse-engineering of database schema objects SQL Assistant depends on Server Management Objects (SMO) installed with Microsoft SQL Server Management Studio (SSMS) and with Microsoft SQL Server Management Studio Express (SSMSE). The Cumulative Update Package 7 or later is required for supporting SQL Azure databases. If you are using an older versions of SSMS or SSMSE, please take the following actions:

If you use SSMSE 2008 R2 or an older version, download and install the latest version from http://go.microsoft.com/fwlink/p/?LinkId=220170.

If you use an edition of SSMS 2008 R2 other than Express, either install Cumulative Update Package 7 for SQL Server 2008 R2 http://support.microsoft.com/kb/2507770 or install Service Pack 1 http://www.microsoft.com/en-us/download/details.aspx?id=26727.

Apache Hive Connections and Settings

SQL Assistant supports Apache Hive versions 2.x and later. It uses ODBC connectivity to communicate with a Hive system. ODBC connectivity requires that Apache Thrift server be installed on the Hive cluster master node.

Hive ODBC drivers are available from a number of vendors including Microsoft, Amazon AWS, HortonWorks Cloudera, and others. In theory ADO.NET interface can be also used for the connections. However at the time of the publishing of this user's guide SQL Assistant has not been formally tested with Hive ADO.NET connections.

Here are references to ODBC drivers available at the time of SQL Assistant version 11.0 release:

Amazon AWS EMR ODBC driver: https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-bi-tools.html
Microsoft Hive ODBC driver: https://www.microsoft.com/en-us/download/details.aspx?id=40886
Cloudera, Inc: https://www.cloudera.com/downloads/connectors/hive/odbc/2-6-1.html

Apache SparkSQL Connections and Settings

SQL Assistant supports Apache SparkSQL versions 2.2 and later. It uses ODBC connectivity to communicate with a Apache Spark system. ODBC connectivity requires that Apache Thrift server be installed on the Spark cluster master node.

Spark ODBC drivers are available from a number of vendors including Microsoft, HortonWorks Cloudera, and others. In theory ADO.NET interface can be also used for the connections. However at the time of the publishing of this user's guide SQL Assistant has not been formally tested with Spark ADO.NET connections.

Here are references to ODBC drivers available at the time of SQL Assistant version 11.0 release

Microsoft SparkSQL ODBC driver: https://www.microsoft.com/en-us/download/details.aspx?id=49883
Simba Technologies, Inc: https://www.simba.com/drivers/spark-jdbc-odbc/
Databricks, Inc.: https://databricks.com/spark/odbc-driver-download

MySQL Database Connections and Settings

SQL Assistant presently supports MySQL database server versions 5.1 and up. To connect to a MySQL database server, SQL Assistant uses either the MySQL native client interface, if available, or the MySQL ODBC or ADO.NET driver. These MySQL drivers are not part of SQL Assistant. The connector and/or driver must be preinstalled before you can use SQL Assistant with MySQL targets. MySQL client and ODBC driver can be obtained from Oracle Corporation http://www.mysql.com.

Important Notes:

For ODBC-based connections, SQL Assistant requires the MySQL ODBC driver version 3.5. Please note that MySQL ODBC driver v5.0, which is available as a beta version download at the time this manual was produced, is not fully ODBC compliant and does not support certain standard ODBC features required by SQL Assistant.

When working with pre-configured targets for MySQL, such as MySQL Query Browser, SQL Assistant silently intercepts the existing connection and uses that connection to access MySQL catalog tables and views. When working with user-defined targets, SQL Assistant opens a new connection to the database server using an ODBC driver or native **libmysql** client. This secondary connection is not shared with the target application. It can be opened using the same or different user credentials. The secondary connection is closed automatically if the target editor or SQL Assistant is closed.

MariaDB Database Connections and Settings

SQL Assistant presently supports MariaDB database server versions 5.5 and up. To connect to a MariaDB database server, SQL Assistant uses either the MariaDB native client interface, if available, or the MySQL or MariaDB ODBC or ADO.NET driver. These drivers are not part of SQL Assistant. The connector and/or driver must be preinstalled before you can use SQL Assistant with MariaDB targets. MariaDB client and ODBC driver can be obtained from MariaDB Foundation http://www.mariadb.org/.

Important Notes:

For ODBC-based connections, SQL Assistant requires the MySQL ODBC driver version 3.5.

When working with pre-configured or user-defined targets for MariaDB, targets, SQL Assistant opens a new connection to the database server using an ODBC driver or native **libmariadb** client. This secondary connection is not shared with the target application. It can be opened using the same or different user credentials. The secondary connection is closed automatically if the target editor or SQL Assistant is closed.

DB2 Database Connections and Settings

SQL Assistant presently supports DB2 UDB for Linux, Unix and Windows versions 7 and up, and DB2 for iSeries. It can be also used with DB2 for zSeries, but the latter requires that minor adjustments be made to the database catalog queries provided in SQL Assistant settings. To connect to a DB2 database server, SQL Assistant can use either the DB2 ODBC driver or the DB2 ADO.NET client, neither of which is part of SQL Assistant. One of these drivers must be preinstalled before you can use SQL Assistant with DB2 targets. DB2 drivers are typically installed with DB2 client software and are part of your DB2 license. The drivers can be obtained from IBM Corporation http://www.ibm.com.

Important Notes:

When working with user-defined targets for DB2, SQL Assistant opens a new connection to the database server that is not shared with the target application. This secondary connection can be opened using the same or different user credentials. The secondary connection is closed automatically in the event the target editor or SQL Assistant is closed.

Netezza Database Connections and Settings

SQL Assistant presently supports IBM Netezza versions 7,0 and up. Older versions are not supported. To connect to a Netezza database server, SQL Assistant can use Netezza ODBC driver, which is not part of SQL Assistant. The ODBC driver must be preinstalled before you can use SQL Assistant with Netezza targets. Netezza drivers are typically installed with Netezza client software and are part of your Netezza license. They are not publicly available, you need to contact Netezza's support to download their ODBC driver.

Important Notes:

When working with user-defined targets for Netezza, SQL Assistant opens a new connection to the database server that is not shared with the target application. This secondary connection can be opened using the same or different user credentials. The secondary connection is closed automatically in the event the target editor or SQL Assistant is closed.

Sybase Database Connections and Settings

SQL Assistant presently provides support for Sybase Adaptive Server Enterprise versions 12.5 and up as well as for Sybase Adaptive Server Anywhere v9.0 and up (also known as Sybase SQL Anywhere). In order to connect to a Sybase database server, SQL Assistant uses the Sybase ODBC driver which is not part of SQL Assistant. The driver must be preinstalled before you can use SQL Assistant with Sybase targets. Sybase ODBC drivers are typically installed either with Sybase client software or separately. They are part of your Sybase license. The drivers for ASE and ASA can be obtained from Sybase Corporation http://www.sybase.com.

Important Notes:

When working with Sybase targets, SQL Assistant opens a new connection to the database server that is not shared with the target application. This secondary connection can be opened using the same or different user credentials. The secondary connection is closed automatically in the event the target editor or SQL Assistant is closed.

PostgreSQL Database Connections and Settings

SQL Assistant presently provides support for PostgreSQL version 8.0 and up. It is likely that SQL Assistant can also be used with PostgreSQL version 7.x.; however, no specific testing has been done for 7.x versions, and there is no guarantee that SQL Assistant is fully compatible with them.

To connect to a PostgreSQL database server, SQL Assistant can use either the PostgreSQL ODBC driver, the ADO.NET driver, or the LibPQ native interface. Neither ODBC or ADO.NET are part of SQL Assistant. If you choose ODBC or ADO.NET, the appropriate driver must be preinstalled before you can use SQL Assistant with PostgreSQL targets. PostgreSQL drivers can be freely downloaded from the PostgreSQL Foundry project web site http://pgfoundry.org/projects/psqlodbc/.

In comparison, the LibPQ native interface is preinstalled with SQL Assistant, both 32-bit and 64-bit versions are

available, and it is ready to use out of the box. It is also the most capable interface and that is why it is recommended as a preferred connectivity method. Using LibPQ interface SQL Assistant is able to process COPY command with STDOUT/STDIN options, asynchronously output diagnostic messages, such as notes, warnings, and exceptions while executing SQL queries, and so on.... there are so other features that are better handled with LibPQ.

Important Notes:

When working with PostgreSQL targets, SQL Assistant opens a new connection to the database server that is not shared with the target application. This secondary connection can be opened using the same or different user credentials. The secondary connection is closed automatically in the event the target editor or SQL Assistant is closed.

Redshift Database Connections and Settings

Amazon Redshift database server is based on PostgreSQL 8.0.2. Even though it has a number of very important differences from PostgreSQL, the connectivity to Amazon Redshift databases is the same as to PostgreSQL databases. To connect to an Amazon Redshift database , SQL Assistant can use either the Amazon Redshift ODBC driver, PostgreSQL ODBC driver, the ADO.NET driver, or the LibPQ native interface. Neither ODBC or ADO.NET is part of SQL Assistant. If you choose ODBC or ADO.NET, The driver must be preinstalled before you can use SQL Assistant with Amazon Redshift targets. Amazon Redshift ODBC drivers can be freely downloaded from Amazon Web Services web site http://docs.aws.amazon.com/redshift/latest/mgmt/install-odbc-driver-windows.html. PostgreSQL drivers can be freely downloaded from the PostgreSQL Foundry project web site http://pgfoundry.org/projects/psqlodbc/.

On contrary, the LibPQ native interface is preinstalled with SQL Assistant, both 32-bit and 64-bit versions are available. It is also the most capable interface and that is why it is recommended as a preferred connectivity method. Using LibPQ interface SQL Assistant is able to process COPY command with STDOUT/STDIN options, asynchronously output diagnostic messages, such as notes, warnings, and exceptions while executing SQL queries, and so on.... there are also other features that are better handled with LibPQ.

Important Notes:

When working with Amazon Redshift targets, SQL Assistant opens a new connection to the database server that is not shared with the target application. This secondary connection can be opened using the same or different user credentials. The secondary connection is closed automatically in the event the target editor or SQL Assistant is closed.

Greenplum Connections and Settings

Pivotal Greenplum database servers are based on PostgreSQL versions. The connectivity to Greenplum databases is the same as to PostgreSQL databases. To connect to an Greenplum database , SQL Assistant can use either the PostgreSQL ODBC driver, the ADO.NET driver, or the LibPQ native interface. Neither ODBC or ADO.NET is part of SQL Assistant. If you choose ODBC or ADO.NET, The driver must be preinstalled before you can use SQL Assistant with Greenplum targets. PostgreSQL drivers can be freely downloaded from the PostgreSQL Foundry project web site http://pgfoundry.org/projects/psqlodbc/.

On contrary, the LibPQ native interface is preinstalled with SQL Assistant, both 32-bit and 64-bit versions are available. It is also the most capable interface and that is why it is recommended as a preferred connectivity method. Using LibPQ interface SQL Assistant is able to process COPY command with STDOUT/STDIN

options, asynchronously output diagnostic messages, such as notes, warnings, and exceptions while executing SQL queries, and so on.... there are also other features that are better handled with LibPQ.

Important Notes:

When working with Greenplum targets, SQL Assistant opens a new connection to the database server that is not shared with the target application. This secondary connection can be opened using the same or different user credentials. The secondary connection is closed automatically in the event the target editor or SQL Assistant is closed.

Teradata Database Connections and Settings

SQL Assistant presently supports Teradata versions 14 and 15. To connect to a Teradata database server, SQL Assistant can use either the Teradata ODBC driver or the Teradata ADO.NET client, neither of which is part of SQL Assistant. One of these drivers must be preinstalled before you can use SQL Assistant with Teradata targets. Teradata drivers are typically installed with Teradata client software and are part of your Teradata license. The drivers can be freely downloaded from Teradata Developer Exchange web site http://downloads.teradata.com/download/connectivity.

Important Notes:

When working with user-defined targets for Teradata, SQL Assistant opens a new connection to the database server that is not shared with the target application. This secondary connection can be opened using the same or different user credentials. The secondary connection is closed automatically in the event the target editor or SQL Assistant is closed.

Microsoft Access Database Connections and Settings

SQL Assistant presently provides direct integration and support for Microsoft Access databases, versions 2003, 2007, and 2010. It can also be used with earlier versions of Microsoft Access database using ODBC connections. To connect to a Microsoft Access database, SQL Assistant uses the Microsoft Access ODBC driver which is not part of SQL Assistant. The driver must be preinstalled before you can use SQL Assistant with Microsoft Access targets. Microsoft Access ODBC drivers are typically pre-installed on all Windows systems. The latest ODBC driver versions can be obtained from Microsoft Corporation http://www.microsoft.com.

Important Notes:

When working with Microsoft Access targets, SQL Assistant opens a new connection to the database file, which is not shared with the target application. This secondary connection can be opened using the same or different user credentials. The secondary connection is closed automatically in the event the Microsoft Access IDE, or non-Access target editor or SQL Assistant is closed.

SQLite Database Connections and Settings

To connect to a SQLite database, SQL Assistant can use either of the following, SQLite wire driver, SQLite ODBC driver or SQLite ADO.NET driver. SQLite wire driver is part of the SQL Assistant software, the others are not. The latest ODBC driver for SQLite can be obtained from Christian Werner http://www.ch-werner.de/sqliteodbc/.

Important Notes:

Using SQLite wire driver connections is recommended in most cases. The driver pre-instaleld with SQL Assistant provides the most features, and is also a requirement for using regular expressions and JSON functions in SQL queries. If you already have saved SQLite connections created in previous versions of SQL Assistant using third party ODBC drivers or SQLite ADO.NET driver, you can continue using them unmodified, but please be aware that some SQL Assistant features may malfunction or don't work at all when using other drivers.

MongoDB Database Connections and Settings

Prerequisites. To connect to a MongoDB database, SQL Assistant requires MongoDB Connector for Business Intelligence (BI) connection which supports using SQL queries for working with MongoDB data. The connection can be made to a cloud-hosted MongoDB Connector for BI which is available in MongoDB Atlas, or to a locally running instance of MongoDB Connector for BI. The MongoDB Connector for BI accepts SQL queries using the MySQL protocol, which enables you to query MongoDB the same way you would query a MySQL database.

To connect to a MongoDB Connector for BI, SQL Assistant uses the ODBC driver, which isn't part of SQL Assistant installation. The driver must be preinstalled before you can use SQL Assistant with MongoDB targets. The latest ODBC driver version can be obtained from MongoDB site ttps://www.mongodb.com/blog/post/odbc-driver-for-the-mongodb-connector-for-business-intelligence.

Important Notes:

In the connection settings be sure to specify the host and port number of your MongoDB Connector for BI and not your MongoDB server.

Apache Cassandra Database Connections and Settings

SQL Assistant supports Apache Cassandra 3.0 and later, and also supports cloud based DataStax Enterprise and DataStax AstraDB databases versions 6.x and later built on Apache Cassandra. To communicate with an Apache Cassandra cluster SQL Assistant requires either a native driver preinstalled with SQL Assistant or a compatible ODBC driver for using ODBC connectivity.

Both the Cassandra native driver provided with SQL Assistant and third-party ODBC driver may provide you with an extended set of functions not available natively in Cassandra, which you may require for your applications, and use with SQL Assistant too. The features are vary by driver. For example, some ODBC drivers provide support for SQL expressions, INSERT-SELECT operations and others SQL functions that are not supported by CQL language out of the box. Choose the driver that supports the features you require. Cassandra ODBC drivers are available from a number of vendors including CData Software, Progress Software, and others. Free ODBC driver is also available from DataStax Corporation https://downloads.datastax.com/#odbc-jdbc-drivers   Most ODBC drivers for Cassandra support various

authentication methods including basic authentication with login credentials, Kerberos, and LDAP. Please consult your ODBC driver documentation for more details and check with your Cassandra administrators on which authentication method you should use.

Snowflake Database Connections and Settings

SQL Assistant supports Snowflake 6 and later. SQL Assistant uses ODBC connectivity to communicate with an Apache Cassandra cluster, which requires a compatible ODBC driver.

Cassandra ODBC drivers are available from a number of vendors including CData Software, Progress Software, and others. Free ODBC driver is also available from DataStax Corporation https://downloads.datastax.com/#odbc-jdbc-drivers

Most ODBC drivers for Cassandra support various authentication methods including basic authentication with login credentials, Kerberos, and LDAP. Please consult your ODBC driver documentation for more details and check with your Cassandra administrators on which authentication method you should use.

Database Connection Settings and Security

By default, SQL Assistant remembers used connection settings and user credentials and stores them in the configuration file SqlAssist.sas file. Your connection settings, user names and passwords are stored in encrypted form so they cannot be read or changed without your authorization.

If, for whatever reason, you do not want to save connection settings between work sessions, you can uncheck the **Save Connection Data** option on the **SQL Assistant - Connect** dialog. This option is not visible by default. To display it, click the **Options >>** button.

If a database connection attempt fails because your user name or password has changed since it was last saved, SQL Assistant automatically prompts you to re-enter your connection settings and updates them in the configuration file.

You can use SQL Assistant's Options dialog to change your connection password proactively and to modify other connection settings. See the "Managing Database Connections" topic in CHAPTER 48 for more information.

Tip: Certain types of database drivers support connection encryption features. Use the **Encrypt Connection** property available in SQL Assistant options to activate that feature. See the "Overview" topic in CHAPTER 2 for more information.

Shared, Automatic, and Interactive Database Connections

Shared Connections

SQL Assistant supports a database connection sharing feature that can be used with certain types of target editors. The connection sharing feature allows SQL Assistant to automatically intercept and join an existing connection established by the target, which eliminates the need for you to set up a separate connection for SQL Assistant. From a database server point of view, a shared connection is represented by one physical connection established to the database that is used jointly by the target editor and SQL Assistant.

Another great advantage of a shared connection is that the current database context is also shared. All changes to the context (for example, database switches and user impersonations) are fully transparent to SQL Assistant. As a result, all SQL Assistant prompts and suggestions always match the current context and are always correct.

As of SQL Assistant version 7.0, the following targets are compatible with this feature:

- SQL Server 2005/2008 Management Studio, Full and Express editions
- SQL Server 2012/2014/2016 Management Studio, Full and Express editions
- Certain editions of Visual Studio.NET/2005/2008
- Visual Studio 2010, 2012, 2013, 2015 Professional and Ultimate editions
- SQL Server Query Analyzer
- MySQL Query Browser
- Oracle SQL*Plus
- DB Tools for Oracle
- SQL Assistant's SQL Editor Professional and Standard editions

Automatic and Interactive Database Connections

For all other target editors, SQL Assistant supports regular types of database connections requiring you to fill in the connection properties in the SQL Assistant's interactive **Connect to Database** dialog. See the "Overview" topic in CHAPTER 2 for more details. Alternatively, such connections can be pre-configured in advance, before they are needed for accessing the database during code editing. Connections can be pre-configured on the DB Options tab in SQL Assistant's **Options** dialog. See the "Managing Database Connections" topic in CHAPTER 48 for more information.

From a database server point of view, a non-shared connection is represented by two physical connections established to the database, with one connection established and used exclusively by the target editor and one connection established and used exclusively by SQL Assistant. Note that when SQL Assistant is used with generic text and code editors such as Notepad, Notepad++, UltraEdit, and other editors not supporting database connectivity, only one physical database connection is established.

By default, SQL Assistant remembers entered connection parameters and saves them for future use. It prompts only once to enter connection parameters. Next time you open the same editor and start typing the code, SQL Assistant loads previously saved parameters and automatically establishes a new database connection without prompting you again for connection parameters. This **Auto-connect** behavior is optimal for most people who work with a single database system and always connect to the same server and database.

If you work with multiple database servers, you can disable the **Auto-connect** feature. When this feature is disabled, SQL Assistant prompts for a new database connection whenever it needs one to complete its first database operation. When you get that prompt, you can choose either of the previously saved connections from the server drop-down or you can enter a new set of connection parameters.

The **Auto-connect** feature can be customized individually for each registered target editor. Different settings can be chosen for different targets. See CHAPTER 49, Registering and Configuring Targets for SQL Assistance topic for more details.

Custom Connection Parameters

SQL Assistant enables you to define custom connection parameters. You can use custom parameters to fine tune your database connections and make working with your databases more efficient.

For ad-hoc database connections, use the raw connection settings section on the Connect to Database dialog. First click the **Options >>** button to expand the connection options, and then click the **Show Raw Connection String** hyperlink to show the raw connection settings as demonstrated on the following diagram.
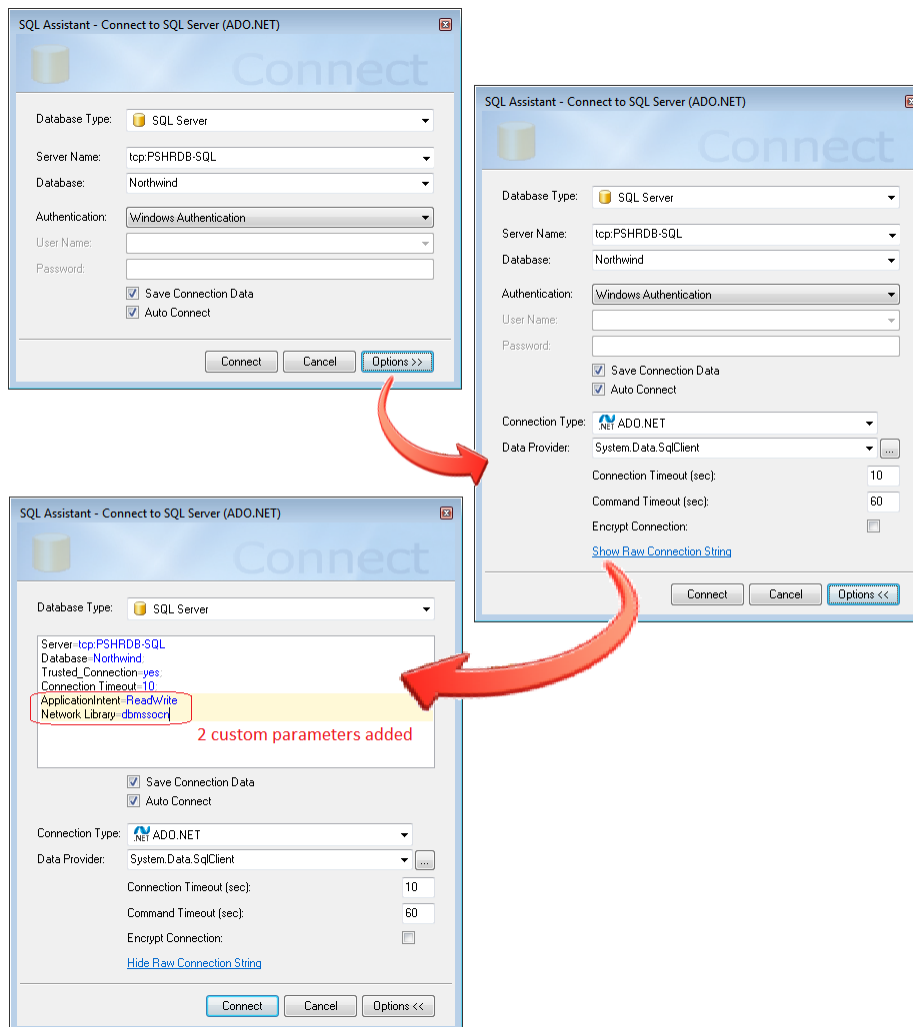
Add each custom connection parameter on a new line below the standard parameters. Note that the lines with custom parameters are highlighted with light beige background to make it easier to distinguish them from the lines with standard parameters.

Important Note: You must ensure that the custom parameters specified in the raw connection settings are supported by the database driver and client software and by your database server version. Consult your database driver and client software documentation for a list of supported parameters and their formats.

For remembered connections, use the Custom Settings option in database connection settings saved in the SQL Assistant Options.
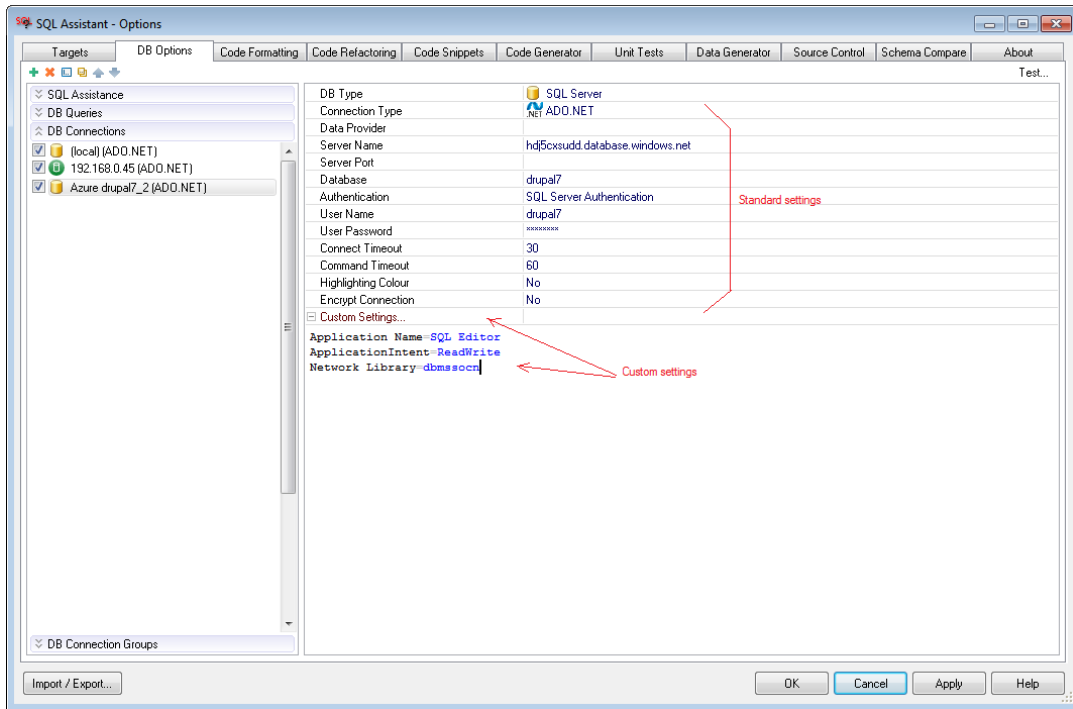
Important Note: You must ensure that the custom parameters specified in the connection settings are supported by the database driver and client software and by your database server version. Consult your database driver and client software documentation for a list of supported parameters and their formats.

You can use the **Test...** button available in the right-top corner of the **DB Options** tab to test the connection and to verify that the entered connection parameters are compatible with the selected database driver and supported by your database server.

Automatic Recovery of a Broken Database Connection

Because of SQL Assistant's unique ability to share connections with certain types of SQL editor targets, it is heavily dependent on the availability and speed of the database connection. In certain situations, if a database connection breaks, SQL Assistant may attempt to automatically repair the broken connection in order to maintain uninterrupted code entry. If an attempt to repair broken connection fails, SQL Assistant displays its own **Connect to Database** dialog independent of the target editor. You can use this dialog to enter the connection details as described in other topics of this chapter. For more information, see topics specific to your database type:

Oracle Database Connections and Settings

SQL Server Database Connections and Settings

MySQL Database Connections and Settings

DB2 Database Connections and Settings

Sybase Database Connections and Settings

PostgreSQL Database Connections and Settings

Amazon Redshift Database Connections and Settings

Microsoft Access Database Connections and Settings

SQLite Database Connections and Settings

CHAPTER 3, Code Assistants and SQL Intellisense

Starting and Stopping SQL Assistant

During installation, the setup program places a SQL Assistant shortcut in the Windows Startup folder so that when you logon to the system Windows, loads SQL Assistant automatically. To start SQL Assistant manually, simply click the SQL Assistant shortcut in the SQL Assistant Program Folder or run the SqlAssist.exe file.

When SQL Assistant is running, its icon appears in the Window system tray.
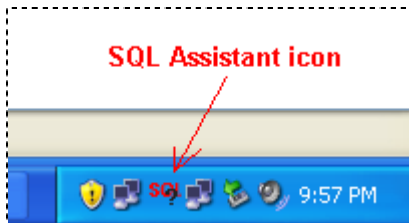
To exit SQL Assistant:

1. Right-click the SQL Assistant icon in the Window system tray.

2. Select **Exit** command from the popup menu.

Tip: If SQL Assistant is registered as an add-on, the host program loads the add-on automatically on startup or when a new instance of SQL editor is opened from within the host program. The host program automatically unloads the add-on when the program quits. If SQL Assistant is registered as an add-on, you have the option of running SQL Assistant as an icon in the system tray. In this case you can disable the "Load on Windows Startup" option in SQL Assistant options. See the "Managing SQL Assistant Load Methods" topic for more information.

Temporarily Pausing SQL Assistant

To pause SQL Assistant services without exiting the program:

1. Right-click the SQL Assistant icon in the Window system tray.

2. Select the **Suspend** command from the popup menu. SQL Assistant icon in the system tray changes color from red to gray.

To resume SQL Assistant services:

1. Right-click the SQL Assistant icon in the Window system tray.

2. Select the **Resume** command from the popup menu. SQL Assistant resumes its normal activities. SQL Assistant icon in the system tray changes color from gray back to red.

Tip: If you have SQL Assistant and target editor menu integration enabled, you can use SQL Assistant's Suspend and Resume commands available from the editor's right-click context menu and/or top level menu.

SQL Intellisense

SQL Assistant provides advanced code Intellisense features to make your SQL programming experience more productive. When you are working in the SQL code editor, SQL Assistant suggests code matches for auto-completing code items you type into the editor. It may also display additional informational about SQL keywords, statement syntax, referenced objects, columns, parameters, and much more. SQL Intellisense uses various types of popup prompts, menus, and tooltips. To accept an Intellisense suggestion for code completion, press the item completion character (by default, the Enter key). Alternatively, you can ignore the Intellisense suggestion and continue typing the code. To dismiss the prompt immediately, press the Esc key.

In addition, the SQL Intellisense features allow you to graphically construct complete SQL queries using just your computer mouse. For example, when you type the keyword "SELECT," Intellisense displays a list of databases, schemas, and tables. You can click on the [+] sign before the table name (or expand the database or the schema name first, and then expand the table), then the table columns are listed. You can mark the checkboxes in front of the column names that you want for the query. You now have a complete SQL SELECT statement with column names and tables completed for you. Similarly, you can click through JOINS and select table joins, WHERE conditions, and other options and SQL operation types.
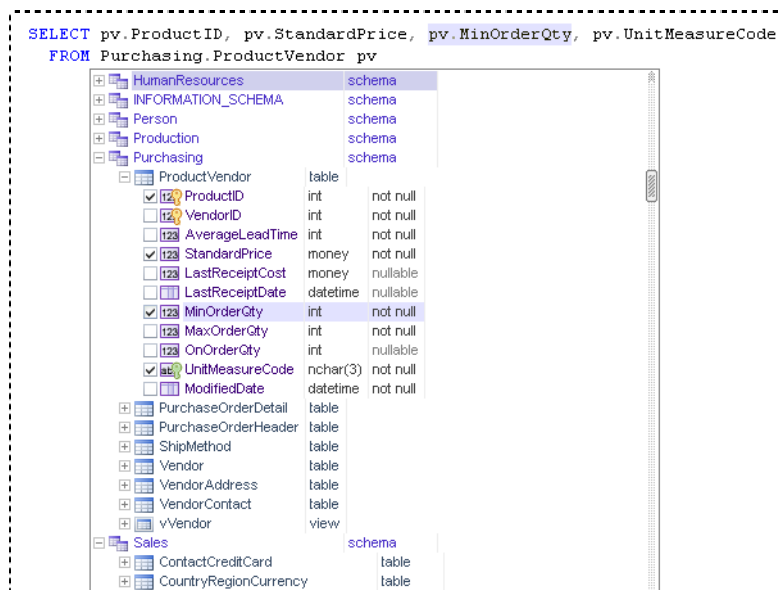
Following are some examples of SQL Intellisense popups provided by SQL Assistant.

The remainder of this chapter explains these features in more detail.

Tips:

- Intellisense behavior is dependent on database type and on many other factors. SQL Assistance provides numerous customization options you can use to tune Intellisense behavior options to match your coding habits as well as to queries and options specific to your database type .

- Most of the options controlling Intellisense behavior can be found either on the **DB Options** tab in the **Options** dialog within the General section or within the **Auto Complete** section.

- SQL Assistant Intellisense supports several different name matching methods it can use to match database entries, keywords, and other text you type in the editor. They described in detail in CHAPTER 6, Database Explorer, in Content Filtering and Sorting topic. In the **Auto Complete** section in the Options you can choose the method that you would like to use as the default method.

You can select different methods for different databases. You can also change the method for the current session on the fly. In SQL Assistant popup, right-click anywhere within the popup box, and in the context menu select the **Name Matching Method** menu and then the desired method..

SQL Assistant Windows and Appearance

SQL Assistant windows and popups have a consistent look in all target editors. However their availability, position, and management functions depend on the target editor type and the chosen SQL Assistant options. For example, when working within SQL*Plus, which comes as one of the pre-registered targets, the SQL Reference window appears on the right hand side of the screen while in all other pre-registered targets, it is displayed, by default, as a docked window on the left side of the target editor screen.

Below is an example SQL*Plus screenshot demonstrating several active SQL Assistant windows.
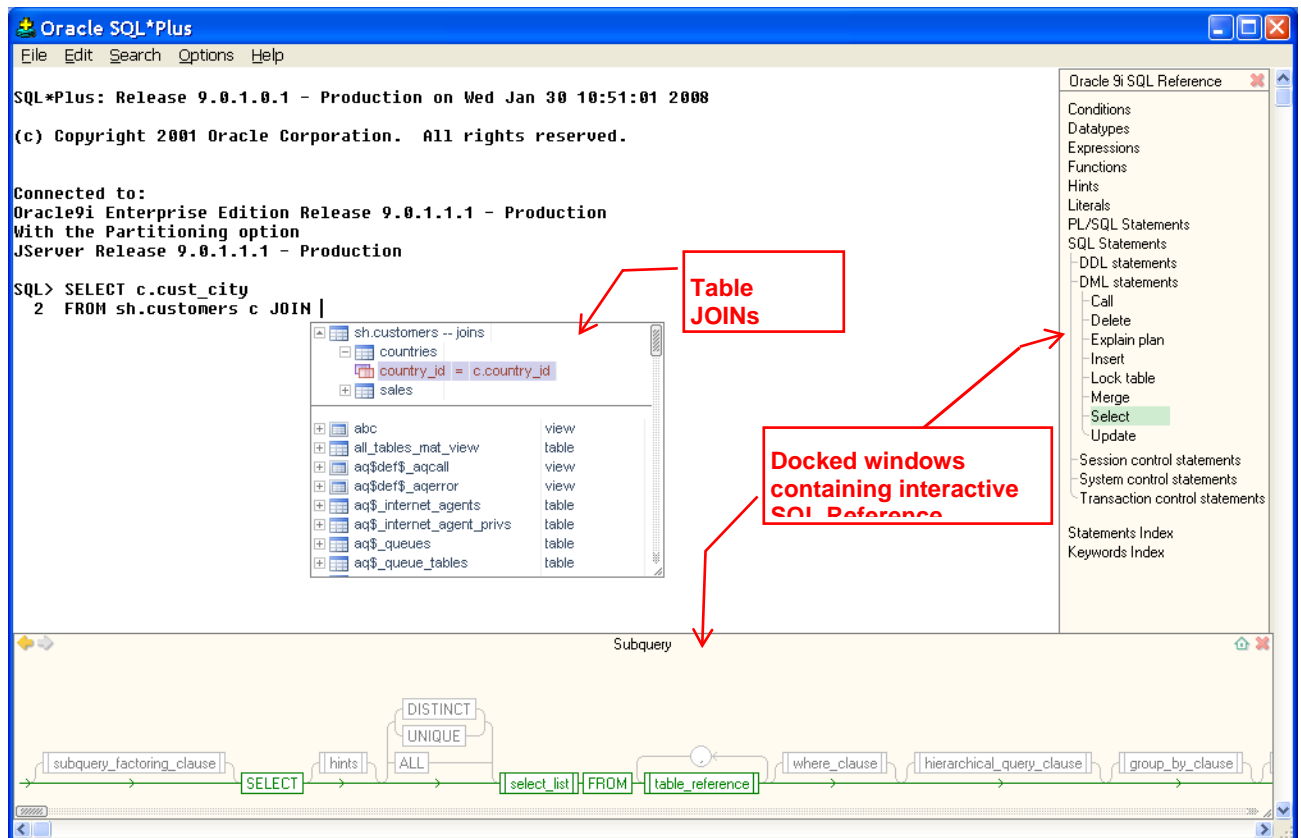
In comparison, a similar set of SQL Assistant windows in Microsoft SQL Server Query Analyzer may look like in the following example.
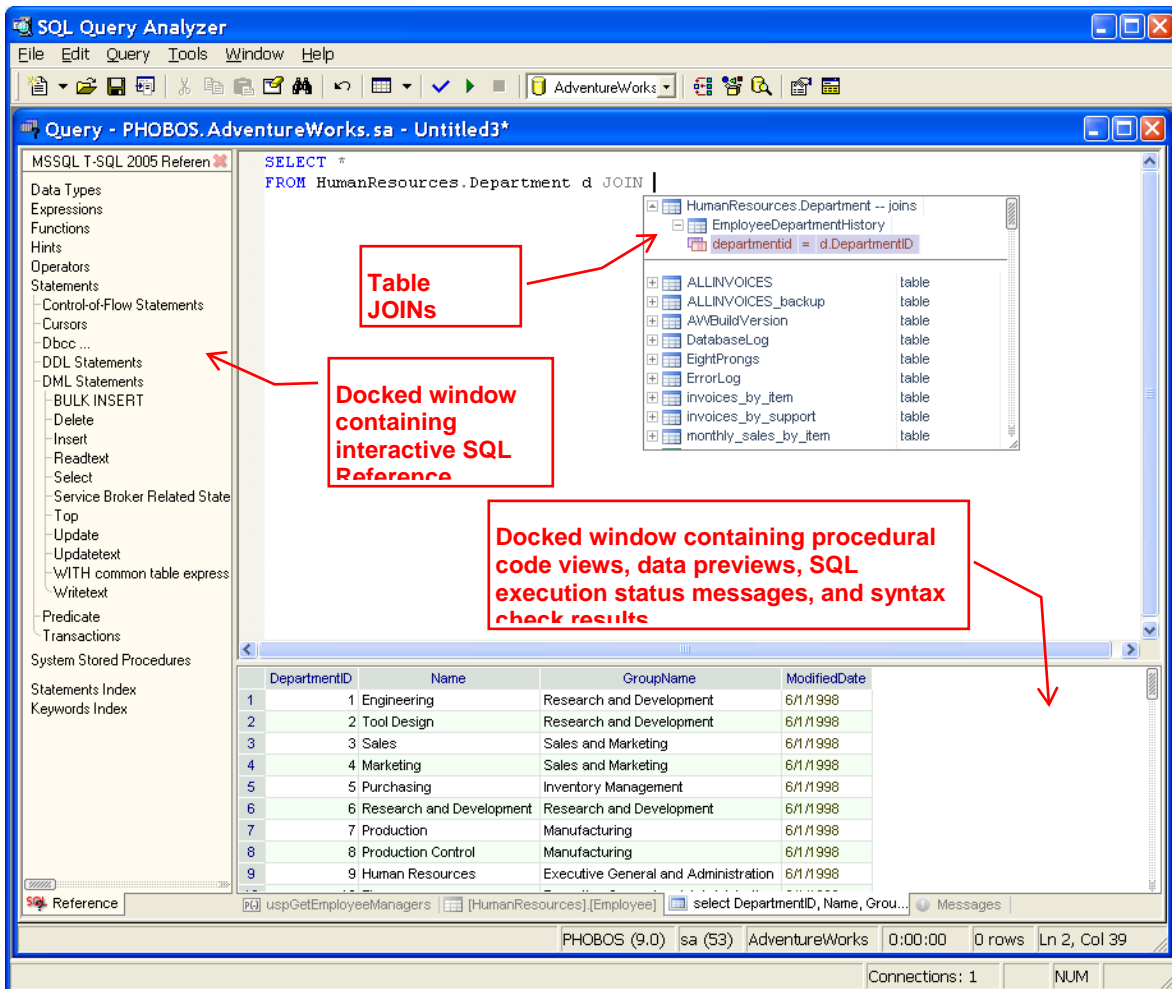
There are two types of SQL Assistant windows that can appear in the target application workspace:

- **Docked windows** are windows docked to the sides of the editor workspace area. SQL Assistant supports both vertical and horizontal docked Windows. The SQL Reference System and Code Structure View are always displayed in the vertically docked window. They share the same window and appear as two different tab pages. All other SQL Assistant windows appear as tabs in the horizontally docked window at the bottom part of the editor workspace area.

  The SQL Assistant's Integrated Development Environments (IDE) supports multiple editor windows that can be arranged in a multi-tab and multi-document interfaces. Each editor window can have its own different set of docked windows attached to it.

  Docked windows can be resized separately or together with the target editor. Docked windows can be repositioned and attached to different sides of the editor using the context menu over the tab, as demonstrated on the following screenshot.
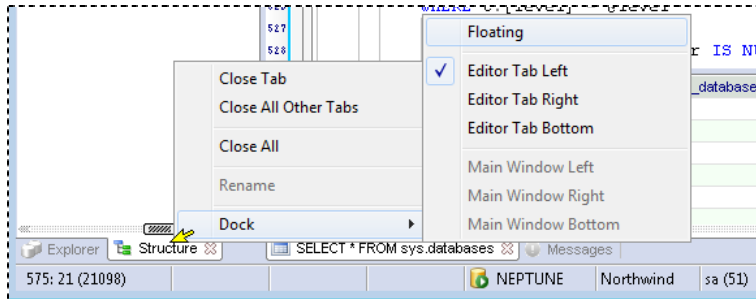
Certain types of docked windows can be attached to the main window which is not editor tab/window specific and appears visible all the time.

Docked windows can be also floated to make them target editor window independent. They can be moved to a second monitor or stacked up if desired.. To make a window floating, right-click the tab of the docked window and from the content menu choose **Dock -> Floating** menu. You can resize the floated window as needed. Note that the floating windows by default appear on top of all other windows. This feature is by design. If they obscure other applications, you can resize them, move them, or restore their docked state within the target editor window. You can also disable the top-most feature using the right-click menu over the floating window title bar, as shown on the following screenshot.
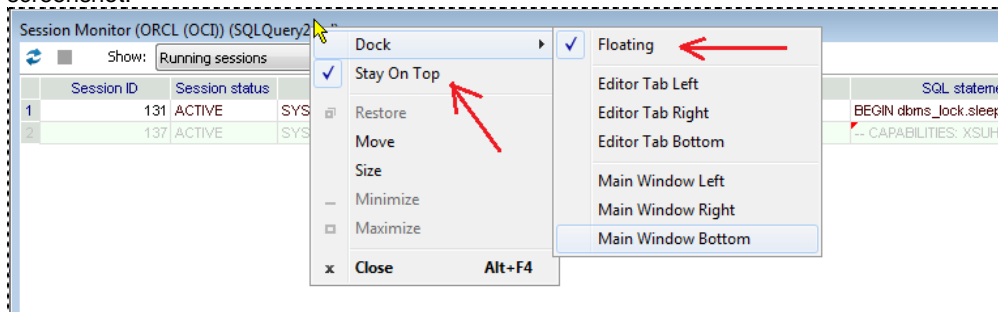
To dock a floating window, right-click the window's title bar and choose the appropriate docking command from the right-**Dock** menu.

- **Popups** are kind of floating windows displayed next to the editing position. They are part of SQL Intelligence features and unlike other floating windows they appear and disappear automatically depending on the position of your mouse cursor in the SQL code. Popup windows can be moved and resized.

Manually Invoking SQL Assistant Popups Using Keyboard Hot Keys

If, for whatever reason, a SQL Assistant popup is not displayed at the current cursor position and you want to display the popup, you can use the global Ctrl+Space hot key or a custom hot key if you changed the default key to something else. Read the following topics for more details. The hot key used to open a SQL Assistant popup can be changed on the Options dialog. For more information, see the "Customizing Hot Keys" topic in CHAPTER 48.

Press the Esc key at any time to immediately close an SQL Assistant popup.

Manually Invoking SQL Assistant Popups Using Context and Top-level Menus

To avoid the need to remember numerous hot key sequences, SQL Assistant supports direct integration with top-level right-click popup menus available in your development environments. To use SQL Assistant functions, right-click on the text in the editor where you want to invoke SQL Assistant, then select the top-most item in the context menu. The item text should read **SQL Assistant** and lead to the next menu level containing specific SQL Assistant functions.

Note: In editors like SQL*Plus that do not support right-click context menus, SQL Assistant creates its own right-click menu. The following example, demonstrates SQL Assistant commands displayed in Microsoft SQL Query Analyzer context menu.
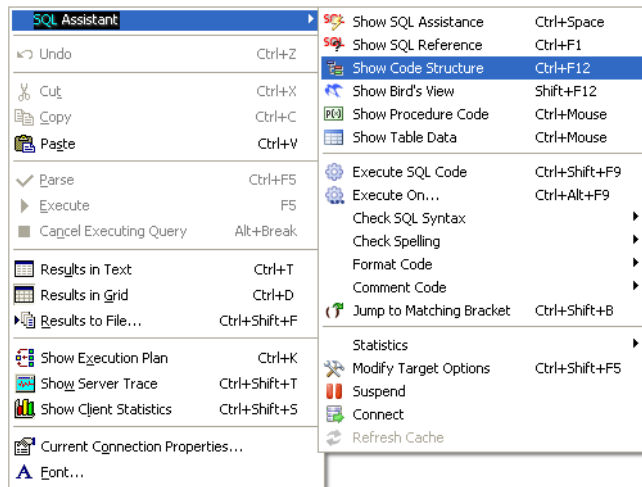
Right-click context menu integration is enabled by default in all pre-configured and new targets, while top-level menu integration is disabled by default. See the "Customizing Target Editor Menu Integration" topic in CHAPTER 8 for more information on how to customize SQL Assistant integration with target editor menus.

Manually Invoking SQL Assistant Popups from the System Tray

All commands available in the SQL Assistant menus are can also be invoked from the SQL Assistant icon in the Windows system tray. Right-click on the system tray icon to display the menu, then choose the **Target** menu branch as shown in the following screenshot.
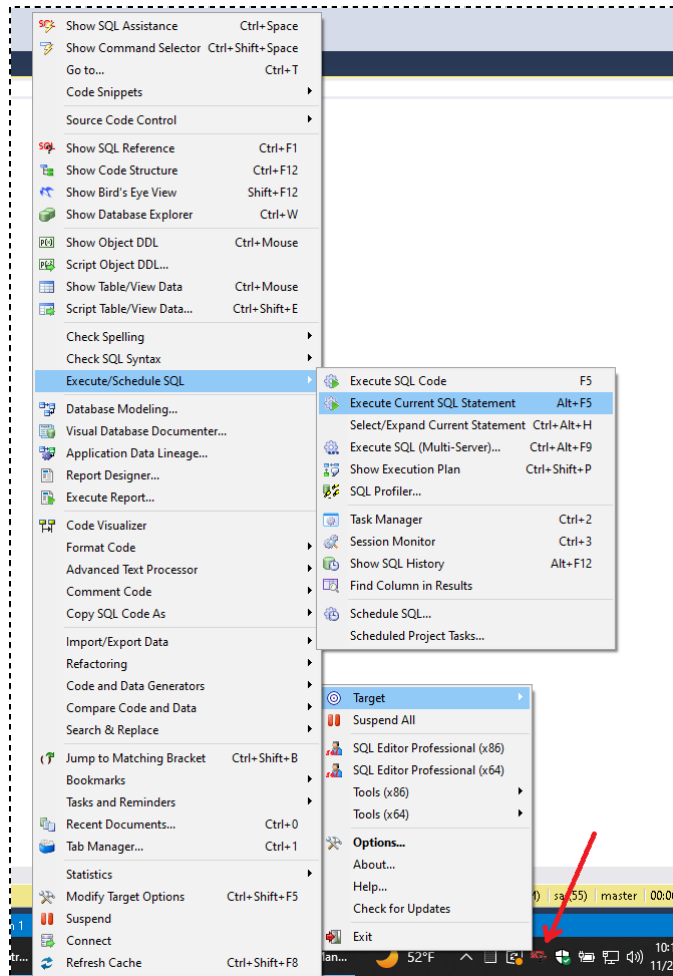
Viewing SQL Assistant Usage Statistics from the System Tray

In addition to invoking various SQL Assistant commands and features, you can use the system tray icon to view a summary of SQL Assistant usage statistics. Rest the mouse pointer over the SQL Assistant's icon in the system tray area to display a usage statistics balloon as demonstrated on the example screenshot.
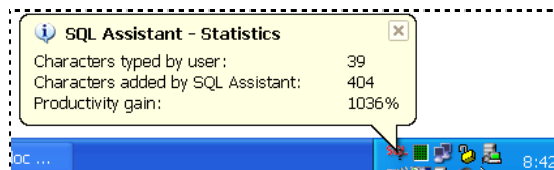
See the following topic for information about usage statistics and how they are calculated.

Understanding and Using SQL Assistant's Usage Statistics

SQL Assistant gathers usage statistics in all target editors where it is active. Usage statistics provide you with information for analyzing which coding methods and SQL Assistant usage techniques provide the most gains. For example, you can use them to compare which of the following strategies for building SQL queries is most efficient:

- Using SQL Assistant's Help, start by defining FROM and JOIN clauses, then add a SELECT clause to the query skeleton created by SQL Assistant

- Start with the SELECT clause first, and then JOIN statements

SQL Assistant provides three types of usage statistics:

Global-level statistics for all target editors and all editing sessions – these statistics are displayed on the About tab page in SQL Assistant's Options dialog. They are also displayed as a balloon for SQL Assistant's system tray icon. Keyboard usage in all target editors ever used with SQL Assistant in all editing sessions is displayed.

Editor-level statistics for all editing sessions – these statistics are available in SQL Assistant's menus – both in the right-click context menu and in the top-level menu (if target editor top-level menu integration is enabled). Keyboard usage for all instances of the current target editor in all editing sessions is displayed.

Instance-level statistics for the current sessions – these statistics are available in SQL Assistant's menus – both in the right-click context menu and in the top-level menu (if target editor top-level menu integration is enabled). Keyboard usage is displayed for the current target editor instance only, and only for the current editing session.

Types of Statistics Collected

All statistics are collected and aggregated in real-time. The following metrics are available:

Typed by User – this is the number of characters typed by the user in the target editor. This number includes all characters and digits, carriage returns, tabs and other characters typed as part of the code.

Added by SQL Assistant – this is the number of characters that SQL Assistant added to the code. It includes text inserted using SQL Assistant's popups, text generated by code snippets, tabs inserted by the auto-indent feature, and other SQL Assistant's dynamic code auto-formatting features applied in the background as you enter the code.

Typing Speed – this is a measure of your average typing speed calculated as an average number of characters added to the code per minute. This number counts all characters, including both characters typed and characters added to you code by SQL Assistant. Note that this metric calculates typist speed during active periods of code entry only. Periods of typist inactivity for 30 seconds and longer are ignored.

Productivity Gain - this is the ratio of how much code you entered with the help of SQL Assistant expressed as a percentage. This number is only available for the global statistics level. For example, if you type 50 characters, and 100 more characters are added for you by SQL Assistant's intellisense and code snippets functions, your productivity gain is 200%.

Tip: An average database developer who is well familiar with their database schema and who remember most table and column names can enter anywhere from 20 to 50 characters per minute. Note that this time includes breaks, mouse activities, data lookup, An average coder working with an unfamiliar database schema can enter from 1 to 5 characters per minute because they have to spend much time researching schema object names, columns and parameters and typing unfamiliar names. In both cases SQL Assistant can dramatically improve their productivity and increase their coding speed by 300% or more.

Disabling and Enabling Statistics Collection

If you do not want SQL Assistant to gather keyboard usage statistics, you can disable this feature:

1. Use any available method to display SQL Assistant's menu.

2. Click the **Statistics** submenu and then click the **Active** command.

To re-activate this feature, repeat these steps.

Resetting Statistics

To reset statistics and begin anew:

1. Use any available method to display SQL Assistant's menu.

2. Click the **Statistics** submenu and then click the **Reset** command.

How to Build Advanced SQL Commands With Only a Few Keystrokes

The following examples demonstrate several important techniques for quickly generating SQL code, complete with multiple table joins, column selection lists, joins and other SQL features.

Example 1: Building complete SELECT statement starting with column names

1. Type **SELECT** and press spacebar. The SQL Assistant's Objects popup will appear.

2. Press the **Right Arrow** keyboard key to expand columns of the actor table. The result should look like the following.
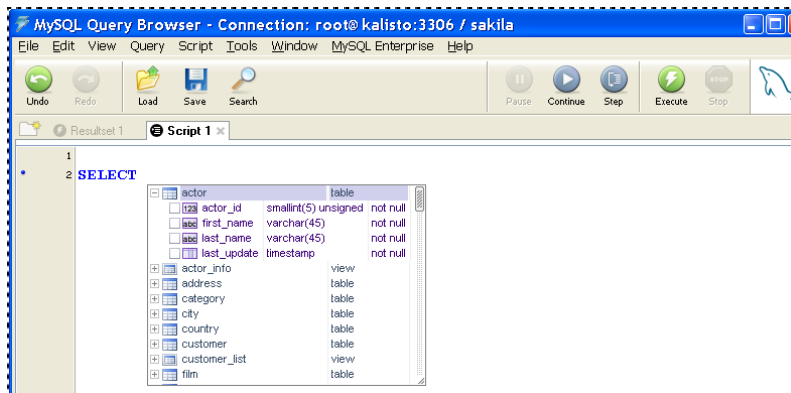
3. Press the **Down Arrow** twice to move the selection to the **first_name** column. Press the **Right Arrow** key to select this column. The SELECT statement in the editor will change to the following:

```
SELECT a.first_name FROM actor a
```
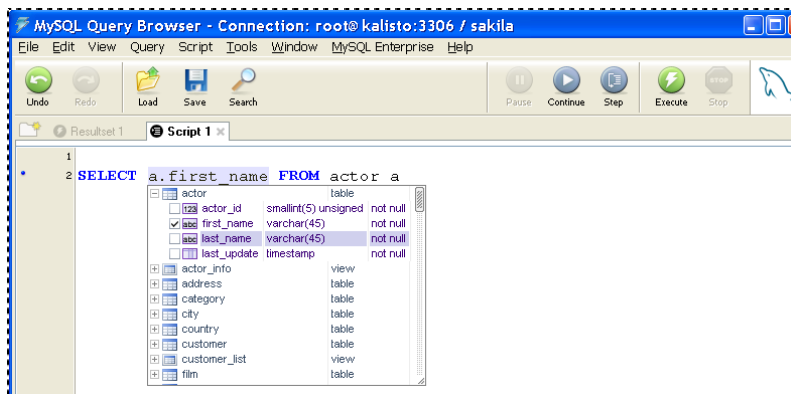

Press the **Down Arrow** to select the next column, **last_name.** Press the **Right Arrow** again to add this column to the SELECT list.

4. Using **Down Arrow** and **Right Arrow** keys, scroll to the next table, **actor_info,** and expand that table too. Then using the same the technique, select the **film_info** column from the **actor_info** table.
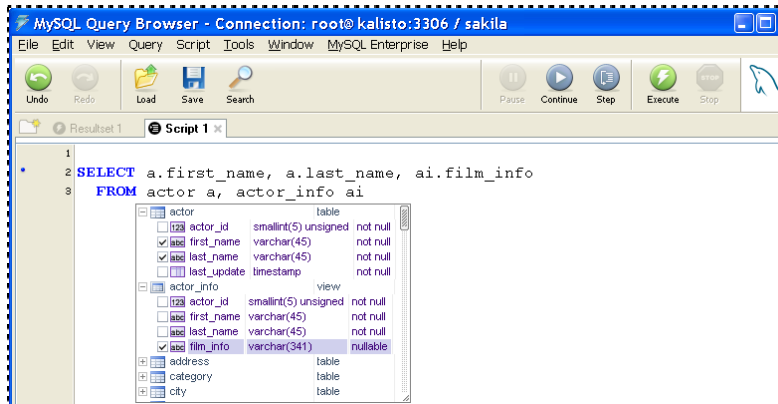
5. Click at the end of the generated query and add a WHERE clause. In other words, click after "ai" and press the Enter key. Type the "WHERE" keyword and press spacebar. The SQL Assistant's Columns popup will appear.
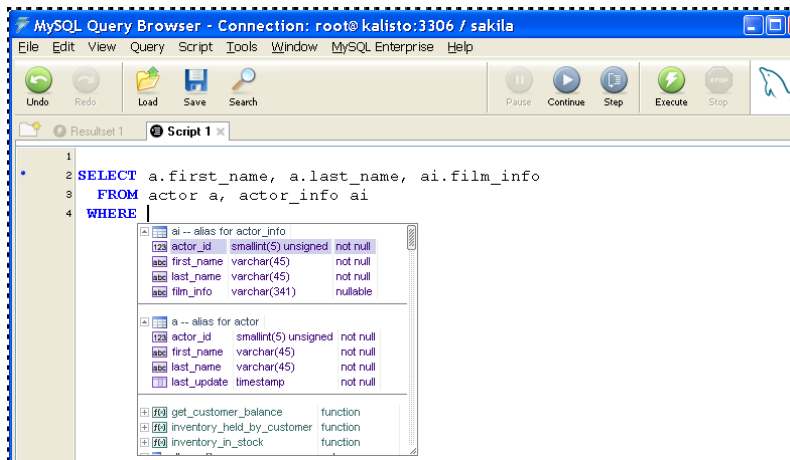
6. Using the **Down Arrow,** scroll to the second line and press the Enter key to paste **ai.actor_id.** The SQL Assistant column popup will disappear.

7.   Type the equal sign. The SQL Assistant's columns popup will appear again. Scroll down to the **a.actor_id**
     column and press Enter or simply double click that column.
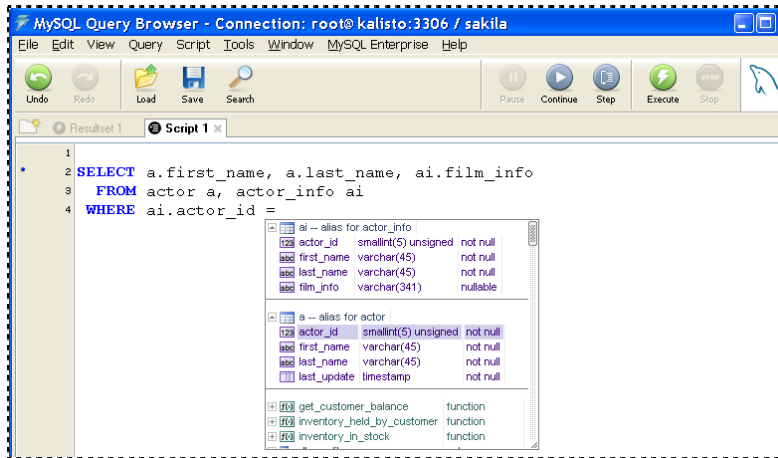
You now have the complete SELECT statement with columns from multiple tables and a complete WHERE
clause.

Note: This result required only 22 key-presses using mostly two adjacent navigation keys and two mouse
clicks. Compare this to the length of the text, that is 106 characters long, and you come up with 79% saving.
And the coding of this query with 3 joined tables took only a few seconds.

Example 2: Building complete SELECT statement starting with joins

1.   Type **SELECT * FROM** and press spacebar. The SQL Assistant's Objects popup will appear.

2.   Type the letter "F". The object list will be automatically filtered to show objects beginning with the letter F.
     The result should like the following image:
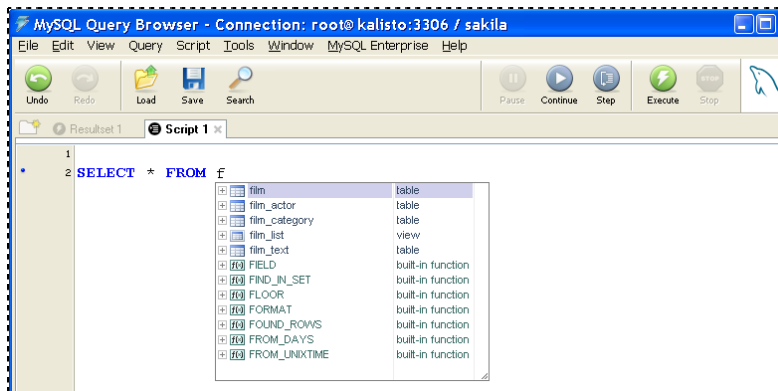
Press the Enter key to select the **film** table. SQL Assistant will modify the SQL query in the editor to the
following:

```
SELECT * FROM film f
```

3. Now type **JOIN** and press spacebar. The SQL Assistant's object-joins popup will appear.
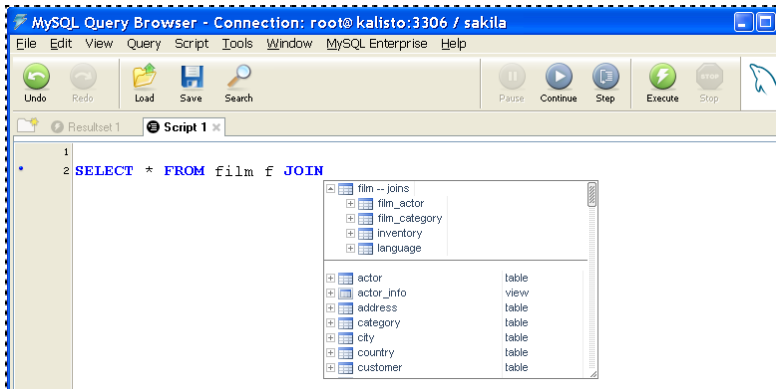
Using the **Down Arrow** and **Right Arrow** keys, expand the **film_category** table join The result should like the below image:
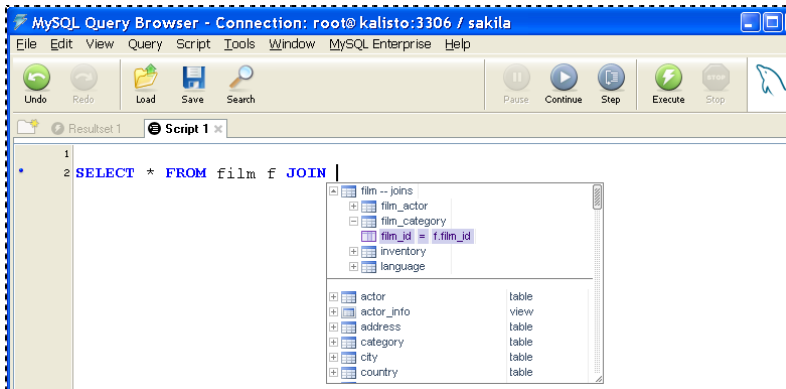
Now press the Enter key to add the **film_category** table to the query. In the editor you now have the following query:

```
SELECT *
FROM film f JOIN film_category fc ON fc.film_id = f.film_id
```

4.   Type **JOIN** again at the end of the query to add another join. Follow same steps as described above, but this time pick the **category** table join:
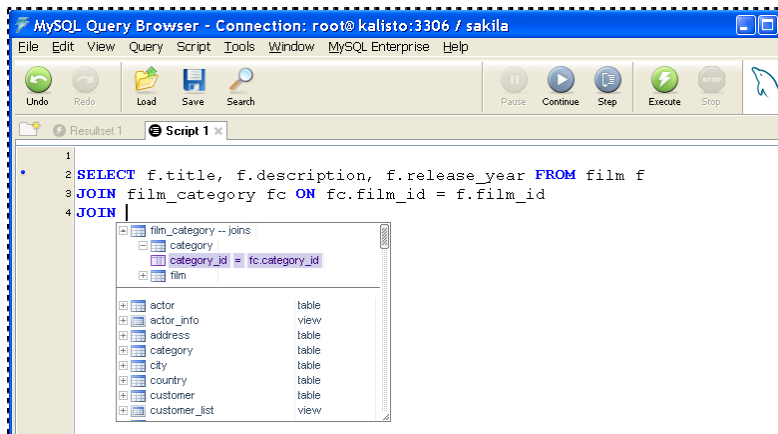
You now have:

```
SELECT *
FROM film f
JOIN film_category fc ON fc.film_id = f.film_id
JOIN category c ON c.category_id = fc.category_id
```

5.   Click before or after the asterisk and press Ctrl+Space to open SQL Assistant's column expansion popup (see the Advanced Code Expansion for Object Columns and Arguments" topic for more information) You should get the following:
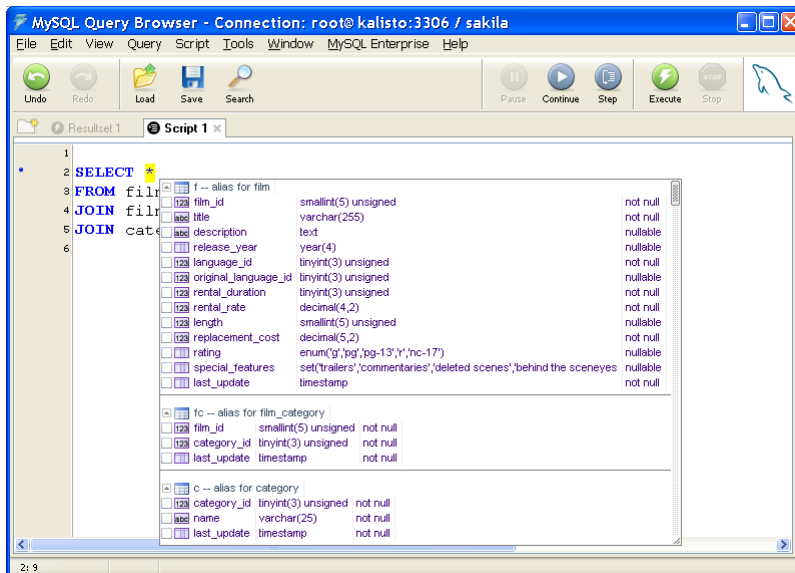
6.   Using the mouse or navigation keys, select the appropriate columns from the referenced tables by clicking the check boxes.
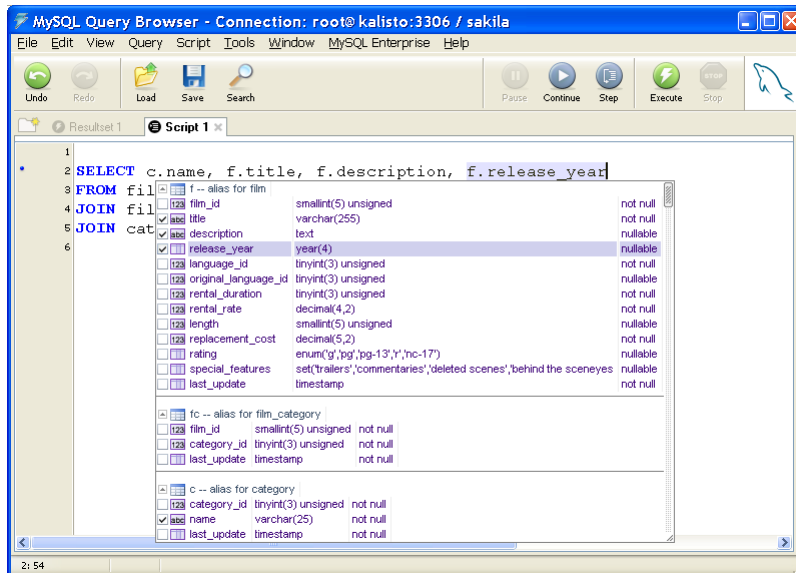
The resulting query may look like below

```
SELECT c.name, f.title, f.description, f.release_year
FROM film f
    JOIN film_category fc ON fc.film_id = f.film_id
    JOIN category c ON c.category_id = fc.category_id
```

Note: This result required only 35 quick key-presses, including spacebar presses. Compare this to the length of the text, that is 146 characters long, and you come up with 76% saving. And the coding of this query with 3 joined tables took only a few seconds.

Example 3: Creating multi-line comments with 4 keys

1.   Type **/**** and then press the Ctrl+Enter hot key sequence. The result will be as follows. The cursor will be positioned on the second line with the exact position indicated by the pipe | character.

```
/*****************************************
*  |
*****************************************/
```

Tip: This simple example involves one of the pre-defined code snippets that are part of SQL Assistant's standard configuration. SQL Assistant is packed with a large number of pre-defined snippets. Some snippets are very simple. Examples are the automatic adding of the END keywords to every BEGIN and indenting the code. Others are more complicated and can be used to generate body of a stored procedure or complete cursor logic along with table references, variables and loops. Code snippets provide very efficient method for quick code entry.

Example 4: Generating complete-cursor logic with 7 keys and 1 click

1. Type **cfetch** and then press the Ctrl+Enter hot key. The Object popup will appear.

2. Click any of the tables or views available in the popup. The result is displayed on the following picture:
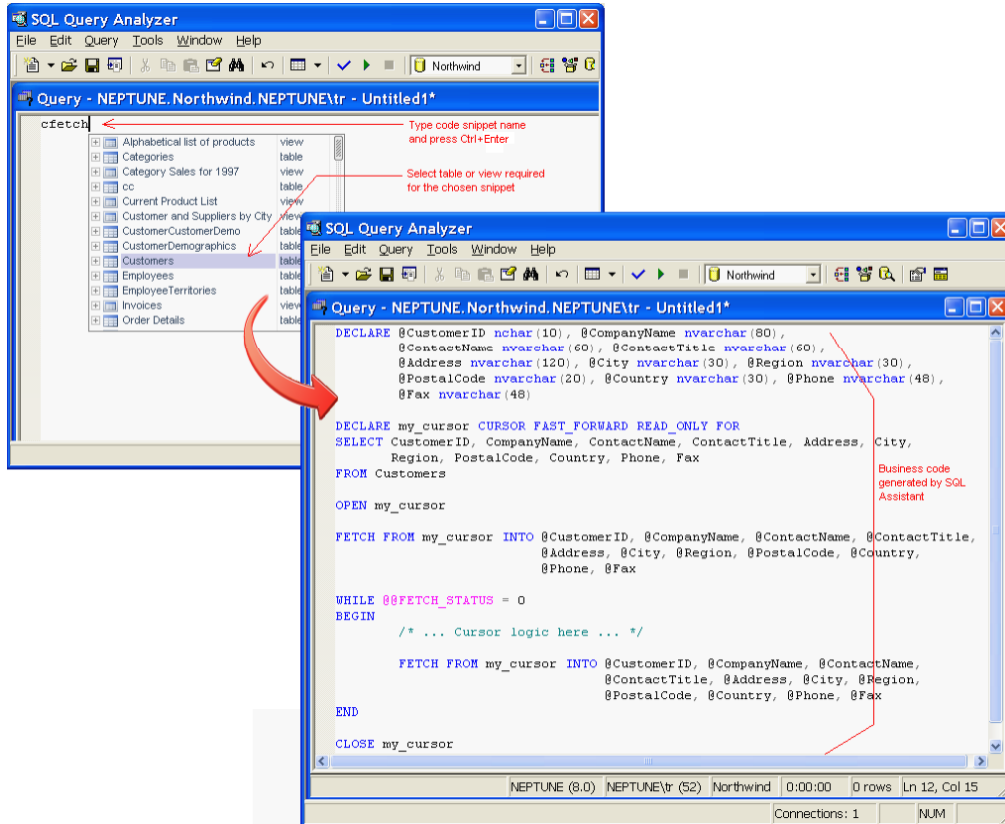
🖈 **Tips:**

- This simple example involves one of the pre-defined code snippets that are part of SQL Assistant's standard configuration.

- You can create your own snippets to quickly generate common code used in your queries and procedures.

- For more information for how to use advanced code snippets, see CHAPTER 7, Code Entry Automation using Code Snippets

Using Object Name Code Completion Features

The content of SQL Assistant popups is context and database-driven. For example, when you type FROM keyword within SELECT statement, or UPDATE keyword, the popup list is populated with items you may want to insert into the text. These items may include table and view names, table function names, schema names, and so on.

For your convenience, items of different types are displayed in different colors and indicated by different icons displayed on the left side of the popup list.

If the item you want is below the visible area of the popup, scroll through the list to locate the item and then double-click or press the Enter key to insert it into the text. Alternatively, you can start typing the first few characters of the item to display only items that begin with those characters.

If the item you want is not in the popup list, continue typing normally and the popup will disappear automatically.

The popup containing object, schema and database names can appear when you type space character after a SELECT, FROM, JOIN, TRUNCATE TABLE, DECLARE, EXEC, EXECUTE, CALL, or USE keywords. It can also appear after a database or schema name followed by a dot character. The set of keywords that trigger the automatic popup is controlled by the "SQL Assistance" type you select in SQL Assistant options for the current target type, such as T-SQL or PL/SQL.

Tip: The order of different items and types of items listed in the popup can be customized in SQL Assistant's settings in the **List Items** section of the **DB Options** tab. You can set additional filters using customized versions of database catalog queries available on the on the **DB Options** tab in the **DB Queries** section. For more information, see the "Using Object Type Filtering" and "Customizing Database Catalog Queries" topics in CHAPTER 48, Customizing SQL Assistant's Behavior.

Tip: The Object Names popup list typically contains names of objects in the current user schema, names of database schemas in the current database, and names of databases in the current server. Schema names and database names represent multi-level items that can be expanded. If you need to select an object in a different schema or in a different database, you can do it in multiple steps selecting database or schema level item then typing dot and selecting next level item, then typing dot again, and so on. Or you can do it in one step expanding levels in the popup using mouse or keyboard keys and selecting the required final item as in the following example:
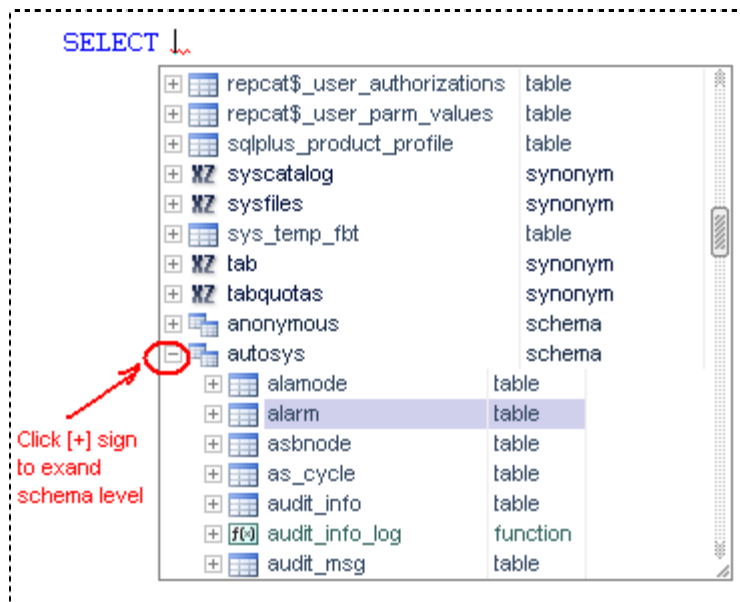
Note that in certain situations, SQL Assistant automatically expands multiple levels if it can uniquely identify the schema and table. For example, if you start typing the schema name with the characters "au" and there is only one schema name that begins with those characters, the list will be narrowed to one schema and the schema level will expand automatically as in the following example,
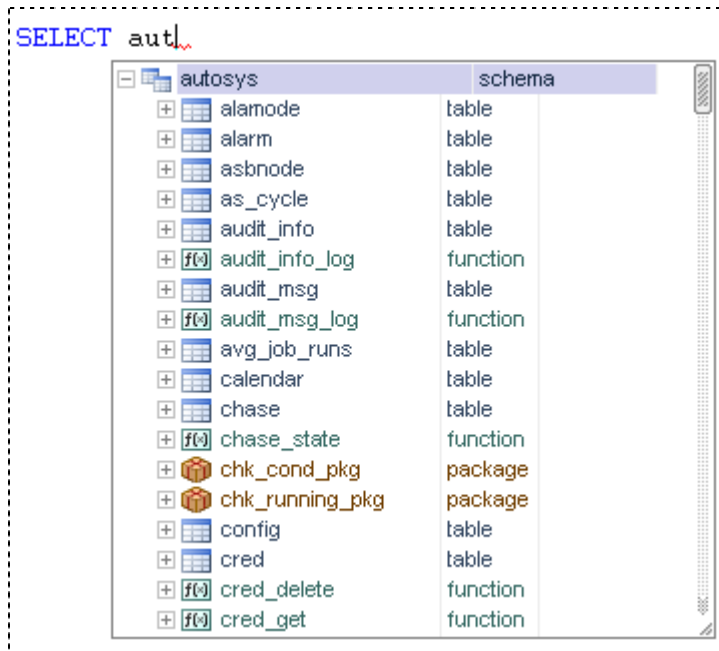
Note that levels can be expanded and collapsed using either the mouse or keyboard. Using the mouse, click on the [+] plus sign to expand the level, click on the [-] minus sign to collapse an expanded level. Use the Right Arrow and Left Arrow navigation keys to achieve the same effect using the keyboard.

Another helpful feature is multi-level item filtering. Here is an example of how this feature works. Suppose you are coding a SQL Server procedure and in the procedure code you want to reference, **Inventory.dbo.SpareParts,** is in a table in another database**.** To quickly enter this table in the FROM clause:

1. After keyword FROM, type space. The SQL Assistant's objects popup will appear on the screen. Start typing the characters "in". SQL Assistant will filter the content of the object names popup so the **Inventory** database name appears on top.
2. Use the Right Arrow key to expand this database level and type the character "d". Now the second level containing schema names should show only schemas whose names begin with letter "d." Most likely the "dbo" schema will be the first in list.
3. Use the Down Arrow key to select "dbo" schema and then use Right Arrow key to expand this schema level. Then type "sp". Now the third level containing schema names should show only objects whose names begin with letters "sp" and most likely "Spare Parts" table will be the first one in list.
4. Use Down Arrow key to select "SpareParts" table and then press the Enter key to insert the table name into the code (or use whatever key you have chosen in options as a list item selection key)

As you can see in this example, you can quickly type 5 characters and use 3 navigation key presses, plus the Enter key, to quickly insert the **Inventory.dbo.SpareParts** table name into your code. Compare this to the length of the fully qualified table name, which is 24 characters long.

Object Name Aliases

When completing table and view names, by default SQL Assistant automatically adds short aliases. In the previous topic you learned how to quickly insert object names into your code. The actual text that is inserted when you press the item selection key or use mouse includes the selected table name or view followed by the AS keyword and further by table aliases. For example, if you select **Inventory.dbo.SpareParts** table name in the popup window after typing SELECT * FROM, the inserted text would be
SELECT * FROM Inventory.dbo.SpareParts AS sp

Tips:

- If you don't want SQL Assistant to automatically add an alias, hold down Shift key when selecting a name. That will result in the name being inserted without trailing AS <alias> element.

- You can predefine preferred aliases for commonly used tables in SQL Assistant options, as well as customize how automatic aliases are calculated. For more details see Customizing Code Auto-completion Options in CHAPTER 48, Customizing SQL Assistant's Behavior.

Using Object, Schema, and Database Name Auto-Completion

Using the Ctrl+Space shortcut, you can instruct SQL Assistant to auto-complete the partially entered name of an object, schema, or database name. For example, in the Microsoft SQL Server Query Analyzer, if you type

SELECT * FROM sysda

and then press Ctrl+Space hot key, SQL Assistant automatically completes the text as

SELECT * FROM sysdatabases

If there is only one name beginning with the partially entered text, that name can be matched unambiguously and completed automatically. If multiple names match the entered text, a SQL Assistant object popup appears with the list of objects, schema, and databases whose names begin with "sysda" prefix.

Using Variable Name Auto-Completion

This feature is available for SQL Server and Sybase ASE and ASE targets in which variable names begin with @ symbols. Using the Ctrl+Space shortcut, you can make SQL Assistant auto-complete partially entered variable names name. For example, if in Microsoft SQL Server Query Analyzer, if you type

SET @somev

and then press Ctrl+Space hot key, SQL Assistant automatically completes the text as

SET @somevariable

If there is only one variable declared in script whose name begins "somev " text, that variable name can be matched unambiguously and completed automatically. If multiple names match the entered text, a regular SQL Assistant variable names appears with the list of variables whose names begin with "somev " prefix.

Using Column and Parameter Names Completion Features

The table/view column list popup appears expanded automatically when SQL Assistant is invoked after a dot character, comma, equal sign or end of procedure name. The popup item list is normally limited to column names of the referenced table or parameters of the referenced procedure or function.

Below are several examples demonstrating the column and parameter names completion feature:
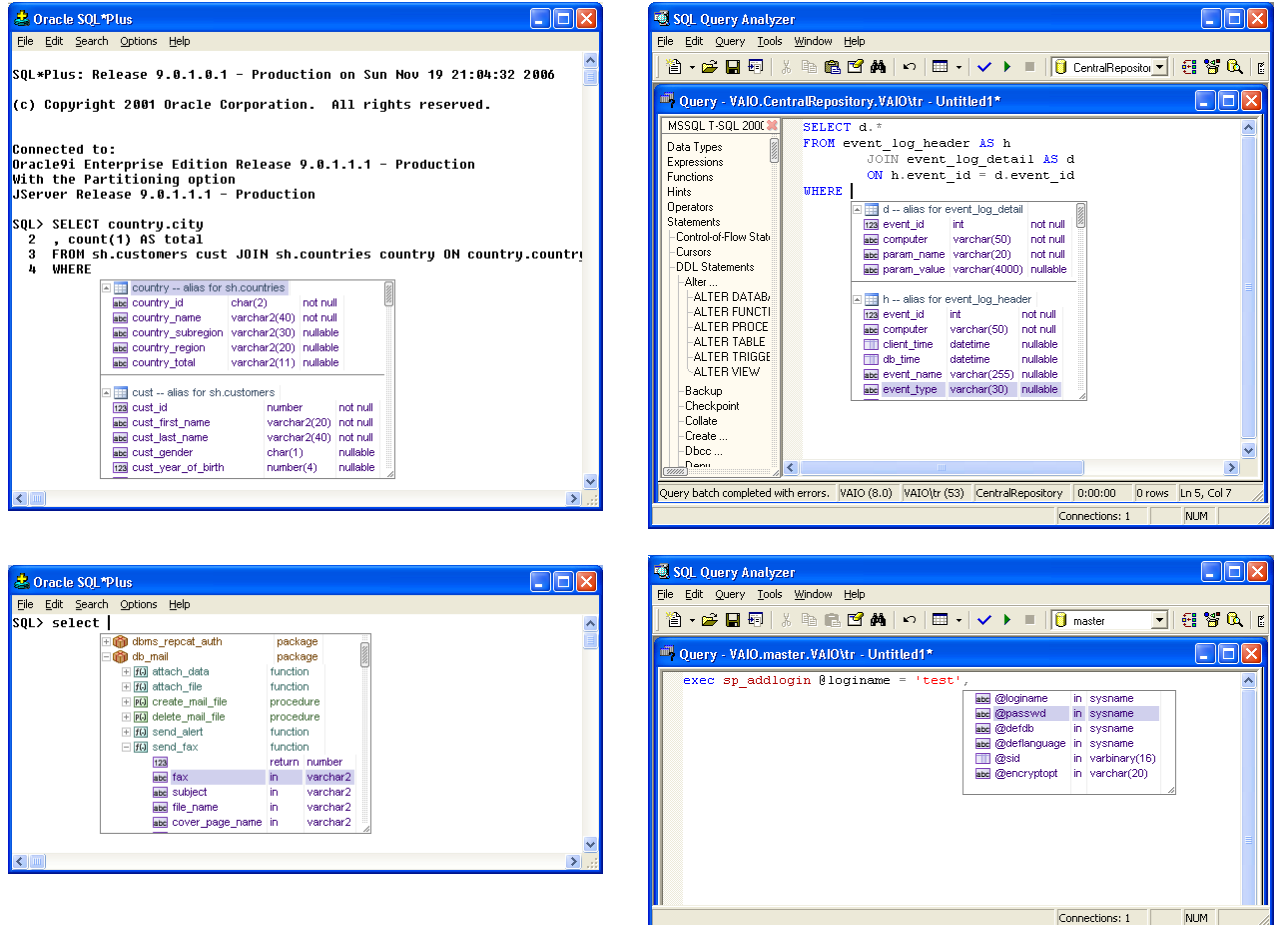
SQL Assistant popup window content depend on the popup invocation context and the target environment. Depending on the context, a series of checkboxes might appear next to column names in the list so you can select multiple columns at once. See the "Using Multiple Columns Selection in DML Statements" topic in CHAPTER 3 for more details.

In addition to column names and their data-types, SQL Assistant column popups may display optional icons indicating primary key columns and indexed columns. See the following topic for information on how to enable this optional feature.

Enabling Display of Key Columns and Indexed Columns

By default, SQL Assistant is pre-reconfigured to display simple table column lists containing only column names, their data types and nullability states. Simple lists allow SQL Assistant to use a relatively small amount of memory for its internal in-memory data caches and, most importantly, they allow fast response times when

SQL Assistant queries database catalog tables that store table column information.

In addition to displaying simple column information, SQL Assistant can be setup to display icons in column lists as well as overlay icons indicating table primary key columns, foreign key columns, unique key columns, and indexed columns. If you work with a sufficiently fast small to medium size database and you are not concerned with the amount of memory SQL Assistant uses for its internal data caches, you can enable this optional feature.

Tip: By the "small", "medium", or "large" database size quantifiers, we are referring to the number of tables, functions, procedures, packages, and other objects available in the database, not the amount of data stored in database tables. From this perspective, a database containing tens of thousands of objects and hundreds of thousands of table columns, procedure parameters, etc… is considered as "large." A good example of a large database would be a database hosting an Enterprise Resource Planning (ERP) application such as SAP, JD Edwards, and similar.

Important Notes: We encourage you to enable the optional **Display Keys and Indexed Columns** feature and test whether the performance and response times you experience are acceptable. If they are not, you can always switch back to the default simple Column display option.

See the "Two Database Catalog Queries for Getting Column Information and How to Change Them" topic in CHAPTER 48, "Customizing SQL Assistant's Behavior," for instructions on how to switch between two available Column display options.

Table or key and index icons and their meaning

Icon	Description
	This icon indicates that the column is part of a primary key. The primary key icon takes precedence over other icons If the same column is part of multiple keys or indexes.
	This icon indicates that the column is part of a foreign key. The foreign key icon takes precedence over unique key and index icons if the same column is part of multiple keys or indexes.
	This icon indicates that the column is part of a unique key. The unique key icon takes precedence over the index icon if the same column is part of a unique key and one or more indexes keys or indexes.
	This icon indicates that the column is part of an index.

The following screenshots demonstrate the differences between simple table column lists and table mouse-over hints and column lists and mouse-over hints with additional icons indicating table keys and indexes:

Simple column list
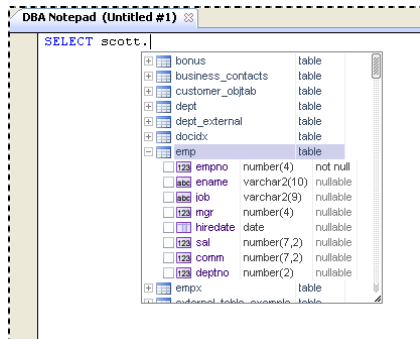
Column list with key and index icons
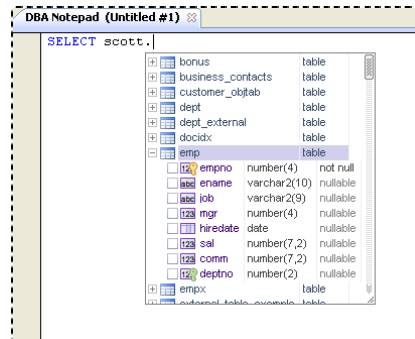
Table columns hint
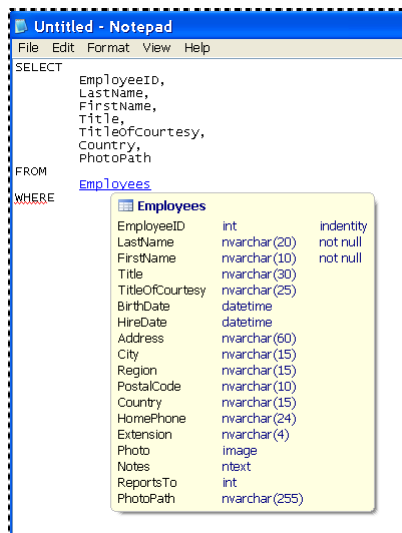
Table columns hint with key and index icons
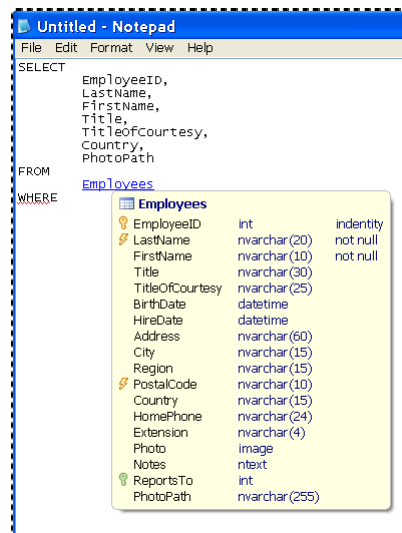
Using JOIN Clause Completion Features

When typing a JOIN clause into the editor, a code completion popup displays each time you type the JOIN keyword, the ON keyword, or an equal sign that is part of the JOIN clause. A similar popup may also appear after you enter a WHERE clause with a correlated sub-query.

In the popup appearing after the JOIN keyword, you are presented with a list of suggestions based on the analysis of the referential integrity constraints defined for the tables referenced in the JOIN clause as well as the list of other tables in the database The list of referential integrity based suggestions appears on top of the popup and a horizontal line separates it from the rest of the popup content.

You can select an object name to add it to the current SQL statement and then type the ON keyword and select

a condition for the join. However, it is more efficient to expand the table required for the join and then select a join condition from the list of displayed conditions. SQL Assistant will automatically generate the entire JOIN clause, including table names, aliases and columns.

The following screenshot demonstrates how to use the JOIN clause completion feature when using referential integrity-based suggestions
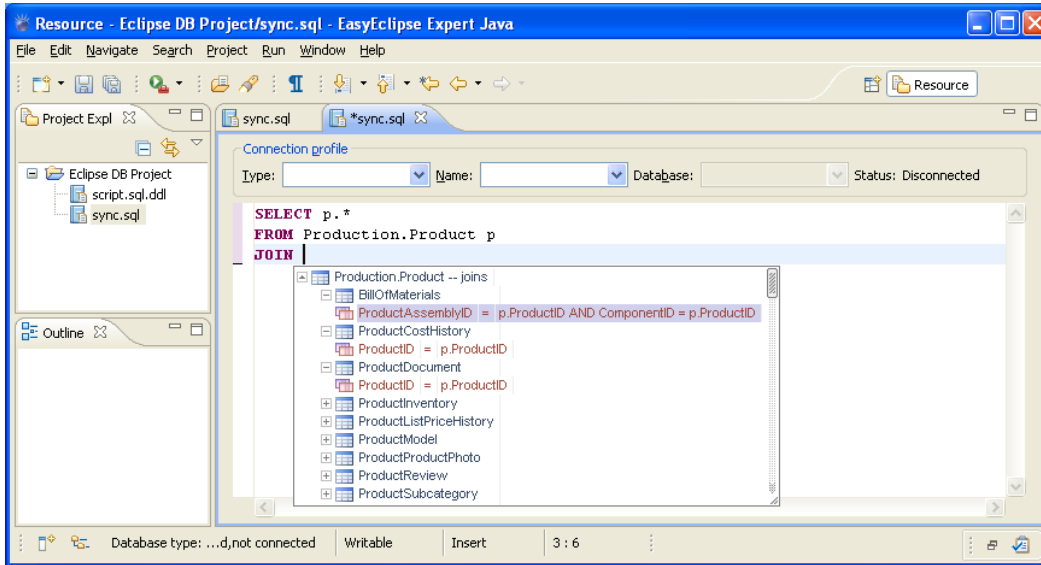
In addition to using referential integrity definitions for JOIN clause suggestions, SQL Assistant can use column-name matches. SQL Assistant uses color-coded styles to distinguish between different suggestion types. The following screenshot shows JOIN suggestions based on column name matching.
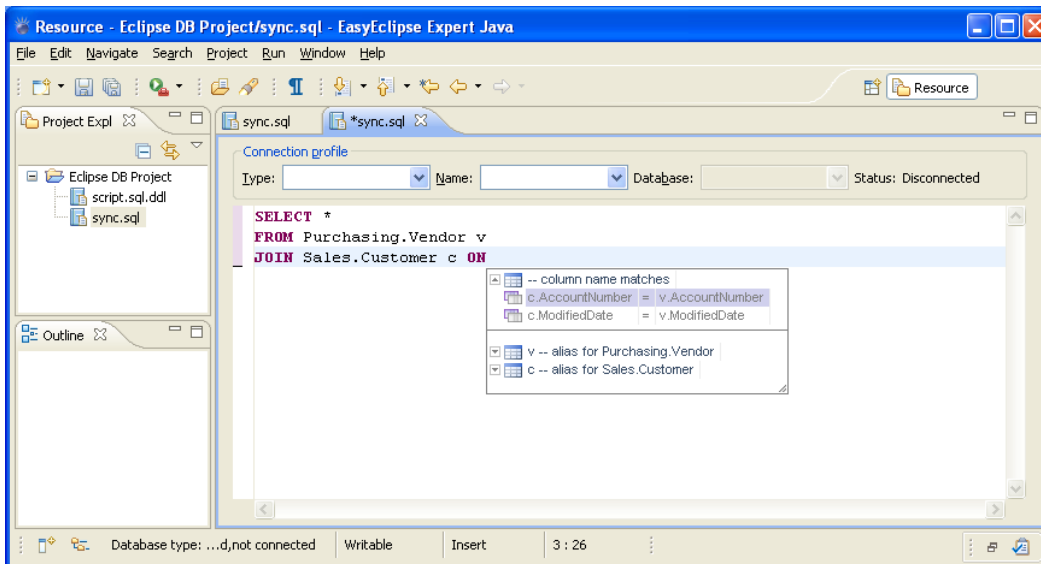
Note that both types of JOIN suggestions can appear in the same popup list. The purple color suggestions indicate referential constraints; gray color suggestions indicate column name matches.

Tips:

- If a referential constraint consists of multiple columns, all columns referenced in that constraint appear on the same line. If the line is long, some of the columns may appear cut off and may not

be visible. To display these columns, you can increase the width of the popup window by dragging its right edge further to the right. See the "Working with SQL Assistant Popups" topic for details on how to resize and manipulate SQL Assistant's popups.

- You can use keyboard navigation keys to quickly expand and collapse tables suggested for a JOIN and show/hide their columns. Use the Right Arrow key to expand the selected table level and the Left Arrow to collapse the level.

Using Multiple Columns Selection in DML Statements

When using SQL Assistant to generate DML statements such as SELECT, INSERT and UPDATE, you can choose which columns to include in the statement as soon as you type the first keyword. For example, if you want to use a SELECT statement to retrieve data from several columns of the *customer* table, use the following steps:

1. Type SELECT and press the space bar. A SQL Assistant objects popup will appear.

2. Select the *customers* table from the popup.

3. Click the plus sign ⊞ in front of the *customers* table name to expand the table and display its columns. Alternatively, you can highlight the table and then press the Right Arrow navigation key to expand the *customers* table.
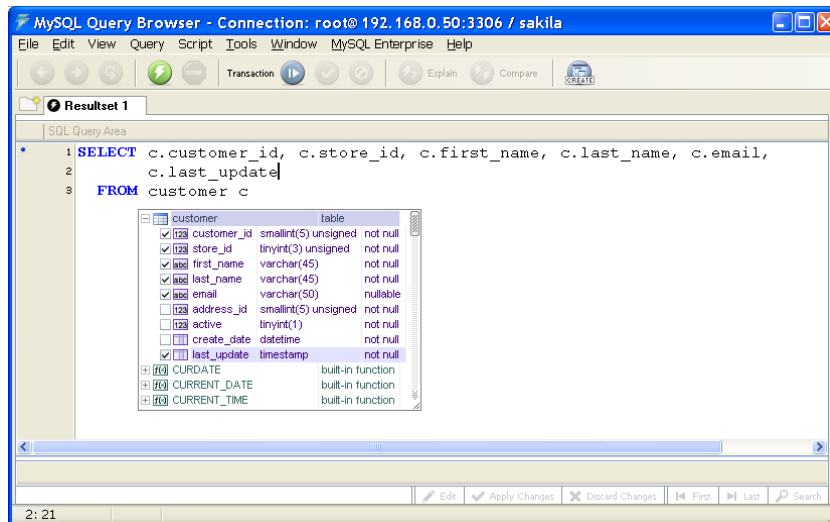
4. Select the checkboxes on front of columns you want to include with the SELECT statement. Alternatively, use the Up Arrow or Down Arrow keys to scroll the list, then use the Right Arrow and Left Arrow keys to select or deselect the appropriate columns. As you select or deselect columns, you will see the SELECT statement for the current column selection automatically generated for you behind the SQL Assistant's popup.

5. Press Esc or Enter to close the popup.

Tip: The same technique can be used for INSERT and UPDATE statements as shown in the following screenshot.
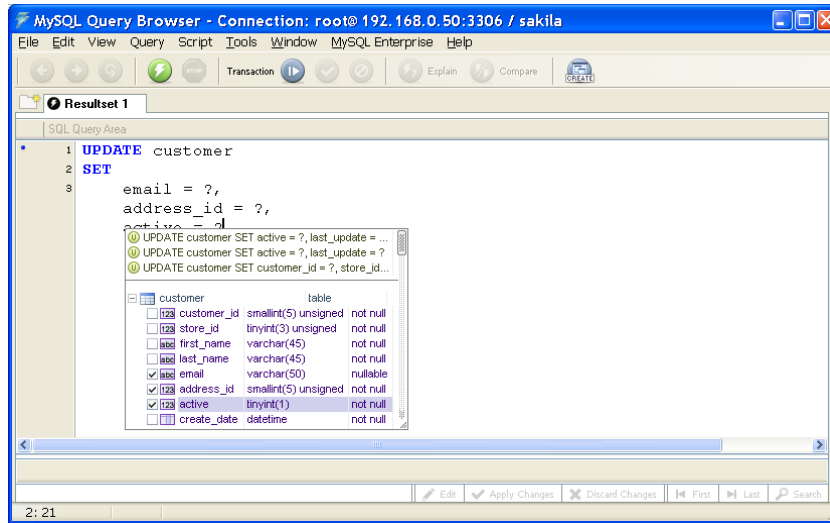
Tip: To quickly select all columns in a table, right-click the table name in the popup menu and click the **Check All** command. The following screenshots demonstrates how to use that command.
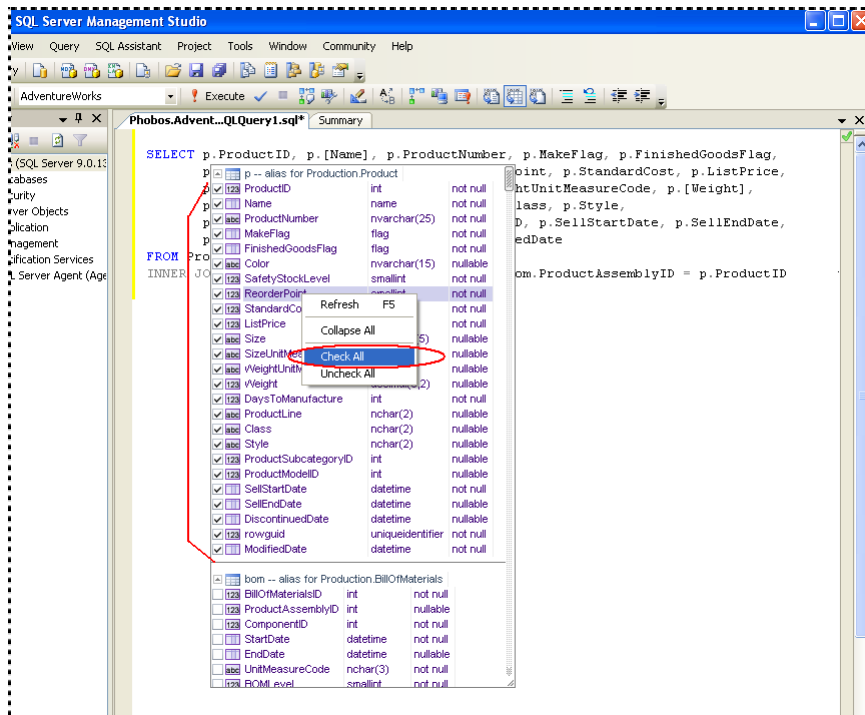
Using Context-based Suggestions Based on Historical Coding Patterns

SQL Assistant caches SQL DML statements code you have previously entered, either manually or with the help

of one of SQL Assistant's code-suggestion or code-generation features. These statements are saved in a **History Cache** file and are ranked according to the frequency with which each statement has been used. When you begin typing one of these statements at a later time, the editor retrieves the relevant portion of the cache file and displays it in a selection window as shown in the screenshot below. To select a line of code from the selection window, use the UP and DOWN arrow keys to select a statement, then press the Enter key to insert the line into your editor code.
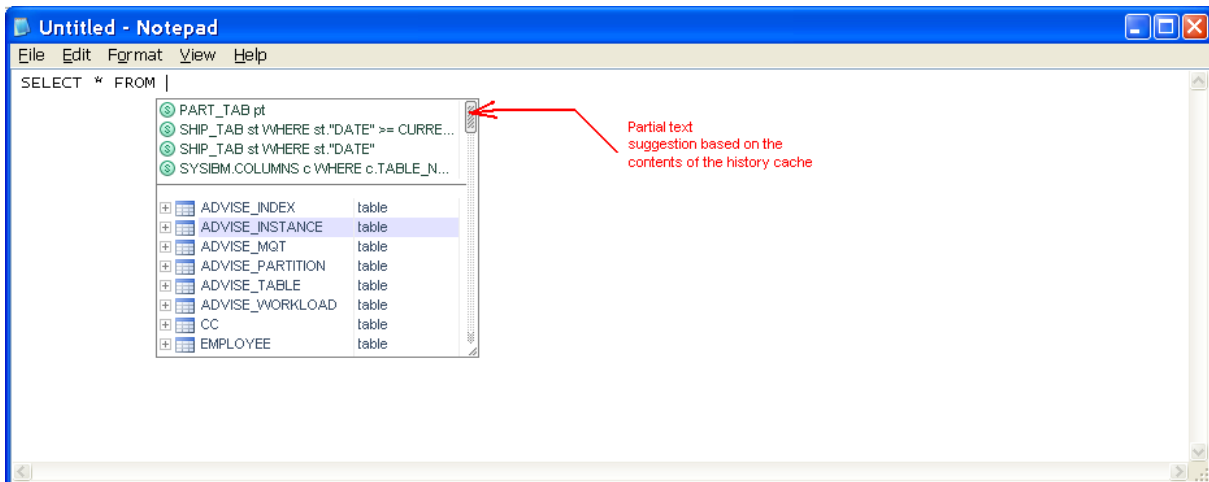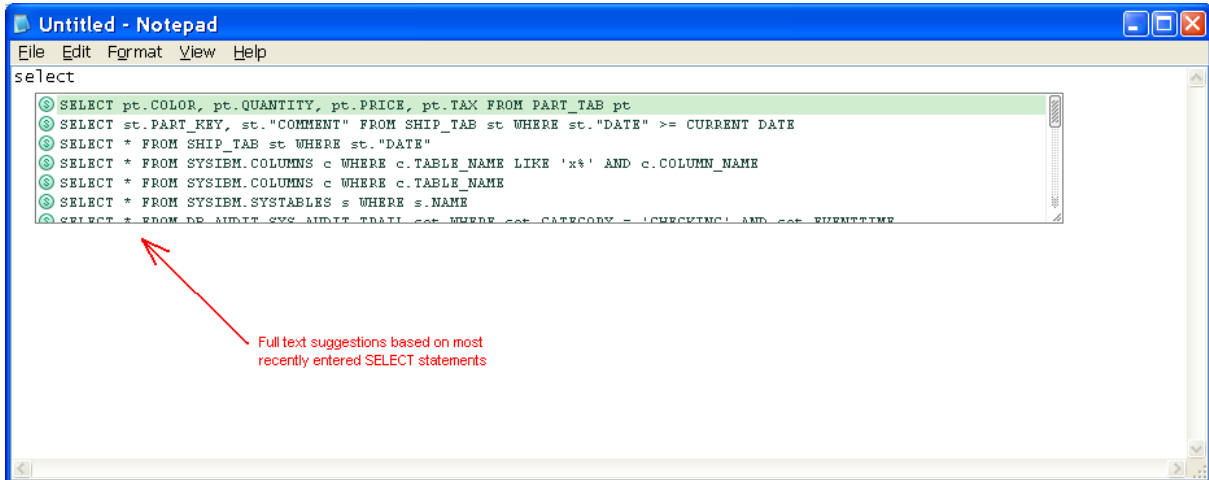
Depending on the context, the complete query text from the **History Cache** can be suggested exactly as it was entered, or only a part of the original query can be used for the suggestions, if the beginning of the query matches what you have already entered into the editor. In the screenshots above, you can see both full and partial text suggestions based on the text of previously entered SQL SELECT queries cached in the **History Cache**.

The size of the history cache is controlled by SQL Assistant settings. See CHAPTER 48, Customizing SQL Assistant's Behavior for more details. The default cache size is 32 Kbytes. SQL Assistant uses a mix of Most Frequently Used (MFU),, Most Recently Used (MRU) and First In First Out (FIFO) rules to maintain cache data. When choosing the right cache size, evaluate your system performance and check how it affects SQL Assistant response times when you enter new code. A larger cache size allows SQL Assistant to save more history data, and more history data permits improved SQL statement ranking and, thus, better context-based suggestions. Smaller cache sizes allow faster processing of SQL cache data which can provide better response times at the cost of less accuracy and fewer choices for suggestions.

Another factor to consider when using code suggestions based on your historical coding patterns is which statements to show in the suggestion part of the SQL Assistant popups. The following choices are available:

- **Most Recently Used** – This is the default option. If selected, this option causes SQL Assistant to show suggestions matching the text you type against the most recently used SQL statements.

- **Most Frequently Used** – If selected, this option causes SQL Assistant to show suggestions matching the text you type against the most frequently used SQL statements.

- **No Sort** – If selected, this option causes SQL Assistant to show suggestions matching the text you type against records in the cache in the order they were saved or updated. With this option, the exact physical order of lines in the History Cache is unknown and depends on your coding patterns.

The number of **History Cache** based suggestions appearing in SQL Assistant's popups is controlled by **History Items Shown in Lists** option that can be configure in SQL Assistant system options for type of assistance. The default value is 3 for all types.

If you would like to hide **History Cache** based suggestions in popups, set the value to 0 (zero).

When the popup is already displayed on the screen, you can dynamically adjust the number of displayed items by dragging the horizontal line separating history-based suggestions appearing in the popup.
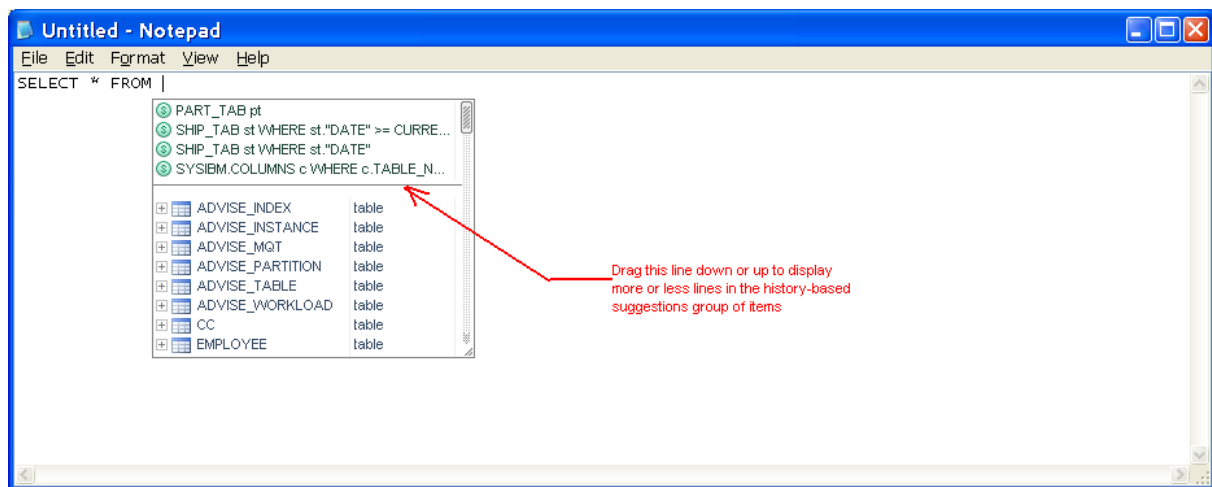
The adjusted size will be effective for the lifetime of the target editor instance. To set the size permanently, use the available system options as described in CHAPTER 48, Customizing SQL Assistant's Behavior. The position of the historical items within the popup is controlled by the **List Items** option group in the SQL Assistant system options. You can use this options group to organize which types of items appear in popups, in which order, and position of the historical items within the popups.

Tip: SQL Assistant supports a separate cache for the Code Execution History, which is used for recalling recently executed SQL statements. For more information, see the Using Code Execution History topic in CHAPTER 14, Executing SQL Scripts.

Using Function Argument Hints Features

The function argument list popup displays when you type an open parenthesis character "(" or commas

separating function arguments. The popup item list is normally limited to hints describing function arguments and the function return code, and also in a separate section, list of column names of tables and views referenced in the same SQL statement.

Below is an example popup demonstrating function argument completion feature.
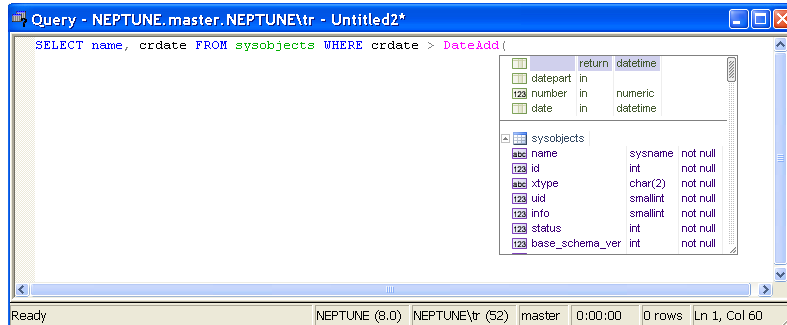
Tips:

- Function argument names and the return code are displayed in a different color and provided as hints to help you enter values having correct data types and in the correct order. After inserting the text from the argument popup, you should replace it with the actual values or valid script variables

- Optional arguments are displayed in [ ] brackets.

- When multiple overloaded versions of a function have different argument types or a different number of arguments, each function version appears in a separate section as shown on the following screenshot:
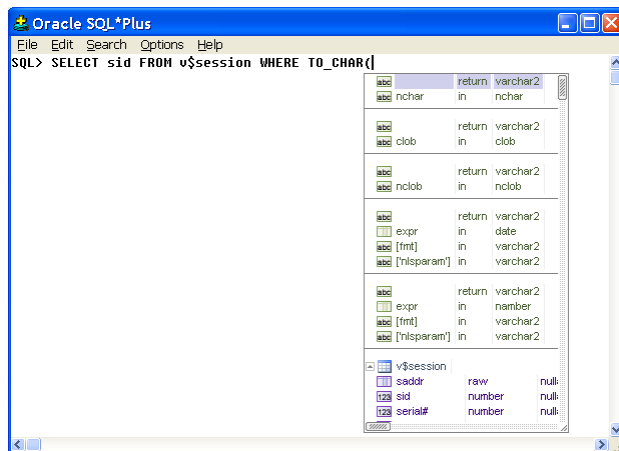
Using Advanced Oracle Package and Object Type Attribute Completion Features

The advanced popup for Oracle packages and object types appears after package and type names. The popup contains lists of attributes and functions available in the referenced objects, providing graphical visibility for what is available in the referenced packages and types. The red text notes on the following example screenshot describe how to read the popup content.
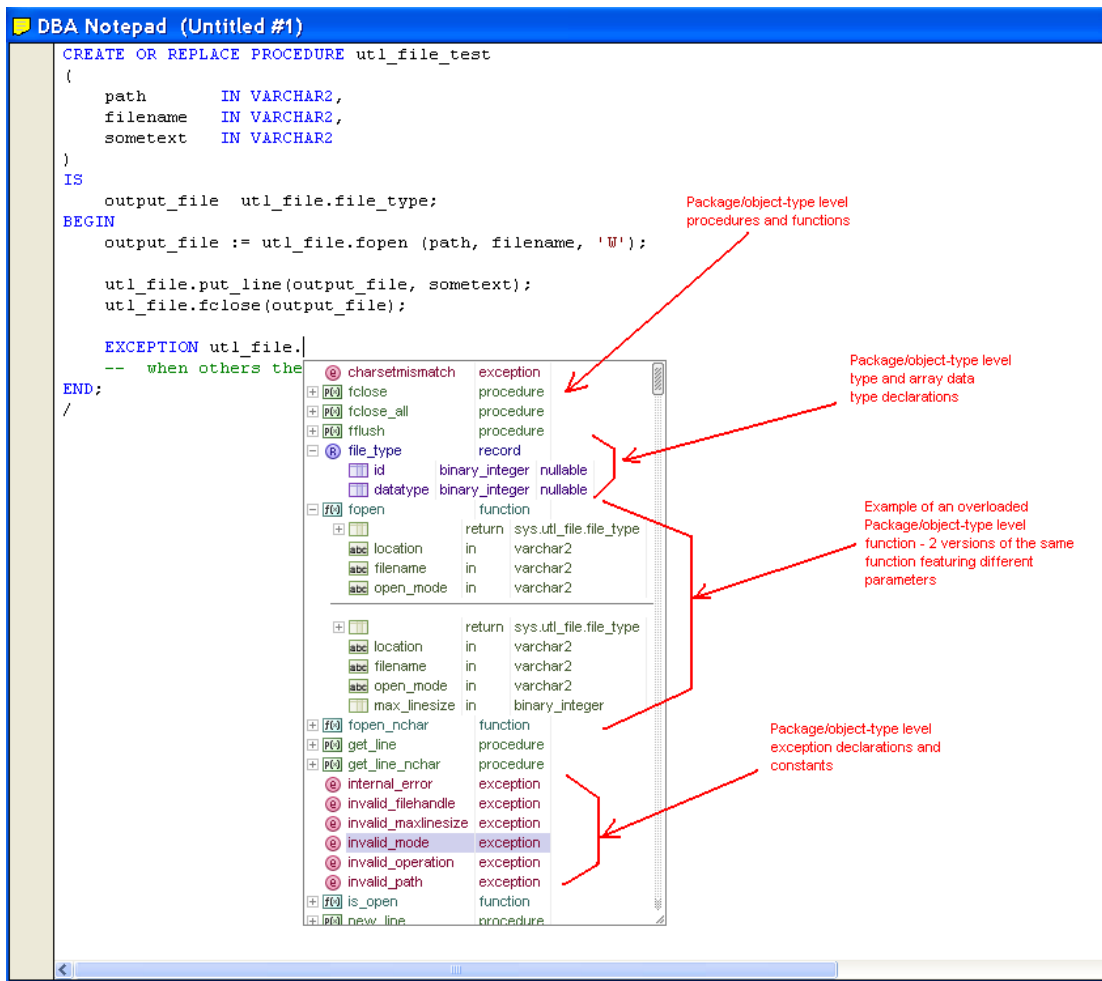
Tips:

1. The popup list may contain multiple levels of information. Items prefixed with a plus sign ⊞ contain sub-items such as object attributes, functions, and function parameters. To expand a sub-item list, click on the plus sign. Similarly to expand sub-items of sub-items, click on the plus sign displayed next to the sub-items.

2. If you don't want to paste or auto-generate code when the popup appears after the initial SQL statement keyword, just continue typing the code normally. The popup will disappear automatically.

3. When multiple overloaded versions of a function have different argument types or a different number of arguments, each function version appears in a separate section.

Using Keywords Completion and Syntax Hints Features

The keywords completion feature is context based. The keywords popup appears automatically when SQL Assistant senses that a keyword might be used in the current edition position. By default, the keywords popup appears after typing the first two characters of a keyword. The popup can also appear in places where a keyword or some other required SQL syntax element is expected. For example, when using default settings,

typing **SELECT *** causes SQL Assistant to display keywords popup whose suggestions list contains FROM keyword and other context based items. The following screenshot demonstrates another case of context based keyword prompts.

![Tips icon] **Tips:**
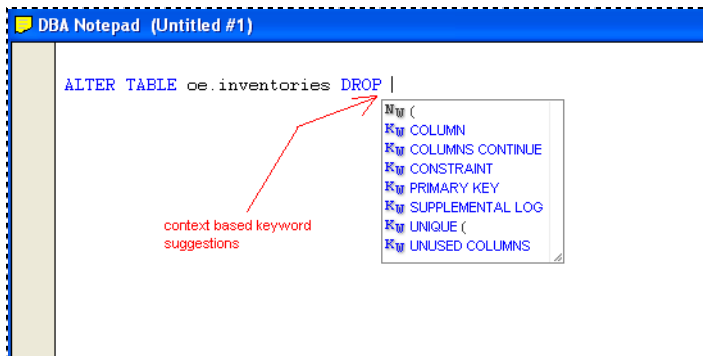
1. The popup list may contain two types of items: keywords and syntax hints. In certain cases, item text may contain both keyword and the associated following syntax hint part. The keywords are always displayed in blue, while suggested syntax hints are displayed in black.

2. Keyword lists may contain individual keywords and groups of keywords that are typically used together; for example, in Oracle targets, both "CREATE" and "CREATE OR REPLACE" keyword suggestions appear in the popup displayed after typing letters CR as well as some other common keyword combinations beginning with the CR text suffix, such as CREATE TABLE and other.

3. Generally, the most commonly used keywords appear at the top of the list in a separate section.

4. The keywords appearance in the popup list and their treatment depends on the settings in the current formatting style associated with the target editor. For example, if in the current formatting style, keywords options are set for keywords to be converted to upper-case, in the popup keywords also appear in the upper-case. In other words, what you see is what you get (WYSIWYG). For more details see the "Keywords" topic in CHAPTER 5.

Using Local and Global Variable Names Completion Features

The variable names suggestions and completion feature is implemented for all supported database systems. However, the implementation differs for database systems using the T-SQL dialect. This includes Microsoft SQL Server, Sybase ASE, and Sybase ASA. In these systems, the variable names popup is activated after typing the @ and @@ symbols. The variable name list popup appears expanded when SQL Assistant is invoked after single @ character or after double @@ characters. The following picture shows an example popup with local variable names and parameters for T-SQL based database systems.

```
ALTER PROCEDURE [HumanResources].[uspUpdateEmployeeP
    @EmployeeID [int],
    @NationalIDNumber [nvarchar](15),
    @BirthDate [datetime],
    @MaritalStatus [nchar](1),
    @Gender [nchar](1)
AS
BEGIN
    UPDATE [HumanResources].[Employee]
    SET [NationalIDNumber] = @NationalIDNumber
        ,[BirthDate] = @BirthDate
        ,[MaritalStatus] = @MaritalStatus
        ,[Gender] = @Gender
    WHERE [EmployeeID] = @
```

Ⓥ	@EmployeeID	int
Ⓥ	@NationalIDNumber	nvarchar(15)
Ⓥ	@BirthDate	datetime
Ⓥ	@MaritalStatus	nchar(1)
Ⓥ	@Gender	nchar(1)

Type @ symbol to display popup with variable names

For local variables prefixed with a single @ character, the popup item list includes all variables declared above the current line in the same unit of code (such as a stored procedure, trigger, function, or other procedural object), in other words, all variables declared between the preceding CREATE or ALTER command and the current cursor position.

For global variables prefixed with double @@ characters, the popup item list includes all standard SQL Server global variables. For Sybase based targets, the popup item list includes all standard Sybase ASE (ASA) global variables.

For all other database systems not using T-SQL dialect, the variable names appear as a special expandable item within content of the object names popup and column names popup. The name of this special item is "**local variables.**" The following picture shows an example popup with local variable names and parameters for Oracle, DB2 and other database systems.

```
CREATE OR REPLACE FUNCTION OE.GET_PHONE_NUMBER_F
  (p_in INTEGER, p_phonelist phone_list_typ)
RETURN VARCHAR2 AS
  TYPE phone_list IS VARRAY(5) OF VARCHAR2(25);
  phone_out varchar2(25) := null;
  v_size INTEGER;
BEGIN
    IF p_phonelist.FIRST IS NULL OR
       p_in > (p_phonelist.LAST + 1) - p_phonelist.FIRST THEN
       RETURN phone_out;
    ELSE
       phone_out := p_phonelist(p_in);
       RETURN
    END IF;
END;
```

This is how the popup appears after "variables variables" branch is expanded

press Arrow Down and then Arrow Right keys to expand "local variables" branch

▣ Ⓥ	local variables	
⊞ 🗒	access$	table
⊞ 🗒	alert_qt	table
⊞ 🗏	all_all_tables	view
⊞ 🗏	all_apply	view
⊞ 🗏	all_apply_conflict_columns	view
⊞ 🗏	all_apply_dml_handlers	view
⊞ 🗏	all_apply_enqueue	view
⊞ 🗏	all_apply_error	view
⊞ 🗏	all_apply_execute	view
⊞ 🗏	all_apply_key_columns	view
⊞ 🗏	all_apply_parameters	view
⊞ 🗏	all_apply_progress	view
⊞ 🗏	all_apply_table_columns	view

▣ Ⓥ	local variables	
Ⓟ	p_in	integer
Ⓟ	p_phonelist	phone_list_typ
Ⓥ	phone_out	varchar2(25)
Ⓥ	v_size	integer
⊞ 🗒	access$	table
⊞ 🗒	alert_qt	table
⊞ 🗏	all_all_tables	view
⊞ 🗏	all_apply	view
⊞ 🗏	all_apply_conflict_columns	view
⊞ 🗏	all_apply_dml_handlers	view
⊞ 🗏	all_apply_enqueue	view
⊞ 🗏	all_apply_error	view

The "local variables" section also appears in the keyword popup after you type **SET** keyword.

Like any other SQL Assistant list, as you type the code, the context of the variable names list is automatically filtered to display matching items only.

Note that for non T-SQL based systems, the collapsed "local variables" branch is expanded automatically if you continue typing when the popup is already displayed and your typing matches beginning of a variable name available in the branch.

Using Package Variable Names and Type Names Completion Features

This feature is similar to Local Variables names completion feature except that it is specific to Oracle database servers only. Refer to the description of the Local Variables completion feature in the previous section for instructions on where to find and how to use these features.

Tip: When looking for package variables, SQL Assistant parses the content of the current file as well as analyzing package code stored in the database server system catalog tables. If you are working on a package body for which its package header has not been compiled yet and the package header code is not available in the current file, do not expect SQL Assistant to find package level variable declarations. Always compile package headers before you reference their attributes in package body. This will not only help SQL Assistant find the declarations, but it will also help SQL Assistant validate procedure and function declarations appearing in the package body and check for correct syntax.

Using User/Role Names Completion Features

The user/role name list popup appears when SQL Assistant is invoked after GRANT, REVOKE and DENY commands. For example, if you type the following line:

GRANT SELECT ON MyTable TO

SQL Assistant will display a list of users and roles to which you can grant the SELECT privilege. Note that SQL Assistant also automatically displays object list popup after the ON keyword.

The following screenshot demonstrates the user name completion feature.
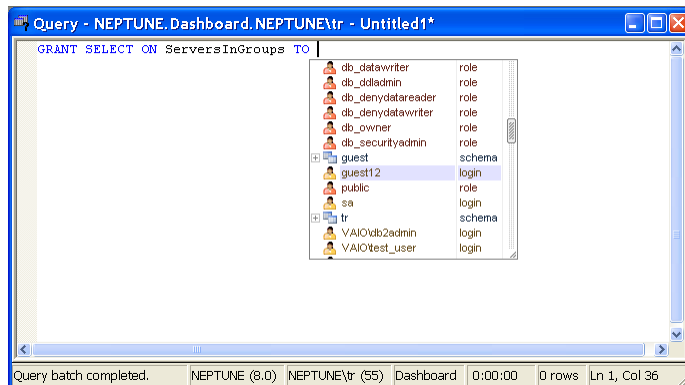
Using Code Auto-Expansion and Auto-Generation Features

Automatic Generation of DML Statements

SQL Assistant supports several handy code auto-generation features. Code auto-generation is context-based and is triggered in certain places within the SQL code. Several examples are provided below.

If you select a table, view or function name from the popup list that displays after you type the SELECT keyword, SQL Assistant inserts into the code the complete SELECT statement for the selected object. Object columns are inserted after the SELECT keyword following the FROM clause containing the selected object, table or view. Similarly, if you choose an object that appears after typing an INSERT or UPDATE keyword, SQL Assistant generates and inserts the complete text of the INSERT or UPDATE statement into the code.

The following SQL Server specific example demonstrates the SQL statement code generation feature.
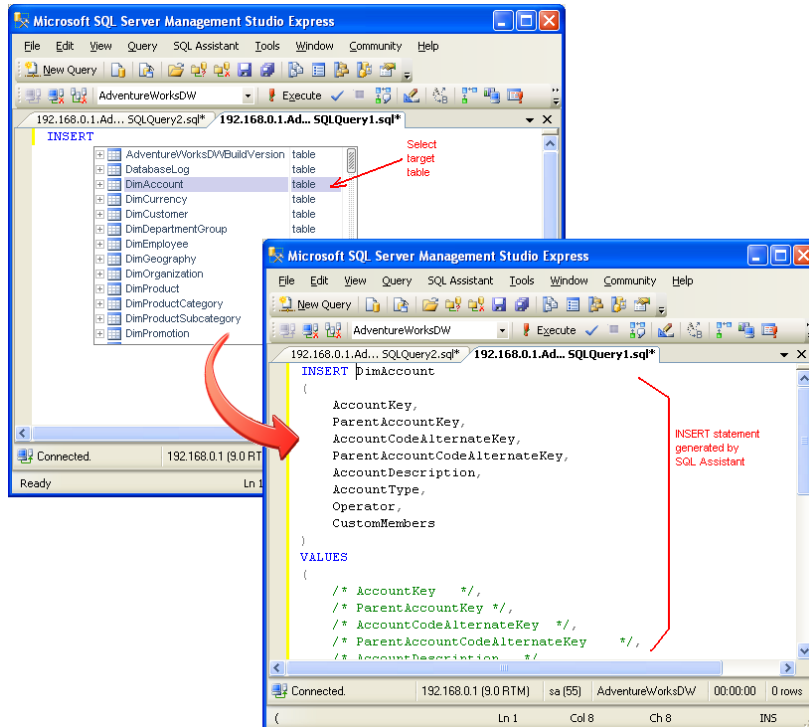
Tips:

- Hold down the SHIFT key while choosing an item in the SQL Assistant popup appearing after the initial SQL statement keyword to paste just the selected object name without generating additional code.

- If you don't want to paste or auto-generate code when the popup appears after the initial SQL statement keyword, just continue typing normally. The popup will disappear automatically.

- You can also press the Esc key to dismiss the popup.

Automatic Generation of Variable Declarations

SQL Assistant can automatically generate variables for table, view or table function columns. This feature can be useful when coding cursors, batch selects, and updates as well as similar SQL operations requiring declaration of a place holder variable for each column in a given object.

To use this feature:

1. Type the DECLARE keyword then type a space. SQL Assistant will display a list of objects in the database.

2. From the popup list, select the object you want to target. SQL Assistant will automatically insert as many

variable declarations into the code as there are columns in the selected object.

Note: The names and data types of the declared variables exactly match the names and data types of corresponding columns in the selected database object.

Following is a SQL Server specific example that demonstrates the automatic code generation feature.
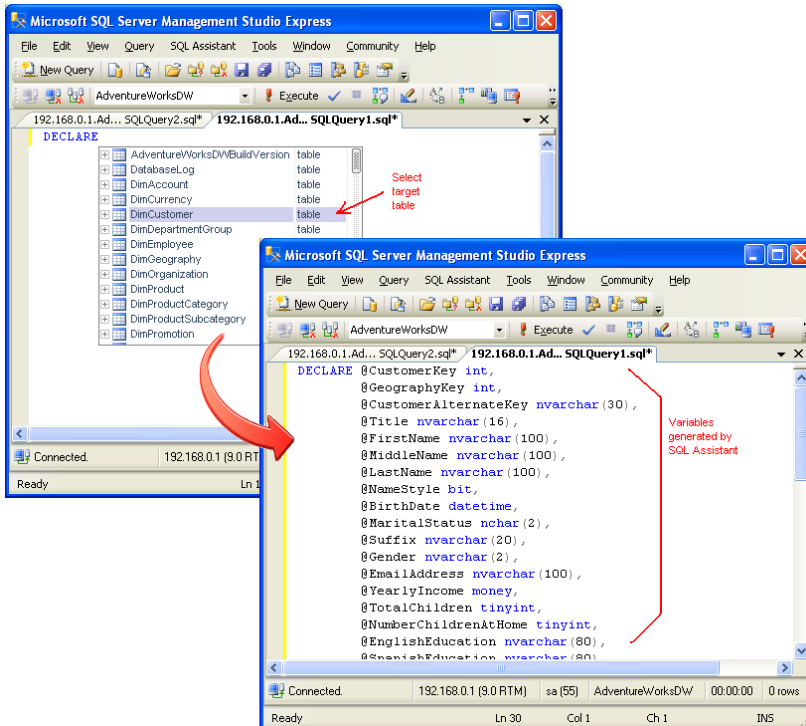
Tips:

- If you don't want to paste or auto-generate any code when the popup appears after the DECLARE keyword, just continue typing the code normally. The popup will disappear automatically.

- Hold down the SHIFT key while choosing an item in the SQL Assistant popup appearing after the DECLARE keyword to paste just the selected object name without generating additional code.

- You can also press the Esc key to dismiss the popup.

Advanced Interactive Code Snippets

Using SQL Assistant's advanced code snippets features, you can automatically generate entire code blocks with complete procedural and business logic based on the popup selections. Interactive code snippets can take your object selection and for the selected objects obtain from the database its attributes, parameters, columns and so on and then plug this information into snippet placeholders, For example, you can use the predefined *cfetch* snippet to generate a complete block of code with table variable declarations, cursor declaration and other cursor loop and handling commands as on the following picture.
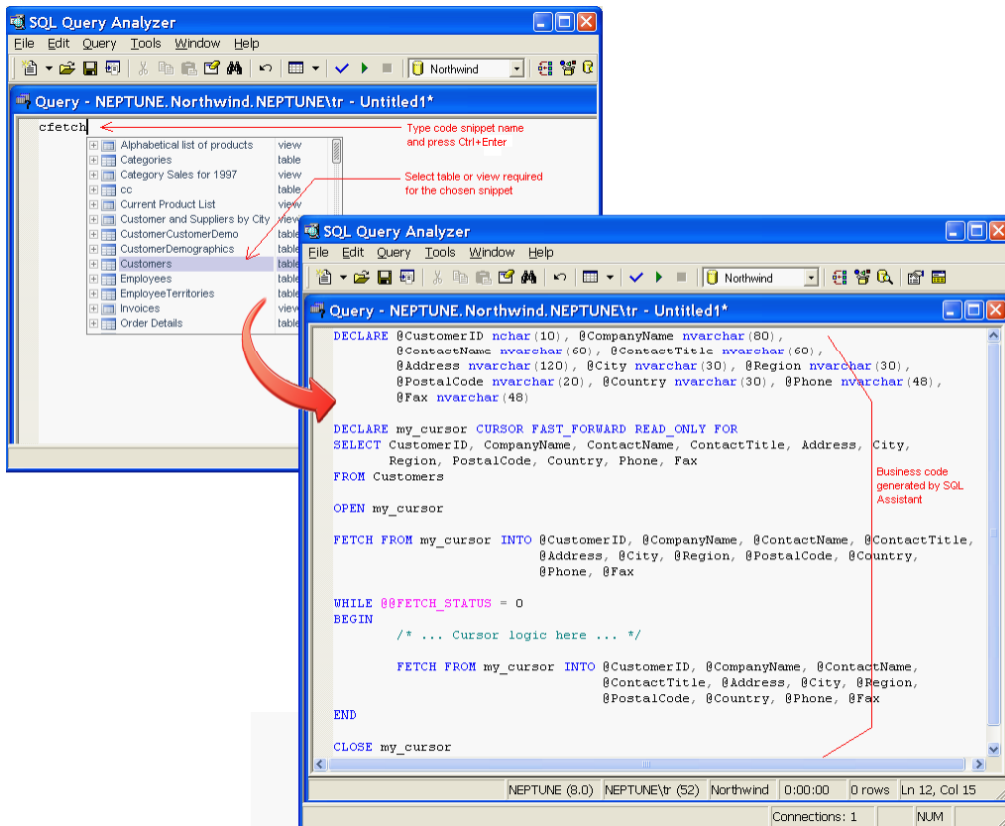
For more information for how to use advanced code snippets, see CHAPTER 7, Code Entry Automation using Code Snippets

Advanced Code Expansion for * and Object Columns and Arguments

SQL Assistant constantly monitors SQL queries entered into the editor and senses SQL syntax elements that you might need to expand. For example, after you enter the following code line

```
SELECT * FROM Categories
```

If you place the cursor over the asterisk symbol, SQL Assistant will highlight that symbol and display a hint. suggesting the expansion of the asterisk wildcard. If you click on the hint or simply use the default Ctrl+Space hot key, SQL Assistant will display a list of columns for the table referenced in the SELECT statement. You can then use the column popup to choose columns that you want to use in place of the asterisk. The following example image demonstrates how it works in the editor.
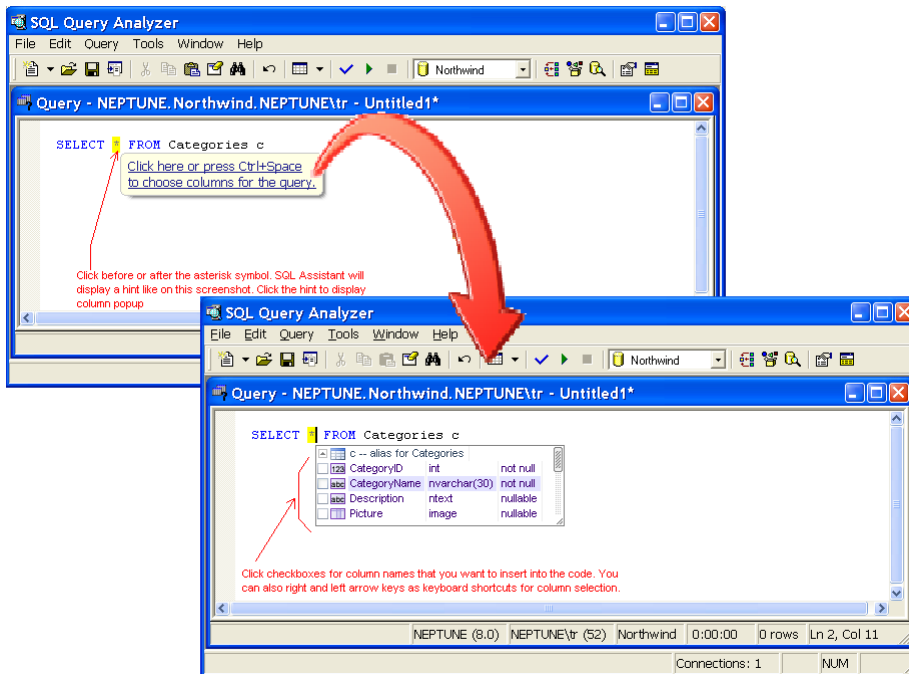
Tips:

- Use the Right Arrow or Left Arrow navigation keys to select columns using the keyboard only. Use Up Arrow and Down Arrow keys to scroll the popup list.

- You can delete an item from the code by selecting it in the popup and then deselecting it.

- Use the Esc key to dismiss the popup at any time.

- Code Auto-Expansion is available for various SQL code elements. For example, to expand a query by adding additional columns to the SELECT clause, click after the name of the table to which you want to add columns and press Ctrl+Space. Alternatively, you can wait a second or two for the hint and then click the hint.

- To get assistance with DDL commands like CREATE TABLE, DROP STATISTICS, ALTER TRIGGER and many others, click near the keywords of the statement for which you need help and then press the hot key for SQL Reference. The default hot key for SQL Reference is Ctrl+F1.

Advanced Code Expansion and Reference for DDL Commands

SQL Assistant constantly monitors SQL queries entered into the editor and senses DDL commands that you might need help with. For example, if you type a code like the following:

```
ALTER TABLE Categories ADD CONSTRAINT
```

If you pause for a couple of seconds after entering this text, SQL Assistant will display a hint as demonstrated on the following screenshot.
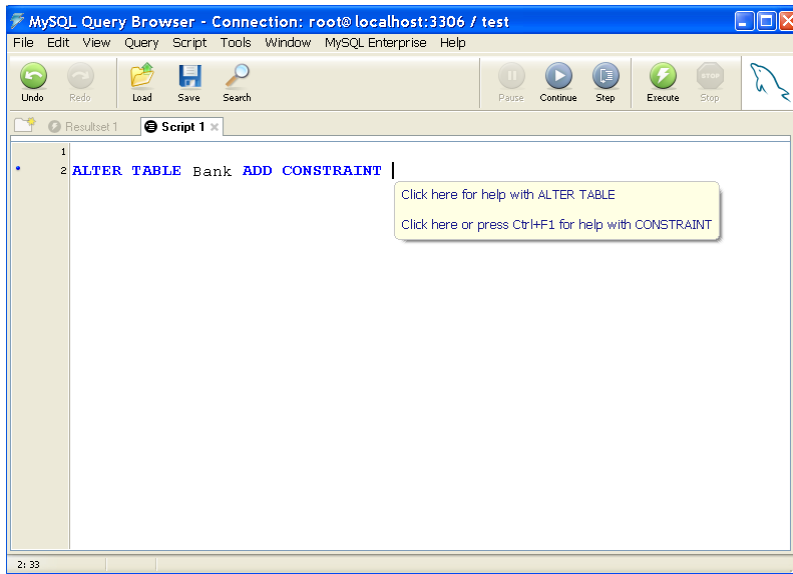
If you click the first hyperlink offering help with the ALTER TABLE syntax, SQL Assistant will popup a context-based SQL Reference window for the ALTER TABLE command. You can use the SQL Reference window to interactively build the ALTER TABLE command.
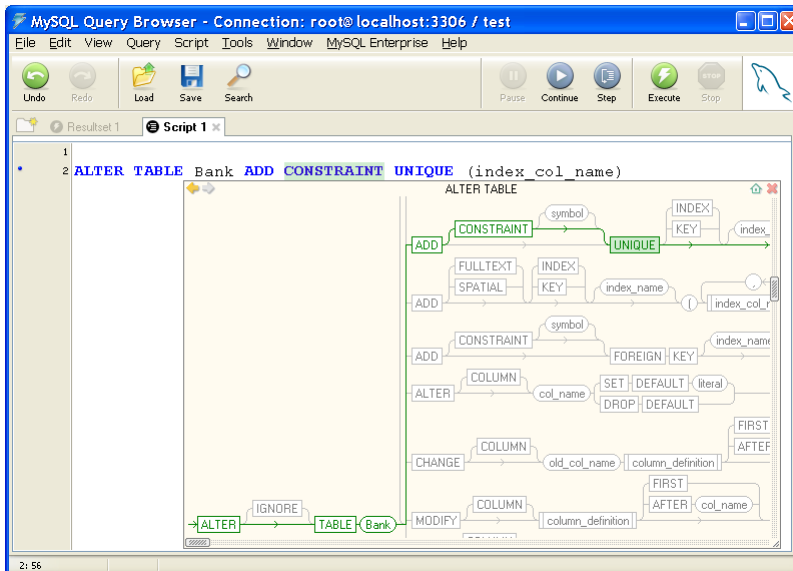
Similarly if you pick the second hyperlink offering help with the CONSTRAINT syntax, SQL Assistant will popup SQL Reference topic with description of CONSTRAINT related types and available options.

For details on how to use SQL Reference, see CHAPTER 9, Interactive SQL Reference System

Working with SQL Assistant Popups

The following screenshot shows a typical SQL Assistant popup.
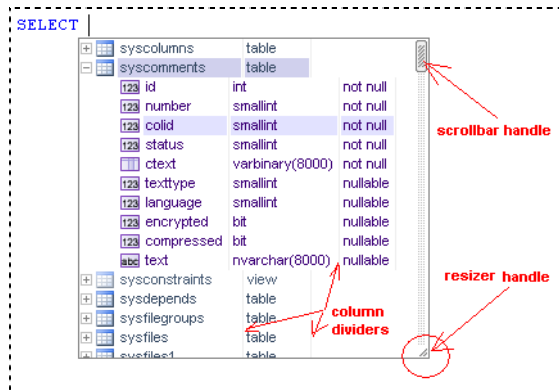
The following topics describe how you can use the keyboard and mouse actions to work with the popup.

Navigation Keys

The following navigation keys are supported in the SQL Assistant popups.

Down Arrow key moves the logical selection to the next item. If pressed immediately after the popup appears on the screen, it selects the top item. If pressed while the last visible item is selected and there are more items available in the list, then it scrolls the content down to the next item and selects it. Can be repeated until the last available item is reached.

Up Arrow key works just like the Down Arrow but in an opposite direction.

Down Page key scrolls the content by one logical page and moves the logical selection to the top visible item. The logical page size is controlled by the popup list size and can be adjusted as described in the Resizing Content topic later in this chapter.

Up Page key works just like the Down Page but in an opposite direction.

ALT+End keys, when pressed simultaneously, move the logical selection to the last item in the popup list.

ALT+Home keys, when pressed simultaneously, move the logical selection to the first item in the popup list.

Left Arrow and **Right Arrow**– select an item in the current line and works with column and argument popups with multiple-choice options. If the selected item is expandable, they expand or collapse the branch.

Tips:

- The most efficient way to use SQL Assistant popups is to start typing the item you want so that only items beginning with that text remain in the list and then using the Down Arrow key move the selection to required item and then hit the Enter key to paste the selected item text into the editor.

- You can also use the computer mouse to scroll the SQL Assistant popup content and then click the item you want to paste into the editor.

Selection Keys

The SQL Assistant supports several alternative item selection keys:

- The Enter key is the default key, which is the standard key used in all Windows controls to select an item in a list and other multiple-choice selection control. This key also allows you to tab through the text while SQL Assistant popup remains displayed on the screen.

- Tab key is an alternative key, which is supported for compatibility with Microsoft development environments featuring Intellisense® technology. This key requires two-hand typing – right hand for scrolling and selecting items in a list and left key for pressing the Tab key.

Tip: Using SQL Assistant options you can customize which selection keys can be used with SQL popups and other features. You can choose to use a single key such as Enter or Tab or you can choose to use both keys. See the Customizing SQL Assistance Types topic for more information.

Scrolling Content

The content of the SQL Assistant popup can be scrolled using either the keyboard navigation keys Arrow Up/Down and Page Up/Down or using the mouse. When using the mouse you can scroll the popup content by dragging its scrollbar handles or clicking little arrows available at the extremes of scrollbars to scroll in small increments. See the sample screenshot with comments at the beginning of the Working with SQL Assistant Popups topic for information on where to locate scrollbar handles.

Resizing Content

To resize the SQL Assistant popup, drag the resizer handle in the bottom-right corner of the popup window. See the screenshot at the beginning of the Working with SQL Assistant Popups topic for information on where to locate the resizer handle.

Note that the new popup window size will be remembered and used in the future for the popup windows with the same kind of content. To re-enable automatic popup window sizing, right-click the popup window and its context menu choose **Automatic Window Sizing** menu.

Resizing Individual Columns

Depending on the popup type, several columns of text can be displayed in the item list. Moreover different parts of the list can have different number of columns and column width and positions. If the column width is insufficient and some parts of text appear cropped, you can resize these columns to see the complete text. Note that thin gray vertical lines indicate column boundaries. To change size of a column, rest the mouse pointer over the vertical line indicating the right boundary of that column. The mouse pointer should change to ╫. Press the left mouse button and while holding it pressed drag the line to the desired position. See the marked screenshot in the beginning of the Working with SQL Assistant Popups topic for information on where to locate lines indicating column boundaries.

Moving Content

If the SQL Assistant popup covers part of the editor screen that you want to see, click on an empty area within the popup window and drag the popup window to a more convenient location on the screen.

Refreshing Content

For performance reasons, SQL Assistance uses in-memory cache for the catalog data retrieved from the database so that it does not need to query the database each time it needs to display SQL Assistant popups with the same content. This internal cache is not updated automatically when changes occur in the database catalog data during active SQL Assistant sessions. For example, when new stored procedures or tables are created in the database or when table columns are altered, SQL Assistant is not automatically made aware of these changes so they are not reflected in the content of popups. You can use any of the following methods to refresh the internal SQL Assistant cache:

- Method 1: Select the **Refresh Cache** command from one of SQL Assistant menus.

- Method 2: While the popup is displayed on the editor screen, press the **F5** hot key. Note that this Refresh method cannot be used if F5 key is also used as a hot key in the editor.

- Method 3: While the popup is displayed on the editor screen, right click on the popup and click the **Refresh** command in the context menu.

Important Notes: The cache is reloaded automatically after DDL operations so that changes in the schema objects can be seen immediately in the Intellisense popups, and in other places. SQL Assistant monitors SQL queries being executed in the editor, and whenever it encounters ALTER, CREATE, or DROP commands in the current batch, it marks the cache dirty, and then reloads it after the SQL execution is complete. When working with large databases and performing frequent schema changes, for example, executing ALTER PROCEDURE and similar commands for database code changes, frequent cache reloading might not be a convenient feature as it may create undesired performance effect. You can use the Disable/Enable Automatic Refresh control in the Database Explorer toolbar to temporarily turn off the automatic refresh feature.
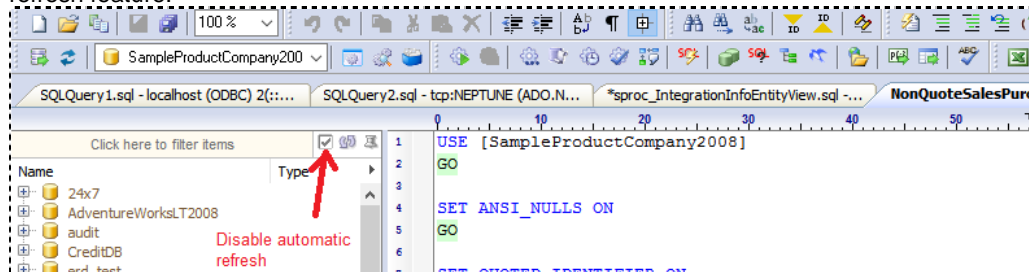
Using Mouse-over Hints

Mouse-over balloons provide you with an informative hint pertaining to an object referenced in the target editor code, such as stored procedure name, table name, variable name and so on. You can use either of the following methods to display mouse-over hints:

- **Timed hint** - Hover mouse pointer over some object name, column, or variable name referenced in the code and leave it there for a couple of seconds. If the object under the mouse pointer is a valid object in the current database context or a valid variable declared somewhere in the current code context, a mouse-over hint will appear as an informational balloon above or below the object name.

- **Immediate hint** - Use the hot-track feature to force immediate display of the hint. While holding the Ctrl key down, hover mouse pointer over an object name, column, or variable name referenced in the code. The name under the mouse pointer should turn into a hyperlink. If this name is a valid object name in the current database context or a valid variable name declared somewhere in the current code context, a mouse-over hint will appear immediately as an informational balloon above or below the object name.

There is an important difference between mouse-over hints displayed for already entered code and interactive popups appearing while you are typing new code. Mouse-over hints are designed to aid in reviewing and analyzing SQL code, while interactive popups are designed to aid in entering SQL code and performing various code completion functions.

The following screenshots demonstrate examples of a mouse-over hint displayed for a stored procedure and an interactive prompt for code completion displayed for the same stored procedure during code typing:

Mouse-over hint
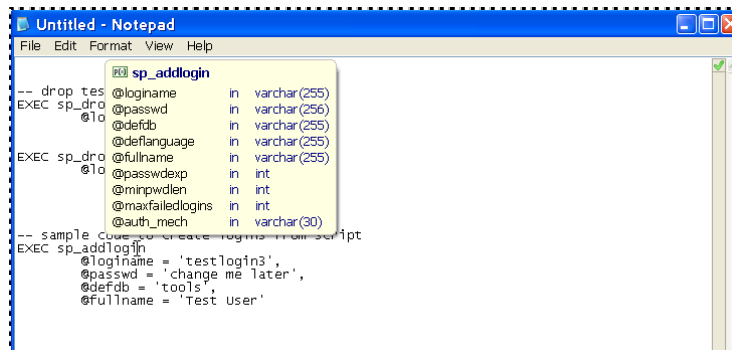
Interactive prompt for code completion and parameters
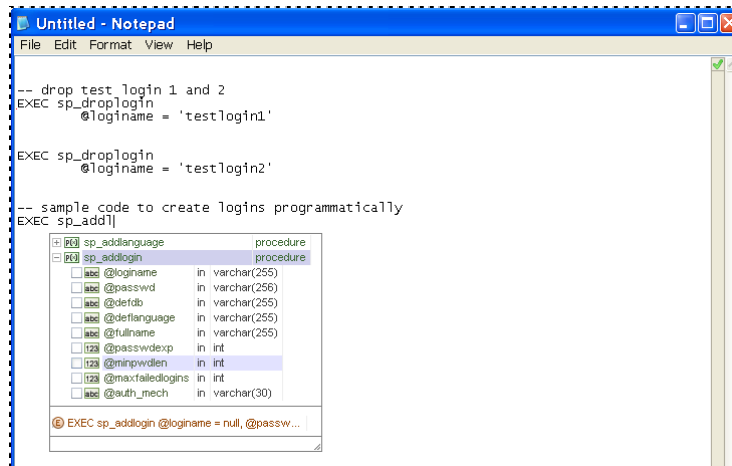
Mouse-over hints are available for references for the following types of objects and variables:

- Script variables – this hint displays declaration of the variable data type with a hyperlink that can be used to jump to the actual line where the variable is declared. This type of mouse-over hint also supports references to arguments declared in user-defined functions and stored procedures.
- User-defined functions and stored procedure signatures, including package functions and procedures in Oracle – this hint displays declaration of function / procedure arguments and its

return type, as demonstrated on the previous screenshot.

- Tables – this hint displays declaration of table columns and their data types.
- User-defined table-functions – this hint displays declaration of table-function arguments, columns and their data types.
- Views – this hint displays declaration of table columns and their data types.

Using Data Preview and Code Preview Hyperlinks in Mouse-over Hints

Data Preview and **Code Preview** hyperlinks are available in certain types of mouser-over hints. Data Preview hyperlinks are available in hints displayed for tables and views. Code Preview hyperlinks are displayed in hints for most types of procedural objects, such as stored procedures, functions, view, packages and types in Oracle, as well as for tables and views.
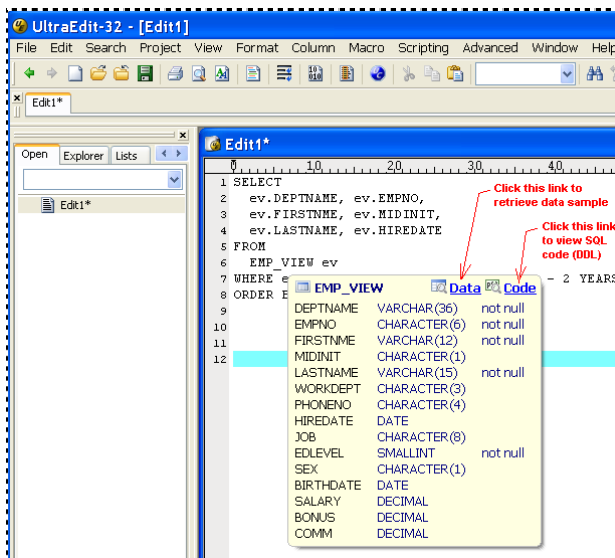
Click the **Data** hyperlink or click icon to retrieve sample data and display it in the Data Preview pane.

Click the **Code** hyperlink or click icon to reverse-engineer DDL code of the object and display it in the Code View pane.

Using the Column and Variable Data-type Hints Feature

SQL Assistant supports interactive mouser-over hints for column, variable, and argument data-types.

To trigger a hint for a particular table or view column referenced in the code, rest the mouse pointer over that column name. After a brief delay, SQL Assistant will display a balloon with the column declaration. Note that the code syntax must be valid in order for SQL Assistant to recognize which table or view owns the column under the mouse pointer.

To trigger a hint for a particular variable referenced in the code, rest the mouse pointer over that variable. After a brief delay, SQL Assistant will display a balloon with the variable declaration and a hyperlink.
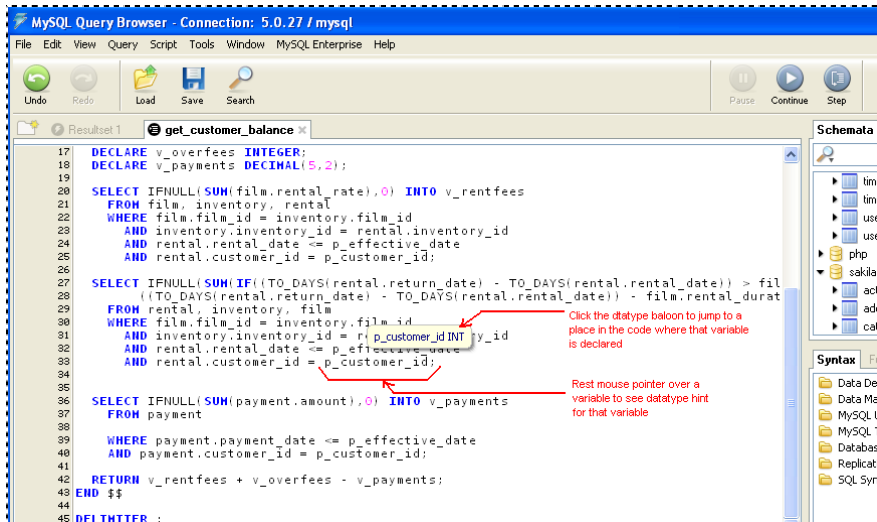
To jump to the actual variable declaration, move the mouse pointer over the balloon. A hyperlink will appear within the hint. Click the hyperlink and SQL Assistant will scroll the text in the target editor and place the cursor in front of the variable declaration.

Using the Keyword Capitalization and Formatting Feature

SQL Assistant can be configured to automatically format keywords as you type them. The set of keywords recognized by the program is controlled by the Code Formatting settings you select in SQL Assistant options.
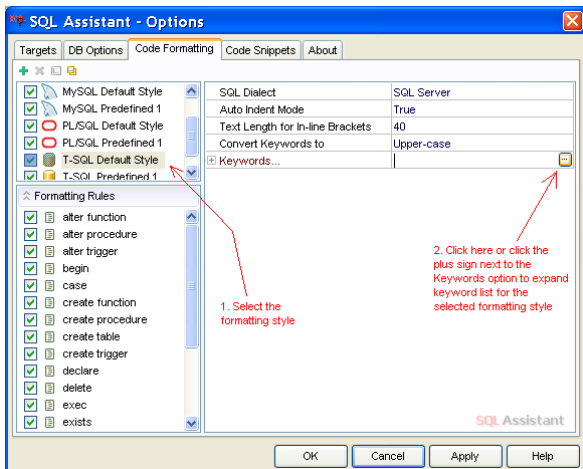
If the Keyword Capitalization feature is enabled, SQL Assistant automatically formats keywords you type into the editor using the case conversion rules specified in the **Convert Keywords** rule. For example, if you type `alter table mytable` and the **Convert Keywords** rule is set to **Uppercase,** SQL Assistant automatically updates the entered text to `ALTER TABLE mytable`.

You can customize the case conversion rules used to format individual keywords in SQL Assistant Options. You can also disable the automatic keyword formatting feature. For more information, see the Customizing Keywords Used With the Keyword Capitalization Feature topic in CHAPTER 48.

Using the Automatic Tab-Replacement Feature

SQL Assistant can be configured to automatically insert spaces in place of tab characters. Turning this option on will cause SQL Assistant to replace tab characters with spaces as you type tabs in your code. It will also replace tabs you enter in code snippets and formatting rules with spaces when these features are used.

The number of spaces used for tab replacements is determined by the **Number of Spaces per Tab** option which can be configured on the **Targets** tab in SQL Assistant's Options dialog. This option can be set separately for each type of editor. Use the **Replace Tabs with Spaces** option in the same place to enable / disable tab replacement.

Using the Smart Auto-Indent Feature

If the Auto-Indent option is enabled, pressing the Enter key in the target editor adds a new line with the caret positioned on it, along with the indent which SQL Assistant assumes to be reasonable in the current code point. The size of the indentation (amount of spacing) is based on the code formatting patterns configured for the current SQL dialect. It is calculated from the left side of the editor screen in accordance with the SQL code format patterns for the current SQL statement and relative to the indent of the previous line. For example, if you are writing an Oracle PL/SQL procedure and typing some text like in the following examples within a BEGIN…END construct:

```
BEGIN
    IF my_variable > 0 THEN
    END IF;
END;
```

and press Enter key after the THEN keyword, SQL Assistant will automatically insert a new line and will enter as many spaces (or tabs) as configured in the format patterns. The resulting code will look like the following:

```
BEGIN
    IF my_variable > 0 THEN
        |
    END IF;
END;
```

Here, the pipe sign represents new position of the editing caret. In comparison, if you press the Enter key after a text line like "*my_variable := 55;*" the indent of the inserted line will be the same as the indent of the previous line.

```
BEGIN
    my_variable := 55;
    |
END;
```

Tip: SQL Assistant also automatically un-indents closing brackets, END, END IF, END LOOP, LEAVE and some other compound SQL statement closing keywords so that the indent of these SQL code closing keywords matches the indent of the opening keywords;

See the Customizing Code Formatting Patterns topic in CHAPTER 48 for details on where and how to change code formatting patterns.

Using the Smart Undo Feature

SQL Assistant can be configured to alter the behavior of your SQL Editor's undo operations activated after use of Ctrl+Z hot key. It can make the undo behavior smarter.

SQL Assistant intercepts Ctrl+Z hot key press and analyzes the position of the last text change and the current cursor position. If the position of the last change is not visible on the screen, the editor window scrolls to that position and the edit caret is positioned at the undo point. This behavior allows you the opportunity to review the code and decide whether that place is the place where you actually want to undo your changes.

To enable or disable the Smart Undo feature, use the Smart Undo option available on the **Target** page in SQL Assistant Options. Note that this feature can be set selectively for specific editors only.

Using the Smart Text Navigation Feature

SQL Assistant can be configured to alter the behavior of your SQL Editor's handling of the Home and End keys.

SQL code is typically a well-formatted document with blocks of code recursively indented to improve text structure and to simplify reading and understanding of the implemented business logic. While working with SQL code, you may need to add text at the start of a text line. Normally if you press the Home key, the editor moves the caret to the 0 position at the beginning of the line. You then press the Right Arrow key several times to skip some number of white-space characters at the beginning of the line. With SQL Assistant and smart code navigation enabled, a single press of the Home key places the edit caret just in front of the first non-white space character in the line. Similarly, the End key places the edit caret after the last non-white space character.

If you press the Home key a second time, the caret moves to position 0 at the beginning of the line. Similarly, pressing the End key a second time places the caret after the very last character in the line, including any trailing white-space characters.

To enable or disable Smart Text Navigation feature, use the Smart Text Navigation option available on the **Target** page in SQL Assistant Options. Note that this feature can be set selectively for specific editors only.

Highlighting of Trailing White-space Characters

This feature works in tandem with Smart Text Navigation. In addition to improved start and end-of-line navigation, SQL Assistant implements highlighting of trailing white-space characters. The highlighting occurs in

the current line after you press the End navigation key once. The highlighting disappears as soon as you press any key. See the Using Smart Text Navigation Feature topic for more details.

Highlighting of Matching Column/Value Pairs in INSERT Statements

In addition to standard assistance with auto-generation of code for DML statements, SQL Assistant supports two additional features for helping with manual coding of VALUES clauses in INSERT statements. These features help to avoid misplaced values by ensuring that values are inserted into correct table columns.

As you type the values, after each comma entered, SQL Assistant displays the Columns popup synchronized with the column list in the INSERT part of the current statement and filtered to match the current position of the value. The following screenshot demonstrate how this feature works.
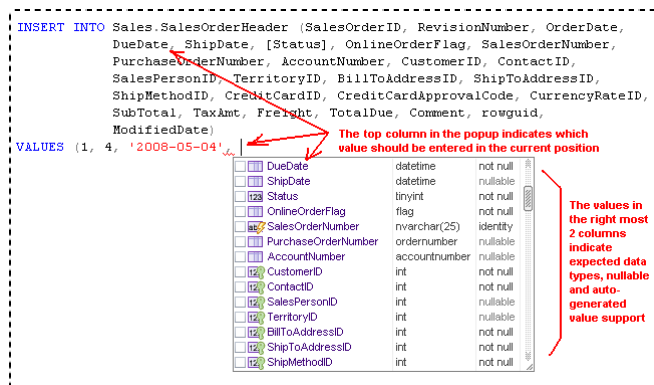
Note: The order of suggested columns in the popup is based on the physical order of columns in the referenced table. It is assumed that the columns are listed in the same order in the INSERT INTO column list.

You can use Column/Value pair highlighting feature to verify columns and values match properly. To see pairs of matching values and column names, click anywhere within the value of interest. SQL Assistant will highlight the value in the VALUES list and its matching column name in the INSERT INTO list. You can also click on a column in the INSERT INTO list, and SQL Assistant will highlight that column name and its matching value in the VALUES list.

The following screenshot demonstrate how this feature works.
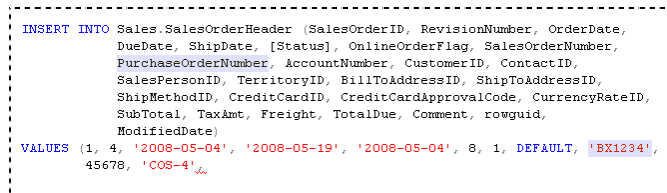
Highlighting of Current SQL Statement with a Single Keypress

By default, the Ctrl+Alt+H hot key can be used to highlight the current SQL statement. The hot key can be customized in SQL Assistant Options.

The primary purpose of the statement highlighting function is to support quick syntax checking and execution of the current SQL statement. The highlighting statement can be syntax checked or executed using SQL Assistant's hot keys Ctrl+F9 and Ctrl+Shift+F9 or similar hot keys in the target editor. For example, many SQL editors such as the SQL Server Management Studio and Toad support the F5 hot key for executing highlighted blocks of SQL code. In the DB Tools, the Ctrl+Shift+R hot key performs the same function. Consult your editor's documentation for details on the supported hot keys.

Tips:

- The current statement is defined as a SQL statement having its text surrounding the edit caret. This definition is also code context sensitive. For example, if the caret is within a SELECT statement contained within a CREATE PROCEDURE body that contains multiple nested SQL statements, pressing Ctrl+Alt+H will highlight the SELECT statement. If the caret is within the CREATE PROCEDURE line itself, pressing Ctrl+Alt+H will highlight the entire procedure code, starting with the CREATE PROCEDURE keyword and ending with the last statement within the procedure body.

- Repeated use of the Ctrl+Alt+H triggers incremental expansion of the current selection> For example, if CREATE PROCEDURE body contains nested BEGIN...END construct and within BEGIN...END there is SELECT statement with the edit caret within the text of the SELECT statement, then the first time the Ctrl+Alt+H is the pressed, the SELECT statement is highlighted. The second time it is pressed, the BEGIN...END is highlighted with all its nested statements. The third time it is pressed, the entire CREATE PROCEDURE is highlighted, and so on...

One-Click Actions for Specially Formatted Comments

SQL Assistant recognizes single line **/*{ }*/** comments containing text between curved brackets as specially formatted comments. Specially formatted comments are treated as placeholders for entry fields. In SQL Assistant options, you can customize the action of these fields. By default, a click anywhere within a specially formatted comment causes SQL Assistant to select the entire comment text so it can be immediately replaced by typing new text.

SQL Assistant generates specially formatted comments when completing INSERT, UPDATE, and CALL statements. Following is an example:

```
INSERT INTO HumanResources.JobCandidate
(
    -- JobCandidateID -- this column value is auto-generated
    EmployeeID,
    [Resume],
    ModifiedDate
)
VALUES
(
    125,
    /*( [Resume] )*/,
    /*( ModifiedDate )*/
)
```

You can use SQL Assistant's Options dialog to customize the default behavior of specially formatted comments.

1. Open the **Options** dialog.

2. Activate the **DB Options** tab.

3. On the left, select the **SQL Assistance** type that is matching your database type.

4. On the right, expand the **Auto Complete…** group of options. Modify the **Action for Specially Formatted Comments** option as required. The following mouse click action types are supported:

   - **None** -- Processes clicks normally; no special action is required.

   - **Select** – Select the entire comment text including the opening and closing tags. This is the default action.

   - **Delete –** Automatically delete the entire comment text including the opening and closing tags.

Converting SQL Queries to Application Code

In many programming languages SQL commands are stored in string variables and tedious to code, they sometimes pages long and need special symbols and quotes escaped. It's also easy to misspell schema object names and columns without SQL syntax validated in design time. Consider the following Java example:

```
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery("String sql = "SELECT c.CustomerId, c.FirstName,\n"
          + "c.LastName, c.Company, e.EmployeeId, e.Phone  \n"
          + "FROM employees AS e \n"
          + " JOIN customers AS c \n"
          + "    ON c.SupportRepId = e.EmployeeId \n"
          + "WHERE c.City = 'Paris' \n"
          + "ORDER BY c.FirstName, c.LastName"; ");
```

SQL Assistant can help you generate well formatted query strings.

1. Highlight text of the SQL query you want to convert.

2. In the right-click menu select **Copy SQL As…** command.

3. Choose the desired programming language in the next level menu.

The text of the query will be converted and copied to the Clipboard. You can then paste it into your development environment.

A number of programming language specific conversion patterns are preconfigured in SQL Assistant settings. If your programming language is not listed, use the same **Copy SQL As…** command as described above, and then choose the top command **Modify Language Formats…** This command will open the **Language Formats** dialog in which you can add additional conversion rules and modify the existing rules too if you want to customize them.

CHAPTER 4, Code Structure View and Bird's Eye View

Overview of Code Structure View

The **Code Structure View** tool displays a hierarchical view of the code in a nested, tree-like form adjacent to the left side of the target editor window. The **Code Structure View** lists the most important SQL commands in the order they appear in the SQL query. Click on any command in the **Code Structure** window to display its code in the editor window on the right.

To open the **Code Structure View,** press the Ctrl+F12 hot key. **Code Structure View** can be also opened using SQL Assistant's menu in the system tray or from menus available in the target editor. To use the target editor's menus, the menu integration option must be enabled. For details, see the Manually Invoking SQL Assistant Popups topic in CHAPTER 3.
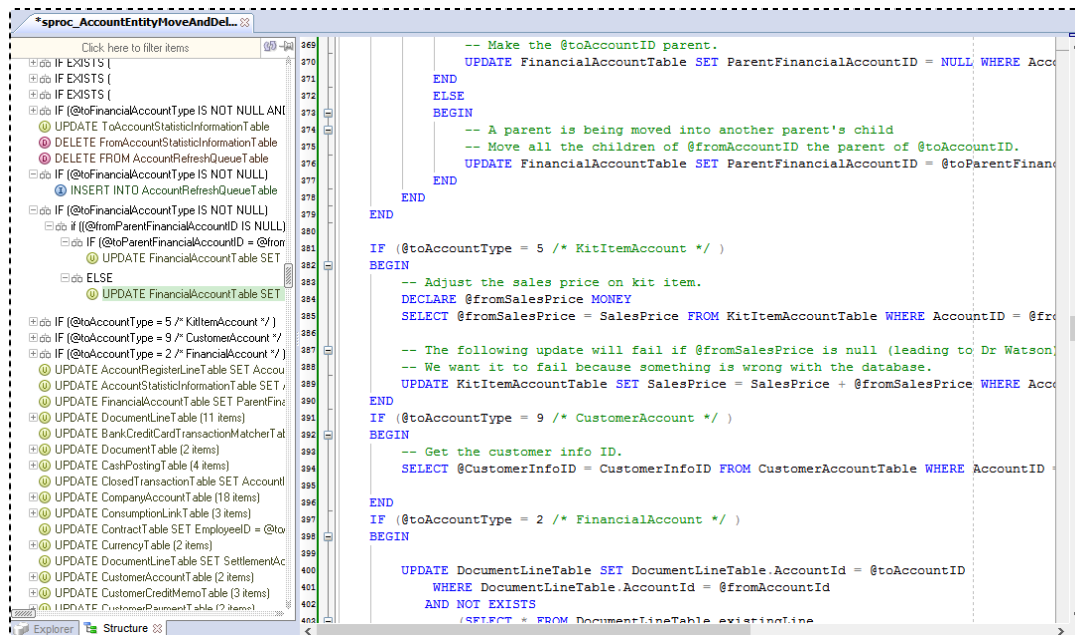
Code Structure View can show references to many different types of SQL commands, including but not limited to the following: CREATE, DROP, EXECUTE, SELECT, INSERT, DELETE, UPDATE, TRUNCATE, PRINT, USE, GRANT, REVOKE, and DENY.

Tips:

- For CREATE PROCEDURE, CREATE FUNCTION and CREATE PACKAGE commands, a plus sign is displayed next to the command. Click the plus sign to expand the CREATE command and display the nested SQL commands which are part of the expanded procedure, function or package.

- A plus sign is also displayed for various loop structures such as FOR, WHILE, UNTIL and generic LOOP loops as well as for conditional IF statements. Just like for CREATE commands, you can click the plus sign to expand the nested SQL commands.

- Different icons and colors are used for different types of SQL statements to improve the visual appearance of the code structure and to allow users to locate the required SQL commands faster.

Working with the Code Structure View Interface

Code Navigation

You can instantly navigate to any command in the target editor by clicking the command name in the **Code Structure View** or by selecting the command name and pressing the Enter key. The editor window will scroll to the selected SQL command and the caret will be set to the beginning of that command line.

Another helpful feature of the **Code Structure View** is dynamic code highlighting. When you move the mouse pointer over command hyperlinks displayed in the Code Structure View, SQL Assistant automatically highlights the corresponding code lines in the editor window.

Expanding / Collapsing Multiple Levels

The **Code Structure View** supports right-click context menus, which can be used to quickly expand or collapse all levels or specific levels only. Commands are available to expand or collapse up to four levels or all levels at once.

Grouping Similar Commands

The Code Structure View automatically groups similar commands and shows them as a single item. A count of grouped commands appears at the end of the line

To see individual commands in a group, click the plus sign displayed in front of the group.

Filtering Content by Schema Object References

To quickly filter code structure view to show only hyperlinks to places in the code referencing specific schema object name or names containing certain substring, click the **Filter** bar above the **Code Structure View** pan. Start typing the name of the object to show only lines in the code related to that object.

To undo the filter, simply erase the text in the **Filter** bar.

Scrolling Content

To scroll the **Code Structure View** window content using the mouse, drag the scrollbar handles as needed, or click the small arrows at the extremes of scrollbars to scroll in small increments. See the screenshot at the beginning of the Working with SQL Assistant Popups topic for information on where to locate scrollbar handles.

Resizing Content

To resize the **Code Structure View** pane, drag the vertical bar separating the pane and the code editor. Note that when you place mouse pointer over the right edge of the **Code Structure View** pane the cursor shape

changes to a double arrow shape

Persisting Code Structure View

By default, the **Code Structure View** pane is not persistent and is not displayed automatically when you open the target editor's code windows. Follow the steps described in the Overview of Code Structure View topic for information on invoking the **Code Structure View** manually.

To persist the **Code Structure View** and make it appear automatically in every code editor window, click the *pushpin* icon in the right-top corner of the **Code Structure View** pane. To disable the **Code Structure View** persistence, click the *pushpin* icon again. Note that the persistence state is indicated by the toggled state of the *pushpin* icon.

Overview of Bird's Eye View

The **Bird's Eye View** tool displays a miniature view of the code loaded in the editor. The view appears adjacent to the left side of the target editor window. The sole purpose of the **Bird's Eye View** is to ease the code navigation while working with large scripts.

You can open the **Bird's Eye View** using the Shift+F12 hot key. **Bird's Eye View** can be also opened using SQL Assistant's menu available in the system tray or from menus available in the target editor. To use the target editor's menus, the menu integration option must be enabled. For details, see the Manually Invoking SQL Assistant Popups topic in CHAPTER 3.
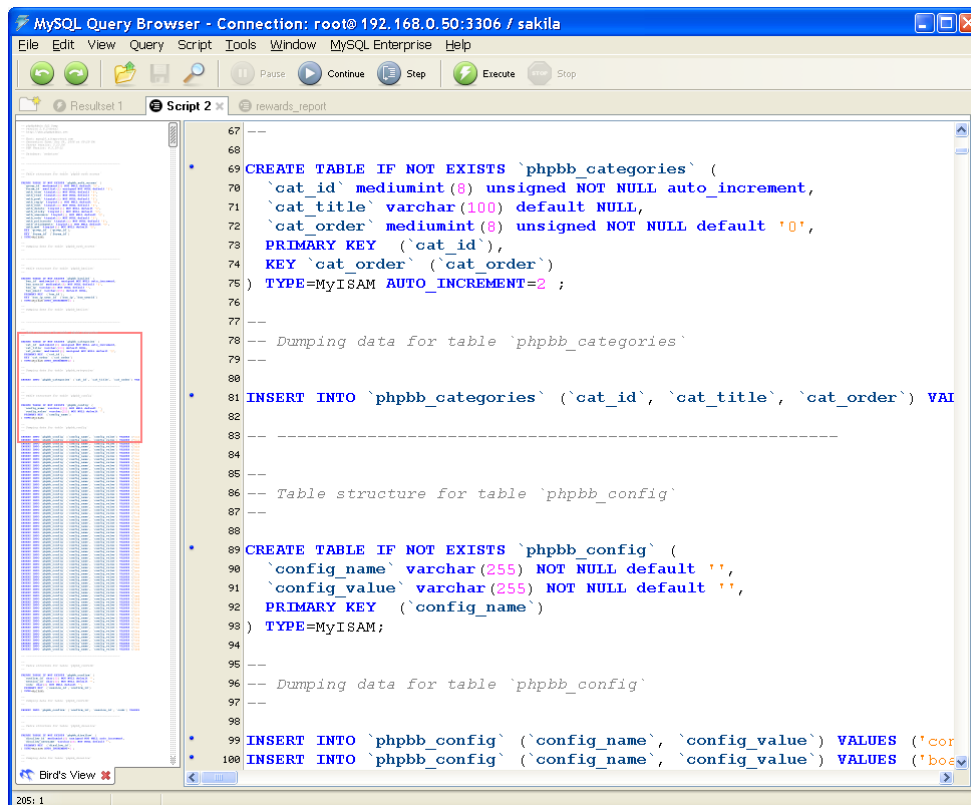
Using Partial Code Display vs. Full Code Display

Fully loading very large scripts into the **Bird's Eye View** and creating scaled down page views may take a while especially on old slow computers. Note that each page view requires taking a separate screenshot. SQL Assistant uses several performance optimization techniques to load the pages faster. On opening of the view, SQL Assistant scans the first 1000 lines of script and automatically builds page views for those lines only. In most editors opened full screen, the first 1000 lines typically represent about 15 to 20 pages of code. If the entire script file is longer than 1000 lines, additional lines are scanned and page views are added to the **Bird's Eye View** as you scroll through the file or enter new code. This Partial Code Display method allows SQL Assistant to work efficiently and quickly update the **Bird's Eye View** in the background**.**

If you would like to force scanning of the entire file right away, right-click the **Bird's Eye View** area and select the **Refresh** command from the context menu. SQL Assistant will scan the entire file and load all page

snapshots into the **Bird's Eye View**. This will provide you with a Full Code Display and allow fast navigation within any part of the loaded file.

Working with the Bird's Eye View Interface

Code Navigation

In **Bird's Eye View,** a red rectangle is used to identify the code lines currently displayed in the editor window. Dragging the red rectangle up or down causes the editor window to scroll up or down to display the code section covered by the rectangle.

You can also use the scroll bars to navigate forward or backward through the code, and you can instantly navigate to any page in the target editor by clicking that page in the **Bird's Eye View.**

Refreshing Content

The **Bird's Eye View** content is refreshed automatically as you make changes in the code and as you scroll through the code pages. However, in certain cases SQL Assistant might be unaware of code changes in the target and as a result will not refresh the view automatically. For example, this can happen if you open an existing file in the same editor window replacing the previous text. This can also happen if you do a global search and replace in the same editor window or multiple windows at once. These kinds of changes are performed in the editor's memory without necessarily updating output to the computer screen. In these cases, you may want to open the **Bird's Eye View** refresh manually not waiting for the screen updates so that the content you see in the **Bird's Eye View** matches what you got in the file.

To force refresh of the view, right-click anywhere in the **Bird's Eye View** pane and click the **Refresh** command on the context menu.

Important Notes: A forced refresh causes SQL Assistant to rescan the entire file. If the file is very large or you are running SQL Assistant on a slow computer, it may take more than a few seconds to rescan the file and create a snapshot of every page. For this reason, if your file is longer than 4000 lines, SQL Assistant displays a prompt asking you to confirm the refresh operation. For more information, see the Using Partial Code Display vs. Full Code Display topic in this chapter.

Scaling the View

The **Bird's Eye View** automatically scales the text size based on the file content type and length. The scaling factor is represented as a fraction of the original text size. For example, the x4 scaling factor means that text in the **Bird's Eye View** will appear four times smaller then the text displayed in the target editor. The actual text size is dependent on the font type and font size used in the editor and may vary for different environments and target types.

To change the default text scaling, right-click anywhere in the **Bird's Eye View** to display the context menu. In the context menu, click the **Scale** command to display its submenu and then choose the desired scaling factor.

Note that the scaling factor is represented as a fraction of the original text size. For example, **x4** scaling factor means that the text in the **Code Bird's View** will appear 4 times smaller then the text displayed in the target

editor. The actual text size is dependent on the font type and font size used in the editor and may vary for different environments and target types.

Scrolling Content

To scroll the **Bird's Eye View** window content, using the mouse drag window scrollbar handles as needed, or click little arrows available at the extremes of scrollbars to scroll in small increments. See the screenshot at the beginning of the Working with SQL Assistant Popups topic for information on where to locate scrollbar handles.

Note that by default SQL Assistant scrolls the content the **Bird's Eye View** automatically and synchronously with the scrolling of the target editor window. The red rectangle indicating the current page is also moved automatically to reflect the current position in the file. If you would like to freeze the automatic content scrolling, right-click anywhere in the **Bird's Eye View** to display the context menu. In the context menu, click the **Auto Scroll** command. To restore automatic scrolling, repeat the same operation.

 Tips:

- You may want to disable Auto Scroll in case you need to make a quick change in some part of the file and then return to the previous editing position. After making the change, click on the red rectangle in the **Bird's Eye View** and SQL Assistant will return the editor to the previous position.

- The **Bird's Eye View** freezing can be also used along with a small scaling factor as way to visually compare different parts of the code in the same file, having one part of the file displayed in the **Bird's Eye View** area and another part in a different place of the same file displayed in the editor area.

Resizing Content

To resize the **Bird's Eye View** pane, drag the vertical bar separating the pane and the code editor. Note that when you place mouse pointer over the right edge of the **Bird's Eye View** pane the cursor shape changes to resize shape as on the following screenshot.

Make sure the cursor takes the right shape before dragging the pane edge.

Persisting Bird's Eye View

By default, the **Bird's Eye View** pane is not persistent and is not displayed automatically when you open the editor window. You can manually open the **Bird's Eye View** pane by following the steps described in the Overview of Bird's Eye View topic.

To persist the **Bird's Eye View** so that it appears automatically in every code editor window, click the *pushpin* icon in the right-top corner of the **Bird's Eye View** pane.

To disable the **Bird's Eye View** persistence, click the *pushpin* icon again. Note that the persistence state is indicated by the toggled state of the *pushpin* icon.

CHAPTER 5, Code Formatter and Beautifier

Overview

In addition to the automatic, on-the-fly code indentation, wrapping, and keyword format-as-you-type features, SQL Assistant provides a full featured code formatting utility that can be used for formatting blocks of code, as well as for formatting all code in a file. If you highlight a portion of the code in the editor—for example, a single SQL statement or a group of SQL statements—and then invoke the SQL Assistant's Code Formatter utility, it will format only the highlighted code. If you invoke this utility with no code highlighted, the utility will format the entire body of code contained in the editor.

The Code Formatter supports multiple predefined formatting styles that can be customized and chosen as needed. It also allows you to create new, custom formatting styles.

Each formatting style is specific to a particular SQL dialect associated with the style. Formatting rules associated with the style can be used to control and customize general formatting behavior which will then be applied to all SQL code formatted using that style. Code formatting patterns can be defined for specific types of SQL statements and SQL syntax elements. Statement-level formatting patterns override formatting behavior specified in style-level general formatting rules. You can also define new patterns for new types of SQL statements and SQL elements. If formatting patterns are specified for nested elements, formatting specified for higher level elements overrides formatting to be applied to lower level elements. For example, you can specify separate formatting patterns for IF statements requiring multi-line text flow and text indents, as in the following example:

```
IF ...

    AND ...

    OR ...

BEGIN

    <stmtList>

END...

ELSE

    ...

BEGIN

    <stmtList>
END
```

And also specify a separate formatting pattern for BEGIN...END elements requiring single line text flow, as in the following example:

```
BEGIN   <stmtList>   END
```

In this case, when formatting standalone BEGIN…END blocks of code, the Code Formatter utility will use the second pattern. However, when formatting IF statements using IF…BEGIN…END syntax, it will use the first the first pattern, ignoring the pattern defined for BEGIN..END element.

Applying Formatting Styles to Code

To format a block of SQL code, for example, a single SQL statement or a group of SQL statements, highlight

the required block, and then invoke the SQL Assistant's Code Formatter utility, it will format only the highlighted code. To format the entire body of code contained in the editor, make sure no code is highlighted, and then invoke the SQL Assistant's Code Formatter utility

There are several ways to invoke the Code Formatter:

- Use the default Ctrl+F11 hot key or a custom hot key, if you changed the default key. This will apply the default formatting style.

- Use the default Ctrl+Shift+F11 hot key or a custom hot key, if you changed the default key. If you have multiple custom formatting styles saved in SQL Assistant's Options, you will be prompted to select the formatting style you want to apply.
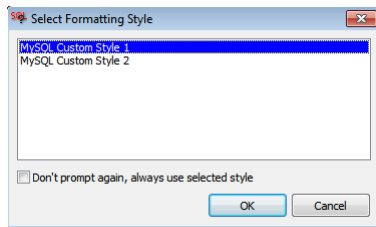
If you have only a single custom style, that style will be applied automatically without additional prompting to select it. If you have no custom styles saved in the Options, the default style will be used.

- Use SQL Assistant's system tray icon menu (see the Using System Tray Icon Menu topic for details)

- Use the target editor's context or top-level menus, if the menu integration option is enabled (see the Using Context and Top-level Menus topic for details).

Note that the Code Formatter formats the code in accordance with the current preferences in SQL Assistant's Options for the chosen formatting style. The currently selected formatting style is indicated by a checkmark in the SQL Assistant's menus displayed next to the formatting style name. To select a different style, click on that style name in the menu. The checkmark will appear next to the name and that style will be used for all code formatting in the same target editor.
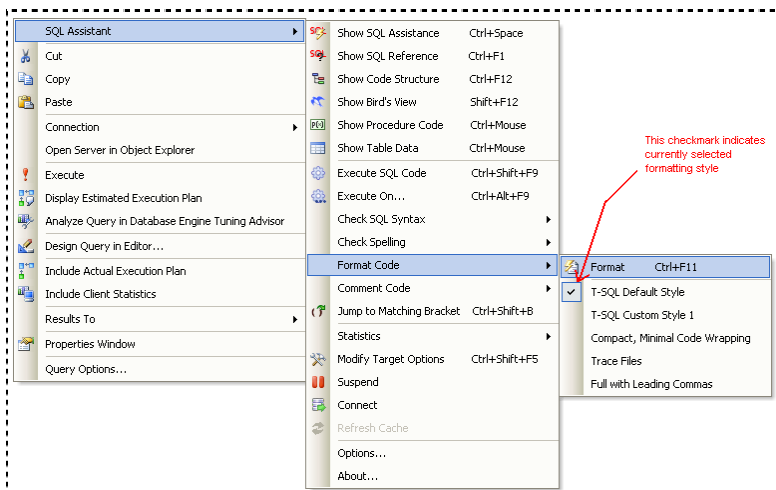
Tip: Only enabled formatting styles are displayed in the menus. You can enable and disable styles in SQL Assistant's Options.

See the following help topics for more information on code formatting rules and patterns and how to customize

them.

Important Notes:

When you make changes in the editor, only the text in the editor is changed. If that text is associated with a disk file, the file is not updated until you use your editor's **File | Save** feature.

If, for whatever reason, you want to undo the formatting, use your editor's **Edit | Undo** feature to undo the changes. If the entire text is reformatted, you may need to use the Undo feature twice, first to undo the insertion of the formatted code and again to retrospectively undo the editor reset command. This will restore text in the editor as it was before you invoked SQL Assistant's Code Formatter feature.

Tip: You can format off-line SQL files using SQL Assistant's command line interface. See the Command Line Interface topic in this chapter for more information.

Formatting Styles, Rules and Options

Formatting styles, rules and options can be accessed on the Code Format tab of the SQL Assistant's Options dialog. To modify an existing formatting style:

1. Double-click the SQL Assistant icon in the Windows system tray. The Options dialog will appear.

2. Click the **Code Format** tab.

3. In the style names list displayed in the top-left corner of the dialog, select the style you want to modify. Style specific configuration options will appear on the right side of the dialog.

Tips:

- Use the formatting style management icons available in the top-left corner of the Options dialog to create new styles or rename, duplicate or delete existing styles.

  Note that the icon functions are sensitive to the location of the focus in the Code Formatting tab. For example, if a formatting style is selected in the top-left window when you click the X button, the selected formatting style will be deleted entirely, including all associated formatting rules and patterns. However, if a pattern name is selected in the bottom-left window when you click the X button, the selected SQL pattern is deleted.

- The content of the window on the right side of the Code Formatting tab is also context sensitive. If an SQL pattern is selected, the pattern definition is displayed. If a formatting style is selected, properties of that formatting style are displayed.

- The default style and formatting patterns for each supported SQL dialect are protected and may not be deleted; however, you can still change their definitions as you see the fit.

- You can drag-and-drop formatting style names in the top-left window to rearrange their order.

General Options

The following general formatting options can be modified:

SQL Dialect – use this drop-down list to choose the type of SQL language associated with the selected formatting style.

Convert Keywords to – use thus the drop-down list to choose how you want SQL Assistant to auto-format keywords as you type. Choose the **Upper-case** function to automatically convert keywords to upper case. Choose the **Lower-case** function to automatically convert keywords to lower case. Choose the **InitCaps** function to automatically format keywords with the first character in upper case and all remaining characters in lower case. Choose the **Custom-case** option to have the keywords formatted exactly as they are entered in the keywords list.

Note that choosing the **None** option disables the automatic keyword formatting feature.

Auto-indent Mode – use this option to control whether SQL Assistant is allowed to automatically indent your code as you type it in the editor.

Spacing, Line Breaks, and Text-Wrapping Options

The following sections describe the SQL Assistant's Code Formatter and Beautifier features for controlling explicit text flow and wrapping:

Text Length for In-line Parentheses – this option controls the treatment of sub-queries and expressions within parentheses and allows keeping short expressions on a single line. In other words, if the text within parentheses is shorter than the specified maximum value, the entire text within brackets is kept on the same line. If the text is longer than the specified value, parentheses are moved to new lines and the text within parenthesis is reformatted using regular formatting rules.

The following example demonstrates how this parameter affects code wrapping. Note that wrapping could also be affected by the formatting patterns. This example uses the default settings.

```
SELECT id as customer_id, store_id, first_name, last_name,
       (SELECT max(id) FROM orders o WHERE o.cust_id = c.id) AS last_order_id,
       email, address_id, active,  create_date, last_update
FROM cust c;
```

If the maximum value for this parameter is set to 60 characters, you should get the following result after formatting:

```
SELECT   id as customer_id,
         store_id,
         first_name,
         last_name,
         (SELECT max(id) FROM orders o WHERE o.cust_id = c.id) AS last_order_id,
         email,
         address_id,
         active,
         create_date,
         last_update
FROM cust c;
```

Note that the entire sub-query appears on a single line. However, if the option is set to a shorter value such as 30 characters, the result would look different:

```
SELECT     id as customer_id,
           store_id,
           first_name,
           last_name,
           (
              SELECT max(id)
              FROM orders o
              WHERE o.cust_id = c.id
           ) AS last_order_id,
           email,
           address_id,
           active,
           create_date,
           last_update
FROM cust c;
```

Line Length for Code Wrapping – this option defines the maximum length, in characters, of code lines in your scripts. The default value is 60 characters. Use this option for controlling overall line wrapping for long lines of text. For example, consider the WHERE condition in the following screenshot:

In this example, the WHERE condition exceeds the default 60-character maximum line length by 11 characters. During formatting, the text in the WHERE clause will therefore be wrapped to two or more lines (depending on the type of expression) as shown in the screenshot below. Note that other formatting options, such as the Text length for in-line parenthesis option, could also affect code wrapping.

Note: To disable automatic line wrapping, set value of **Line Length for Code Wrapping** option to zero

In-line Parenthesis Spacing – use this option to set how "white" space is treated in your code around parenthesis. The values in the drop-list for this option are self-explanatory.

Align Data-types – use this option to control data-type alignment in type declarations for function and procedure parameters as well as for tables and views column definitions. See the following screenshot for details.
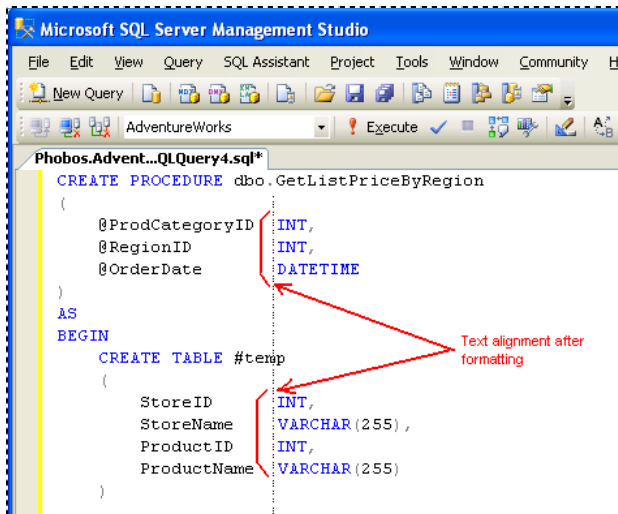
Align Assignments – use this option to control alignment of values in value assignment. The indenting is applied to values following the equal "=" sign, as in the following example
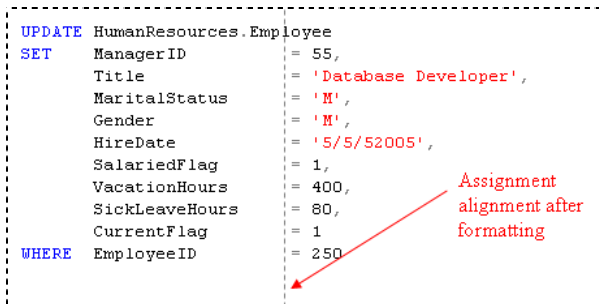
Separate SQL Statements – use this option to define how empty lines are treated between individual SQL statements. The names of values in the drop-list for this option are self-explanatory:

- Don't Modify

- Multi-line Statements Only

- All Statements

- All Statements (Except Consecutive)

 Notes:

> **Multiline statements** are SQL statements allocating more than one line of code.
> **Single-line statements** are statements allocating a single line.
> **All statements (Except consecutive)** means both multiline and single line statements, except consecutive single line statements as in the following example.

```
DECLARE @ProdCategoryID INT
DECLARE @RegionID INT
DECLARE @OrderDate DATETIME = NULL
```

Operators Spacing – use this option to define how white space is treated around arithmetic, binary and string concatenation operators. The values in the drop-list for this option are self-explanatory.

Commas and Logical Operators Formatting Options

Formatting options for commas and logical operations are used to control positioning of these syntax elements within the text of SQL queries. The values are self-explanatory and the effect of different options can be immediately seen in the examples displayed next to each option.

Important Notes: Changes made to positioning options for commas and logical operators impact both style-level formatting rules and statement-level formatting-patterns defined for individual SQL statements. This is by design and for your convenience.

For example, if you change logical operator positioning from STACKED to WRAPPED, SQL Assistant automatically updates the formatting of all individual statements affected by the style change, eliminating the need for you to update formatting patterns for individual statements. If necessary, you can override style-level formatting for individual SQL statements by modifying individual statements directly.

Keywords

The **Keywords** section allows you to customize keyword list for each formatting style. In this section you can also control keyword casing. This list is used in conjunction with the **Convert Keywords to** option. Only keywords listed in the **Keywords** section are processed by the Code Formatter. In case the **Convert Keywords to** option is set to **None,** no keywords are changed, In case it is set to **Custom-case** value, keywords are converted to match the case in which they are entered in the Keywords section.

The Keywords list is also used for keyword suggestions. See the Using Keywords Completion and Syntax Hints Features topic in CHAPTER 3 for more details. Note that some items in the Keywords list, such as SELECT TOP, appear as concatenated keywords. These items are used for keyword popups and should not be modified.

Statement-level Formatting Patterns

To a great extent, you can customize how code is formatted for specific types of SQL statements and SQL syntax elements. SQL Assistant uses the concept of formatting patterns, which consist of anchoring keywords and syntax elements and other text in between.

Use SQL Assistant Options to edit code formatting patterns:

1. On the Options screen activate the **Code Formatting** tab.

2. In the top-left style names list, choose the formatting style you want to modify.

3. In the bottom-left **Formatting Rules** list, click the type of SQL statement whose formatting pattern you want to modify. The screenshot below shows the formatting pattern for a T-SQL SELECT statement.
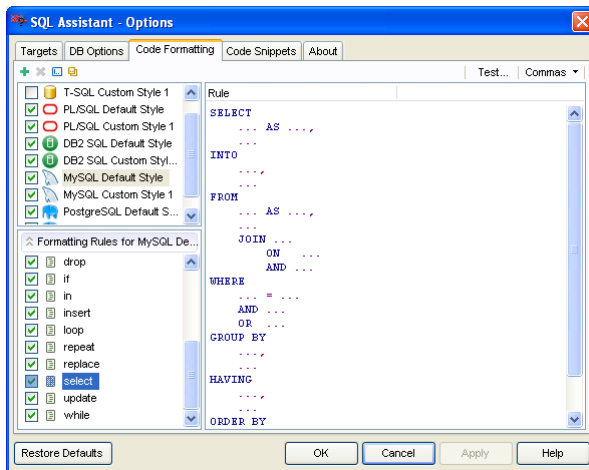
4. Edit the selected pattern as required. For more information see the Customizing Code Formatting Patterns topic in CHAPTER 48.

5. [Optional] You can use the Test Formatter function to preview the results of formatting. See the Testing Code Formatter Effective Settings topic for detailed instructions.

6. Click the OK button to save changes and close the Options dialog.

Special Formatting Rules

SQL Assistant supports a special formatting rule named **(header)** that is used to insert code-refactoring comments at the beginning of a reformatted SQL script, rather than to modify the SQL code within the script. This special rule supports several macro-variables for inserting dynamic information at the time of code formatting. This special rule can be customized just like any other code formatting rule. Follow the instructions in the previous topic for step-by-step instructions.
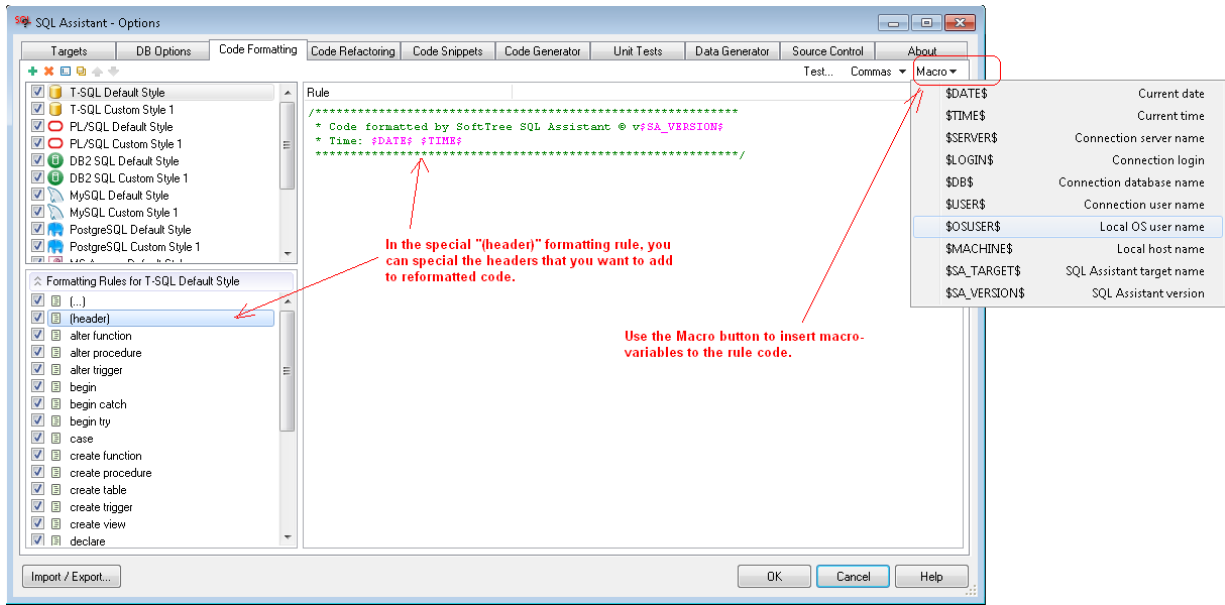
If you do not want to add code formatting headers to the reformatted SQL code, simply disable the **(header)** rule by clearing the checkbox in front of the rule name.

The following macro-variables are supported in the **(header)** rule.

Variable	Meaning
$DATE$	Current system date
$TIME$	Current system time
$LOGIN$	Login name for the current database connection
$USER$	Database user name for the current database connection
DB	Name of the current database (in the context of the current database connection)
$SERVER$	Name of the current database server (in the context of the current database connection)
$OSUSER$	Network name of the current user
$MACHINE$	Name of the computer where macro-variable is processed
SA_TARGET	SQL Assistant current target caption, for example, "SQL Server Enterprise Manager"
$SA_VERSION$	SQL Assistant version

Testing Code Formatter Effective Settings

SQL Assistant allows you to test current effective settings of selected formatting options before saving changes to an SQL script. You can use this feature to make changes in settings and immediately preview their effects.
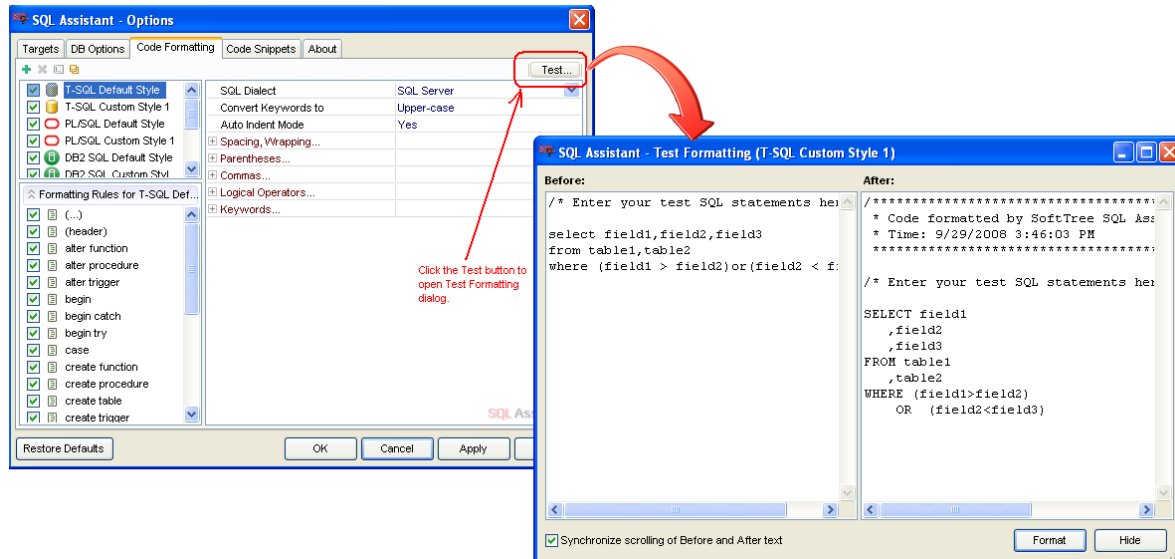
You can enter or paste test SQL queries into the **Before** edit box of the **Test Formatting** window and then press the **Format** button to see how they will be formatted. If the results are not what you want, you can adjust the formatting options and try again.

Tips:

- The **Test Formatting** dialog is non-modal; you don't need to close it in order to modify the options. If you want, you can position the Options and Test Formatting dialogs side by side so that you can modify both at the same time.

- The **Test Formatting** dialog is resizable. Drag the window resizer control in the top-bottom corner of the dialog to change the dialog size.

- Scroll bars appear in the Test Formatter's **Before** and **After** edit controls if the text length does not fit in the visible area.

- By default, SQL Assistant performs synchronizes scrolling of both edit controls so that you can see the original text and the results of the formatting side-by-side. However, in some cases the length and layout of the formatted text can be very different from the original test text such that synchronous scrolling might be inappropriate. To disable synchronous scrolling, uncheck the checkbox in the left-down corner of the Test Formatting dialog.

Commenting and Uncommenting Code Blocks

The SQL Assistant code formatter provides two functions for formatting code comment blocks.

To quickly format a block of code as a code comment:

1. Using your mouse or keyboard, select the block of code you want to comment out.

2. Right-click anywhere in the editor workspace. From the right-click menu, select **SQL Assistant,** then select the **Comment Code** submenu. From the **Comment Code** submenu, click one of the following comment formatting commands:

    a. **Comment with – –**

    b. **Comment with /* */**

To uncomment a currently commented block of code:

1. Using the mouse or keyboard, select the block of code you want to uncomment.

2. Right-click in the editor workspace. From the right-click menu, select **SQL Assistant,** then select the **Comment Code** submenu. From the **Comment Code** submenu, click the **Uncomment** command.

Tips:

*   If the commenting functions are activated when no text is highlighted in the editor, the entire script is commented out. Similarly if the uncommenting functions are activated when no text is highlighted in the editor, the entire script is parsed and comments are removed from all lines with comments.

*   Note that the uncommenting functions only remove leading comments that appear at the beginning of a line of text. They do not touch comments appearing in the middle and at the end of line text.

*   By default, during commenting or uncommenting blocks of text, SQL Assistant copies the highlighted text to an internal memory buffer, transforms it as required, and then replaces the selected text in a single pass. If you invoke the editor's "undo" function after that operation, the previous block of text is restored as it was before the commenting action was performed. However, some code editors do not allow SQL Assistant to perform operations on contiguous blocks of text. Certain editors only allow line-by-line changes. In this case, SQL Assistant is forced to replace each line in the selected block of text separately. As a result, using the 'undo' function restores only the last line. To undo the entire operation, you must invoke the "undo" function multiple times to restore the entire block.

*   To customize code commenting method compatible with your editor, use **Commenting Method** option available in the target options in SQL Assistant's **Options** dialog. See the **Advanced** section in the target options.

Formatting SQLCMD Scripts

When working with SQL Server 2008 and SQL Server 2012 based editors, the SQL Assistant code formatter automatically recognizes SQLCMD scripts and skips lines containing non-SQL commands, such as lines beginning with double-exclamation point characters.

Command Line Interface

To format off-line SQL files from command line console, use the following command:

```
sacmd fmt:"path-to-sql-file " sas:"path-to-sa-settings-file" fpref:"format-style-name"
```

In the above command, replace **path-to-sql-file** with the full file name of the SQL file to format; replace **path-to-sa-settings-file** with the full file name of the SQL Assistant settings file containing the required database connection parameters; and replace **format-style-name** with the required formatting style name.

Important Notes: The SQL Assistant settings file location is version and user profile specific. See the notes in the Overview topic in CHAPTER 51 for details on how to find out the location of that file.

Example:

```
cd "C:\Program Files (x86)\SQL Assistant 12"
sacmd fmt:"C:\Projects\App Code\procedure1.sql" sas:"%APPDATA%\SQL Assistant\13.0\sqlassist.sas" fpref:"MySQL
Custom Style 1"
```

Using DOS Batch Processing to Format Multiple SQL Files

The following example demonstrates how to use standard DOS commands in a batch file to recursively invoke SQL Assistant's command line interface and reformat multiple SQL Files:

1. Save the following text as **format_sql_files.bat** file in any folder on your system:

   ```
   @ECHO OFF

   SET curr_dir=%CD%
   CD "C:\Program Files (x86)\SQL Assistant 12"

   FOR %%f IN (%1\*.sql) DO (
       ECHO  Processing file %%~ff
       SACMD fmt:"%%~ff" sas:"%APPDATA%\SQL Assistant\13.0\sqlassist.sas" fpref:%2 > NUL
   )

   CD "%curr_dir%"
   ECHO  Done
   ```

2. Copy all SQL files you want to reformat to the folder **C:\SQL Scripts**.

3. Open command line console, navigate to the folder where you saved **format_sql_files.bat** file, and execute the following command:

   ```
   format_sql_files.bat "C:\SQL Scripts" "T-SQL Default Style"
   ```

Note: The preceding script example has been tested on a Windows XP system. Changes might be required before running it on other versions of Windows. Before attempting to run this script on other versions of Windows, please verify and, if necessary, correct the SQL Assistant installation path.

CHAPTER 6, Database Explorer

Overview

The **Database Explorer** pane is a graphical interface to your database with a hierarchical tree representation of your database schema objects, tablespaces, files, full-text catalogs, security principles, assemblies and extensions, and other features. In the hierarchical tree the information is groupepd into folders. The Database Explorer makes navigating and managing databases easy, enabling you to change database and schema object properties, create new objects, monitor activities and performance, perform database backups and restores, execute various database and schema scope reports, update statistics, and utilize many other functions.

The databases and their schema objects are displayed for the currently active database connection. Each target editor and editor tab have their own instance of the Database Explorer docked, making it easy to work with many different databases simultaneously. The Database Explorer also enables you to use drag-and-drop interface for quickly building your code from existing database schema objects, columns, parameters and other items.

The Database Explorer pane can be moved around and docked to the left or right side of the parent editor, or it can be undocked and moved to a separate monitor
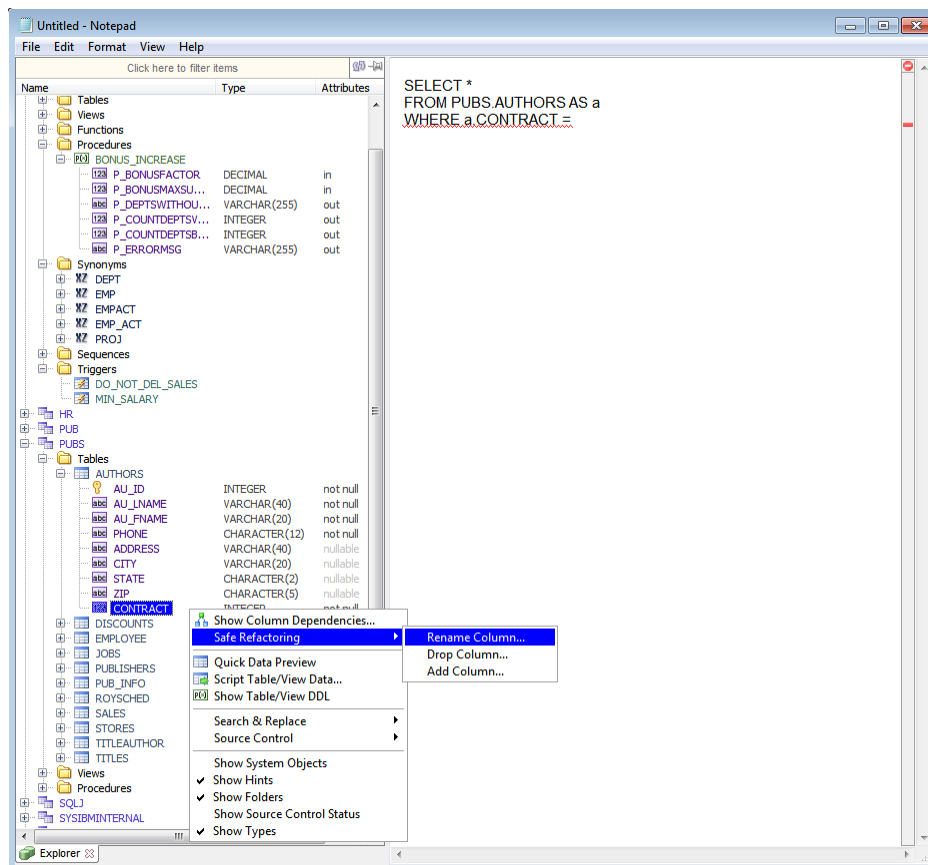
To open the Database Explorer pane, press the Ctrl+W hot key. The **Database Explorer** pane e also opened using SQL Assistant's menu in the system tray or from menus available in the target editor. To use the target

editor's menus, the menu integration option must be enabled. For details, see the Manually Invoking SQL Assistant Popups topic in CHAPTER 3.

To make the **Database Explorer** sticky and open automatically in every editor instance, click the *pushpin* icon.

To expand folders, click the plus sign (+) or double-click the folder. Expand folders to show more detailed information. Right-click folders or objects to show their contect menu for performing common tasks.

Persisting Database Explorer Pane

By default, the **Database Explorer** pane is not persistent and does not display automatically in target editor's code windows. You can manually open the it following the steps described in the About topic.

If you want the Database Explorer appear automatically in every code editor window, click the *pushpin* icon in the right-top corner of the Database Explorer window. This will make the Database Explorer pane persistent in the current target editor and all future instances of the same editor type and its code windows.

To disable the Database Explorer persistence, click the *pushpin* icon again. Note that the persistence state is indicated by the toggled state of the *pushpin* icon.

Content Filtering and Sorting

The Database Explorer offers super-fast content filtering. Type the substring you want to use as a filter for database objects into the filter box available above the database-tree.

To control which objects appear in the database tree, you can either use the right-click context menu or the SQL Assistant's Options dialog. For example, to hide or show system schemas and objects in the Database Explorer, you can right-click any database and choose **Show System Objects** menu.

The Database Explorer content filleting supports the same item name matching ordering methods that you use in SQL Intellisense popups.
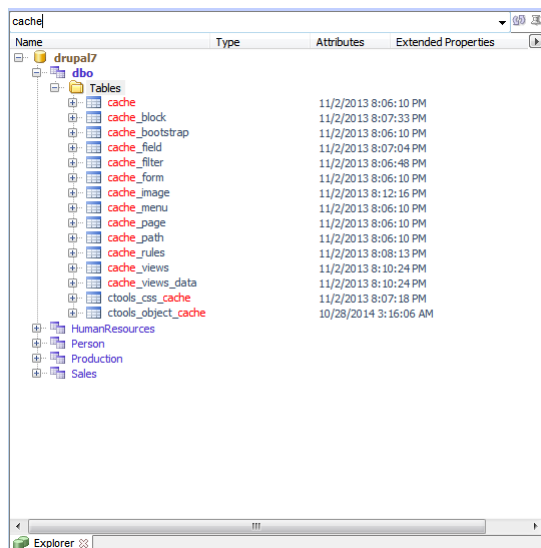
You can customize Database Explorer content filtering and sorting method in the Options dialog:

1. Open the Options dialog

2. Select DB Options tab.

3. Select SQL Assistance type for your database server on the left side of the dialog.

4. Expand Database Explorer option group on the right side of the dialog.

5. Select desired Item Name Matching Method.

6. Click Ok to save changes and close the dialog.

Item Name Matching Methods - Specifies the method that governs the way Database Explorer name matching responds to names you type in the filter box. Note that text matching is **case insensitive**. The methods are:

- **Name Starts with Key String** – the name must begin with the text you typed into the filter. For example, if you typed "Order", names like "OrderHeader", "OrderDetail" would be shown in the Database Explorer objects tree.

- **Name Contains Key String, Order Alphabetically** – the name must contain the text you typed into the filter. The text string could be anywhere within the name. For example, if you typed "Order", names like "OrderHeader", "OrderDetail", "fnOrderData", "prDeleteOrder" would be shown in the Database Explorer objects tree. The matching names are filtered and then sorted alphabetically. In the example here, the resulting order is going to be "fnOrderData", "prDeleteOrder", "OrderDetail", "OrderHeader".

- **Name Contains Key String, Order by Best Match** – the name must contain the text you typed into the filter. The text string could be anywhere within the name. For example, if you typed "Order", names like "OrderHeader", "OrderDetail", "fnOrderData", "prDeleteOrder" would shown in the Database Explorer objects tree. The matching names are filtered and then sorted in order of the best match. In the example here, the resulting order is going to be "OrderDetail", "OrderHeader", "fnOrderData", "prDeleteOrder".

- **Name Contains Characters from Key String, Order Alphabetically** –the name must contain the characters from the text you typed into the filter. The characters appear in the same order but do not need to be sequential. For example, if you typed "Ordr", names like "OrderHeader", "OrderDetail", "fnOrderData", "prDeleteOrder" , "vwOrdrWklyReport", as well as "vwOrdYearlyReport" will be shown in the Database Explorer objects tree. The matching names are filtered and then sorted alphabetically.

- **Name Contains Characters from Key String, by Best Match** –the name must contain the characters from the text you typed into the filter. The characters appear in the same order but do not need to be sequential. For example, if you typed "Ordr", names like "OrderHeader", "OrderDetail", "fnOrderData", "prDeleteOrder" , "vwOrdrWklyReport", as well as "vwOrdYearlyReport" will be shown in the Database Explorer objects tree. The matching names are filtered and then sorted in order of the best match.

Using Context Menus

Right-click context menus in the Database Explorer provide quick access to frequent commands. The contents of the right-click menu vary for different types of database schema objects, for databases, for table and view columns and for other types of database items. The right-click context menus in the Database Explorer are also context driven. For example, if you want to create a new schema object, in the Database Explorer you can navigate to the node for the schema in which the object is to be created, right-click that schema node, and then select **New…** in the right-click menu. A prompt will appear asking you to choose type of the schema object you want to create. You can choose a Table, or a Security Policy, or a Function, or any other type of the schema object. However, if you expand the schema folder and select Tables node, and after that you select **New…** in the right-click menu, there will be no prompt shown for the object type, as the only applicable choice is a new

table. It works similar for all other hierarchical levels. To create a new schema, right click the database in which the schema is to be created. To create a new database or new user, right click the server level node, and so on. The description of the actual commands and their usage are described in detail in other topics of this User's Guide.

Using Drag-and-Drop

Drag-and-drop provides an innovative method for fast code entry. Drag items from the Database Explorer and drop them into the code editor at a position where you want to insert the dragged item's name. What is actually inserted depends on the insertion point context. For example, If you drop table name after FROM keyword where another table is already referenced, SQL Assistant will replace the referenced table name with the dragged table name. If you drop the table name in a middle of string value, SQL Assistant will expand the string inserting table name where specified.

Managing Database and Schema Objects

To review or modify properties of an existing object, locate that object in the tree, right click it and select the **Properties…** menu. For procedural objects and view you can also work with their code in a full featured editor. To open an editor for the selected object, choose the **Edit** menu.

The **New…** menu will open an object creation dialog specific to the type of the selected object and type of the database you are connected to. The **Properties…** dialog menu will open an object properties dialog specific to the type of the selected object and type of the database you are connected to which you can use to modify the selected object, add or remove objects features and sub-objects. For example, a Table Properties dialogs can be used to modify, add and drop table columns, modify, create and drop table indexes, triggers, constraints, rules, permissions, tags, manage row-level security policy, and so on…

Hundreds of different schema object management dialogs are available to help you with the database schema management for most of the database systems supported by SQL Assistant. The schema object management dialogs display mouse-over hints for all their fields and features providing in-line help and suggestions, which are database type and database version specific. Please use the mouse over hints and also consult with your database documentation for help with the database and schema management.
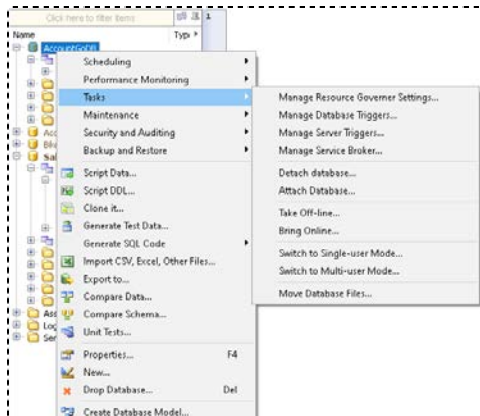
The Database Explorer supports many management commands. The menus and available commands are database type and database version specific. Only database type specific commands are shown in the menus.

Commands that in the same category or handle related tasks are grouped in sub-menus. The sample image on the left side of this page demonstrates "Database" and "Server" specific management tasks for a SQL Server database. There are also separate sub-menus for database maintenance, for backup and restore operations, for performance monitoring, and so on…

Source Code Control Integration
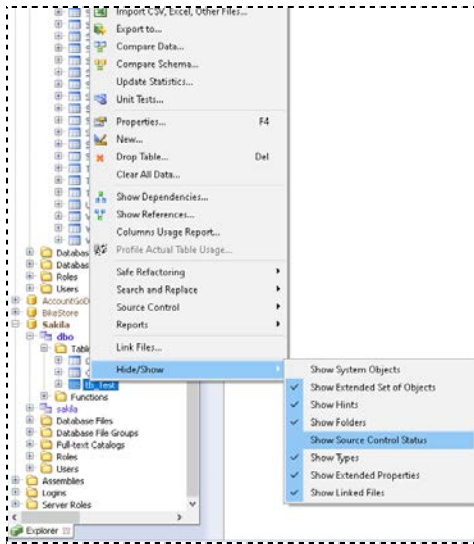
The Database Explorer can be directly linked to the source control Interface and show statuses of schema objects in the Source Control System. To enable that feature, select **Hide/Show -> Show Source Control Status** command in the right-click context menu for the database node or any schema object. See CHAPTER 23, Database Source Code Control Interface for instructions on how to configure and use the Source Code Control

Database and Schema Scope Aggregated Statistics

The mouse-over hints for databases and schemas shown in DB Explorer provide additional utilities and reports that simplify database management. They can be used to quickly count number of schema objects by type in a schema or entire database, or display row counts for all tables.

Tips:

- The row counts are approximate. They are obtained from statistics stored in the system catalog tables.

- To update schema and database stats using **Update Statistics…** command which is available in DB Explorer's right-click menu.

- You can also use right-click menu as demonstrated on the following screenshot to export and analyze the results further.
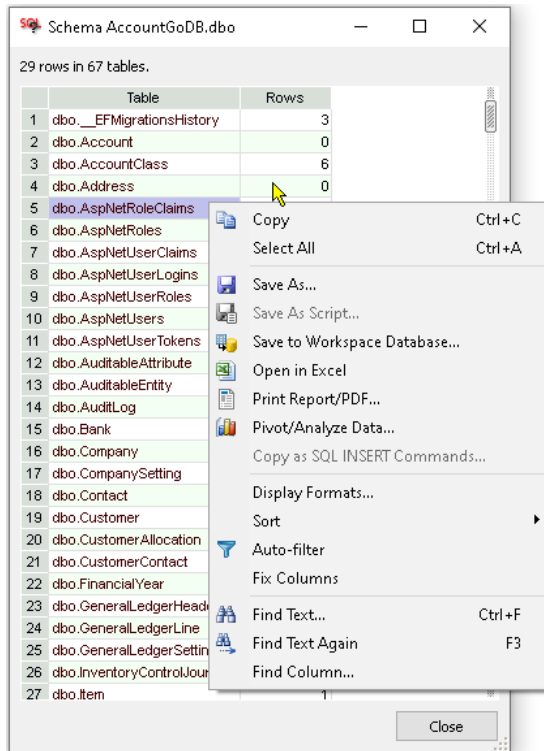
- SQL Assistant provides a large number of reports that can be also used to show aggregate database and schema scope statistics. The reports are accessible using several different methods including right-click menu in DB Explorer and then selecting **Reports** menu. Among available reports are reports showing biggest tables in the database by row counts, by space usage; tables without indexes, and so on…

Updating Table and Index Statistics

To maintain good database performance, you should update table and index statistics after the content of tables changes significantly; for example, after deleting or adding a large amount of data.

It's often that the statistics are refreshed infrequently using scheduled statistics collection jobs, or automatic statistics collection options get disabled for certain reasons, and so the statistics do not get refreshed automatically. There could be many reasons for why you may want to update statistics in demand in order to resolve an immediate performance issue, or perhaps update statics in a different manner using non default options.

You can use schema scope Update Statics menus in the Database Explorer to update table and index statistics for all tables in the selected schema. This command will open **Update Statistics** dialog with database type specific methods and options. For example, options are available for shrinking free space and eliminating space fragmentation from tables and indexes, and then collecting fresh statistics, for collecting statistics using data sampling, which is a good option for very large tables, or using a full table scan.

Refactoring Existing Schema Objects

The **Safe Refactoring** menus in the Database Explorer provide commands to safely modify, rename, and drop schema objects in your database. The Safe Refactoring commands are also available to rename, drop, and add columns in database tables and views, to rename, drop, and add parameters in database procedures and functions.

See CHAPTER 8, Smart Database Refactoring for more information and description of how the safe database refactoring is implemented and how you can use it and customize too.

Using Table Information Reports

SQL Assistant provides a number of table information reports for analyzing table and index space allocation, table fragmentation, and other table metrics. You can access them using right-click menu in the Database Explorer and then selecting the **Object Info** command, or alternatively use the mouse-over event to display table info balloon and in the balloon click the **Info** hyperlink in the top-right corner. This will show the Table Info pane, and the reports will be accessible as hyperlinks on the right side of the pane as demonstrated on the following screenshot.
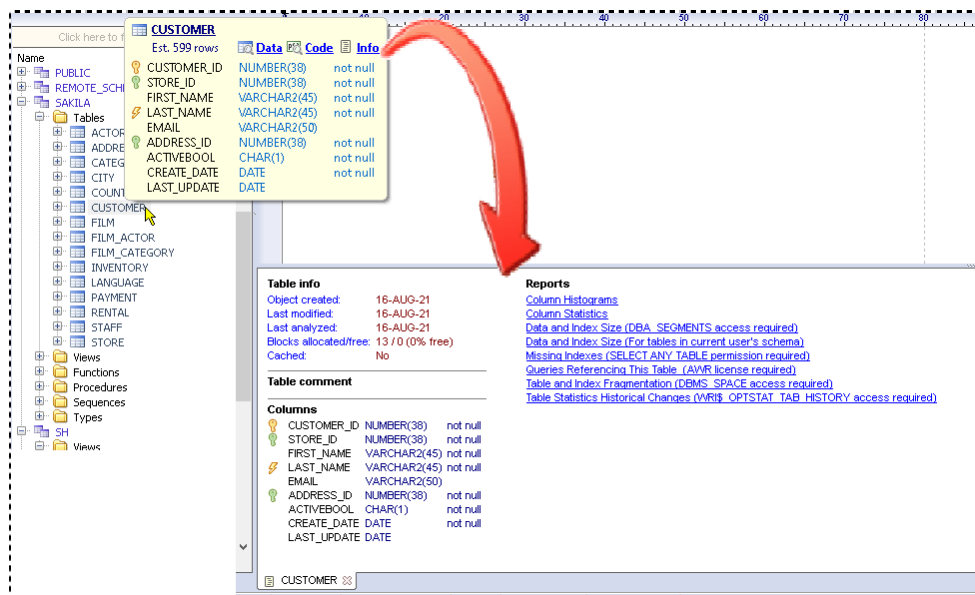
Tip: You can add custom reports to the Info page. See CHAPTER 24, Reporting, Data Pivot and Analytics for more information

Performing Database and Schema Scope Backup and Restore

For many types of databases SQL Assistant provides graphical interfaces for backup and restore operations.

The graphical backup and restore dialogs can be accessed using several different methods. The simplest though is right-click context menus for database and schema nodes in the Database Explorer. You can then use commands available in the Backup and Restore menus as pictured below. The available backup and restore commands, methods, and options are database type and database version specific.
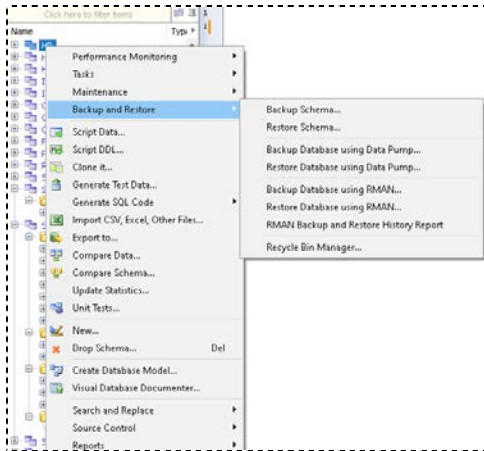
CHAPTER 7, Code Entry Automation using Code Snippets

Overview

SQL Assistant's Code Snippets feature enables you to quickly insert static, ready-made snippets of code into your SQL scripts, as well as dynamically generated SQL code. For example, Code Snippets can be used to insert small chunks of code such as BEGIN … END, and also generate many pages of code and even entire stored procedures implementing complex business logic. Macro variables are used to program code snippets when dynamic code generation is required.

Tip. Code snippets provide the most efficient way to enter SQL code. A number of ready to use code snippets are installed with SQL Assistant. We encourage you to define your own snippets for code structures, and queries you use most often. Use the **Code Snippets** tab in the SQL Assistant options to create new snippets and manage existing ones.

To generate SQL code using code snippet and insert the resulting code into your SQL code editor, in the editor type the name of the snippet you want to use and then press Ctrl+Enter hot key. If the snippet is configured for a non-default hot key, press that hot key instead of Ctrl+Enter. Alternatively, you can use SQL Assistant's context menu to select a snippet. Right-click in the editor to bring up the context menu, select **SQL Assistant → Code Snippets →[here you will find a list of available snippets to choose from]**. Choose name of the snippet you want to use.

SQL Assistant comes with a number of ready to use code snippets. For example, if you use a PL/SQL editor and you type letters "for" without quotes and then press Ctrl+Enter default hotkey, the following text will be inserted at the current caret position:

```
FOR i IN 0..| LOOP

END LOOP;
```

Use the SQL Assistant Options dialog to customize pre-configured snippets and to create new snippets.

1. On the Options screen, activate **Code Snippets** tab.
2. Choose SQL dialect for which you want to configure code snippets.
3. Select a snippet as shown in the following screenshot.
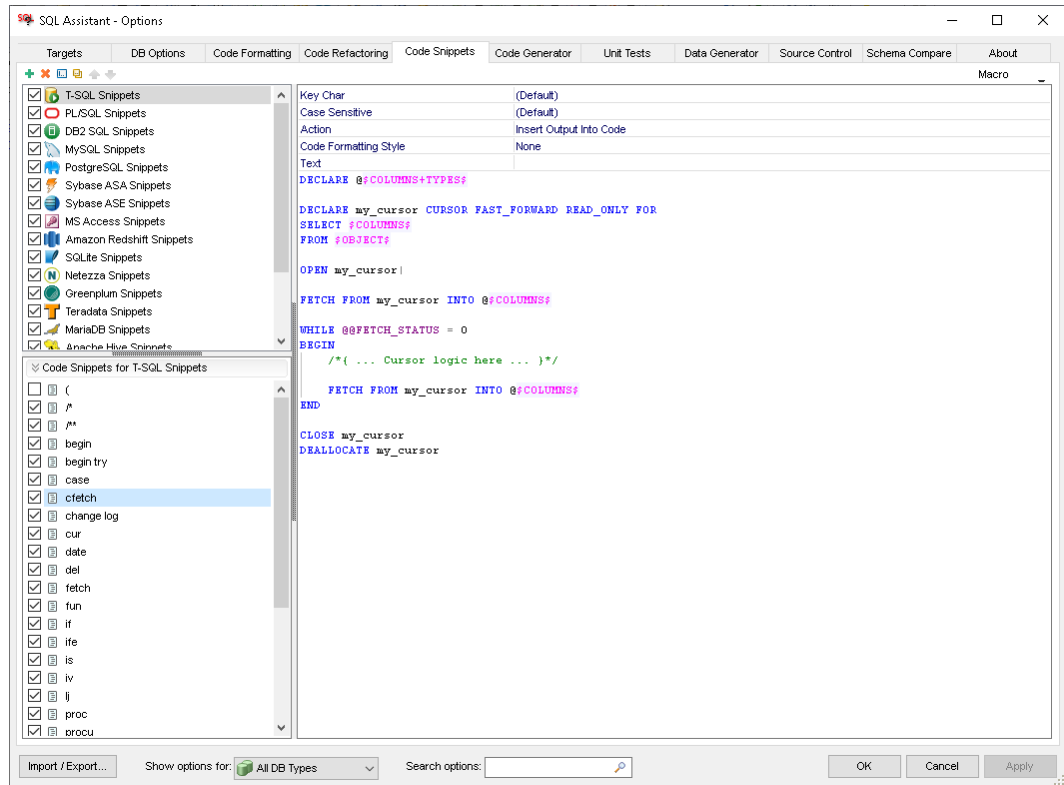
4. Edit the snippet code as required.

To disable a snippet, you can uncheck the box next to the snippet name. If a snippet is disabled, its definition remains in the SQL Assistant options but the snippet is not active and cannot be used. Disabled snippets also do not appear in SQL Assistant's context menus

To delete a snippet, click on the snippet name and then press the Del key on the keyboard.

For more information on how to use and change SQL Assistant options, see CHAPTER 48.

Tip: The vertical bar **|** in the snippet code indicates where SQL Assistant will place the edit caret after snippet code is inserted into the editor. Note that this predefined caret position does not work in buffered text editors like SQL*Plus where the caret is always placed at the end of the last line.

To create new code snippets and manage existing ones, use SQL Assistant's Options dialog.

You can assign different hot keys to different code snippets, or use the same hotkey. The choice of hotkeys depends on the snippet purpose, snippet execution mode and on your personal preferences. For more information, see the Customizing Existing and Creating New Code Snippets topic in CHAPTER 48.

Tip: Code snippets can be also accessed via SQL Assistant menus. For example, if right-click menu integration option is enabled, you can right-click in the target editor, choose the SQL Assistant submenu, then choose the Code snippets submenu. From the code snippets submenu, choose the snippet you want and the SQL Assistant will insert the snippet code into the editor.

Note that the Code Snippets submenu displays only the initial characters of the Code Snippet code along with the snippet name and hot key. The snippet hot key is displayed in brackets. For example, **case+(Ctrl Enter)**. If the hot key is a composite key, all keys must be pressed at the same time. In the Code Snippets submenu menu, individual keys are separated by the pipe "|" symbol.
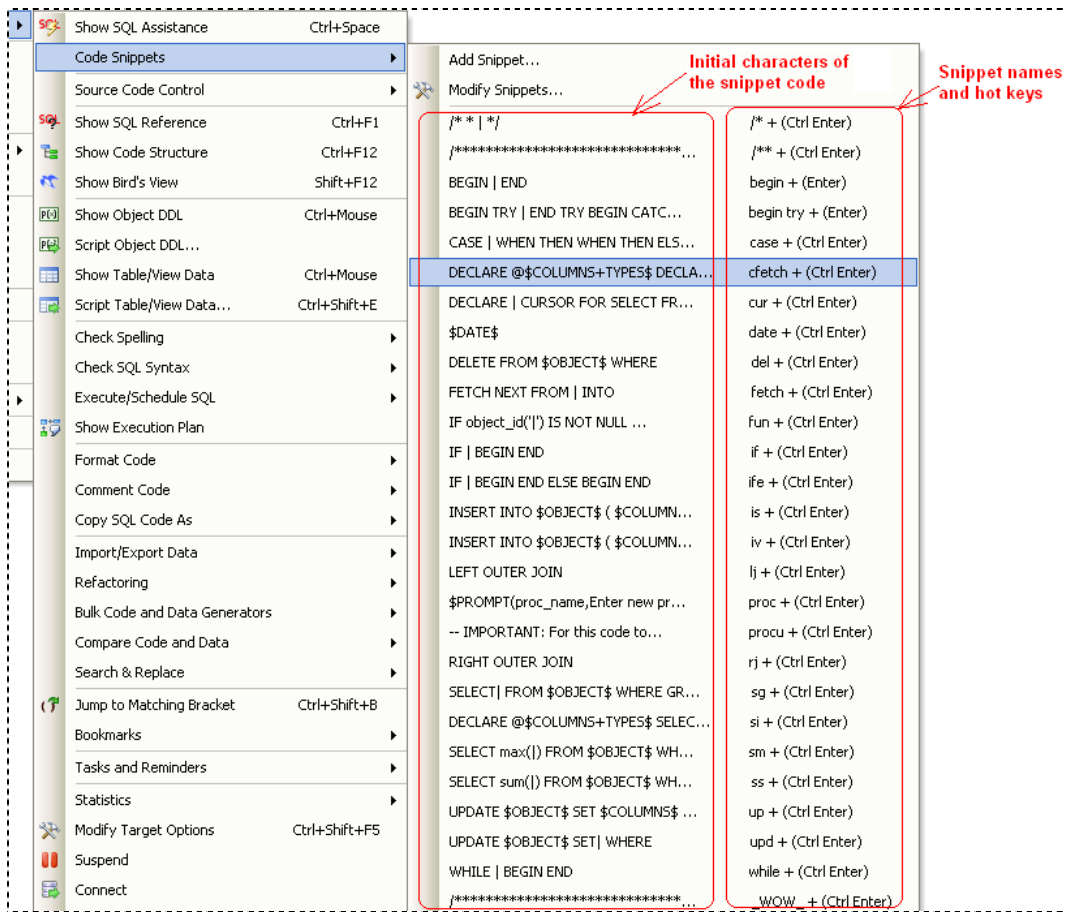
Additional Tips:

- Use the code snippet management icons available in the top-left corner of the Options dialog to create new snippets or to rename, duplicate or delete existing snippets.

  Note that the icon functions are sensitive to the location of the focus in the Code Snippets tab. For example, if a snippet interface name is selected in the top-left when you click the X button, the selected interface name is deleted entirely, including all associated code snippets. However, if a snippet name is selected in the lower left window, clicking the same button will delete the selected snippet.

- The content of the right side of the Code Snippets tab is also context sensitive. If a snippet interface is selected in the top right window, the interface definition is displayed in the right window. If a snippet name is selected in the lower left window, the right window displays properties of that snippet and its code.

- Drag-and-drop snippet interface names in the left-top list to rearrange their order. You can use this method to push most commonly used interfaces to the top of the list and minimize the amount of scrolling and clicking required for customizing code snippets.

- Code snippets can contain macro-variables whose values are dynamically replaced when the snippet code is inserted into the code editor. To insert a macro-variable into the snippet code, you can type its name, including $ sign delimiters, into the code window on the right. Alternatively, you can use the **Macro** button available in the top-right corner of the tab to select a macro-variable from a menu containing a list of available macros, as shown in the illustration below. This method also provides interactive assistance for macro-variables supporting multiple options such as, for example, the $OBJECT(…)$ macro variable.

  See the "Macro-variables and Dynamic Code Generation" section below for a listing of available macro-variables.
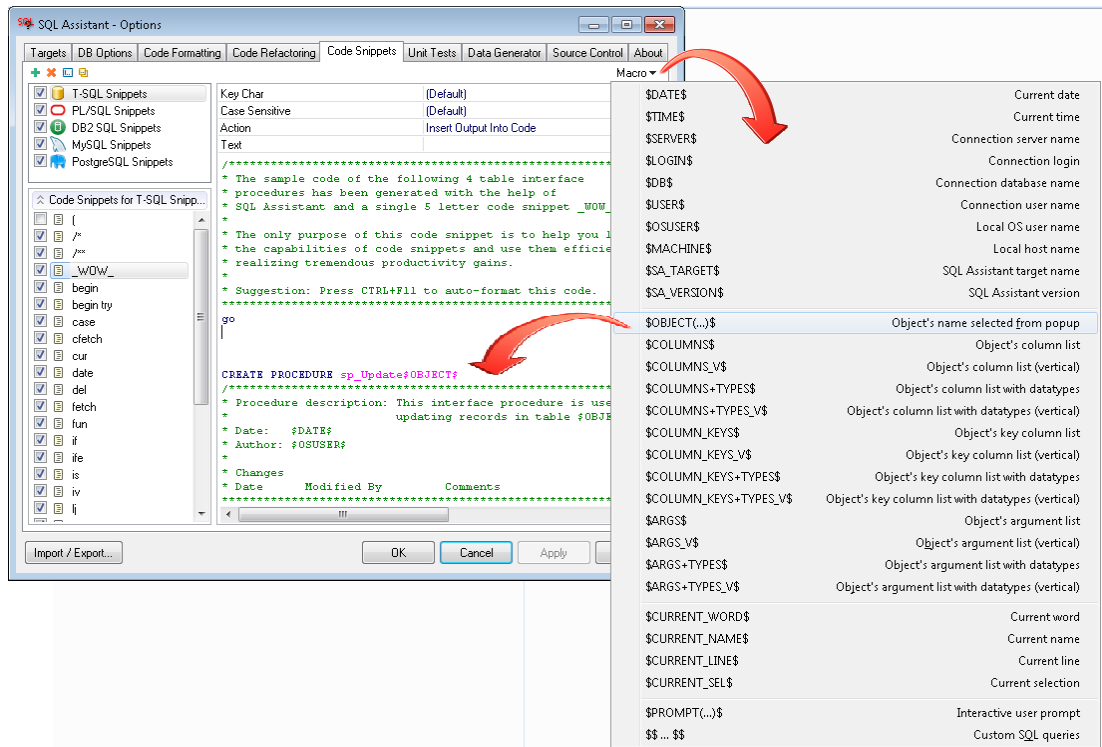
Auto-formatting Generated Code

By default the code generated by code snippets is inserted into the editor as it is entered in the snippet and generated by the database. You can have the generated code automatically formatted before it is inserted. Use the **Code Formatting Style** option and change the default **None** value to one of the formatting styles compatible with the selected SQL Assistance type.
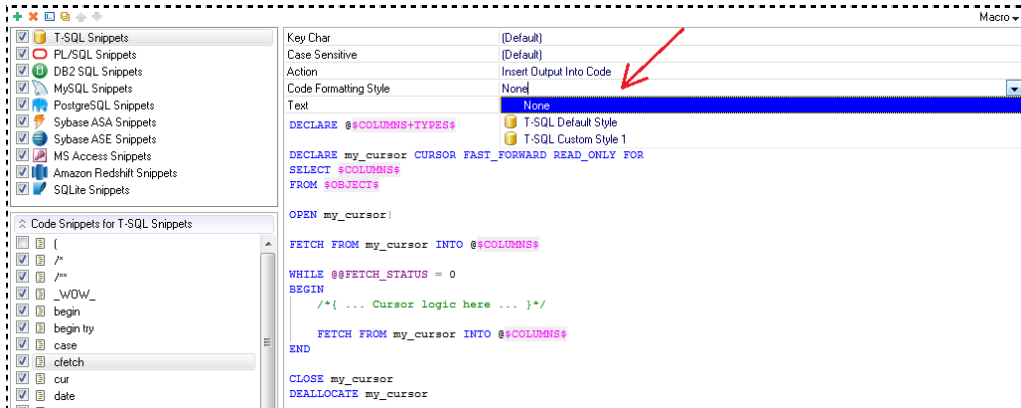
Macro-variables and Dynamic Code Generation

Code snippets can contain certain macro-variables whose values are dynamically replaced when the snippet code is inserted into the code editor. There are two types of macro-variables: passive and active.

Using Passive Macro-Variables

Passive variables produce easy to guess results and do not require additional user input. The following passive macro-variables are supported in all snippets:

Variable	Meaning
$DATE$	Current system date
$TIME$	Current system time
$LOGIN$	Login name for the current database connection
$USER$	Database user name for the current database connection
DB	Name of the current database (in the context of the current database connection)
$SERVER$	Name of the current database server (in the context of the current database connection)
$OSUSER$	Network name of the current user

$MACHINE$	Name of the computer where macro-variable is processed
SA_TARGET	SQL Assistant current target caption, for example, "SQL Server Enterprise Manager"
$SA_VERSION$	SQL Assistant version

For example, if you have a code snippet named "fun" having the following text:

```
CREATE OR REPLACE FUNCTION |
(
    v_in IN <data type>
)
RETURN <data type>

/***********************************************************
* Function description:
* Date:   $DATE$
* Author: $OSUSER$ connected as $LOGIN$
*
* Changes
* Date     Modified By       Comments
***********************************************************
*
**********************************************************/
IS
DECLARE
     v_ret <data type>;
BEGIN
    v_ret := ...;

    RETURN v_ret;
END;
```

If you type word "fun" without quotes in the SQL editor and press Ctrl+Enter (or whatever you have selected as a hotkey for that code snippet), the text of the code snippet will be inserted at the current caret position in the editor and the $DATE$ macro-variable will be automatically substituted with the current system date. Similarly, the $OSUSER$ macro-variable name will be replaced by your Windows login name, for example, *MyDomain\MyName,* and the *$LOGIN$* macro-variable will be substituted with your database login name, for example, *SomeLoginName.* The caret will be then placed at the point marked with the pipe | symbol so you can type your function name immediately after the snippet code is inserted into the editor.

Tip: In Oracle, MySQL, and DB2 targets, $LOGIN$ and $USER$ variables always generate the same value.

Using Active Macro-Variables

In comparison to passive macro-variables, an active macro-variable referenced in snippet code will cause SQL Assistant to display a prompt that lists additional options. Based on your selection from the prompt, the macro-variable will be replaced with dynamically generated code appropriate for your input selection. The following active macro-variables are supported in all snippets:

Variable	Meaning
$OBJECT(…)$	This will be replaced with the name of the selected object or object's sub-item in case an object level is expanded in the popup and a sub-

	item is selected. See the separate $OBJECT(…)$ macro topic for more information.
$OBJECT$	This will be replaced with the name of the selected object. $OBJECT$ is a shortcut version of the more advanced $OBJECT(…)$ macro-variable. The result is the same as choosing $OBJECT(…)$ macro-variable in the menu and then choosing **Insert Object Name** option with **All Objects** filtering option. Note: The $OBJECT$ macro is maintained for backward compatibility. When developing new code snippets, please use the new $OBJECT(…)$ macro.
$COLUMNS(…)$	This will be replaced with a list of columns of the selected object or a list of columns and their data-types. Column names and data-types, can be inserted on a single line, multiple lines, or a vertical list as specified in the macro parameters. This macro-variable can be used with tables, views, and table functions only. See the separate $COLUMNS(…)$ macro topic for more information.
$COLUMNS$	This will be replaced with a comma-separated list of columns of the selected object. If there are more columns than can fit on a single line, additional lines with columns will be added as needed. This macro-variable can be used with tables, views, and table functions only. Note: The $COLUMNS$ macro is maintained for backward compatibility. When developing new code snippets, please use the new $COLUMNS (…)$ macro.
$COLUMNS_V$	This is virtually the same as the $COLUMNS$ macro, except that each column will be inserted on a separate line. The inserted text will appear as vertical comma-separated list of column names. Positions of commas and elements before and after each inserted column name are controlled by pre and post macro-text elements. See the Tips section below for more details. Note: The $COLUMNS_V$ macro is maintained for backward compatibility. When developing new code snippets, please use the new $COLUMNS (…)$ macro.
$COLUMNS+TYPES$	This will be replaced with a comma-separated list of columns of the selected object and their data types. This macro-variable can be used with tables, views, and table functions only. Note: The $COLUMNS+TYPES$ macro is maintained for backward compatibility. When developing new code snippets, please use the new $COLUMNS (…)$ macro.
$COLUMNS+TYPES_V$	This is virtually the same as $COLUMNS+TYPES$ macro, except that each column/type pair will be inserted on a separate line. The inserted text will appear as a vertical comma-separated list of column name/type pairs. Positions of commas and elements before and after each inserted column name are controlled by pre and post macro-text elements. See the Tips section below for more details. Note: The $COLUMNS+TYPES_V$ macro is maintained for backward compatibility. When developing new code snippets, please use the new $COLUMNS (…)$ macro.
$COLUMN_KEYS$	This will be replaced with a comma-separated list of primary key columns of the selected object. If there are more columns than can fit

	on a single line, additional lines with columns will be added as needed. This macro-variable can be used only with tables having primary keys. Note: The $COLUMN_KEYS$ macro is maintained for backward compatibility. When developing new code snippets, please use the new $COLUMNS (…)$ macro.
$COLUMN_KEYS _V$	This is virtually the same as the $COLUMN_KEYS$ macro, except that each column will be inserted on a separate line. The inserted text will appear as a vertical comma-separated list of column names. Positions of commas and elements before and after each inserted column name are controlled by pre and post macro-text elements. See Tips section below for more details. Note: The $COLUMN_KEYS _V$ macro is maintained for backward compatibility. When developing new code snippets, please use the new $COLUMNS (…)$ macro.
$COLUMN_KEYS+TYPES$	This will be replaced with comma-separated list of primary key columns of the selected object and their data types. This macro-variable can be used only with tables having primary keys. Note: The $COLUMN_KEYS+TYPES$ macro is maintained for backward compatibility. When developing new code snippets, please use the new $COLUMNS (…)$ macro.
$COLUMN_KEYS+TYPES_V$	This is virtually the same as $COLUMNS+TYPES$ macro, except that each column/type pair will be inserted on a separate line. The inserted text will appear as a vertical comma-separated list of column name/type pairs. Positions of commas and elements before and after each inserted column name are controlled by pre and post macro-text elements. See Tips section below for more details. Note: The $COLUMN_KEYS+TYPES_V$ macro is maintained for backward compatibility. When developing new code snippets, please use the new $COLUMNS (…)$ macro.
$ARGS(…)$	This will be replaced with a comma-separated list of arguments of the selected object or arguments and their data-types. This macro-variable can be used with stored procedures and user-defined functions only. See the separate $ARGS(…)$ macro topic for more information.
$ARGS$	This will be replaced with comma-separated list of arguments of the selected object. Note: The $ARGS$ macro is maintained for backward compatibility. When developing new code snippets, please use the new $COLUMNS (…)$ macro.
$ARGS_V$	This is virtually the same as the $ARGS$ macro, except that each argument will be inserted on a separate line. The inserted text will appear as a vertical comma-separated list of arguments. Positions of commas and elements before and after each inserted argument name are controlled by pre and post macro-text elements. See Tips section below for more details. Note: The $ARGS_V$ macro is maintained for backward compatibility. When developing new code snippets, please use the new $COLUMNS (…)$ macro.

| $ARGS+TYPES$ | This will be replaced with comma-separated list of arguments of the selected object and their data types. This macro-variable can be used with stored procedures and user-defined functions only.

Note: The $ARGS+TYPES$ macro is maintained for backward compatibility. When developing new code snippets, please use the new $COLUMNS (…)$ macro. |
|---|---|
| $ARGS+TYPES_V$ | This is virtually the same as $ARGS+TYPES$ macro, except that each argument/type pair will be inserted on a separate line. The inserted text will appear as a vertical comma-separated list of arguments and their types. Positions of commas and elements before and after each inserted argument/type pair are controlled by pre and post macro-text elements. See Tips section below for more details.

Note: The $ARGS+TYPES_V$ macro is maintained for backward compatibility. When developing new code snippets, please use the new $COLUMNS (…)$ macro. |

For example, if you have a code snippet named "cfetch" having the following text

```
DECLARE v_$COLUMNS+TYPES$;

DECLARE my_cursor CURSOR FOR
SELECT $COLUMNS$
FROM $OBJECT$;

OPEN my_cursor;
fetch_loop:

LOOP
    FETCH FROM my_cursor INTO v_$COLUMNS$
    IF at_end <> 0 THEN
        LEAVE fetch_loop;
    END IF;

    /* ... Cursor logic here ... */

END LOOP fetch_loop;

CLOSE my_cursor;
```

If you type "cfetch" without quotes and then press Ctrl+Enter (or whatever you have selected as a hotkey for this code snippet), you will be presented with a prompt for an object name. If, for example, you select "Customers" table, the text of the code snippet will be inserted at the current caret position and the $OBJECT$ macro-variable will be automatically substituted with the selected table name. The columns of the selected "Customers" table and their data types, will be inserted as variable declarations along with the defined **v_**.prefix for variable names and the following DECLARE CURSOR statement will be generated using Customer's table columns and the results output to the generated SQL variables:

```
DECLARE v_CustomerID nchar(10), v_CompanyName nvarchar(80),
        v_ContactName nvarchar(60), v_ContactTitle nvarchar(60),
        v_Address nvarchar(120), v_City nvarchar(30), v_Region nvarchar(30),
        v_PostalCode nvarchar(20), v_Country nvarchar(30), v_Phone nvarchar(48),
        v_Fax nvarchar(48);

DECLARE my_cursor CURSOR FAST_FORWARD READ_ONLY FOR
SELECT CustomerID, CompanyName, ContactName, ContactTitle, Address, City,
```

```
        Region, PostalCode, Country, Phone, Fax
FROM Customers;

OPEN my_cursor;
fetch_loop:

LOOP
    FETCH FROM my_cursor INTO INTO v_CustomerID, v_CompanyName, v_ContactName,
                               v_ContactTitle, v_Address, v_City, v_Region,
                               v_PostalCode, v_Country, v_Phone, v_Fax;

    IF at_end <> 0 THEN
        LEAVE fetch_loop;
    END IF;

    /* ... Cursor logic here ... */

END LOOP fetch_loop;

CLOSE my_cursor;
```

Macro-variables Execution

Macro-variables code execution is optimized for minimal possible interaction with the user. If a code snippet refers to multiple related macro-variables, only a single prompt is displayed at macro execution time and the results of that prompts are shared for all referenced related macro-variables. The $OBJECT$ macro has highest priority. It can be used to filter the contents of the popup menu and to limit it to certain types of items only. See the $OBJECT(…)$ Macro topic in this chapter for more information.

In the previous example for the "cfetch" code snippet, the snippet code refers to a series of four macro-variables, **$COLUMNS+TYPES$**, **$COLUMNS$,** **$OBJECT$,** and **$COLUMNS$** (again). During execution of the "cfetch" snippet, SQL Assistant displays a single Objects prompt. The results of that prompt are then used to replace all four macro-variables with the appropriate items.

Also note that the text leading or trailing macro-variable names is repeated for each item returned as a result of the macro-variable expansion. See the following topic for more details.

Using Macro-variables with Text Prefixes and Text Suffixes

The prefix and suffix text entered before or after an active macro-variable is honored in all expanded elements. For example, in the SQL Server editor, if you use a snippet that includes the macro @$COLUMNS$, when the snippet code is expanded, each column will be prefixed with the @ sign, effectively inserting variable names into the code. Similarly, if you want to generate text that is automatically expanded from the selected text, and if some word is appended to each expanded item, add that text immediately after the macro-variable name.

See the code snippet referenced in the Special Cases for Column/Variable and Argument/Value Pairs topic in this chapter as an example of using text prefixes with the $COLUMNS(…)$ and $ARGS(…)$ macro variables and their derivative macros. See the predefined code snippets in SQL Assistant options for more examples of using text prefixes and suffixes with other types of macro-variables.

Escaping $ Symbols in Snippet Codes

When executing the snippet code the code processors searches for syntax tokens enclosed in a pair of $...$ symbols. For all tokens identified as known macro-variables it substitutes their references with the generated macro code. All unknown tokens enclosed in a pair of $...$ symbols are removed from the snippet text before code execution. If you need to enter $ symbols as literals within the snippet code, you must escape them using ^ suffix, for example,

```
CREATE OR REPLACE FUNCTION "p_Get$OBJECT(ins_object, table)$"
(
    a_$COLUMNS(vertical,types,keys)$
)
RETURNS REFCURSOR
AS
$^body$^
    /***********************************************************
    * Code generated by SoftTree SQL Assistant
    **********************************************************/
$^body$^
LANGUAGE plpgsql;
```

Note the use of ^body^ tags to escape $ symbols surrounding the **body** token. During snippet code execution, the snipped engine will generate **$body$** code.

Custom Interractive Prompts
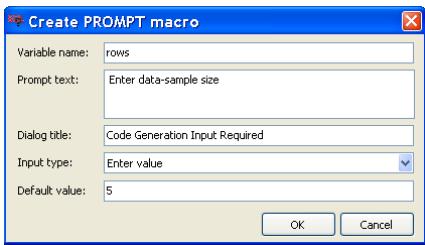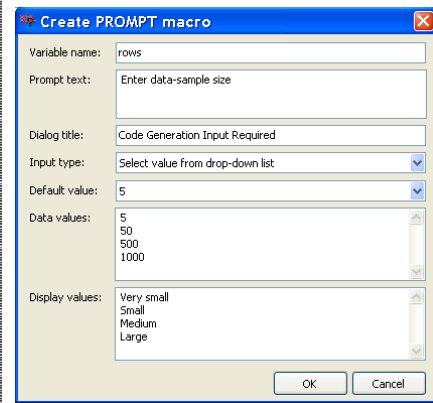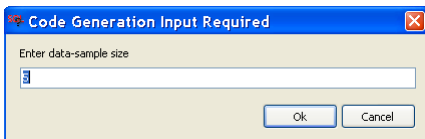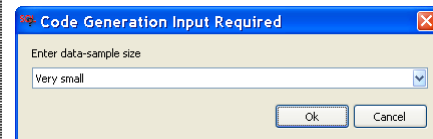
Custom prompts can be used to obtain user input for generating dynamic SQL code using code snippets. Custom prompts can be created using the special **$PROMPT(…)$** macro.

Variable	Meaning
$PROMPT(…)$	This macro creates a data input dialog that displays a custom prompt. The prompt definition is saved in a user-defined macro-variable that can be referenced in the snippet code. The macro code requires entering the following macro-parameters as a comma-separated list: **Variable name** – name of the macro variable in which the result will be saved. This parameter is required. **Prompt text** – The text of the prompt asking user to enter or choose some value. Note that the prompt text may not contain commas. This parameter is required. **Default value** – The default value. This parameter is optional and can be omitted. **Dialog title** – The title of the prompt dialog. Note that the title text may not contain commas. This parameter is optional. **List values** – The list of values for multiple-choice prompts. Values in the list must be space separated. If a value contains spaces, the entire value must be enclosed in double quotes. This parameter is optional and can be omitted. **Descriptive names for list values** – The optional list of descriptive

	value names. Use this parameter if you need to display lookup value names instead of the actual values. Names in the list must be space separated. If a name contains spaces, the entire name must be enclosed in double quotes. This parameter is optional and can be omitted.

Tip: There are two different methods for entering a $PROMPT(…)$ macro into a code snippet when you create new snippet or alter an existing snippet. You can either type in the entire macro code manually or you can use the **Macro** menu available in the top-right corner of the **Code Snippets** tab. Using the **Macro** menu is the recommended method. With the menu, SQL Assistant provides an interactive $PROMPT(…)$ macro build dialog. This dialog makes designing $PROMPT(…)$ macros both safe and simple, ensuring that the macro syntax is correct.

The following screenshots demonstrate how the interactive menu works. In the examples in the left column, "Enter Value" is selected from the **Input Type** drop-down. On the right, "Select value from drop-down list" is selected for **Input Type**.

Input Type presentation style: edit box	Input Type presentation style: drop-down value list
Here is what will be inserted into the snippet code: `$PROMPT(rows,Enter data-sample size,5,Code Generation Input Required)$`	Here is what will be inserted into the snippet code: `$PROMPT(rows,Enter data-sample size,,Code Generation Input Required,5 50 500 1000,"Very small" Small Medium Large)$`
Here is the prompt dialog that displays on the user screen when this snippet is executed: 	This snippet creates the following prompt dialog and drop-down list:

The value obtained with the help of the $PROMPT(…)$ macro should be referenced in the snippet code using a variable name enclosed in $ symbols. The following example shows a snippet named "proc" that is designed for use in the T-SQL dialect. Please note the highlighted text.

```
$PROMPT(proc_name,Enter new procedure name)$

USE [$DB$]
go
```

```
IF object_id('$proc_name$') IS NOT NULL
BEGIN
    PRINT 'Dropping procedure $proc_name$'
    DROP PROCEDURE [$proc_name$]
    IF @@ERROR = 0 PRINT 'Procedure $proc_name$ dropped'
END
go

CREATE PROCEDURE [$proc_name$]
/***********************************************************
* Procedure description:
* Date:    $DATE$
* Author: $OSUSER$
*
* Changes
* Date    Modified By        Comments
***********************************************************
*
***********************************************************/
(
    |<parameters>
)
AS
BEGIN
    /* code business logic here */

    RETURN @@ERROR
END
go

IF @@ERROR = 0 PRINT 'Procedure $proc_name$ created'
go
```

In this example, if you type the characters "proc" and press Ctrl+Enter, SQL Assistant first displays an Input Prompt asking you to enter a new procedure name. Once you have entered the name and clicked the **OK** button, the macro inserts the specified name in each location referenced by the $proc_name$ macro-variable. It then insert the resulting code into the editor replacing "proc" code snippet name.

Tip: The snippet code containing a **$PROMPT(…)$** macro can be also programmed to call other supported macros. Additional interactive prompts may appear on the editor screen during snippet execution if required by other macros referenced in the snippet code.

 Important Notes: Do not use names of built-in macro variables as names of variables referenced in the $PROMPT(…)$ macros because that can lead to unpredictable results. The following example demonstrates type of code that should be avoided:

```
$PROMT(CURRENT_WORD,Enter some word)$
PRINT 'You entered $CURRENT_WORD$'
```

As you can see, the second line in the snippet code refers to $CURRENT_WORD$. This reference is ambiguous, as it is unclear whether this code refers to the predefined $CURRENT_WORD$ macro name or to the $CURRENT_WORD$ macro variable defined in the $PROMPT(…)$ macro.

Also note that snippet code execution methods and the order of macros executed within the snippet code can change in future SQL Assistant versions.

Special Cases for Column/Variable and Argument/Value Pairs

To simplify programming of dynamic code generation, SQL Assistant supports special use cases for code snippets containing active macro-variables referenced on the same line and separated by an equal sign. For example, consider a predefined "si" snippet for T-SQL having the following definition:

```
DECLARE @$COLUMNS+TYPES$

SELECT
    @$COLUMNS$ = $COLUMNS$
FROM $OBJECT$
WHERE
```

When this snippet is invoked, the SELECT clause will be generated as a list of pairs of "variable name = column name" syntax tokens. If you type "si" without quotes and then press Ctrl+Enter (or whatever you have selected as a hotkey for this code snippet), you will be presented with a prompt for an object name. If, for example, you select "Customers" table, the text of the code snippet will be inserted at the current caret position and the $OBJECT$ macro-variable will be automatically replaced by the selected table name.

The columns of the selected "Customers" table and their data types, will be inserted as variable declarations along with the specified @.prefix for variable names. The following SELECT statement will be generated using Customer's table columns and the results output to the generated SQL variables:

```
DECLARE @CustomerID nchar(10), @CompanyName nvarchar(80),
        @ContactName nvarchar(60), @ContactTitle nvarchar(60),
        @Address nvarchar(120), @City nvarchar(30), @Region nvarchar(30),
        @PostalCode nvarchar(20), @Country nvarchar(30), @Phone nvarchar(48),
        @Fax nvarchar(48)

SELECT
    @CustomerID = CustomerID, @CompanyName = CompanyName,
    @ContactName = ContactName, @ContactTitle = ContactTitle,
    @Address = Address, @City = City, @Region = Region,
    @PostalCode = PostalCode, @Country = Country, @Phone = Phone, @Fax = Fax
FROM Customers
WHERE
```

After this code is inserted into the target editor, the editor's cursor will be positioned immediately after the WHERE keyword so that if you need to add a WHERE clause for this generated SELECT-INTO-VARIABLES query, you can quickly generate one by pressing the spacebar and selecting one or more columns from the popup menu that displays.

Similar snippets can be used for other operations. SQL Assistant comes with several sample snippets for each SQL dialect. You can customize them as well as define your own active code snippets.

Note:

Any text entered between an active macro variable and the equal sign, will repeat in every generated value pair. This is true regardless of whether the text is adjacent to the variable name or is separated by a tab or space character. The following example shows how inserted text is handled:

```
EXEC $OBJECT$
    $ARG$ /* test suffix 1 */ = /* test suffix 2 */ $ARG$_var
```

When invoked, this test snippet will cause SQL Assistant to prompt you for the procedure name. Assume that you pick the "SalesByCategory" procedure in the sample "Northwind" database. The text of the resulting code may look like the following:

```
EXEC SalesByCategory
    @CategoryName /* test suffix 1 */ = /* test suffix 2 */ @CategoryName_var,
    @OrdYear /* test suffix 1 */ = /* test suffix 2 */ @OrdYear_var
```

Code Snippet Execution Modes

Code snippets can be processed in either of two execution modes:

Insert Output Into Code – In this mode, text and results returned during the processing of snippet code are inserted into the SQL editor. For example, if a snippet contains static text BEGIN END, this text is inserted into SQL editor when the snippet is activated. If a snippet contains text with macros for dynamic code generation, the macros are executed during snippet code processing, macro references are replaced by the appropriate values, and the resulting code is inserted into the SQL editor. For specific examples, see the Macro-variables and Dynamic Code Generation topic in this chapter. The **Insert Output into Code** mode is intended for automating SQL coding processes.

Execute and Display Output Results – In this mode, text and results returned during the processing of snippet code are displayed in SQL Assistant's Results and Messages panes, while code in the SQL Editor remains unmodified. In this mode, the data output methods and formats are the same as formats used to display output results returned during regular SQL code execution and data preview. See examples in Reading and Understanding Code Execution Output for more details. Note that SQL Assistant's Results and Messages panes are displayed automatically whenever they are required. In the case where snippet code does not return any results, the code is executed in the database and only the Messages pane is displayed with code execution status messages. The Execute and Display Output Results mode is intended to automate execution of repetitive SQL queries, data lookup queries, and other auxiliary functions.

Advanced Code Entry Automation

Advanced Snippet Programming

SQL Assistant provides special coding **$$…$$** macro, and several supporting macros for programming advanced code snippets. This enables you to use SQL that you are already familiar with as a macro programming language. Using **$$…$$** macro you can create code snippets for automating SQL code generation for your custom projects. No job is too big or too small for advanced code snippets. They can be used to generate elementary code, for example, for getting a data value from the database and inserting it into the SQL editor code. The can be used to generate complex code, for example, to generate an entire set of CRUD stored procedures for a given table or project.

Programming advanced snippets is like building development tools for yourself and other members of your team. The results can be reused multiple times and applied to multiple database objects in multiple projects. Developers can share their code snippets using the built-in **Settings Export/Import** feature. See the Sharing SQL Assistant Settings Between Team Members topic in CHAPTER 48 for more details.

$$…$$ Macro

Using the $$…$$ macro is the primary method for creating advanced snippets for code entry automation. You can enter any valid SQL code between the double dollar $$ symbols. If snippet's execution mode is set to **Insert Output Into Code**, it's expected that the SQL code executed by the $$…$$ macro returns some text based results which can be inserted into the code editor as dynamically generated text. If snippet's execution mode is set to **Execute and Display Output Results**, the the SQL code in the $$…$$ is executed in the database, and if returns any results, the results are redirected to a separate **Results** pane; and the code in the editor is not changed. For information about supported snippet execution modes, see the previous topic Code Snippet Execution Modes

Variable	Meaning
$$…$$	This macro contains executable SQL code you want to run when the macro is invoked. The results of the executed code can be optionally output to the editor replacing the macro name.

Note that predefined macros can be referenced inside the body of custom $$….$$ macros. SQL Assistant executes predefined macros first and replaces their references before executing a custom $$…$$ macro. For example, you define a code snippet named "do-it" with the following code, note the highlighted text of custom $$…$$ macro:

```
SET @my_var = 'As of $DATE$ @ $TIME$ table $OBJECT$ had $$ SELECT count(*) FROM $OBJECT$ $$ rows'
```

To invoke this snippet, in the SQL editor type "do-it" (without the double quotes) and press Ctrl+Enter. This will trigger the following activities:

1. Replace the $DATE$ and $TIME$ predefined macros with the current date and time, for example, `5/25/2009` and `9:32:45`.

2. Display Object Name popup at the current edit caret position in the editor.

3. After you select a table from the popup, for example, `dbo.Accounts`, that table name replaces the reference to the $OBJECT$ macro.

4. Execute the custom query enclosed in the $$…$$ symbols. In this example, the query is: `SELECT count(*) FROM dbo.Accounts` query.

5. Replace the `$SELECT count(*) FROM $OBJECT$$` with the result of the above query For this example, assume the result is:12055.

6. If the snippet execution mode is Insert Output Into Code, SQL Assistant inserts the code generated by the snippet into your SQL editor. In this example, the inserted code is:

   ```
   SET @my_var = 'As of 5/25/2009 @ 9:32:45 table dbo.Accounts had 12055 rows'
   ```

Important Notes:
- Multiple $$…$$ macros can be used in the same code snippet.

- Macro code for $$…$$ macros can include other macros such as $OBJECT$, $DATE$, $CURRENT_WORD$ and others.

- Nested $$…$$ macros are not supported, and $$…$$ macros may not be used within $PROMPT(…)$ macros.

- Result set processing rules:

  1. If a $$…$$ macro contains a SELECT statement or stored procedure returning a multi-column result set, multiple column values are concatenated as tab-separated text. The resulting text row

is then processed by the snippet

2. If a $$…$$ macro contains a SELECT statement or stored procedure returning multi-row result set, text of all rows is concatenated using the carriage return symbol as a line separator. Columns in each row are processed as described in rule 1 above.

3. If a $$…$$ macro contains multiple SELECT statements or stored procedures returning multiple result sets, each result set is processed as described in rules 1 and 2 above. The resulting text from all result sets is then concatenated using the carriage return symbol as a result set separator.

The following graphical examples demonstrate how to use $$…$$ macros in your custom code snippets to automate repetitive tasks and to generate required SQL code dynamically:

Example 1 (DB2): Running a report for a selected account number – This example demonstrates how to set up a code snippet that can be used to automate reports retrieval for an expense account. The snippet can be invoked later with a single key press. Note that this method is not specific to the DB2 or UltraEdit editor pictured on the screenshots.
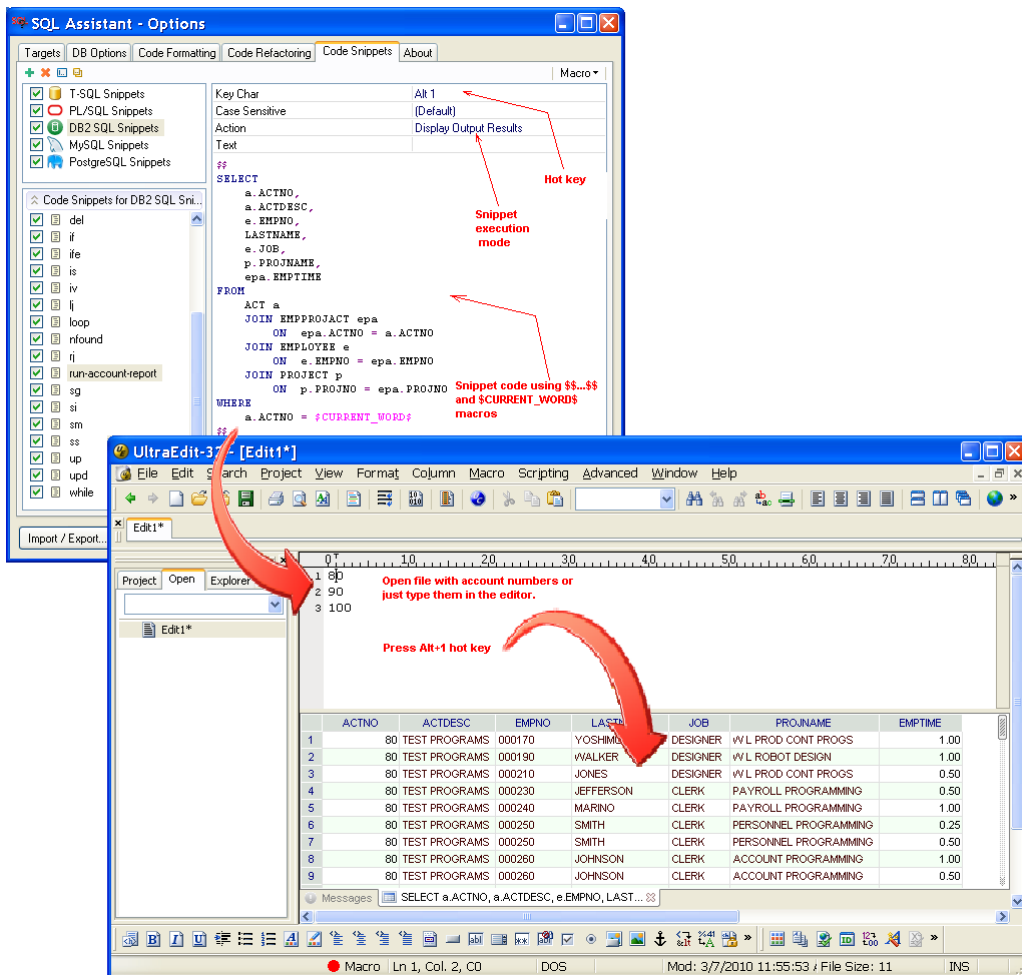
Code for this example snippet:

```
$$
SELECT
    a.ACTNO,
    a.ACTDESC,
    e.EMPNO,
    LASTNAME,
    e.JOB,
    p.PROJNAME,
    epa.EMPTIME
FROM
    ACT a
    JOIN EMPPROJACT epa
        ON   epa.ACTNO = a.ACTNO
    JOIN EMPLOYEE e
        ON   e.EMPNO = epa.EMPNO
    JOIN PROJECT p
        ON   p.PROJNO = epa.PROJNO
WHERE
    a.ACTNO = $CURRENT_WORD$
$$
```

**Example 2 (SQL Server): Reverse-engineering CREATE VIEW syntax with additional comments – ** This
example demonstrates how to set up a code snippet that can be used to automate reverse-engineering of DDL
syntax of a view with additional comments. The snippet can be invoked later with a single key press. Note that
this method is not specific to the SQL Server or SQL Server Management Studio editor pictured on the
screenshots.
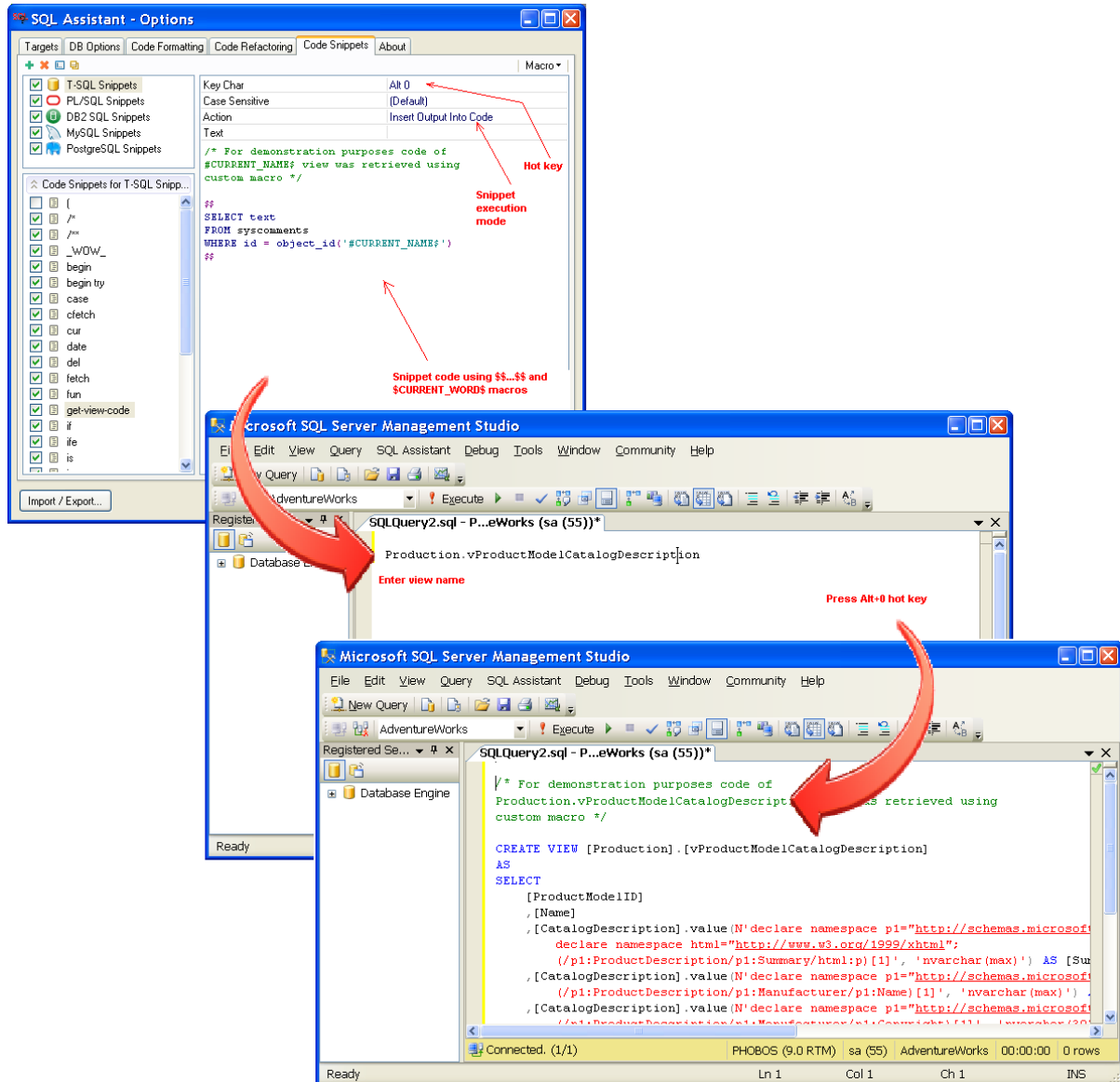
Code for this example snippet:

```
/* For demonstration purposes code of
#CURRENT_NAME$ view was retrieved using
custom macro */

$
SELECT text
FROM syscomments
WHERE id = object_id('#CURRENT_NAME$')
$
```

Example 3 (Oracle): Generating "Delete" stored procedure for a table with primary key – This example demonstrates how to set up a code snippet to automate generation of stored procedure code for any table in the database. The snippet can be invoked later with a single keystroke. This example also demonstrates using multiple $$…$$ macros within a single code snippet. Note that this method is not specific to the Oracle or DB Tools for Oracle editor pictured on the screenshots.
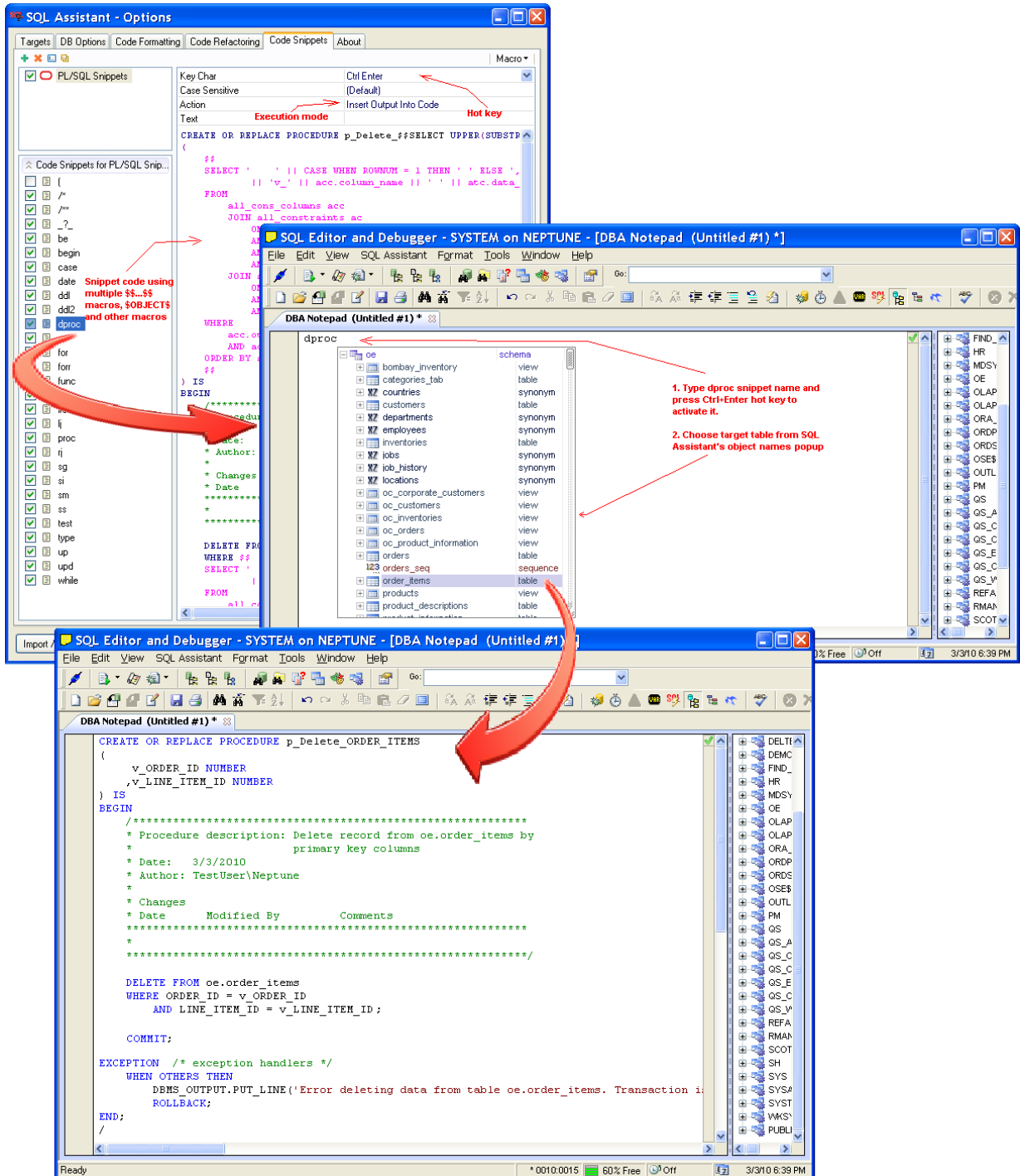
Code of this example snippet:

```
CREATE OR REPLACE PROCEDURE p_Delete_$$SELECT UPPER(SUBSTR('$OBJECT$', INSTR('$OBJECT$', '.') + 1,
30)) FROM dual$$
(
```

```
      $$
      SELECT ' ' || CASE WHEN ROWNUM = 1 THEN ' ' ELSE ',' END
              || 'v_' || acc.column_name || ' ' || atc.data_type
      FROM
          all_cons_columns acc
          JOIN all_constraints ac
              ON   ac.owner = acc.owner
              AND ac.table_name = acc.table_name
              AND ac.constraint_name = acc.constraint_name
              AND ac.constraint_type = 'P'
          JOIN all_tab_cols atc
              ON atc.owner = acc.owner
              AND atc.table_name = acc.table_name
              AND atc.column_name = acc.column_name
      WHERE
          acc.owner = UPPER(SUBSTR('demo.order_items', 1, INSTR('demo.order_items', '.') - 1)) /*
parse schema */
          AND acc.table_name = UPPER(SUBSTR('$OBJECT$', INSTR('$OBJECT$', '.') + 1, 30)) /* parse
table name */
      ORDER BY acc.position
      $$
) IS
BEGIN
      /*************************************************************
      * Procedure description: Delete record from $OBJECT$ by
      *                        primary key columns
      * Date:   $DATE$
      * Author: $OSUSER$
      *
      * Changes
      * Date        Modified By        Comments
      *************************************************************
      *
      *************************************************************/

      DELETE FROM $OBJECT$
      WHERE $$
      SELECT ' ' || CASE WHEN ROWNUM = 1 THEN ' ' ELSE '  AND ' END
              || acc.column_name || ' = v_' || acc.column_name
      FROM
          all_cons_columns acc
          JOIN all_constraints ac
              ON   ac.owner = acc.owner
              AND ac.table_name = acc.table_name
              AND ac.constraint_name = acc.constraint_name
              AND ac.constraint_type = 'P'
          JOIN all_tab_cols atc
              ON atc.owner = acc.owner
              AND atc.table_name = acc.table_name
              AND atc.column_name = acc.column_name
      WHERE
          acc.owner = UPPER(SUBSTR('demo.order_items', 1, INSTR('demo.order_items', '.') - 1)) /*
parse schema */
          AND acc.table_name = UPPER(SUBSTR('$OBJECT$', INSTR('$OBJECT$', '.') + 1, 30)) /* parse
table name */
      ORDER BY acc.position
      $$ ;

      COMMIT;

EXCEPTION  /* exception handlers */
      WHEN OTHERS THEN
          DBMS_OUTPUT.PUT_LINE('Error deleting data from table $OBJECT$. Transaction is going to be be
rolled back.');
          ROLLBACK;
END;
/
```

$OBJECT(…)$ Macro

The special **$OBJECT$** macro-variable provides advanced methods for selecting database names, schemas, schema objects, columns, types and even variables and various structures within programmatic objects such as procedures, Oracle packages and object types. The macro parameters in parentheses control display of the prompt and determine what can be inserted into the editor in place of the macro reference. For example,

$OBJECT(ins_object, view, mview)$ causes the SQL Assistant to display a prompt that lists objects only of type views and materialized views. The following options can be used with the $OBJECT(…)$ macro -variable.

"Insert" options control how $OBJECT(…)$ macro is executed and its reference is replaced in the snipped code during code execution.

- **Nothing** – this option triggers the popup menu during code snippet execution, but instructs SQL Assistant not to insert anything in place of this specific $OBJECT$ reference. Use of this option makes sense only if other macro-variables are used in the same code snippet. The filter options in the $OBJECT$ macro controls content of the prompt and therefore what can be used with other macros in the same code snippet.

- **Qualified name** – this option causes the code snippet to insert the selected items into the code in place of the $OBJECT(…)$ macro-variable. The selected items are named as specified in the "Always Fully Qualify Object Names" option in "SQL Assistance" group of options for your database type. See the Qualify Object Names topic in CHAPTER 8 for more details about object name qualification.

- **Database name** – this option causes the code snippet to insert name of the database containing the selected items. In other words, during snipped code execution $OBJECT(…)$ macro-variable reference is replaced with a database name. This option can be used with SQL Server, Sybase ASE and PostgreSQL database system types.

- **Schema name** – this option causes the code snippet to insert name of the schema containing the selected items belong to. In other words, during snipped code execution $OBJECT(…)$ macro-variable reference is substituted with schema name.

- **Object name** – this option causes the code snippet to insert the selected object name. In this case, the state of "Always Fully Qualify Object Names" option is ignored.

- **Column name** – this option causes the code snippet to insert the selected column name. The result is similar to the result of $COLUMS$ macro-variable, except that just a single column can be selected. To access column names, expand the object level in the popup, by clicking the little [+] sign in front of the object name or using the common keyboard shortcuts
  Note: To select a column, it is enough to click that column in the popup, or alternatively using the keyboard select the required column name and then press the Enter key.

The other options implement filters controlling type of the content that can appear and/or can be selected in the popups triggered by the code snippet. The following item types can be selected for content filtering:

- Objects
- Arguments of functions and procedures
- Columns and arguments having Binary data type
- Built-in Functions
- Columns
- Databases
- Columns and arguments having Date and Date/time data types.
- Functions

- Login Names
- Materialized Views
- Columns and arguments having Number data types
- Object Types
- Packages
- Package Constants
- Package Cursors
- Package Exceptions
- Package Records
- Package Subtypes
- Package Types
- Package Variables
- Procedures
- Public Synonyms
- Records
- Roles
- Schemas
- Sequences
- Columns and arguments having String data type
- Synonyms
- Tables
- Table Functions
- Views
- *In addition you can specify within the $OBJECT(…)$ macro extended database object types, for example, tablespace, certificate, credential, ddl trigger, xml collection, and so on*

Important Notes: Do not use multiple $OBJECT$ macros in the same code snippet with different **Insert** and **Filter** options. The behavior of the code snippet is ambiguous in this case, and the code generation result is unpredictable.

The following example demonstrates use of the $OBJECT(...)$ in SQL Assistant CRUD code generation templates for SQL Server.

```
UPDATE [$OBJECT(ins_schema, table)$].[$OBJECT(ins_object, table)$]
SET $COLUMNS(vertical,updatable)$ = @$COLUMNS(vertical,updatable)$
WHERE "AND "$COLUMNS(vertical,keys)$" = @"$COLUMNS(vertical,keys)$
```

$COLUMNS(…)$ Macro

The special **$COLUMNS$** macro-variable provides advanced methods for selecting table, view, and table function column names and optionally their data types. The macro parameters within brackets following COLUMNS macro-variable name control what can be inserted into the editor in place of the macro. For example, **$COLUMNS(vertical,types,keys)$** causes SQL Assistant to generate a vertical comma-separated list of primary key column names with their data types. The $COLUMNS(…)$ macro can be prefixed or suffixed with additional text that you want to add to each element of the generated list. See the sing Using Macro-variables with Text Prefixes and Text Suffixes topic in this chapter for more details.

To access column names in the popup created by the macro, expand the object level in the popup, by clicking the little [+] sign in front of the object name or using the common keyboard shortcuts. Note: To select a single

column, it is enough to click that column in the popup, or alternatively using the keyboard select the required column name and then press the Enter key. To select multiple columns, using mouse tick the checkboxes in front of the column names and then press the Enter key.

The following options can be used with the $COLUMNS(…)$ macro-variable.

- **Vertical list** – this option causes SQL Assistant to generate vertical lists or columns. If this option is not selected, a horizontal list will be generated. If you choose the horizontal list option and there are more columns than can fit on a single line, additional lines with columns will be added as needed. Line wrapping is controlled by **Spacing, Wrapping** options in the Code formatting settings. See CHAPTER 5, Code Formatter and Beautifier for more information.

- **Column name and datatype** – this option causes SQL Assistant to insert column names along with their data type as elements of the list. If not specified, only column names will be inserted.

- **Don't use datatype precision and length** – this option can be used only in combination with **Column name and datatype** option. When this option is selected, SQL Assistant lists data type names without specifying data type length and precision.

- **Only key columns** – this option causes SQL Assistant to process primary key or unique columns only. Use of this option makes sense for tables with primary key and/or unique key columns only. It will not work for tables not having such keys. It will not work also for views and table functions.

- **Only non-key columns** – this option causes SQL Assistant to skip primary key and unique columns only. Use of this option makes sense for tables with primary key and/or unique key columns only. The result of this option is opposite of **Only key** columns option, they are mutually exclusive and should not be used together.

- **Only updatable columns** – this option causes SQL Assistant to process updatable table columns only. All system generated and computed columns will be skipped.

  Note: Do not use the **Only key columns** and **Only updatable columns** options together. When used together, the result is unpredictable and not guaranteed to be accurate.

- **Don't add comma –** – this option causes SQL Assistant not to add trailing commas after columns.

The following example demonstrates use of the $COLUMNS(...)$ in SQL Assistant CRUD code generation templates for SQL Server.

```
UPDATE [$OBJECT(ins_schema, table)$].[$OBJECT(ins_object, table)$]
SET $COLUMNS(vertical,updatable)$ = @$COLUMNS(vertical,updatable)$
WHERE "AND "$COLUMNS(vertical,keys)$" = @"$COLUMNS(vertical,keys)$
```

$ARGS(…)$ Macro

The special **$ARGS$** macro-variable provides advanced methods for selecting stored procedure and user defined function argument names and, optionally, their data types. The macro parameters within parentheses following ARGS macro-variable name control what can be inserted into the editor in place of the macro reference. For example, **$ARGS(vertical,types)$** causes SQL Assistant to generate a vertical comma-separated list of argument names with their data types. The $ARGS(…)$ macro can be prefixed or suffixed with

additional text to be added to each element of the generated list. See the Using Macro-variables with Text Prefixes and Text Suffixes topic in this chapter for more details.

The following options can be used with the $ARGS(…)$ macro-variable.

- **Vertical list** – this option causes SQL Assistant to generate a vertical list of arguments. If this option is not selected, a horizontal list will be generated. If the horizontal list option is selected and there are more arguments that can fit on a single line, additional lines with arguments will be added as needed. Line wrapping is controlled by **Spacing, Wrapping** options in the Code formatting settings.. See CHAPTER 5, Code Formatter and Beautifier for more information.

- **Argument name and datatype** – this option makes SQL Assistant insert argument names and their data type as elements of the list. If this option is not specified, only argument names will be inserted.

- **Don't use datatype precision and length** – this option can be used only in combination with the **Argument name and datatype** option. It causes SQL Assistant to insert data type names without specifying data type length and precision.

- **Sort arguments alphabetically** – by default arguments are listed in the order they are defined. This option makes SQL Assistant insert arguments in alphabetical ascending order A to Z.

- **Descending order -** This option makes SQL Assistant insert arguments in alphabetical descending order Z to A.

- **Don't add comma –** – this option causes SQL Assistant not to add trailing commas after arguments.

Other Special Macros

SQL Assistant supports several other special macros designed for use in code snippets supporting advanced code generation techniques: Typically these macros should be used with **$$…$$** macro.

Variable	Meaning
$CURRENT_WORD$	The text of current word containing the edit caret. By "word", we mean any series of text characters separated by spaces, commas, periods, brackets, or other non-alphanumeric characters.
$CURRENT_NAME$	The name of current object containing the edit caret. This macro differs from the $CURRENT_WORD$ macro in that the object name may contain dot-separated schema and database parts. The macro will pick these parts automatically along with the object name.
$CURRENT_LINE$	The entire text of the current line containing the edit caret. This text can be parsed within the $$…$$ macro during execution.
$CURRENT_SEL$	The currently highlighted text. Important note: The text is inserted into the snippet code "as is" before the snippet code is executed. If the highlighted text spans several lines, it could potentially break the snippet code. The following is an example of a code snippet in which it is acceptable to insert multiple lines of text returned by the $CURRENT_SEL$ macro: In this example the text highlighted in the

<table>
<tr><td></td><td>editor is inserted into the snippet code within the comments block.</td></tr>
</table>

```
CREATE PROCEDURE my_procedure AS
/*
comments go here: $CURRENT_SEL$
*/
```

$CURRENT(…)$ Macro

The special **$CURRENT$** macro-variable provides advanced methods for tokenizing the text near edit caret and returning various lexical elements. The options within the macro define which lexical elements or words to return. Note that "word" is defined as a series of alphanumeric characters separated by white spaces and/or non alphanumeric characters with an exception of a series of characters enclosed in double quotes. Alphanumeric characters comprise the combination of the twenty-six characters of the Latin alphabet (from A to Z) and the numbers 0 to 9. In addition two special symbols "$" and "_" are treated as alphanumeric characters too because they are often used in schema object names.

The following options can be used with the $CURRENT(…)$ macro.
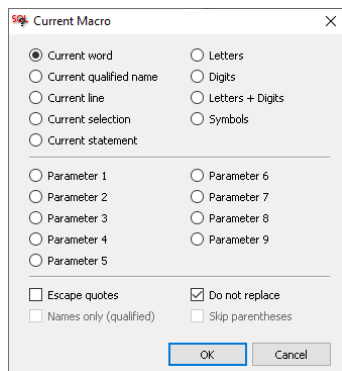
- **Current word** – the result is the same as the result of $CURRENT_WORD$ macro described in the previous topic, unless additional behavior-control options are selected. Additional optional modifiers can be choosen for specific types of "words" that can be recognized.

  Letters – a word is defined by a continuous group of letters a..z only, any other character is a word boundary.

  Digits – a word is defined by a continuous group of digits 0..9 only, any other character is a word boundary.

  Letters + Digits – a word is defined by a continuous group of letters a..z and digits 0..9 only, any other character is a word boundary.

  Symbols – a word is defined by a continuous group of symbols only which are non leters and not digits; white space, any letter or digit is a word boundary.

  If no modifiers are selected, all letters, digits and symbols are included; a word boundary is defined by white space; everything else is part of the word.

- **Current qualified name** – the result is the same as the result of $CURRENT_NAME$ macro described in the previous topic, unless additional behavior-control options are selected.

- **Current line** – the result is the same as the result of $CURRENT_LINE$ macro described in the previous topic.

- **Current selection** – the result is the same as the result of $CURRENT_SEL$ macro described in the previous topic.

- **Current statement** – The entire text of the current statement containing the edit caret. This text can be parsed within the $$…$$ macro during execution.
  Important Note: The text is inserted into the snippet code "as is" before the snippet code is executed. If the current statement text spans several lines, it could potentially break the snippet code.

The following is an example of a code snippet in which it is acceptable to insert multiple lines of text returned by the $CURRENT(statement)$ macro: In this example the statement text is wrapped into CRETE PROCEDURE body

```
CREATE PROCEDTE my_procedure AS
BEGIN
  -- this line along with the current statement in the
  -- eitor will wrapped into CREATE...BEGIN...END
  -- procedure blocks

  $CURRENT(statement)$
END
```

It can be also used inside $$...$$ macro to dynamically insert text of the current statement into $$...$$ macro, and then execute the resulting, and insert final result into the editor. For example:

```
-- this comment line along with the modified results
-- of the dynamic statement in the editor will be
-- inserted into the code
$$
  SELECT 'this value' AS thisColumn, t.*
  FROM ( $CURRENT(statement)$ ) AS t
$$
```

- **Parameter 1 to Parameter 9** – This is the same as **Current qualified name** except that it returns lexical elements preceding the edit caret position. Parameter 1 refers to the element just before the edit caret, Parameter 2 refers to the element before the Parameter 1 element, Parameter 3 - to one before Parameter 2, and so on... The lexical element counting starts backward from the one just before the current edit caret position toward the beginning of the current line.

  If this option is coupled with **Names only (qualified)** option, the macro processors accounts for and returns qualified names only. For example, in the following line

  ```
  CREATE INDEX ind_schema.ind_name ON
  tab_schema.tab_name;|
  ```

  macro $CURRENT(param1, names_only)$ returns tab_schema.tab_name, while macro $CURRENT(param2, names_only)$ returns ind_schema.ind_name. The ON keyword is skipped because it does not represent a qualified name in schema.object or database.schema.object format.

- **Escapes quotes** – this behavior-control option instructs the macro processor to return quotes and brackets as they are specified in quoted names and do not remove them. By default all quotes and brackets are removed. For example, text "here [comes name]" is parsed as "here" and "comes name" without that option, and as "here" and "[comes name]" when that option is chosen.

- **Do not replace** – this behavior-control option makes the macro-parser not to replace the lexical elements in the editor text which are referenced in the snippet code with the code generated by the code snippet. In other words, the results of the snippet code execution are appended to the text in the editor.
  For example, if the text in the editor is

  ```
  CREATE INDEX ind_schema.ind_name ON
  tab_schema.tab_name;|
  ```

  here | symbol designates position of the edit caret in the line

above.

and a code snippet like

```
/* Here we are going to index data in table
$CURRENT(param1, names_only, dont_replace)$ */
```

is applied to it, the result would be

```
CREATE INDEX ind_schema.ind_name ON
tab_schema.tab_name; /* Here we are going to index data
in table tab_schema.tab_name */
```

Note that the highlighted text has been added to the text in the editor after the edit caret. In comparison, if **Do not replace** option is not used, and a code snippet like

```
/* Here we are going to index data in table
$CURRENT(param1, names_only)$ */
```

is applied to the same text, the result would be different

```
CREATE INDEX ind_schema.ind_name ON /* Here we are
going to index table tab_schema.tab_name */;
```

In this case the code generated by the snippet (the highlighted text) has replaced the parameter 1 referenced in the macro.

Important Note: The **Do not replace** option is useful only if the code snippet contains single reference to $CURENT(...)$ macro. If there are multiple references within the same code snippet and **Do not replace** option is unchecked, the results of the code snippet execution are unpredictable.

- **Names only (qualified)** – this behavior-control option can be used with **Parameter 1 to Parameter 9** options. It defines type of the lexical elements to look for, in other words whether to look for simple words or for qualified names only. For example, text "*name3.name4*)name1.name2, name3.name4" is parsed as four separate lexical elements "*name1","name2", "name3", and "name4",* if this option is not chosen, or as two elements "*name1.name2"* and "*name3.name4"* if this option is chosen. A qualified name is a name in schema.object or database.schema.object format.

- **Skip parenthesis** – this behavior-control option makes the macro parser ignore all parenthesis and any text between them. If you want to extract function name from the code ignoring any parameters that may follow it, you can choose this option. For example, if macro

```
[schema].[$CURRENT(param1, skip_parentheses)$]
```

is applied to

```
EXEC [some_function](55, "value 5", 77)
```

The result of the macro execution would change text in the editor to

```
EXEC [schema_name].[some_function](55, "value 5", 77)
```

inserting "[schema_name]." before the function name.

CHAPTER 8, Smart Database Refactoring

Overview

SQL Assistant's refactoring functions allow you to quickly and safely reorganize and restructure your database code. What makes SQL Assistant refactoring different is its ability to scan and parse existing database code and in the code locate and update references to the items being refactored. For example, if you rename a table column, not only can SQL Assistant find all database dependencies associated with that column, it can also locate column references within the code of stored procedures, functions, views and other objects. These referenced objects will also be automatically renamed, and all affected objects will be recompiled as required.

Renaming objects, methods, columns, parameters

SQL Assistant supports several "rename-class", "rename-method", "rename-field", "rename-parameter", and "rename-variable" refactoring methods for restructuring existing database code and schema structures. In other words, this type of refactoring can be used to safely change the name of an existing database object, column, or parameter. SQL Assistant automatically searches for database code dependencies, analyzes code references, and suggests changes in dependent objects so that they remain operational after the root object change. All code change suggestions are previewed before the actual changes take place in the database.

Adding new columns and parameters

SQL Assistant supports several "add-field" and "add-parameter" methods for refactoring and restructuring an existing database code and its dependencies. In other words, this type of refactoring can be used for safely adding a new column to a database table or view, or adding a new parameter to an existing procedure or function. SQL Assistant automatically searches for database code dependencies, analyzes code references, and suggests changes in dependent objects so that they remain operational after the root object change. All code change suggestions are previewed before the actual changes take place in the database.

Deleting objects, columns, parameters

SQL Assistant supports several "delete-class", "delete-field", and "delete-parameter" methods for refactoring and restructuring existing database code and code dependencies. In other words, this type of refactoring can be used to safely drop an existing database object, to drop a column from a database table or view, or to delete a parameter from an existing procedure or function. SQL Assistant automatically searches for database code dependencies, analyzes code references, and suggests changes in dependent objects so that they remain operational after the root object change. All code change suggestions are previewed before the actual changes take place in the database.

Reorganizing code layout and qualifying object names

SQL Assistant supports advanced highly customizable methods for code formatting and layout control. These methods can be used for standardizing code layout of database procedural objects developed by multiple developers and/or ported from legacy applications. These methods can be also used for quick reformatting of hard to read SQL code generated by various automated code generators and applications, as well as for reformatting non-formatted SQL code captured by various database tracing utilities and profilers. Code reformatting can be performed both in on-line editor-only mode and off-line batch mode for reprocessing off-line SQL files. A separate chapter in this manual is dedicated to the usage, interface, and layout customization techniques and options available. See CHAPTER 5, Code Formatter and Beautifier for more details.

The "qualifying-class-names" refactoring method is supported for renaming and fully qualifying object references in procedural SQL code. In other words, this type of refactoring can be used for automatically adding schema name and, optionally, database name qualifiers to the names of objects referenced within procedural code such as SQL bodies of stored procedures and user-defined functions.

Extracting reusable code and code encapsulation

SQL Assistant also supports basic types of "extract-class" and "extract-method" refactoring types for breaking database code into more logical pieces.

Tips:

- The results of refactoring operations can be logged to a file along with backup script containing the original code for altered procedures, functions, and views. The log file can be used to roll back changes if you find the results of refactoring operations to be unsatisfactory.

- Use the refactoring interfaces management icons available in the top-left corner of the Options dialog to create new refactoring interfaces or to rename, duplicate or delete existing refactoring interfaces.

  Note that the icon functions are sensitive to the location of the focus in the Code Refactoring tab. For example, if a refactoring interface is selected in the top-left list box and you click the X button, the selected interface will be deleted entirely, including all associated refactoring rules. However, if a refactoring rule is selected in the left-bottom list box when you click the X button, the selected rule will be deleted.

- The content of the right side of the Code Refactoring tab is context sensitive as well. If a refactoring interface is selected, the definition of the selected interface is displayed in the right window. If a refactoring rule is selected, the rule definition code is displayed on the right.

- You can drag-and-drop refactoring interface names in the top left list to rearrange their order. You can use that to push most commonly used interfaces to the top of the list and minimize the amount of scrolling and clicking required for customizing refactoring rules.

Code Refactoring Macros

Code refactoring macros are special macros intended for use in code refactoring templates that can be used along with other supported macros described in CHAPTER 7, Macro-variables and Dynamic Code Generation topic to customize behavior of refactoring methods.

Variable	Meaning
$REFACTORING_OBJ_NAME$	The fully qualified name of the selected schema object, or the name entered by user in refactoring operations that prompt users to enter a new object name.
$REFACTORING_COLUMNS$	This macro variable can be used in templates for converting queries into database views and for similar refactoring operations. It inserts comma separated list of columns from the result set returned by the selected query.
$REFACTORING_COLUMNS_V$	It's the same as $REFACTORING_COLUMNS$ except that it inserts vertically aligned list of column names.
$REFACTORING_CODE$	The code selected in the editor.
$REFACTORING_ARGS+TYPES$	The suggested argument signature for the database function or procedure for which the refactoring is performed. Typically the signature is generated based on declarations of local variables in the selected code. This macro can be used in code refactoring operations

	for converting selected code to standalone database functions or procedures. For example: SQL Server: @cust_id INT, @eff_date DATETIME, @ret INT OUT MySQL: p_customer_id INT, p_effective_date DATETIME Together with $REFACTORING_OBJECT$ it can be used to generate code for a new function definition or in other operations where full function signature is required.
$REFACTORING_ARGS$	It's the same as $REFACTORING_ARGS+TYPES$ except that it inserts list t of argument names omitting their data types.

Refactoring Wizard Dialog

The Refactoring Wizard dialog is used by virtually all refactoring methods. The wizard dialog guides you through a 3-step refactoring process:

- Step 1: Enter required refactoring parameters and options.

- Step 2: Preview and edit the proposed changes.

- Step 3: Execute the refactoring operations and review the output results and logs.

The labels on some buttons on the Refactoring Wizard dialog are step sensitive. For example, the second button label reads "Next" in step 1, and reads "Refactor" in step 2.

Tip: If the refactoring operation fails for some reason, you can click the Back button to go back to step 2 where you can review and edit the code for any failed items. After making necessary corrections, click the Refactor button again to rerun the process.

Layout

Step 2 of the Refactoring Wizard is where you will spend most of your time reviewing proposed changes and, if necessary, providing additional input. In Step 2, the dialog is split into three parts: the refactoring operation description in the top section of the dialog, the object tree navigator on the left side, and the SQL editor on the right side. Items in the object tree are represented by icons indicating their object type and text labels as illustrated in the screenshot below. See the Object Tree Legend table following the screenshot for a key to icon meanings. Checkboxes in front of item icons can be used to select and unselect objects you want to change.

Object Tree Legend

The following types of checkboxes with optional overlay icons could be displayed in the object tree.

Item is not selected for refactoring, no action will be taken for this item.

Item is selected and ready for refactoring.

Item may need refactoring, but SQL Assistant is unsure how to refactor it correctly. Your input in the SQL editor is required before the object code can be altered in the database. You should select this item and manually correct the code in the SQL editor box before advancing to Step 3.

A reference to the object being refactored is found in the code. However, SQL Assistant assumes that no changes are required in that item. This is an information message only. If you think that changes are required, select this item and manually correct the code in the SQL editor box before advancing to Step 3.

For an expandable schema or database item, this type of checkbox indicates that only a subset of "child" items has been selected in the object tree branch for this item. For a simple item, this type of checkbox indicates that changes in the database completed successfully. This type of checkbox displays only if you click the Back button to return to Step 2 to review and edit an initial refactoring operation.

Item changes in the database failed. This type of checkbox displays only if you click the Back button to return to Step 2 to review and edit an initial refactoring operation.

Tip: The item for which you invoked the refactoring operation is displayed in bold in the object tree.

Navigation Status Bar Legend

The navigation bar is available on the right side of the Refactoring Wizard dialog in step 2. The status bar provides color coded interactive location indicators enabling you to quickly jump to a line of code where a

change is pending or where an important reference has been found. Click on a location indicator to jump to the associated line of code.

The following color codes are used:

Gray   Indicates certain references found in the code, typically references to refactored objects. No input is expected in such places, this type of mark is simply used to help you visually locate refactored object names

Blue   Indicates places in the code that SQL Assistant is changing automatically. If necessary you can use the embedded SQL editor to override the changes.

Red   Indicates places in the code that SQL Assistant is unable to change automatically, typically due to ambiguous definitions or situations in which a refactoring path is unclear. Your manual input is required. You are expected to modify code in such places manually using the embedded SQL editor.

Embedded SQL Editor

The Refactoring Wizard dialog features an embedded SQL Editor you can use to modify code before it is executed. For more information on the supported features, see CHAPTER 3, Code Assistants and SQL Intellisense.

Note that the editor supports a dual-mode interface:

- **Read-only mode with change highlighting** of proposed changes for the current refactoring operation and selected item. In this mode, the right side status bar is used for quick change navigation. Color-coded marks are used to indicate ending changes and other sensitive elements in the code.

- **Edit mode**, a full featured SQL editor with SQL Intellisense and other editor features. This mode is activated automatically as soon as you start typing anything. In this mode the right-hand side status bar is used to show syntax check results. To switch back to **Read-only mode with change highlighting,** select a different object in the object tree on the left.

Important Notes:

- If you start modifying code of an object that is not selected for refactoring in the object tree, SQL Assistant will automatically select it.

- You should review all selected objects and their pending changes before you click the **Refactor** button on the Refactoring Wizard dialog.

Tip: The Refactoring Wizard dialog window is resizable. A border of the window can be dragged to change the size of the window. You can also use the dialog window **Maximize** button to open it full screen and allow more room for the dialog controls, including the SQL Editor

Refactoring Log Legend

The refactoring log is displayed in step 3 of the Refactoring Wizard dialog. The following icons indicate operation status conditions in the refactoring log.

✅	The operation or step completed successfully
⬛	The operation or step failed
❌	Indicates code lines that generated specific error messages returned by the database server during refactoring

Code Dependencies Analyzer

The Code Dependencies Analyzer is a powerful utility that can be used to quickly locate schema and code dependencies for objects of the following type:

- Tables

- Views

- Stored Procedures

- User-Defined Functions

- Table Columns

- View Columns

- Procedure Parameters

- Function Parameters

In other words, the utility can be used to find out all objects that depend on the object selected for the dependencies analysis. It is recommended to use the Code Dependencies Analyzer before dropping or modifying a schema object in the database so that you can review the impact of the pending change and avoid unnecessary application errors resulting from broken dependencies.

In a way, this utility is more sophisticated that many standard database dependencies viewers:

- It can find dependencies across databases.

- It can find dependencies for specific table and view columns.

- It can find dependencies for specific procedure and function parameters.

- It displays a list of found dependent objects and, for each object, displays precise locations of found code references within the dependent code.

- It works anywhere on the selected code; you don't need to browse objects in database navigators to pull their dependencies. You can click on an object, column, or parameter name referenced anywhere in the code to immediately invoke the Code Dependencies Analyzer.

To open the Code Dependencies Analyzer:

1. Position the edit caret on certain types of code symbols, such as an object name, column name or parameter name.

2. With the edit caret positioned over an appropriate code symbol, use the Ctrl+Shift+D hot key to open the Code Dependencies Analyzer. The Code Dependencies Analyzer determines the reference type of the symbol under the edit caret and starts the dependencies search operation. Alternatively, you can right-click a code symbol in the code window and select the **SQL Assistant → Refactoring → Show Dependencies…** command from the right-click menu. This will display the **Code Dependencies** dialog.

3. In the Code Dependencies dialog, choose the dependencies search method and options, then click the **Next** button to start the search.

4. Review the list of dependencies returned by the search. In the object tree-view displayed on the left side of the screen, click an object to view its code in the SQL edit window on the right side. Note that the code references found are underlined by wavy blue lines.

For ease of navigation, the navigation bar on the right displays color coded navigation marks. Each mark represents an occurrence in code of a reference to the selected object. Clicking on a blue navigation mark causes the editor to automatically scroll to the corresponding place in the code. Briefly resting your mouse pointer over a navigation mark, causes a small help window to appear that displays the line number where a reference to the selected object occurs.

5. After reviewing the list of dependencies, you can minimize the dialog and continue working with the editor or you can click the Close button to close the dialog.

Tip: The working of the Code Dependencies Analyzer and the types of results returned can be customized in SQL Assistant **Options** dialog using the options provided within the **Refactoring** tab.

Code References Analyzer

The Code References Analyzer utility is an opposite of the Code Dependencies Analyzer. It can be used to locate recursively all references an object depends on in order to work correctly. It is recommended to use the Code References Analyzer utility before code deployments an application upgrades to ensure that all related objects are deployed and available.

In a way, this utility is more sophisticated that many standard database dependencies viewers:

- It can find references across databases.

- It displays a list of referenced objects and, for each object, displays precise locations of the found code references within the dependent code.

- It provides a function to generate database deployment script capturing all referenced objects.

To open the Code References Analyzer:

1. Position the edit caret on certain types of code symbols, such as an object name.

2. With the edit caret positioned over an appropriate object name, right-click the name and select the **SQL Assistant → Refactoring → Show References…** command from the right-click menu. This will display the **Code References** dialog.

3. In the Code Dependencies dialog, choose additional objects whose code references you want to analyze, then click the **Next** button to start the search.

   Tip If you are working on a deployment script, select top level objects only.

4. Review the list of references returned by the search. In the object tree-view displayed on the left side of the screen, click an object to view its references in the SQL edit window on the right side. Note that the code references found are underlined by wavy blue lines.

For ease of navigation, the navigation bar on the right displays color coded navigation marks. Each mark represents an occurrence in code of a reference to the selected object. Clicking on a blue

navigation mark causes the editor to automatically scroll to the corresponding place in the code. Briefly resting your mouse pointer over a navigation mark, causes a small help window to appear that displays the line number where a reference to the selected object occurs.

For a graphical diagram view visualizing the code references, click the **Diagram** tab.
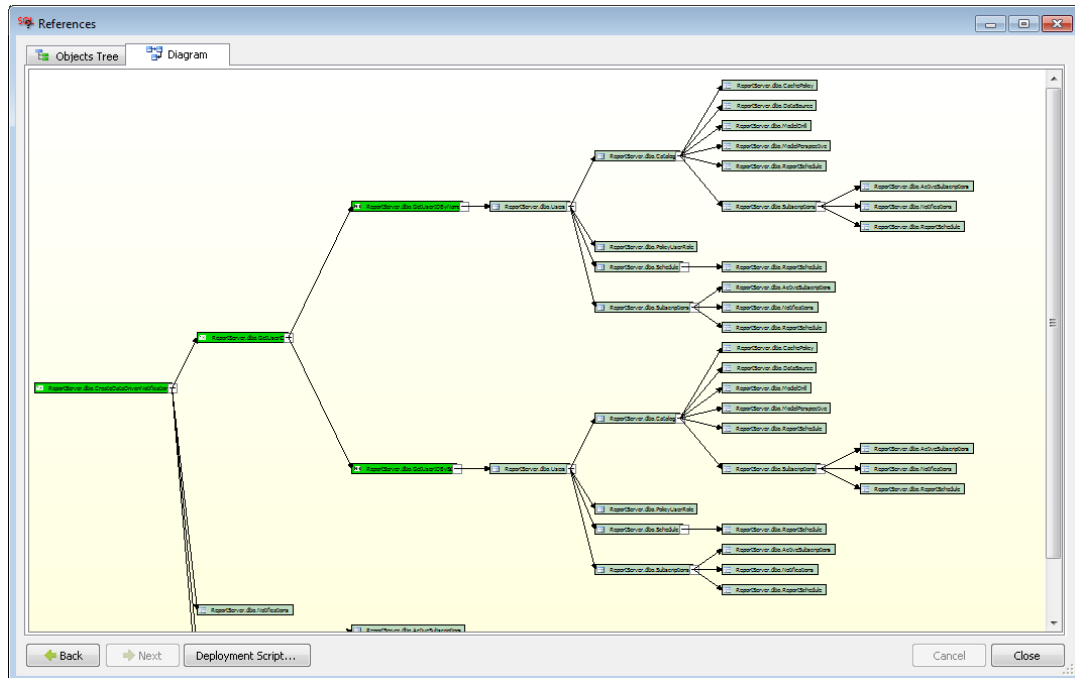
5. The **Deployment Script** button can be used to generate a deployment script for all selected objects.

6. After reviewing the list of references, you can minimize the dialog and continue working with the editor or you can click the Close button to close the dialog.

Tips:

- The working of the Code References Analyzer and the types of results returned can be customized in SQL Assistant **Options** dialog using the options provided within the **Refactoring** tab.

- To collapse and expand multiple levels, change display zoom and other options, use context specific the right-click menus available in different panels of the Code References Analyzer dialog.

Important Notes:

As of SQL Assistant version 13.0, the Code References Analyzer supports the following database systems only:

- SQL Server 2008 R2 and later

- SQL Azure 11 and later

- Oracle 11g and later

- PostgreSQL 9.2 and later

- DB2 UDB 9.5 and later.

An error will be returned for all other database types and versions.

Extract View

This refactoring method allows you to convert complex queries in a large program unit into a set of separate ,reusable database views. This method operates on the selected code in the editor window.

To use the Extract View method:

1. In your SQL editor, highlight the code of the query you want to use for the new database view. Make sure that the highlighted code begins with a valid SELECT statement.

2. Right-click on the highlighted code. In the right-click menu, choose the **SQL Assistant → Refactoring → Extract View** menu command. The **Refactoring – Extract View** dialog will appear. In this dialog, you will find a preview of the CREATE VIEW statement that will be used to create the new view.

3. Enter the view name in the **View Name** field. SQL Assistant automatically updates the CREATE VIEW code using the name entered in the View Name field

4. Click the **OK** button to create the new view. The CREATE VIEW statement will be executed in the database. The **Refactoring – Extract View** dialog will disappear, and the highlighted query in the SQL editor will be replaced by a SELECT statement from the new database view.

Tips:

*   The CREATE VIEW statement, including headers, comments and declaration, is based on the refactoring template **(view create)**. This template can be customized on the Code Refactoring tab in SQL Assistant Options dialog.

*   The code of the new SELECT statement, including column placement, comments, and so on, is based on the **(view call)** refactoring template. This template can be customized on the Code Refactoring tab in SQL Assistant Options dialog.

*   You can click the **Save As…** button on the **Refactoring – Extract View** dialog to save a copy of the CREATE VIEW statement in a SQL file. You can use that file for documentation purposes and also to add the new code to your code source control system.

Extract Procedure

This refactoring method allows you to extract a section of a large, complex SQL script and save it as a separate stored procedure (or function).

To use the Extract Procedure method:

1. In the SQL editor, highlight the block of code you want to convert to a separate stored procedure. The highlighted code can include any number of SQL statements.

2. Right-click on the highlighted code. From the right-click menu, choose the **SQL Assistant →
Refactoring → Extract Procedure** command. The **Refactoring – Extract Procedure** dialog will appear. This dialog displays a preview of the CREATE PROCEDURE statement that will be used for the new procedure. Note that in PostgreSQL based targets, the actual command is CREATE FUNCTION.

3. Enter the procedure name in the **Procedure Name** field. SQL Assistant automatically updates the CREATE PROCEDURE code using the name entered in the Procedure Name field

4. Click the **OK** button to create the new stored procedure. This will execute the CREATE PROCEDURE statement in the database (CREATE FUNCTION in PostgreSQL based targets). The **Refactoring –
Extract Procedure** dialog will disappear and the highlighted query in the SQL editor will be replaced by a SQL statement with the call to the new stored procedure.

Note: All variables that are referenced in the highlighted code but are declared outside of that code will be converted to stored procedure parameters. All variables declared within the highlighted code remain in the new procedure as local variables. You should review the proposed CREATE PROCEDURE code to verify it for correct parameters and variable use, and if necessary, to correct the code before executing it.

Tips:

- The CREATE PROCEDURE statement, including headers, comments and declaration, is based on the refactoring template **(procedure create)**. This template can be customized on the Code Refactoring tab in SQL Assistant Options dialog.

- The code of the new procedure call, including the call statement, parameter placement, comments, and so on, is based on the **(procedure call)** refactoring template. This template can be customized on the Code Refactoring tab in SQL Assistant Options dialog.

- You can click the **Save As…** button on the **Refactoring – Extract Procedure** dialog to save a copy of the CREATE PROCEDURE statement in a SQL file. You can use that file for documentation purposes and also to add the new code to your code source control system

Rename Table or View

This refactoring method provides an easy way to rename existing tables and views in your database and to automatically find and correct dependent database code, such as dependent tables, views, procedures, functions, triggers and so on. It works on the selected object name.

To use this refactoring method:

1. In the SQL editor's code window, right-click the name of the table you want to rename. In the right-click menu, choose the **SQL Assistant → Refactoring → Rename Table/View…** command. The **Refactoring – Rename Table/View** dialog will appear.
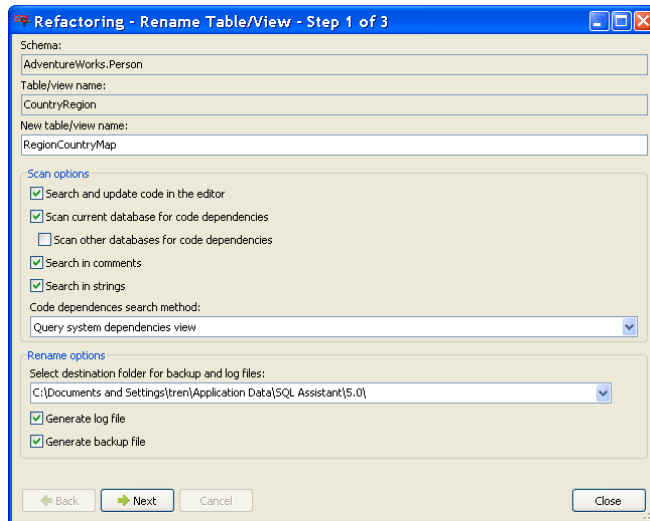
2. In the **New table/view name** field, enter the new table or view name.

3. In the **Scan options** box, select appropriate code dependency search options.

4. In the **Rename options** box, choose logging and backup options for the rename operation.

5. Click the **Next** button to advance to the next step.

6. Review pending changes. In the object tree-view displayed on the left hand side of the screen, click individual objects. The pending code changes will appear in the SQL edit window on the right side of the dialog. Note that the references to the old object name are displayed using strike-through gray color font. The new name references are displayed next to old references but using black bold font.

    If you find an object you don't want to change, deselect the check box in front of the object name in the object tree window.

    To assist you in navigating through the code, a navigation status bar with color coded navigation marks is displayed on the right side of the code window. See the Refactoring Wizard Dialog topic in this chapter for details on the supported types of marks, their colors, and use of other visual elements.

7. If you are satisfied with the proposed changes, click the **Refactor** button to begin the renaming

operation. This will advance the dialog to the next step, update code in the editor, and execute the DDL statements required to update the database code and objects. Progress and status messages will be logged to the screen.

8. Review all logged status messages to ensure that no errors occurred during refactoring.

9. Click the **Close** button to close the dialog.

Important Notes:

- SQL Assistant performs intelligent rename operations. When renaming tables, SQL Assistant uses SQL commands that allow the database server to automatically update all dependent referential constraints and indexes, and to ensure that dependent tables remain intact. For dependent views and procedural objects, SQL Assistant automatically updates their code using SQL ALTER commands whenever possible. In the case where ALTER commands are not available, SQL Assistant uses DROP and CREATE statements to recreate the object. It automatically executes GRANT commands to restore object-level privileges dropped along with the old object.

- SQL Assistant does not support updating code dependencies in encrypted code objects including, but not limited to, encrypted stored procedures in SQL Server or wrapped procedural objects in Oracle.

Rename Table or View Column

This refactoring method provides an easy way to rename columns in existing tables and views in your database and to automatically find and correct dependent database code, such as dependent tables, views, procedures, functions, triggers and so on. It works on the selected column name.

To use this refactoring method:

1. In the SQL editor code window, right-click the column name you want to rename. In the right-click menu, choose the **SQL Assistant → Refactoring → Rename Column…** command. The **Refactoring – Rename Column** dialog will appear.
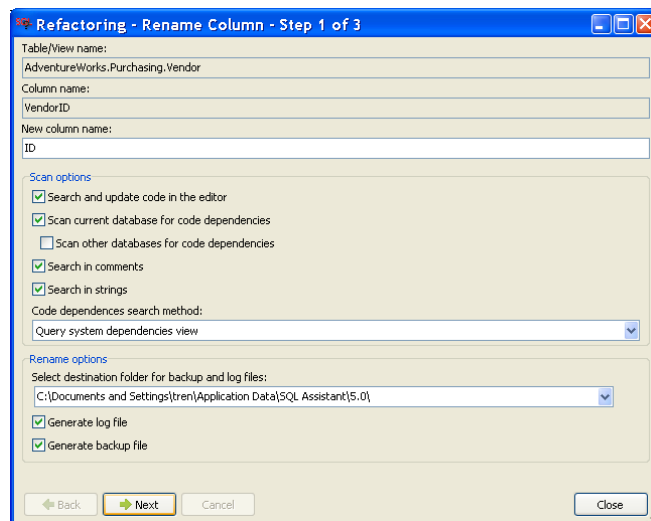
2.   In the **New column name** field, enter the new name.

3.   In the **Scan options** box, choose appropriate code dependency search options.

4.   In the **Rename options** box, choose logging and backup options for the rename operation.

5.   Click the **Next** button to advance to the next step.
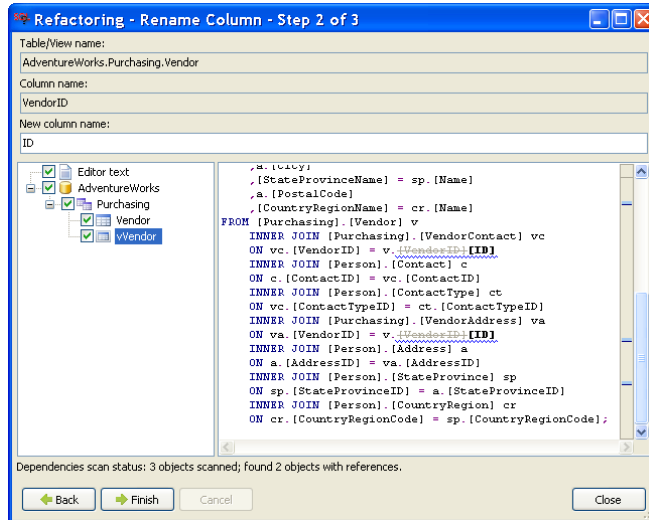
6.   Review pending changes. In the object tree-view displayed on the left hand side of the screen, click individual objects. The pending code changes will appear in the SQL edit window on the right side of the dialog. Note that the references to the old column name are displayed using strike-through gray color font. The new column name references are displayed next to old references but using black bold font.

If you find an object you don't want to change, deselect the check box in front of the object's name in the object tree window.

To assist you in navigating through the code, a navigation status bar with color coded navigation marks is displayed on the right side of the code window. See the Refactoring Wizard Dialog topic in this chapter for details on the supported types of marks, their colors, and use of other visual elements.

7.   If you are satisfied with the proposed changes, click the **Refactor** button to begin the rename operation. This will advance the dialog to the next step, update code in the editor and execute the DDL statements required to update the database code and objects. Progress and status messages will be logged to the screen.

8.   Review all logged status messages to ensure that no errors occurred during refactoring.

9.   Click the **Close** button to close the dialog.

Important Notes:

- SQL Assistant uses intelligent rename operations. For example, when renaming columns, SQL Assistant uses SQL commands that allow the database server to automatically update all dependent referential constraints and indexes and ensure that dependent objects remain intact.

  However, not all database types and versions support intelligent rename operations. If your database does not support them, you may need to manually drop and recreate the affected table indexes and/or

constraints.

- SQL Assistant does not support search and update operations for dependencies in encrypted code objects including, but not limited to, encrypted stored procedures in SQL Server or wrapped procedural objects in Oracle.

Rename Procedure or Function

This refactoring method provides an easy way to rename existing stored procedures and user-defined functions and to automatically find and correct dependent database code such as dependent views, procedures, functions, triggers, and so on. It works on a selected object name.

To use this refactoring method:

1. In the SQL editor code window, right-click the procedure or function name you want to rename. In the right-click menu, choose the **SQL Assistant → Refactoring → Rename Procedure/Function…** command. The **Refactoring – Rename Procedure/Function** dialog will appear.

2. In the **New procedure/function name** field, enter the new name.

3. In the **Scan options** box, choose appropriate search options for code dependencies.

4. In the **Rename options** box, choose logging and backup options for the rename operation.

5. Click the **Next** button to advance to the next step.

6. Review pending changes. In the object tree-view displayed on the left hand side of the screen, click individual objects. The pending code changes will appear in the SQL edit window on the right side of the dialog. Note that the references to the old object name are displayed using strike-through gray color font. The new name references are displayed next to old references but using black bold font.

   If you find an object that you don't want to change, in the object tree-view uncheck the check box displayed in front of the object name.

   For easy of navigation, use the left side object tree and right side status bar with color coded navigation marks. See the Refactoring Wizard Dialog topic in this chapter for details on the supported types of marks, their colors, and use of other visual elements.

7. If you are satisfied with the proposed changes, click the **Refactor** button to begin the renaming operation. This will advance the dialog to the next step, update code in the editor and execute the DDL statements required to update the database code and objects. Progress and status messages will be logged to the screen.

8. Review all logged status messages to ensure that no errors occurred during refactoring.

9. Click the **Close** button to close the dialog.

 Important Notes:

- SQL Assistant uses intelligent procedure and function renaming operations if the database server supports them. However, not all servers support this feature. In the case where intelligent renaming of procedures and functions is not supported, SQL Assistant uses DROP and CREATE commands to

recreate the same procedure with a different name. It automatically executes GRANT commands to restore object-level privileges dropped along with the old object.

- SQL Assistant currently does not support the renaming of user-defined functions referenced in computed table columns, function-based indexes, and packages.

- SQL Assistant does not support search and update operations for dependencies in encrypted code objects including, but not limited to, encrypted stored procedures in SQL Server or wrapped procedural objects in Oracle.

Rename Procedure or Function Parameter

This refactoring method allows you to rename parameters declared in existing stored procedures and functions and to automatically find and correct dependent database code such as dependent views, procedures, functions, triggers, and so on. It works on the selected object name.

To use this refactoring method:

1. In the SQL editor code window, right-click the column name you want to rename. In the right-click menu, choose the **SQL Assistant → Refactoring → Rename Parameter…** command. The **Refactoring – Rename Parameter** dialog will appear.

2. In the **New parameter name** field, enter the new parameter name.

3. In the **Scan options** box, choose appropriate search options for code dependencies.

4. In the **Rename options** box, choose logging and backup options for the rename operation.

5. Click the **Next** button to advance to the next step.

6. Review pending changes. In the object tree in the left window, check each of the individual objects by clicking on an object name and reviewing the object's code in the right code window. Note that references to the original column name are displayed using a gray, strike-through font. The new column names are displayed next to the original ones in black, boldfaced font.

   If you find an object you don't want to change, deselect the check box in front of the object's name in the object tree window.

   To assist you in navigating through the code, a navigation status bar with color coded navigation marks is displayed on the right side of the code window. See the Refactoring Wizard Dialog topic in this chapter for details on the supported types of marks, their colors, and use of other visual elements.

   If you are satisfied with the proposed changes, click the **Refactor** button to begin the renaming operation. This will advance the dialog to the next step, update code in the editor and execute the DDL statements required to update the database code and objects. Progress and status messages will be logged to the screen.

7. Review all logged status messages to ensure that no errors occurred during refactoring.

8. Click the **Close** button to close the dialog.

Important Notes:

- SQL Assistant uses ALTER commands to update the affected code.

- SQL Assistant does not support search and update operations for dependencies in encrypted code objects including, but not limited to encrypted stored procedures in SQL Server or wrapped procedural objects in Oracle.

Rename Local Variable

This refactoring method provides an easy way to safely rename variables declared within bodies of procedures and functions in the SQL Editor. Using this method is safer and better than using text search and replace operations because it affects only the body of the current procedure or function and because it renames only the variable references, avoiding changes to any other text containing the same sub-strings.

To use this refactoring method:

1. In the SQL editor code window, right-click the variable name you want to rename. This could be any reference to the variable anywhere in the code. In the right-click menu, choose the **SQL Assistant → Refactoring → Rename Local Variable…** command. The **Refactoring – Rename Local Variable** dialog will appear.

2. Enter the new name.

3. Click the OK button box to perform the refactoring.

Add Table Column

This refactoring method provides an easy way to add new columns to existing tables in your database and to automatically find and correct dependent database code such as dependent views, procedures, functions, triggers and so on. It works on the selected table name.

To use this refactoring method:

1. In the SQL editor code window, right-click the name of the table you want to modify. In the right-click menu, choose the **SQL Assistant → Refactoring → Add Column…** command. The **Refactoring – Add Column** dialog will appear.
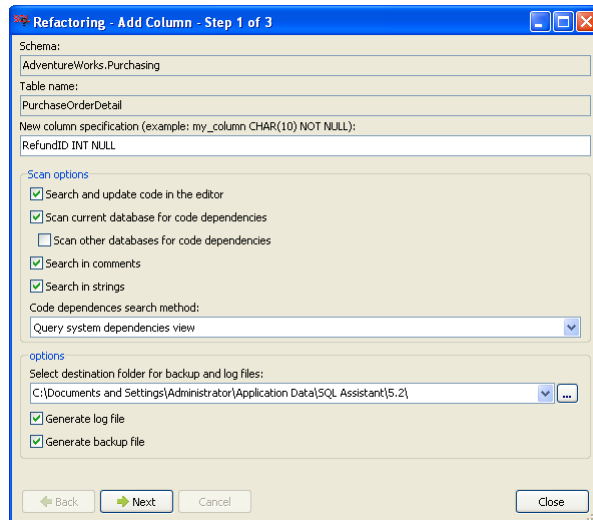
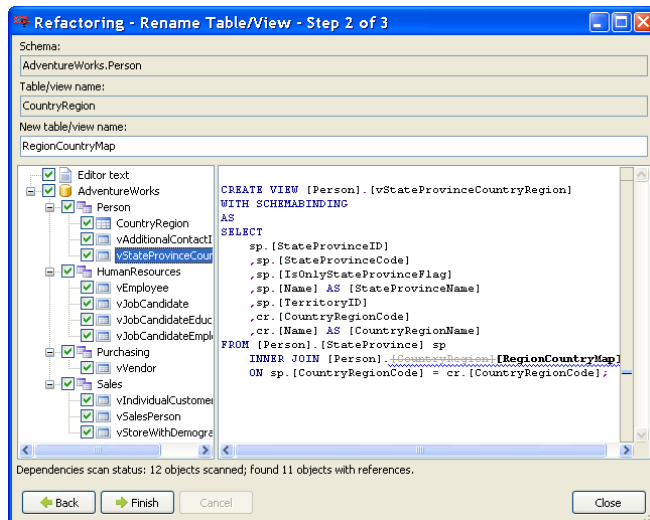2. Enter a new column specification in the **New column specification** field. The specification must include the column name and data type. Other elements such as constraints, defaults, and so on are optional. If the new column name contains spaces or other special symbols, the name must be enclosed in name delimiters supported by your database. Examples:
   ```
   IsCriticalError  BIT  NOT NULL  DEFAULT 0
   [Alert Email]  VARCAHR(50)
   `size_column`  ENUM('small', 'medium', 'large')  DEFAULT  'small'
   ```

3. In the **Scan options** box, choose appropriate search options for code dependencies.

4. In the **Options** box, choose logging and backup options for the rename operation.

5. Click the **Next** button to advance to the next step.

6. Review all references returned. In the object tree-view displayed on the left hand side of the screen, click individual objects to review their specific changes and code references. The found code references will appear in the SQL edit window on the right side of the dialog.

   Important Notes: Check all locations where the refactored table is referenced and verify that all INSERT statements for that table reference specific column names (the code is not like INSERT INTO [table] SELECT and also do not using wildcards in place of specific column names, for example. INSERT INTO …SELECT *). The refactoring operations will likely succeed for that kind of code, but the code might fail in the run-time.

   If you find an object you don't want to change, deselect the check box in front of the object's name in the object tree window.

   To assist you in navigating through the code, a navigation status bar with color coded navigation marks is displayed on the right side of the code window. See the Refactoring Wizard Dialog topic in this chapter for details on the supported types of marks, their colors, and use of other visual elements.

7. If you are satisfied with the proposed changes, click the **Refactor** button to begin the renaming operation. This will advance the dialog to the next step, update code in the editor and execute the DDL statements required to update the database code and objects. Progress and status messages will be

logged to the screen.

8. Review all logged status messages to ensure that no errors occurred during refactoring.

9. Click the **Close** button to close the dialog.

Important Notes:

- SQL Assistant does not support search and update operations for dependencies in encrypted code objects including, but not limited to encrypted stored procedures in SQL Server or wrapped procedural objects in Oracle.

Drop Table Column

This refactoring method provides an easy way to drop existing columns from database tables and to automatically find and correct dependent database code such as dependent views, procedures, functions, triggers, and so on. It works on the selected column name. Whenever possible SQL Assistant will simply remove the dropped column by altering its base table and in the dependent code it will substitute all references to the dropped column with NULL values.

The following example illustrates the changes in the dependent code:

Before:

```
CREATE VIEW ActiveFilteredAccounts AS
SELECT ae.AccountID, ae.ApplicationOwner, ae.AccountType
FROM AccountEntityTable ae
   JOIN AccountFilterTable af
   ON ae.AccountID = af.AccountID
WHERE ae.[Active] = 1
ORDER BY ae.AccountID
```

If the ApplicationOwner column in table AccountEntityTable is selected to be dropped, the following changes will be made in the dependent view ActiveFilteredAccounts:

After:

```
CREATE VIEW ActiveFilteredAccounts AS
SELECT ae.AccountID, NULL AS ApplicationOwner, ae.AccountType
FROM AccountEntityTable ae
   JOIN AccountFilterTable af
   ON ae.AccountID = af.AccountID
WHERE ae.[Active] = 1
ORDER BY ae.AccountID
```

Here, in the dependent view, the dropped column name is replaced with a NULL expression so that the view code remains intact and so all of the view dependencies including dependent application code, dependent stored procedures, etc.. can continue running..

To use the Drop Table Column refactoring method:

1. In the SQL editor code window, right-click the column name you want to rename. In the right-click menu, choose the **SQL Assistant → Refactoring → Drop Column…** command. The **Refactoring – Drop Column** dialog will appear.

2. In the **Scan options** box, choose appropriate search options for code dependencies.

3. In the **Options** box, choose logging and backup options for the pending rename operation.

4. Click the **Next** button to advance to the next step.

    Review all returned references. In the object tree in the left window, check each of the individual objects by clicking on an object name and reviewing the object's code in the right code window.

    Important Notes: Check all locations where the refactored table is referenced. Verify that all highlighted SQL statements no longer refer to the dropped column name.
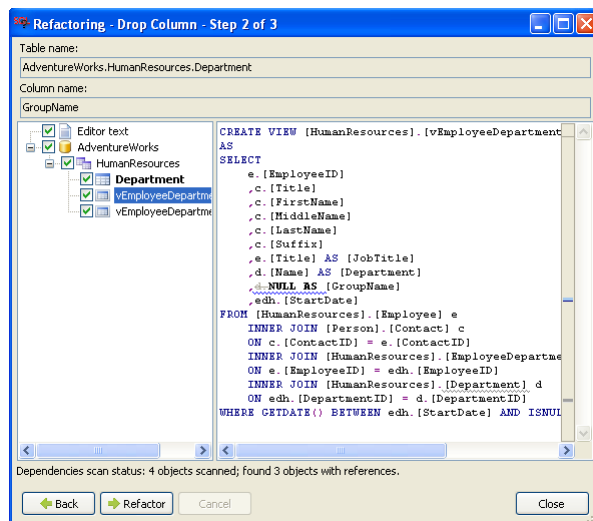
    If you find an object that you don't want to change, in the object tree-view uncheck the check box displayed in front of the object name.

    For easy of navigation, use the left side object tree and right side status bar with color coded navigation marks. See the Refactoring Wizard Dialog topic in this chapter for details on the supported types of marks, their colors, and use of other visual elements.

5. If you are satisfied with the proposed changes, click the **Refactor** button to begin the refactoring operation. This will advance the dialog to the next step, update code in the editor and execute DDL statements required for updating the database code and objects. The progress of work and status messages will be logged to the screen.

6. Review all logged status messages to ensure that no errors occurred during refactoring.

7. Click the **Close** button to close the dialog.

Important Notes:

- SQL Assistant does not support search and update operations for dependencies in encrypted code objects including, but not limited to, encrypted stored procedures in SQL Server or wrapped procedural objects in Oracle.

Add Procedure or Function Parameter

This refactoring method provides an easy way to add new parameters to existing procedures and functions in your database and to automatically find and correct dependent database code such as dependent views, procedures, functions, triggers, and so on. It works on the selected procedure or function name.

To use this refactoring method:

1. In the SQL editor code window, right-click the name of the procedure or function you want to modify. In the right-click menu, choose the **SQL Assistant → Refactoring → Add Parameter…** command. The **Refactoring – Add Parameter** dialog will appear.
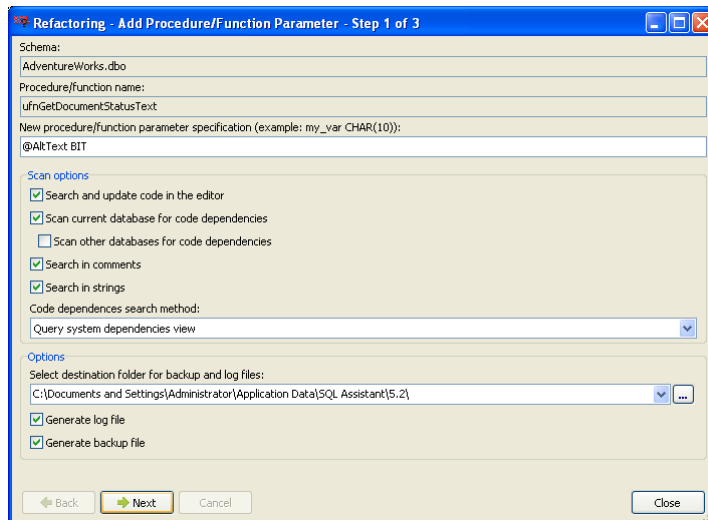
2. Enter a new parameter specification into the **New parameter specification** field. The specification must contain the parameter name and data type. Other elements such as default values, IN/OUT modifiers, and so on are optional. Entry of parameter name prefix @ is optional. SQL Assistant will add it automatically if the prefix is not specified. Examples:
    ```
    @IsCriticalError  BIT
    AlertEmail  VARCAHR(50)
    size_column  INT = 1
    ```

3. In the **Scan options** box, choose appropriate search options for code dependencies.

4. In the **Options** box, choose logging and backup options for the pending rename operation.

5. Click the **Next** button to advance to the next step.
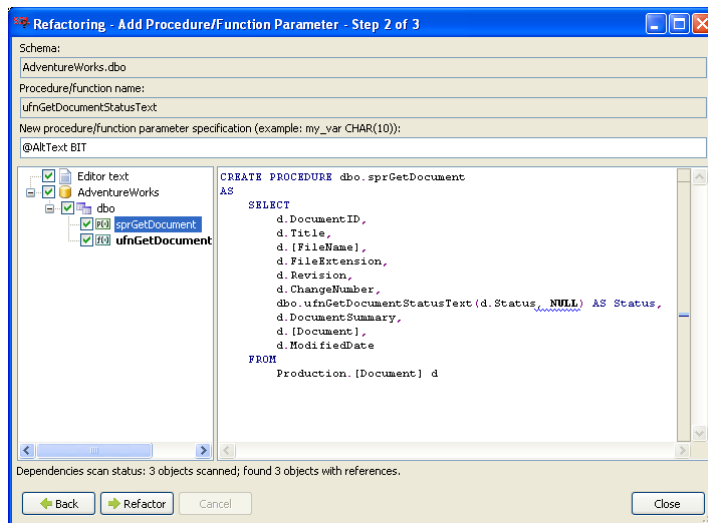
6. Review all returned references. In the object tree in the left window, check each of the individual objects by clicking on an object name and reviewing the object's code in the right code window.

   Important Notes: SQL Assistant takes special care to update object references, adding where necessary a NULL placeholder for the new parameter in all function and procedure calls. This method should work for all input parameters, but may fail for output parameters if your database engine does not support optional output parameters. Make sure to review the pending changes and validate them for usability.

   If you find an object you don't want to change, deselect the check box in front of the object's name in the object tree window.

   To assist you in navigating through the code, a navigation status bar with color coded navigation marks is displayed on the right side of the code window. See the Refactoring Wizard Dialog topic in this chapter for details on the supported types of marks, their colors, and use of other visual elements.

7. If you are satisfied with the proposed changes, click the **Refactor** button to begin the renaming operation. This will advance the dialog to the next step, update code in the editor and execute the DDL statements required to update the database code and objects. Progress and status messages will be logged to the screen.

8. Review all logged status messages to ensure no errors occurred during refactoring.

9. Click the **Close** button to close the dialog.

Important Notes:

- SQL Assistant does not support search and update operations for dependencies in encrypted code objects including, but not limited to, encrypted stored procedures in SQL Server or wrapped procedural objects in Oracle.

Drop Procedure or Function Parameter

This refactoring method provides an easy way to remove parameters from existing procedures and functions in your database and to automatically find and correct the dependent database code such as dependent views, procedures, functions, triggers and so on. It works on the selected parameter name.

To use this refactoring method:

1. In the SQL editor code window, right-click name of the parameter that you want to drop. The parameter may be in a procedure or function call or in a DDL statement. In the right-click menu, choose the **SQL Assistant → Refactoring → Drop Parameter…** command. The **Refactoring – Drop Parameter** dialog will appear.

2. In the **Scan options** box, choose appropriate search options for code dependencies.

3. In the **Options** box, choose logging and backup options for the pending rename operation.

4. Click the **Next** button to advance to the next step.
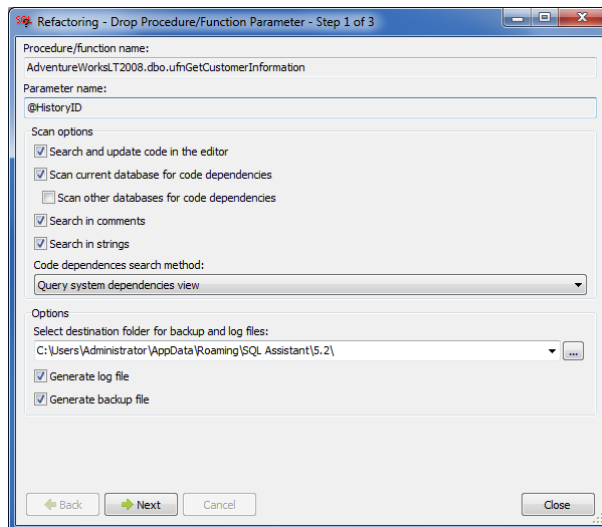
5. Review all returned references. In the object tree in the left window, check each of the individual objects by clicking on an object name and reviewing the object's code in the right code window.
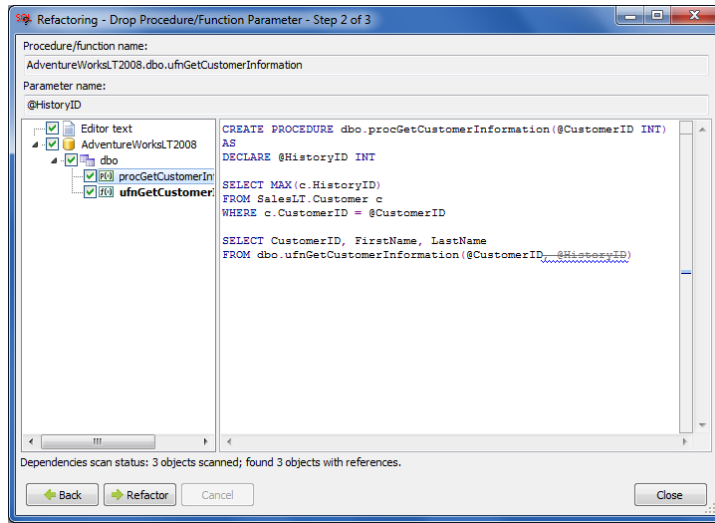
Important Notes: SQL Assistant takes special care of updating parameter references in procedure and function calls. It automatically removes values and expressions for the removed parameters. However, it does not change the business logic of the procedural code. If the logic depends on the values of the parameters, especially in case of the output parameters, the refactoring operation will succeed, but the code will fail in run-time because no variables will not be populated with the expected values. Make sure to review the pending changes and validate them for usability.

If you find an object that you don't want to change, in the object tree-view uncheck the check box displayed in front of the object name.

To assist you in navigating through the code, a navigation status bar with color coded navigation marks is displayed on the right side of the code window. See the Refactoring Wizard Dialog topic in this chapter for details on the supported types of marks, their colors, and use of other visual elements.

6. If you are satisfied with the proposed changes, click the **Refactor** button to begin the renaming operation. This will advance the dialog to the next step, update code in the editor and execute the DDL statements required to update the database code and objects. Progress and status messages will be logged to the screen.

7. Review all logged status messages to ensure that no errors occurred during refactoring.

8. Click the **Close** button to close the dialog.

Important Notes:

- SQL Assistant does not support search and update operations for dependencies in encrypted code objects including, but not limited to, encrypted stored procedures in SQL Server or wrapped procedural objects in Oracle.

Drop Procedure or Function

This refactoring method provides an easy and safe way to drop existing procedures and functions in your database. It automatically finds and highlights dependent views, procedures, functions, triggers, and so on, enabling you to review and, if necessary, modify all dependent procedural objects in a single place. It works on the selected procedure or function name.

To use this refactoring method:

1. In your SQL editor code window, right-click the name of the procedure or function you want to drop. In the right-click menu, choose the **SQL Assistant → Refactoring → Drop Procedure/Function…** command. The **Refactoring – Drop Procedure/Function** dialog will appear.

2. In the **Scan options** box, choose appropriate search options for code dependencies.

3. In the **Options** box, choose logging and backup options for the pending rename operation.

4. Click the **Next** button to advance to the next step.

5. Review all returned references. In the object tree in the left window, check each of the individual objects by clicking on an object name and reviewing the object's code in the right code window.

Important Notes: If SQL Assistant finds one or more objects that are dependent on the procedure or function you want to drop, it displays their icons with an explanation point and highlights their references with red, wavy lines. The dependent code cannot be changed automatically, as SQL Assistant does not know how to modify the business logic of the dependent code. You will need to edit the code manually and make any needed corrections. Alternatively, you may cancel the pending changes and close the Refactoring Wizard dialog using the Close button.

If you find an object you don't want to change, deselect the check box in front of the object's name in the object tree window.

To assist you in navigating through the code, a navigation status bar with color coded navigation marks is displayed on the right side of the code window. See the "Refactoring Wizard Dialog" topic in this chapter for details on the supported types of marks, their colors, and use of other visual elements.

6. If you are satisfied with the proposed changes, click the **Refactor** button to begin the refactoring operation. This will advance the dialog to the next step, update code in the editor and execute DDL statements required for updating the database code and objects. The progress of work and status messages will be logged to the screen.

7. Review all logged status messages to ensure that no errors occurred during refactoring.

8. Click the **Close** button to close the dialog.

Important Notes:

- SQL Assistant does not support search and update operations for dependencies in encrypted code objects including, but not limited to, encrypted stored procedures in SQL Server or wrapped procedural objects in Oracle.

Drop Table or View

This refactoring method provides an easy and safe way to drop existing tables and views in your database. It automatically finds and highlights dependent views, procedures, functions, triggers, and so on, enabling you to review and, if necessary, modify all dependent procedural objects in a single place. It works on the selected table or view name.

To use this refactoring method:

1. In the SQL editor code window right-click name of the table or view that you want to drop. In the right-click menu, choose the **SQL Assistant → Refactoring → Drop Table/View…** command. The **Refactoring – Drop Table/View** dialog will appear.

2. In the **Scan options** box, choose appropriate search options for code dependencies.

3. In the **Options** box, choose logging and backup options for the pending rename operation.

4. Click the **Next** button to advance to the next step.
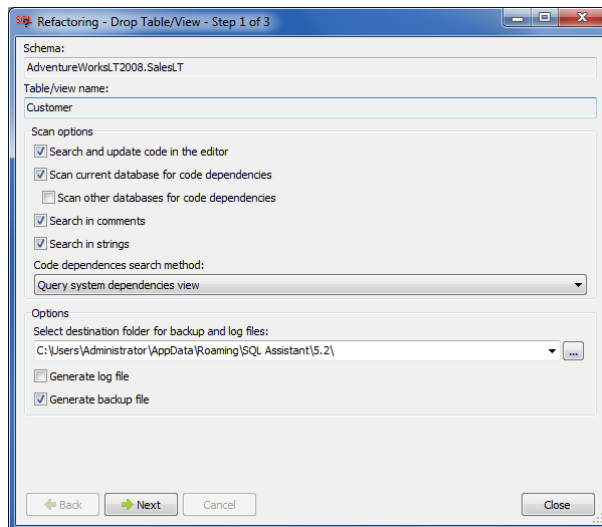
5. Review all references returned. In the object tree in the left window, check each of the individual objects by clicking on an object name and reviewing the object's code in the right code window.
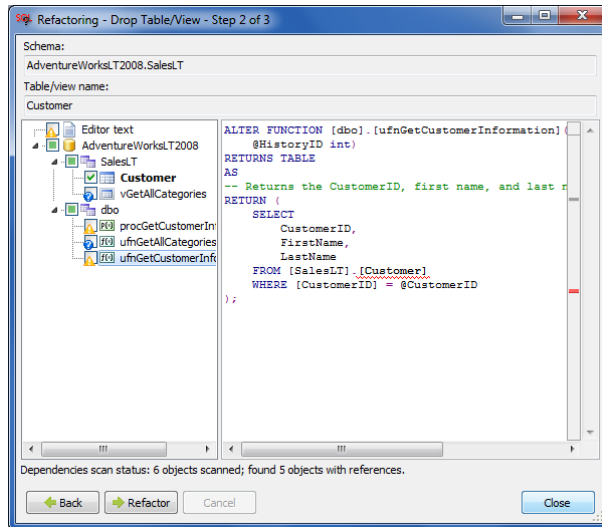
Important Notes: If SQL Assistant finds one or more objects that are dependent on the table or view you want to drop, it displays their icons with an explanation point and highlights their references with red, wavy lines. The dependent code cannot be changed automatically as SQL Assistant does not know how to modify the business logic of the dependent code. You must edit the code manually and make any needed corrections. Alternatively, you can cancel all pending changes and close the Refactoring Wizard dialog using the Close button.

If you find an object you don't want to change, deselect the check box in front of the object's name in the object tree window.

To assist you in navigating through the code, a navigation status bar with color coded navigation marks is displayed on the right side of the code window. See the Refactoring Wizard Dialog topic in this chapter for details on the supported types of marks, their colors, and use of other visual elements.

6. If you are satisfied with the proposed changes, click the **Refactor** button to begin the renaming operation. This will advance the dialog to the next step, update code in the editor and execute the DDL statements required to update the database code and objects. Progress and status messages will be logged to the screen.

7. Review all logged status messages to ensure that no errors occurred during refactoring.

8. Click the **Close** button to close the dialog.

Important Notes:

- SQL Assistant does not support search and update operations for dependencies in encrypted code objects including, but not limited to, encrypted stored procedures in SQL Server or wrapped procedural objects in Oracle.

Qualify Object Names

This simple refactoring method enables you to modify a SQL script so that all object names are qualified with the schema name in the format, schema_name.object_name. For SQL Server and Sybase ASE, this method also supports full name qualifications including the database name in the format, db_name.schema_name.object_name.

The following example illustrates the changes

Before:
```
SELECT ae.AccountID, ae.ApplicationOwner, ae.AccountType
FROM AccountEntityView ae
    JOIN AccountFilterTable af
    ON ae.AccountID = af.AccountID
WHERE ae.[Active] = 1
    AND af.EnumName = 'balance'
ORDER BY ae.AccountID
```

After:
```
SELECT ae.AccountID, ae.ApplicationOwner, ae.AccountType
FROM SampleServiceCompany2008.dbo.AccountEntityView ae
    JOIN SampleServiceCompany2008.dbo.AccountFilterTable af
    ON ae.AccountID = af.AccountID
WHERE ae.[Active] = 1
    AND af.EnumName = 'balance'
ORDER BY ae.AccountID
```

Important Notes:

The **Always Fully Qualify Object Names** option controls how object names are qualified by this refactoring method. If the option value is one of the **With schema name…** values, only the schema name is added to object names. If the option value is **\With database and schema names,** both database name and schema name are added to object names.

For more information on changing the **Always Fully Qualify Object Names** option, see the Customizing Code Auto-completion Options topic in CHAPTER 48

To use Qualify Object Names refactoring method:

1. In the SQL editor, right-click and choose the **SQL Assistant → Refactoring → Qualify Object Names…** command. SQL Assistant will analyze the code in the editor and display the **Text Compare** dialog, showing the code before and after the changes.
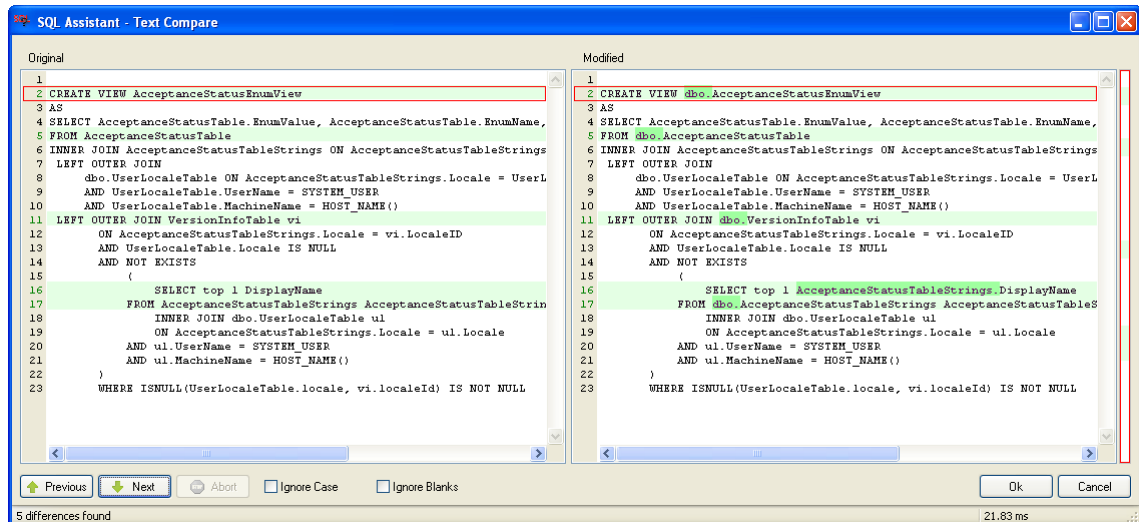
Color legend: Text lines with light green background indicate lines updated using Qualify Object Names refactoring. Sections of text highlighted with bright green background indicate the actual changes. The change status bar on the right represents change map for the entire text.

2. Review the proposed changes. If you are satisfied with the results, click the OK button.

Note: changes are applied to the code in the target SQL editor only .Use the code execution facility in the target editor, or SQL Assistant's code execution facility, to apply the changes to your database.

Reformat and Beautify Database Code

Formatting code of an existing database objects, such as view or procedural objects, can be done in four simple steps:

1. Extract the DDL code of an existing database object using the Procedural Code View methods described in CHAPTER 10, One-click DDL Code View. Copy the extracted code into the SQL editor.

2. Reformat the extracted code using the methods described in CHAPTER 5, Code Formatter and Beautifier.

3. In the code, replace the first CREATE keyword with the ALTER keyword (or CREATE OR REPLACE keyword group in Oracle and PostgreSQL).

4. Execute the updated code to modify it in the database. You can use your SQL editor's code execution facility, if available, or you can use SQL Assistant's code execution facility. See CHAPTER 14, "Executing SQL Scripts for more details.

Batch formatting of multiple database objects can be done in five steps:

1. Extract to flat text files the DDL code that defines the database objects. You can use methods available in your SQL development environment or database administration tool. For example, in SQL Server Management Studio, you can use the right-click menu in the Object Explorer for a database and then choose the Tasks/Generate Scripts... menu to reverse-engineer existing objects and save the generated SQL code. Similarly, in DB Tools for Oracle, you can use the Tools/Reverse-engineer Schema menu. Similar methods are available in other tools.

2. Open the generated script in SQL Editor.

3. Reformat the extracted code using the methods described in CHAPTER 5, Code Formatter and Beautifier.

4. Use the editor's search and replace function to replace in the code every occurrence of the CREATE PROCEDURE, CREATE FUNCTION and similar commands with their ALTER PROCEDURE, ALTER FUNCTION equivalents (replace with CREATE OR REPLACE in Oracle and PostgreSQL).

5. Execute the updated code to modify the database objects. You can use your SQL editor's code execution facility, if available, or you can use SQL Assistant's code execution facility. See CHAPTER 14, Executing SQL Scripts for more details.

CHAPTER 9, Interactive SQL Reference System

Overview

SQL Assistant features a handy interactive SQL reference system. Different SQL Reference versions are provided for different versions of Oracle, SQL Server, DB2, MySQL, and PostgreSQL database servers so that you can lookup and interactively build SQL commands fully compatible with your database server type and version.

The SQL Reference consists of two main parts:

1. Table of Contents, Statements and Keywords Indexes

2. Visual SQL Command Builder and Lookup
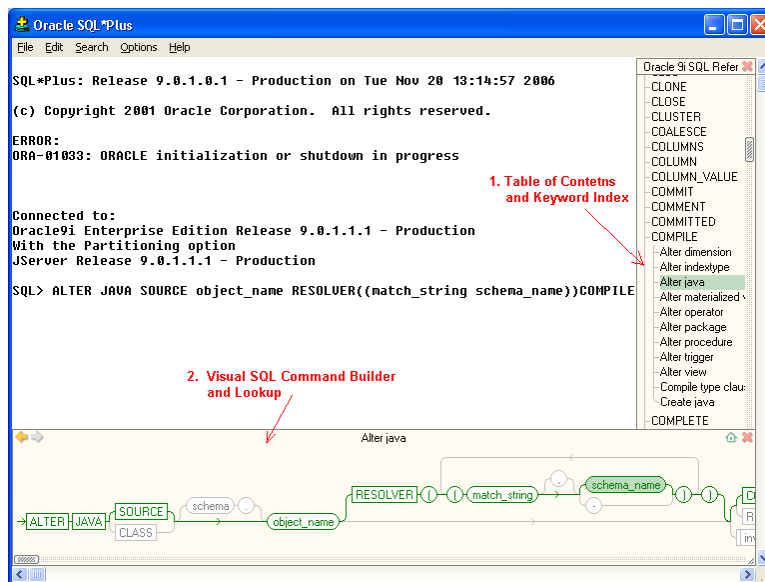
The following topics in this chapter describe how to open and use different parts of the SQL Reference system.

Invoking the SQL Reference System

SQL Reference can be invoked at any time using the keyboard hot key assigned to it. The default hot key is Ctrl+F1. If this key is already reserved for some function in your SQL editor or development environment, you can assign a different hot key in SQL Assistant options. For more information, see the Customizing Hot Keys topic in CHAPTER 48.

SQL Reference can be also opened using SQL Assistant's menu available in the system tray or from menus available in the target editor. To use the target editor's menus, the menu integration option must be enabled. For details, see the Manually Invoking SQL Assistant Popups topic in CHAPTER 3.

Using the SQL Reference Index and Table of Contents

Depending on which editor is used, the SQL Reference Table of Contents may appear on the left or right hand side of the target editor window. The visual SQL Command Builder may appear at the bottom of the editor window or as a popup below the editing caret.

To display the SQL Reference Index and Table of Contents

1. Press the default Ctrl+F1 hotkey.

2. If Step 1 causes only a context sensitive SQL Reference topic popup to appear on the screen, click the Home icon 🏠 in the top-right corner of the topic popup.

To select a specific SQL command, function or topic:

1. In the SQL Reference Table of Contents, click hyperlink for the topic you want to view. If that topic contains subtopics, a list of hyperlink subtopics will appear under the selected topic name.

2. In the Subtopics list, select the required command or function. Visual SQL Command Builder window will appear at the bottom of the editor screen. You can use this window to review the syntax or you can use it to interactively build the required command and paste it into the editor

Tip: If you are not sure which topic or subtopic contains the information you are looking for, you can use either the Statements or the Keywords Index to quickly determine the correct topic. These Indexes are available as the last two items below the Table of Contents.

Persisting SQL Reference Table of Contents

By default, the **SQL Reference Table of Contents** window is not persistent and does not display automatically in target editor's code windows. You can manually open the Table of Contents by following the steps described in the Invoking SQL Reference System topic.

If you want the **SQL Reference Table of Contents** window appear automatically in every code editor window, click the *pushpin* icon 📌 in the right-top corner of the **SQL Reference Table of Contents** window. This will make the **Table of Contents** pane persistent in the current target editor and all future instances of the same editor type and its code windows.

To disable the **Table of Contents** persistence, click the *pushpin* icon again. Note that the persistence state is indicated by the toggled state of the *pushpin* icon.

Searching Contents

To quickly find stuff in **SQL Reference**:

1. Open SQL Reference window or SQL Command Syntax window. For example, you can use

Ctrl+F1 hot key or use available SQL Assistant commands in your editor's right-click menu.

2.   Click the *find* icon 🔍 in the right-top corner of the **SQL Reference Table of Contents** or click the *find* icon in the top-right corner of the **SQL Command Syntax** window.

3.   Type the text you want to find. SQL Assistant will automatically scroll the **Table of Contents** and select the first topic or subtopic containing this text. In case the text is found in a second-level or third-level subtopic, it will automatically expand the entire patch topic and select the appropriate subtopic.

4.   Click on one of the available matches to have the associated topic or subtopic to appear in **SQL Command Syntax** window.

Resizing Table of Contents

To resize the **SQL Reference Table of Contents** pane, drag the vertical bar separating the pane and the code editor. Note that when you place mouse pointer over the edge of the **Bird's Eye View** pane the cursor shape changes to resize shape as on the following screenshot.

Make sure the cursor takes the right shape before dragging the pane edge.

Using SQL Command Syntax and Functions Lookup

To quickly check syntax of a known SQL command:

Method 1

1.   Pres Ctrl+F1 to open the SQL Reference system.

2.   Click the **Statement Index** hyperlink. The Statement Index will expand.

3.   In the expanded Index locate and click the command whose syntax you want to lookup. Visual SQL Command Builder and Lookup windows will appear at the bottom of the editor screen.

Method 2

1.   Pres Ctrl+F1 to open SQL Reference window.

2.   In that window, type the command whose syntax you want to check. The SQL Reference will automatically locate that command in the Table of Contents and scroll it so the required commands appears on top of the visible list of items.

3.   Click the command whose syntax you want to lookup. Visual SQL Command Builder and Lookup window will appear at the bottom of the editor screen.

If you do not know the exact command or function name, you can use hyperlinks in the Table of Contents to

browse topics and subtopics. For example, if you are looking for the topic "Log Shipping Procedures" in Microsoft SQL Server but are not sure about their names or syntax:

1. Pres Ctrl+F1 to open SQL Reference window.

2. Click the **System Stored Procedures** hyperlink. The Topic will expand
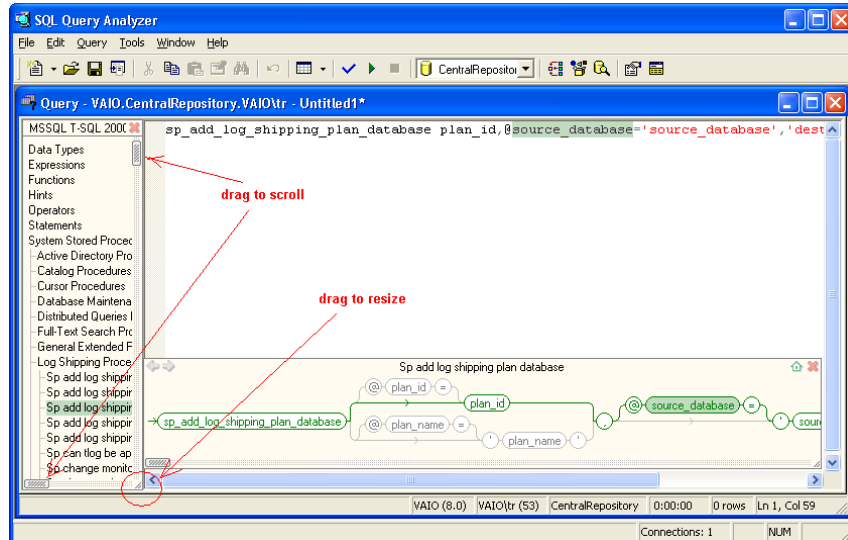
3. Locate the appropriate procedure; for example, *sp_add_log_shipping_plan_database* and click on the hyperlink. The procedure syntax and parameters appear in a new window at the bottom of the editor screen.

To scroll the **Table of Contents** window, drag and its scrollbar handles as pointed on the picture above.

To resize the **Table of Contents** window drag its bottom-right corner as needed.

To pick a different version of the SQL Reference, click the drop-down list displayed at the top of the **Table of Contents** window, then select the required version.

Working with the Visual SQL Command Builder Interface

The previous topic describes how to locate SQL commands and functions in the SQL Reference system and how to open the **Visual SQL Command Builder** window. Once you have the **Visual SQL Command Builder** open, you can use it to graphically build SQL commands or to simply paste command syntax into the editor window.

Most SQL commands support multiple option groups and syntax elements. For example, the ALTER TABLE command can be used to add new referential constraints, to drop a column, to alter table storage attributes and for many other actions each requiring a different command syntax with different group of options.

Syntax groups are displayed as parallel horizontal chains of keywords and hyperlinks. If these syntax groups are non-mutually exclusive and can be used together, a "loop-back line" is displayed around them. The selected syntax group is displayed in bright green color. All other available syntax groups are displayed in grey color.
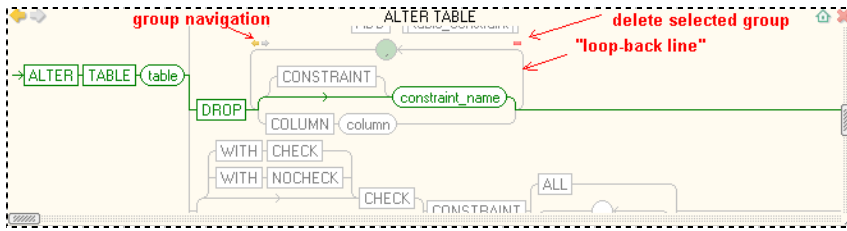
To generate and paste code for a specific syntax group, double-click the last syntax element in the group.

To completely replace text of a previously pasted syntax group, simply click the last element of the required group.

To add another syntax group to already pasted text without replacing the previous group, click the ⊙ circle with comma symbol displayed in a middle of the "loop-back line,"

If you want to add new syntax group in front of the previously pasted group use the little yellow arrows navigation links to navigate syntax groups and highlight their text in the editor.

To delete unneeded group from the text in the editor, single click on that option name displayed in **Visual SQL Command Builder** window and then click the little red minus sign.

If a syntax option contains multiple sub-options, its name is displayed as an underlined hyperlink. If you click on such name the **Visual SQL Command Builder** window will refresh and show available sub-options.

Use yellow arrows in the top-left corner of the **Visual SQL Command Builder** window to navigate displayed screens.

To synchronize the current topic with the Table of Contents click the little home icon in the top-right corner of the **Visual SQL Command Builder** window.

Once you got the correct command syntax pasted, edit non-syntax elements such as specific object names, column names, parameters and so on. You can edit such elements directly in the editor, or you can edit them within the Visual SQL Command Builder Interface.

Tip: Note different shapes of syntax elements displayed in the Visual SQL Command Builder Interface. The elements in rectangular boxes represent SQL keywords. The elements in round-corner boxes represent editable content. For example, the "table" and "constraint_name" elements on the previous screenshot are editable elements. If you click them, you can use in place editing to enter table and constraint names. As you type the names within round-rectangles, the Visual SQL Command Builder enters them same names into the editor. The elements in double-edged rectangular boxes represent SQL constructs whose syntax is documented in a separate SQL Reference topic and has its own Visual SQL Command Builder interface. Elements in circles represent non-keyword elements

CONSTRAINT - a single rectangular box for keyword elements. Clicking this box pastes keywords and all it preceding dependent keywords into the editor.

constraint_name - a round-corner box for user entered values such as table and column names, constraint names, column data type precisions, etc… After clicking box, you can edit the value within the box.

Table_constraint - a double-edged rectangular box for nested SQL constructs. Clicking this box opens another topic.

⟲ ⟲ - a circle for non-keyword type of syntax elements, such as brackets, column separating commas, etc… Clicking this circle pastes the circle content and all it preceding dependent keywords into the editor.

In place code editing interface:
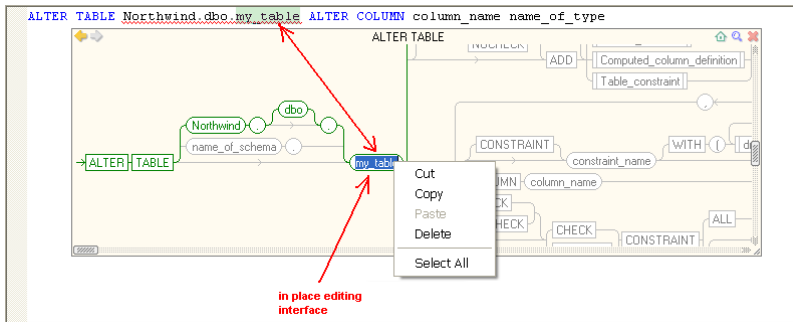
Scrolling Content

To scroll the **Visual SQL Command Builder** window content, use the mouse to drag the scroll bar or click the small arrows at the top and bottom of the scroll bar. See the screenshot at the beginning of the Working with SQL Assistant Popups topic for information on where to locate scrollbar handles.

Resizing the Visual SQL Command Builder window

To resize the **Visual SQL Command Builder** window, drag the resizer handle in the bottom-right corner of the window. See the screenshot at the beginning of the Working with SQL Assistant Popups topic for information on where to locate the resizer handle.

Moving the Visual SQL Command Builder window

If the **Visual SQL Command Builder** window covers part of the editor window that you want to see, click on an empty area within the **Visual SQL Command Builder** window, and while holding the left mouse button pressed, drag the window to the part of the screen where it is convenient for you.

Navigating Recently Visited Topics

To move between recently visited SQL Reference topics, use the Back and Forward arrows displayed in the left top corner of SQL Reference topic window. You can also use Ctrl+ Left Arrow and Ctrl+ Right Arrow keyboard shortcuts. Note that for the keyboard shortcuts to work correctly, the input focus must be in the SQL Reference topic window, but not within round-corner rectangles in the active edit mode.

CHAPTER 10, One-click DDL Code View

Overview

Any SQL developer who programs stored procedures, or functions or anyone who simply writes SQL code using SELECT statements querying database views, frequently needs to review the source code of referenced objects to understand the business logic encapsulated in that object. Typical methods used to access the source code involve using some kind of the database browser utility which may be either external to the editor or integrated with the editor. Typically, these browser utilities require many mouse clicks or keystrokes to find a particular object and then open the DDL of the object in a separate editor window or even save it to a file and then open. This is a very time consuming process, especially when working with large systems containing many thousands of objects.

SQL Assistant supports an elegant single click method to lookup object source code without the need to leave the editor. This method is based on what we call the "hot mouse tracking" feature. Hot tracking is the visual effect whereby text under the mouse pointer reacts to pointer movement and turns into a hyperlink. To activate the hot mouse tracking feature, hold down the Ctrl key and then move the mouse pointer over the name of the object whose source code you want to preview. The object name under the pointer will turn to a hyperlink. Click the hyperlink to display the object's source code in the Code View window.
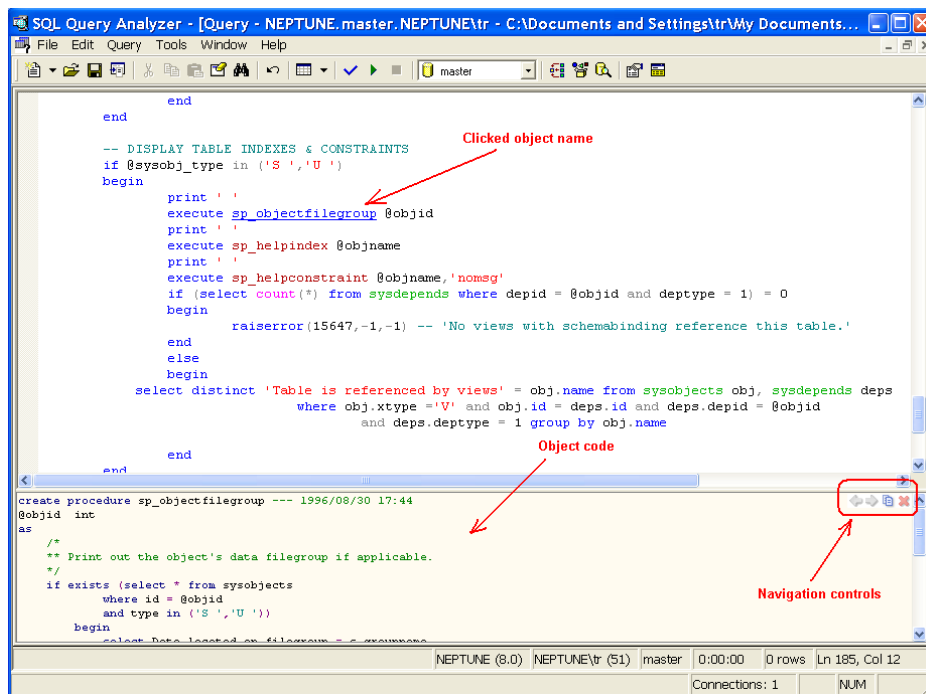
SQL Assistant supports two additional methods for invoking the Code View:

- Highlight the word or simply click the word referring to a procedural object or view in the database, then click the **Target / Show Object DDL** command in SQL Assistant's system tray icon menu (see the Using System Tray Icon Menu topic for details)
- Highlight the word or simply click the word referring to a procedural object or view in the database, then click the **SQL Assistant / Show Object DDL** command in the target editor's context or top-level menus. This method is available only if the menu integration option is enabled (see the Using Context

and Top-level Menus topic for details)

📌 **Important Note:**

The **Show Object DDL** menu can be used with procedural objects of different types including:

- Views (in all supported database systems) *See notes for Oracle versions 7 and 8 below*.

- Stored procedures (in all supported database systems)

- User defined functions (in all supported database systems)

- Oracle packages (applicable to Oracle database systems only)

- Oracle types (applicable to Oracle database systems only)

- SQL Server triggers (applicable to SQL Server database systems only)

📌 **Notes for users of Oracle 8.x:**

In Oracle systems, view definitions are stored in a system table in LONG data type column and therefore cannot be queried using regular catalog queries. The following technique can be used to add support for Oracle views to the **Code View**:

1. First create a new user-defined function in your Oracle database as in the example below:

```
CREATE OR REPLACE FUNCTION view_text(v_owner VARCHAR2, v_name VARCHAR2)
RETURN VARCHAR2
AS
    v_long LONG;
    v_len NUMBER;
    v_text VARCHAR(4000) := 'CREATE OR REPLACE VIEW ' || v_owner || '.'
                            || v_name || ' AS' || CHR(10);
BEGIN
    SELECT text, text_length INTO v_long, v_len
    FROM all_views
    WHERE owner = v_owner AND v_name = view_name;

    v_text := v_text || SUBSTR(v_long, 1, 4000 - LENGTH(v_text));
    IF v_len > 4000 THEN
        v_text := SUBSTR(v_text, 1, 3991) || CHR(10) || CHR(10) || 'more...';
    END IF;

    RETURN v_text;
END;
```

2. Double-click the SQL Assistant icon in the system tray to open the Options dialog, then add the following text to the end of the **DDL Code (Oracle)** query in the DB Queries option group:

```
UNION ALL
SELECT view_text(owner, view_name)
FROM all_views
WHERE owner = :OWNER AND view_name = :OBJECT
```

3. Close the Options dialog and wait several seconds for SQL Assistant to reload options in all open editors and for the new settings to take effect. For more information on how to customize SQL Assistant database queries, see the Customizing Database Catalog Queries topic in CHAPTER 48.

Working with the Code View Interface

Navigating Code Views

The previous topic describes how to open the **Code View** window. You can use the same methods to view code of a different procedural object while the Code View window is already open. Every time you invoke Code View, the view content is automatically refreshed with the source code of the requested object. The Code View window keeps track of requested objects and allows you to navigate their code segments much like you navigate pages in a web browser. Use the yellow arrows in the top right corner of the **Code View** window to navigate back and forth between screens.

Code View window navigation history is only available while the window is open. If you close it and then open the **Code View** again, the previous navigation history is gone.

Tip:

You can use the same methods within the **Code View** window to lookup the source code of objects referenced in the displayed code. This way you can drill-down from the top level source code of a procedure to source code of other objects referenced within that object code. The navigation buttons in the top-right corner can be used to go back to the source code of the previously viewed object.

Scrolling Content

Use the standard scroll bars available in the **Code View** window to scroll the content. In addition you can use regular keyboard navigation keys and the mouse wheel control if available.

Resizing Content

To resize the **Code View** window, drag the top edge of the window up or down. Note that when you place mouse pointer over the top edge of the **Code View** window the cursor shape changes to resize shape as on the following screenshot.

Make sure the cursor takes the right shape before dragging the window edge.

Copying Code

You can use the 'Copy to Clipboard' and 'Copy to Editor' methods to copy the entire content of the **Code View** window or selected portions only. Use the icon in the toolbar area of the Code View pane to copy content of the Code View to the clipboard, or use the icon to copy it directly into the editor. If no text is highlighted in the **Code View,** the entire content is copied; otherwise only the highlighted text is copied to the Clipboard or Editor.

Important Note: The 'Copy to Editor' method replaces the entire editor content with the content of the **Code View** pane. To undo that action, use standard Undo function available in your editor.

Comparing Code Between Code View and Editor

While working on modifying an existing stored procedure or view you may want to compare the current code in the editor against the version stored in the database. Using the technique described in this chapter open the procedure code in the **Code View** pane. Click the icon in the in the toolbar area of the Code View pane. This open the **Code Compare** utility which will show the code differences in textual, logical and program flow diagram formats. For more information see CHAPTER 25, Code Compare Utility. The Code

Comparing Code Between Code View and a File

The usage is the same as comparing code in **Code View** against code in the editor, which is described in the previous topic, except that in this case the code is compared against its version saved in a file. Click the icon in the in the toolbar area of the Code View pane. You will be prompted to select a file to compare to, and then the comparison results will be displayed in the **Code Compare** utility.

Customizing DDL Code Reverse-Engineering for Code View

SQL Assistant supports multiple methods for reverse-engineering DDL code of schema objects. Each method has it's own advantages and disadvantages. For example, using SQL scripts from DDL Code queries from DB Options tab allows fastest code generation, but the queries do not support all types of schema objects. In comparison using database native API and utilities provides the most coverage, but it's also the slowest method which may be performance-wise prohibitive when reverse-engineering DDL code of thousands of schema objects.
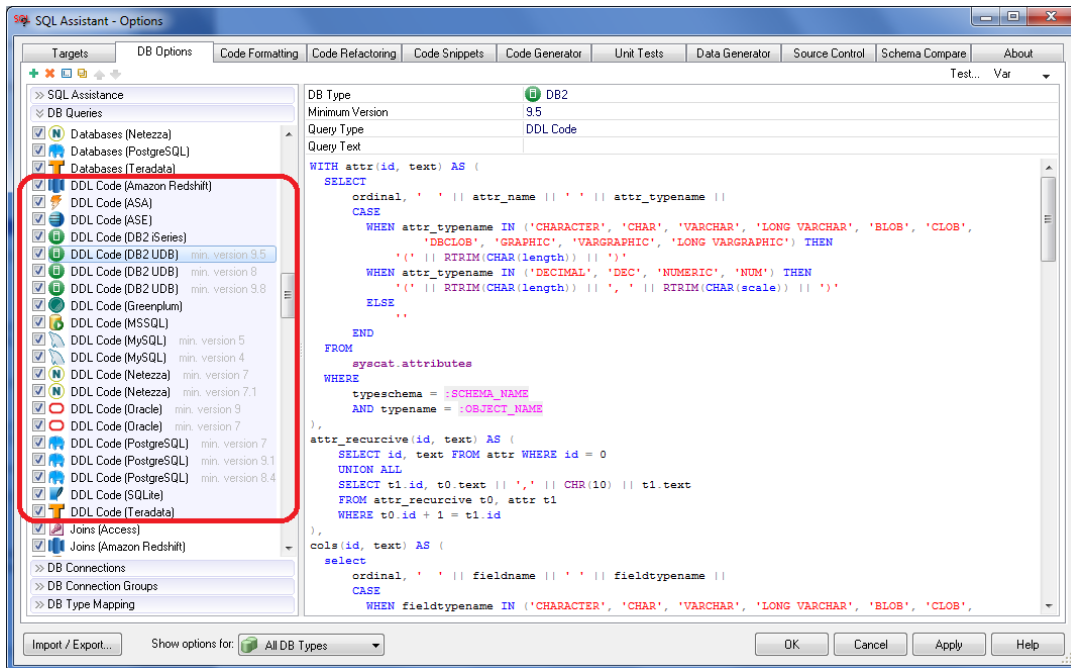
You can choose the most efficient method for your database and environment in SQL Assistant options. You can also configure SQL Assistant to use your own SQL script or external script files and command line utilities. To change the method:

1. Open the **Options** dialog and select **DB Options** tab.

2. Expand **SQL Assistance** section on the left and choose your database type.

3. Change **DDL Extraction Utility** option. The following values are supported:

    Let SQL Assistant choose best method – Enables SQL Assistant to choose the best method based on the object type and environment. For example, for SQL Server, it would use the programmatic SMO interface to reverse-engineer full table DDL, and query dbo.syscomments and other system catalog views to retrieve code of stored procedures and triggers.

    Using SQL scripts only – Instructs SQL Assistant to use DDL Code queries only for reverse-engineering schema objects.

    Using database command line tools and APIs only – Instructs SQL Assistant to use database native tools and APIs only for all kinds of schema objects. For some database types this might be the slowest method, however it ensures full compatibility with the database engine and version.

    Custom utility – Enables you to specify your own batch script or command line utility for reverse-engineering schema objects.. For details on how to use and customize this method, review several sample batch scripts provided in the DDL subfolder under SQL Assistant installation folder.

4. Click the Apply button to save your changes.

CHAPTER 11, Data Display and Editing

Overview

The Table Data Preview function enables you to pick at the data in a table or view referenced in the code with a single click without a need to leave the editor or to type and execute the complete SELECT statement. This method is based on what we call the "hot mouse tracking" feature. Hot tracking is the visual effect whereby the text under the mouse pointer reacts to pointer movement and turns into a hyperlink. To activate that feature, hold down the Ctrl key and then move the mouse pointer over the name of the object whose properties, source code, or data you want to preview. The object name under the mouse pointer will turn to a hyperlink. Click that hyperlink to display sample data in a data grid window.

Tip: For performance reasons, only the first 100 records are displayed in the Table Data Preview. To see the complete table content, right-click the data grid and choose the **Retrieve All** command from the popup menu. If you would like to filter the data or display the data in a certain order, use SQL Assistant's built-in facility for executing SQL queries.
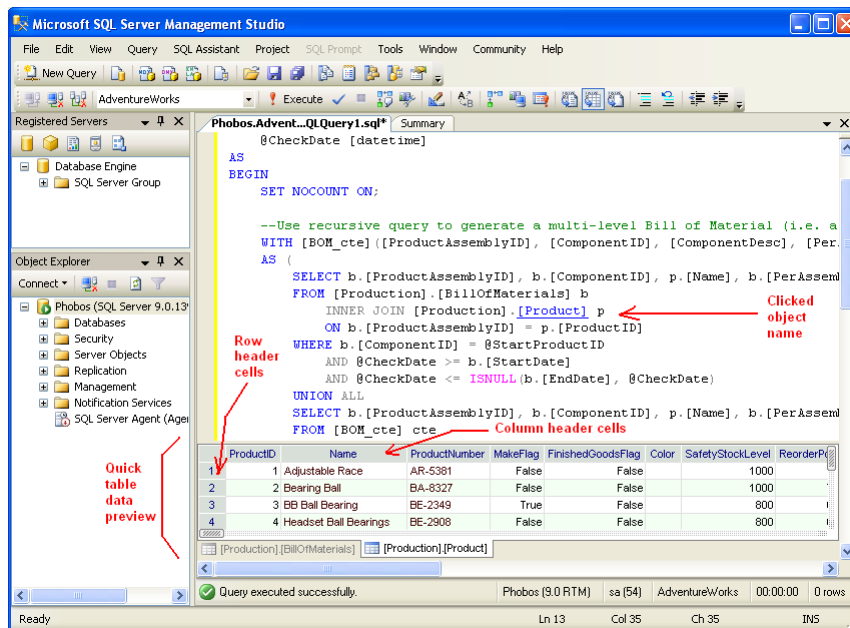
SQL Assistant supports two additional methods for invoking Table Data Preview:

- Highlight the word or simply click the word referring to a table or view object, then click the **Target / Show Table Data** command in the system tray icon menu (see the Using System Tray Icon Menu topic for details)
- Highlight the word or simply click the word referring to a table or view object, then click the **SQL Assistant / Show Table Data** command in the target editor's context or top-level menus. This method is available only if the menu integration option is enabled (see the Using Context and Top-level Menus topic for details)

Important Note:

The **Show Table Data** menu can be used with objects of different types including:

- Tables (in all supported database systems)
- Views (in all supported database systems)
- Aliases (supported in Oracle and SQL Server for aliases referring to table and view objects)
- Materialized Views (applicable to database systems supporting materialized views)

Working with Data Grid Interface

NULL Values

SQL Assistant renders cells containing NULL values as empty cells. A small green mark is displayed in the top left corner of a NULL cell to differentiate it from cells containing empty strings or spaces.

go	IL	USA
	TX	USA
hen		Germany
York	NY	USA
		France

If you mouse over a cell with a green mark, you should see a popup hint with word "NULL."

Long and Multi-line Text Values

Long values that exceed the column width are displayed as truncated values. Truncated values are followed by three dots (called ellipses) to indicate the data overflow effect. To see the entire value, resize the column.

For multi-line text values containing end-of-line and carriage return characters, only the first line is displayed. A small red mark is displayed in the top-left corner of a cell containing multi-line text to differentiate it from cells containing single-line values. To see the entire text value, double-click the cell. The text will be displayed in a separate Cell Value window. See the Expanded Cell View topic below.

essed	text
1	
1	CREATE VIEW syssegments (segment, n;
1	CREATE VIEW sysconstraints AS SELEC
0	([au_id] like '[0-9][0-9][0-9]-[0-9][0-9]-[0...
0	('UNKNOWN')
0	([zip] like '[0-9][0-9][0-9][0-9][0-9]')
0	([pub_id] = '1756' or ([pub_id] = '1622' o...
0	('USA')

If you mouse over a cell with a red mark, you will see a popup hint with instructions for displaying the entire

value.

Tip: To resize a column to display the entire content of all cells, double-click the right-edge of the column header. SQL Assistant resizes the column to fit the content of each cell.

Expanded Cell View

The returned query results do not always fit well in the data grid cells. The values can be longer than the available space or contain multiple text lines. For a better view of the results, double-click the cell whose value you want to see. This will open the **Cell Value** popup window. To see a value in some other cell, double-click that cell. There is no need to close the **Cell Value** popup window, it will refresh automatically.

Tips:

- The title bar of the **Cell Value** popup window indicates the row and column of the cell containing the value.

- The popup window can be moved and resized as needed. It will remember its size and position and stay on top of other windows until it is closed.

- If the value displayed in the **Cell Value** popup window contains long lines, select the **Line Wrap** check box to make text wrap at the right edge of the display area.

Scrolling Content

Use the standard scroll bars in the data grid to scroll the content. You can also use regular keyboard navigation keys and the mouse wheel for scrolling.

Resizing Content

To resize individual columns in the data grid, drag the right-edge of the column header left or right. Note that when you place mouse pointer over the right edge of a column header the cursor shape changes to resize shape as on the following screenshot.

⬌

Make sure the cursor takes the right shape before dragging the column edge.

Tip: Long values that exceed the column width are displayed as truncated values. Truncated values are followed by three dots (called ellipses) to indicate the data overflow effect. To see the entire value, resize the column.

Copying Data to the Clipboard

You can use the 'Copy to Clipboard' and 'Copy to Editor' methods to copy the entire content of the data grid or selected portions only. Use the right-click menu over the data grid to activate the **Copy** function or use the Ctrl+C keyboard shortcut. The copied text can be then pasted into the target editor or any other program using standard **Edit / Paste** menu or Ctrl+V hot key.

Tips: The grid supports both regular column-wise and row-wise data selection and irregular data selection of multiple regions:

- To select an entire row of data, click the row header of the cell containing the row number

- To select an entire column, click the column header

- To select the entire grid content, click the top left cell in the row headers column. Alternatively, you can right-click on the grid, then click the **Select All** command in the right-click menu.

- To select an individual cell, just click on that cell.

- To quickly select several adjacent cells, use "mouse lasso" effect over these cells. Click on the starting cell and then, while holding down the mouse left button, drag it around cells you want to select.

- To select any combinations of the above, for example, to select several columns, hold down the **Ctrl** key on the keyboard and then select the additional areas using methods described above.

Copying Data to a New Excel Worksheet

The data grid supports enhanced copying data to Microsoft Excel function that can be used to copy the entire content to a well-formatted Microsoft Excel worksheet. Use the right-click menu over the data grid to activate the **Open in Excel** function.

1. Right-click the data grid to activate the context menu.

2. In the context menu choose **Open n Excel** command

Tips: If Microsoft Excel is not running, a new instance will be started automatically

Saving Data to Files

The data grid supports data export functions that can be used to save the content to a file.

1. Right-click the data grid to activate the context menu.

2. In the context menu choose the **Save All as…** command

3. Use standard system file browse dialog to choose destination directory and the name of the output file.

4. In the same dialog choose file output format. The extension given to the output file will be used to for the output format. You can use one of the following values:

   - Comma separated format (CSV)
   - Tab-separated format (TXT),
   - Space-padded table column layout format (TBL)
   - Excel workbook (XLS or XLSX)
   - Extensible markup language file format (XML)
   - JavaScript object notation format (JSON)
   - Apache Parquet files (PARQUET)

5. Click the **Save** button to save the file. If the file already exists, SQL Assistant will prompt to overwrite the file.

Note: In order to save data to one of Excel supported formats, Microsoft Except version 2000 or later must be installed on your system.

Printing Data and Saving it as Reports

The data grid enables saving data to PDF reports as well as printing it with a variety of page control options.

1. Right-click the data grid to activate the context menu.

2. In the context menu choose the **Print Report/PDF…** command. This will generate a report and display it in the **Print Preview** dialog as in the following example.
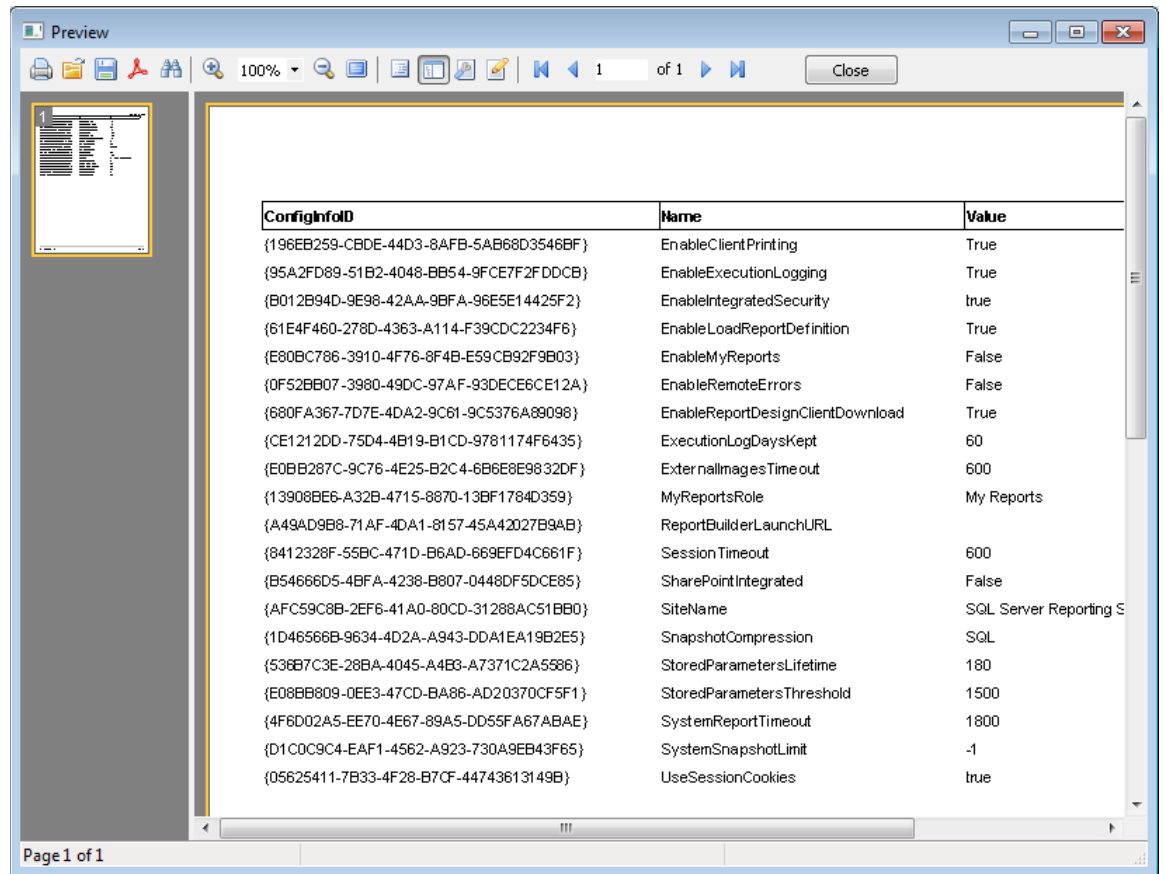
3. Click the **Print** toolbar icon in the Print Preview dialog to print the report.

4. Click the **PDF** toolbar icon in the Print Preview dialog to save report to a PDF file. If the file already exists, SQL Assistant will prompt to overwrite the file.

5. To close the report, click the **Close** button in the top right corner of the dialog

Tips:

- Width of columns in the report matches width of columns in the data grid from which the report was generated. To adjust the width, resize columns in the data grid and then regenerate the report using the same steps as above.

- The report title matches the name of the table data preview (or query results) tab. To change the title, you can rename the tab. Right-click the tab and choose the **Rename** command from the context menu.

Scripting Data as SQL INSERT Commands

The data grid provides an alternative interface for scripting table and query results data. See CHAPTER 13,

Scripting, Exporting, and Importing Data for more information.

To access the scripting interface from the data grid:

1. Right-click the data grid to activate the context menu.

2. If you plan on executing the script immediately or copying it to another program, in the content menu choose **Save as Script …** command. This will display the options dialog in which you can choose where to output the generated script and additional options for the data formatting.

   Important Note: The **Save as Script** command scripts out the complete window content, which in the case of the data grid, may or may not be the complete table data.

Loading All Records

By default, the **Table Data Preview** shows only the first 100 records of the selected table. To make SQL Assistant display the complete table data, right click anywhere in the data grid and select the **Retrieve All** command from the right-click menu.

Sorting Content

The data displayed in the data grid appears in the order returned by the database. Occasionally you may want to sort its content based on a specific column. To sort the data based on a specific column, move the mouse pointer over the column header until a small green arrow (the **sorting hot-spot)** appears. The data is sorted in ascending order based on the content of that column. Click the **sorting hot-spot** again to sort in descending order. A third click restores the original order.

The same technique can be used to sort by multiple columns. After you click the **sorting hot-spot** in one of the columns, hold down the Ctrl key and click the **sorting hot-spot** in another column to apply two-columns based sorting. Repeat the same if you need to add more columns.

Note: The **sorting hot-spot** is displayed as a small green arrow near the column header's right edge. This arrow appears only when the mouse pointer is over the column header. If you click anywhere in the column header away from the **sorting hot-spot**, the content of the column is selected, but the sort order is not changed.

The current sort order is indicated by a fixed green arrow which is permanently displayed over the column header. Refer to the following sample screenshot for details.

You can also sort data in the data grid using the context menu. To use this method, right click on the column you want to sort on and select one of the following options from the **Sort** submenu:

- **Ascending –** sorts data in ascending order based on the content of the column under the mouse pointer.

- **Descending –** sorts data in descending order based on the content of the column under the mouse pointer.

- **Original –** restores the original sort order in effect when the data was retrieved from the database. Note that this option is enabled only when you right-click a column which has been previously selected for the sorting.

Filtering Content

The data grid supports spreadsheet style Auto-filter interface turning values in the grid columns into specific filters based on the cell content. The auto filter supports multi-column filtering as well. To Filter the data:

1. Right-click the data grid and in its context menu choose the **Auto-filter** command. The grid column headers will change to dropdown menus.

2. Column headers will display drop-down arrow to the right If you click the arrow control, you will see all the values for that particular column.

Important Note: The displayed list is **dynamic**. If you choose some filter values in a particular column, then click the arrow control for some other column, the list for that other column will show only values that are still visible.

To show all values for a particular column, click the arrow control, and then select the special **All** value displayed close to the top of the list. In case you also applied custom filters, click the **Delete Filter** top most item.

If you rather prefer to filter values using a value range or some other conditional criteria, select the **Custom Filter** item in the same drop-down items.

Note: Different kinds of conditions are available for different data types. The example above shows conditions available for an numeric column. More choices are available for text based columns including advanced regular expressions. For additional information about regular expressions syntax see Using Regular Expressions topic in CHAPTER 34, Integrated SQL Editors.

Tip: Different icons are used for filtered columns to indicate where filters have been already applied. Headers of filtered columns show while unfiltered columns show .

To turn off the Auto-filter feature, deselect Auto-filter from the same context menu.

Finding Data Values

You can use **Find Text...** and **Find Text Again** commands within the data grid to search for the data in the data grid. To access the search interface from the data grid:

1. Right-click the data grid to activate the context menu.

2. In the context menu choose **Find Text...** command. The **Search in Table** dialog will appear.
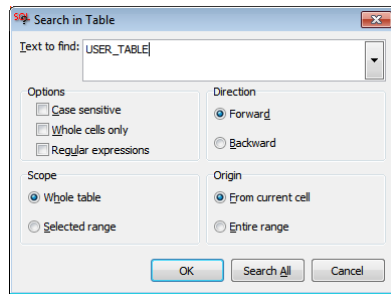
The following search options are supported:

Text to find – A regular expression or plain text to find in the table.

Regular expression – If this checkbox is checked, the value of the **Find text** field is treated as a regular expression. If not checked, the value is treated as a plain text.

> **Tip**: When using plain text searches, you can refer to special symbols using backslash as the escape symbol. For example, to refer to a tab character, specify \t. When using regular expression, use regular expressions compatible syntax to refer to special symbols.

Case sensitive – If this checkbox is checked, the search is case sensitive. This property is only used with plain text searches (e.g. when the **Regular expression** checkbox is not checked)

Whole cells only – If this checkbox is checked, the search is performed only for cell with values exactly the same as the text in the **Find Text**, otherwise a substring search is performed on cell values. This property is only used with plain text searches (e.g. when the **Regular expression** checkbox is not checked).

Direction – The search direction. This is used together with the **Scope** and **Origin** options.

> **Forward –** The search is started forward. Note the search is run in columns left to right and rows top to down.

> **Backward –** The search is started backward. Note the search is run in columns right to left and rows bottom to top.

Scope – The search scope.

> **Whole table** – the entire data grid is searched.

> **Selected range** – only the selected cells are searched. See tips section in this chapter in the Copying Data to the Clipboard topic for description of methods that can be used to select a regular shaped or irregularly shaped region containing multiple cells.

Origin   – The search starting point:

> **From current cell -** the search is started from the cell with the input focus in the direction specified by the selected **Direction** option.

> **Entire range -** the search is started from the top left cell of the selected cell range or the first cell of the table if no cell range is selected.

> **Tip**: Press F3 key to quickly find next sell matching the same search criteria. This is a shortcut for the **Find Text Again** menu..

Finding Columns

Sometimes results of a query or quick table data preview contain many columns spawning multiple horizontal pages, and It is difficult to spot the required column or columns when looking at the grid. In such situation you can use the **Find Column…** command to quickly locate the required columns.

1. Right-click the data grid, and choose **Find Column…** command in the context menu. This command opens **Find Column** dialog listing all columns in the selected grid.
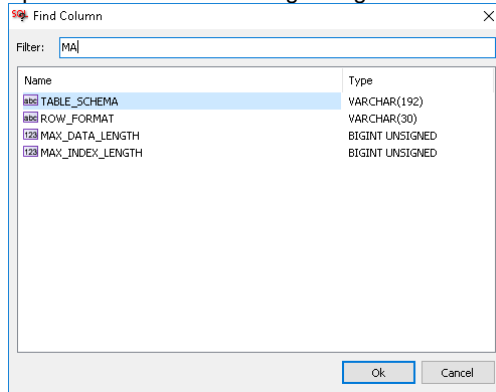
2. To narrow the list you can start typing the name of the column you looking for.

3. To sort column names alphabetically click the right side of **Name** header.

4. Double-click the required column name to highlight it in the data grid.

Tip: The behavior of the **Find Column** command when used with SQL Server Management Studio grids is a bit different. It opens a different kind of **Find Columns** dialog containing additional options for supporting column searches across multiple grids that may appear after executing single or multiple SQL commands single.
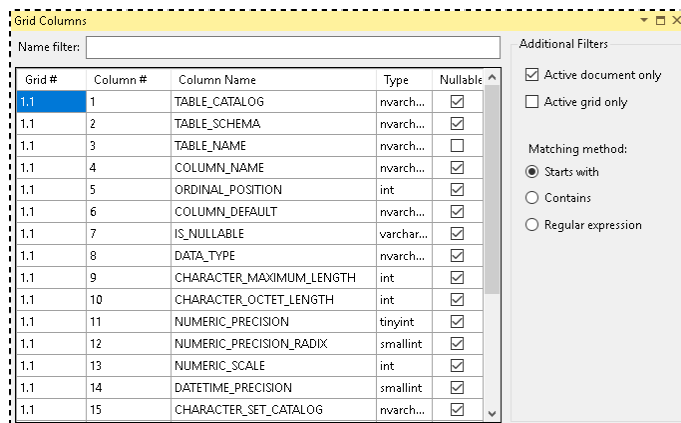
Active document only – this option refers to the active tab. If unchecked, the column search is applied to all data grids currently available in all open SQL Server Management Studio tabs.

Active grid only – If checked, searches only the active grid, otherwise searches all grids available in the result set.

Matching method – the method to use when matching column names to the text you typed n the filter box.

Freezing Grid Columns

In certain cases you may want to keep an area of the data grid visible while you scroll it horizontally. You can use the **Fix Column** feature to divide the data grid into left and right panes. When divided, the left pane is static containing the frozen columns, and right pane is scrollable containing the other columns.

To freeze a column

Right-click the column header of the column you want to freeze and then select the **Fix Column** menu. The selected column is moved to the left most position of the left pane. If you need to reposition the frozen column, unfreeze them first and then select them in a different order or freeze them all at once asa group.

Tip: If you right click header of a frozen column, you will notice that a checkmark is shown in front of the **Fix Column** menu indicating that the column is already frozen.

To freeze a group of columns

While holding down the Ctrl key, click the column header of each column you want to freeze. The selected columns will have their background highlighted. Right click the header of any of the selected column and then select the **Fix Column** menu. All selected columns are moved to the left static pane.

To unfreeze a column

Right click the column header of each column you want to unfreeze and then select the **Fix Column** menu again. The selected column is moved to the right scrollable pane.

Hiding Columns

You can hide one or more column by right clicking anywhere in the data grid and selecting **Hide Columns**… from the context menu. The **Hide Columns** dialog will be shown. Use the left most column with checkboxes to choose which columns to hide and which to show. All columns with their checkboxes selected will be visible, while all columns with their checkboxes cleared will be hidden.

To unhide previously hidden columns, repeat the same procedure again and select checkboxes for the columns you want to be visible again.

Tips:

- Fixed columns cannot be hidden. If you want to hide them, you need to unfreeze them first.

- The hidden columns are removed from display only, but not from the grid's data cache. This feature is provided to help reduce screen clutter. Hiding columns does not impact saving the cached grid's data to files, databases, exporting Excel, converting grid data to a report or pivot table, and so on. However, hidden columns are ignored by the Find Column and Find Text features.

Changing Data

Activating Edit Mode

If the data in the data grid is for a table with primary key, you can make changes in the data and save them back to the database. To activate the **Edit** mode, click the cell whose data you want to change and press F2

key. Alternatively, you can right-click the cell and select Edit command from the context menu.

Once the **Edit** mode is activated, you can make changes in other cells as well.

Note that the changes are not immediately saved to the database. They are saved only when you select the **Save Changes** menu.

> **Tip**: Cells with modified and unsaved yet data have yellowish background for easy of data change identification, as demonstrated on the example screenshot below.

	PROJNO	PROJNAME	DEPTNO	RESPEMP	PRSTAFF	PRSTDATE	PRENDATE
1	AD3100	ADMIN SERVICES	D01	000010	6.50	1/1/2002	2/1/2003
2	AD3110	GENERAL ADMIN SYSTEMS	D21	000070	6.00	1/1/2002	2/1/2003
3	AD3111	PAYROLL PROGRAMMING	D21	000230	2.00	1/1/2002	2/1/2003
4	AD3112	PERSONNEL PROGRAMMING	D21	000250	1.00	1/1/2002	2/1/2003
5	AD3113	ACCOUNT PROGRAMMING	D22	000270	2.50	1/1/2002	2/1/2003
6	IF1000	QUERY SERVICES	C01	000030	2.00	1/1/2002	2/1/2003
7	IF2000	USER EDUCATION	C01	000030	0.95	1/1/2002	2/1/2003
8	MA2100	WELD LINE AUTOMATION	D01	000010	12.00	1/1/2002	2/1/2003
9	MA2110	W L PROGRAMMING	D11	000060	9.00	1/1/2002	2/1/2003
10	MA2111	W L PROGRAM DESIGN	D11	000220	2.00	1/1/2002	12/1/1982
11	MA2112	W L ROBOT DESIGN	D11	000150	3.00	1/1/2002	12/1/1982
12	MA2113	W L PROD CONT PROGS	D11	000160	3.00	2/15/2002	12/1/1982

PROJECT

Cell Value Manipulations

Use regular data input methods to edit cell values. To copy, cut, or paste entire cell value or a part of it, use appropriate commands in the cell's right-click menu.

To set a cell value to NULL value, select the cell, be sure it's not in the Edit mode, and then press the Del key.

.

Row Manipulations

To insert a blank row at the end of the data-grid, use the Ins key, or right click the data grid and select **Add Row** command from the context menu.

To delete one or more rows, select the rows to be deleted and then press Ctrl+Del keys, or right click the selected rows, and from the context menu select Delete Rows menu.

Note that like cell changes, row changes are not immediately saved to the database. They are saved only when you select the **Save Changes** menu.

> **Tip**: You can copy data to the data-grid from other applications, for example, from Microsoft Excel, using the Windows clipboard. If the clipboard contains a set of tab separated values in one or more rows, the content is pasted as new rows appended to the end of the data-grid. If the shape of the data in the clipboard does not match columns in the data-grid, the values from the clipboard are pasted cell by cell. To use this method, press Ctrl+V, or use the right-click menu in data grid and select the **Paste** command.

Undoing Changes

Before the changes are saved to the database, you can undo them incrementally much like you undo changes in the editor. To undo the changes, press Ctrl+U, or use the right-click menu in data grid and select the **Undo** command.

Saving and Scripting Changes

To save your changes to the database table, right click the data grid and select **Save Changes** command from the context menu.

To save your changes as a set of SQL commands without actually updating the database, right click the data grid and select **Save Changes as Script...** command from the context menu. The Save As dialog will appear enabling you to select the SQL script file name.

Customizing Fonts

To change the font used in the data grid

1. Open SQL Assistant's main **Options** dialog.

2. On the **Targets** tab select **Common** section and then click the **[...]** button on the right side of the Table Pan Font option. The standard **Font Properties** dialog will appear.

3. Choose desired font family and size and then click the **OK** button to close the dialog.

    Tip: Only the font family and size are used, all other font properties shown in the Font Properties dialog are ignored.

Customizing Data Display Formats

Column Specific Formats

To change how the values appear in the data grid for specific columns, right-click anywhere within the grid and select the **Display Format**… menu. The **Display Format** dialog window will appear.
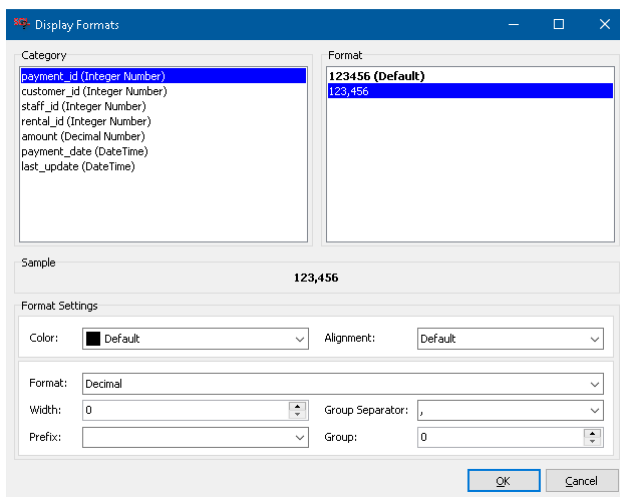
In the Category list, the dialog lists all columns available in the grid and their data types. In the format list it lists the default format used fro the data type as well as all previous customized formats you have used before so that you can quickly pick from that list the one you want to use. Use the controls and options in the Format Settings section to define a new format. Click other columns in the Category and if required modify their formats

too or select formats from the Format list. When ready, click the OK button to close the dialog and apply new display formats to the data in the data grid.

Tip: The customized display formats apply to columns referenced in the current data grid. The customized display formats are sticky. If you execute a query retuning data from the same table and columns, if SQL Assistant can match them to the columns whose display formats have been previously customized, it will automatically apply the customized formats.

Clearing saved customizations

If you want to clear the saved format, right-click in any data grid, choose the **Display Format**… menu and then simply delete them from Display Formats dialog.

Data-type Specific Formats

Using the global options you can customize display formats for various data types that apply to all data grids and all columns with the same data type unless their column specific format is customized as described above.

1.   Open the Options dialog.

2.   On the Targets tab, click the Common section on the left hand side of the dialog.

3.   On the right hand side expand the data grid section.

4.   Click the Display format option, and then click the […] button on the right end of the line.
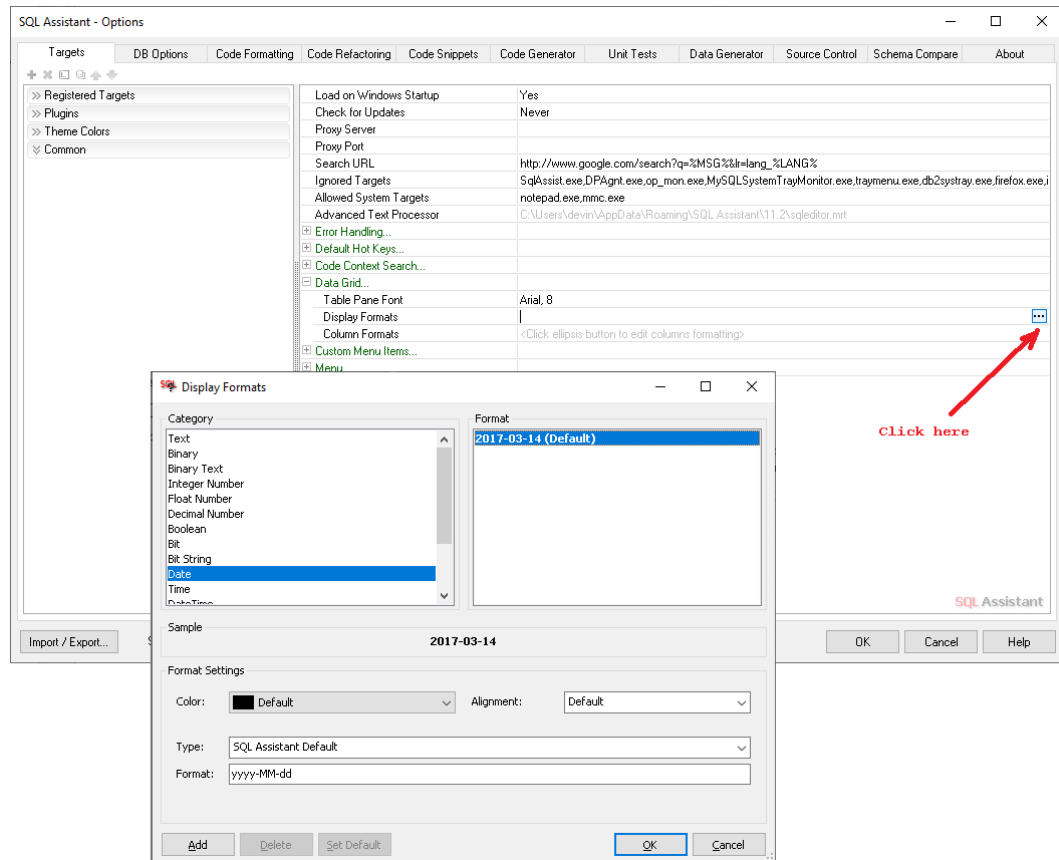
5. Customize formats as needed

6. Click the Ok button to close the Display Formats dialog

7. Click the Ok button to close the Options dialog.

Conditional Formatting

Conditional formatting in the data grids enables you to highlight cells with a certain background or font color, depending on the specified condition. You can use that feature to make some data cells to stand out and to emphasize data patterns.

Multiple conditional formatting rules can be used in the same data grid. For example, you can apply different formatting rules to different columns. Or you can apply multiple conditional rules to the same column to produce composite highlighting effects.

	order_id		item_id		product_id	quantity	list_price	discount
25	10		1		14	1	269.99	0.10
26	11		1		8	1	1799.99	0.05
27	11		2		22	2	269.99	0.10
28	11		3		16	2	599.99	0.20
29	12		1		4	2	2899.99	0.10
30	12		2		11	1	1680.99	0.05
31	13		1		13	1	269.99	0.10
32	13		2		17	2	429.00	0.05
33	13		3		20	2	599.99	0.10
34	13		4		16	2	599.99	0.05
35	14		1		6	1	469.99	0.07
36	15		1		12	2	549.99	0.07
37	15		2		8	1	1799.99	0.07
38	15		3		18	2	449.00	0.05
39	15		4		23	2	299.99	0.20
40	16		1		8	1	1799.99	0.20

stocks | Messages | order_items

Applying Conditional Formatting Rules

1. Right-click the data grid column to which you want to apply conditional formatting.

2. In the popup menu select the **Conditional Formatting** menu. A submenu will expand to the right. Select the type of conditional formatting rule you want to apply.

3. In case the selected rule requires additional input, for example, if you selected a rule for highlighting values greater than certain number, specify the value.

4. You can customize cell background and text colors.

5. Click the OK button.

Tips:

- If the selected column has been already formatted, the new rule by default will be combined with the previously applied rules, as in the example below. To replace the existing conditional highlighting, select the **Override previous rules** checkbox in the Rule properties dialog.

269.99	0.05
3999.99	0.10
499.99	0.05
2999.99	0.10
1549.00	0.07
2999.99	0.20
2999.99	0.07
1549.00	0.07
269.99	0.05
429.00	0.10
1549.00	0.07
529.99	0.20

- If you only want to change cell text color, but not its background, choose **None** color value in the **Background color** drop-down list.

- The color drop-down lists contain common colors from the basic palette as well as various Windows system colors. If you want to choose a custom color, pick the special **Custom …** color value

Color:	Yellow		Custom...
	Custom...		
	B20000		
Preview:	Black		
	Maroon		
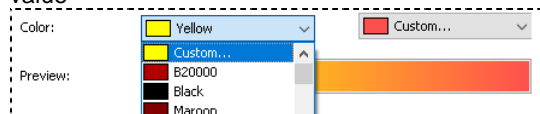
Highlighting Cells with Unique or Duplicate Values

To highlight cells with unique values, choose the formatting rule named **Format only unique or duplicate cells** and then in the **Format all** drop-down list, choose the **Unique values** option.

Similarly, to highlight cells with non-unique values, choose the formatting rule named **Format only unique or duplicate cells** and then in the **Format all** drop-down list, choose the **Duplicate values** option.

Clearing Rules

1. Right-click a column with conditional highlighting.

2. In the popup menu select the **Conditional Formatting** menu. A submenu will expand to the right.

3. Select **Clear Rules** menu and then choose whether to clear conditional formatting in the selected column only, or in the entire data grid.

Tip: If you have multiple conditional formatting rules applied to a single column and you want to remove or change only a subset of the applied rules, use the **Manage Rules** submenu. See the following topics for more information on formatting rules management.

Using "Stop if True" Option

The **Stop If True** option can be used with multiple rules applied to the same column(s). It prevents the current rule from processing other rules when a condition in the current rule is met. In other words, if two or more rules are set for the same columns and **Stop if True** is enabled for the first rule, the subsequent rules are disregarded after the first rule is activated.
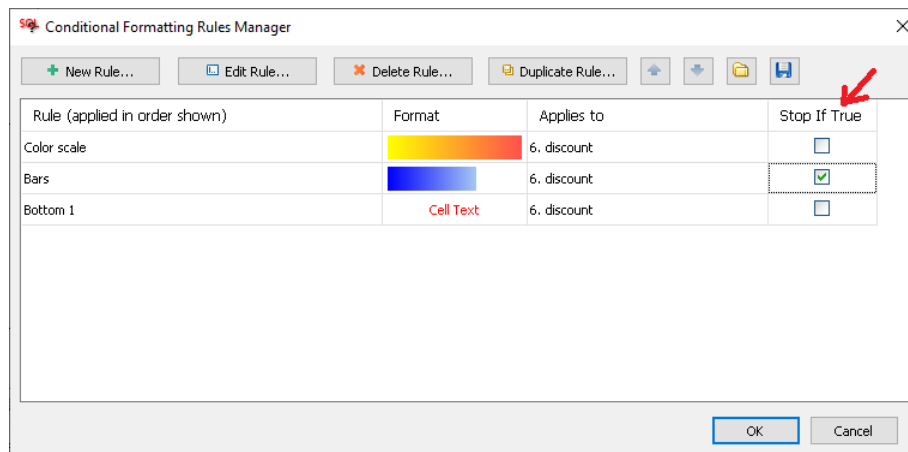
Tip: The rule order is important when **Stop If True** option is used. To change the order, use the Arrow Up and the Arrow Down icons in the right-top corner of the **Conditional Formatting Rules Manager** dialog. The dialog is accessible through the **Conditional Formatting -> Manage Rules…** menu.

Managing Rules

Use the **Conditional Formatting Rules Manager** dialog to modify criteria for a formatting rule already applied to the current data grid column, and to quickly add multiple rules in case you want to highlight different cells based on a set of specific values, rather than a single value range, and to selectively clear

some of the rules already applied to the current column. The dialog is accessible through the **Conditional Formatting -> Manage Rules…** menu. A screenshot of the **Conditional Formatting Rules Manager** dialog can be found in the previous topic. The New Rule… and the Edit Rule… buttons will open the **Formatting Rule** dialog, which can be used to enter new or edit an existing rule definition
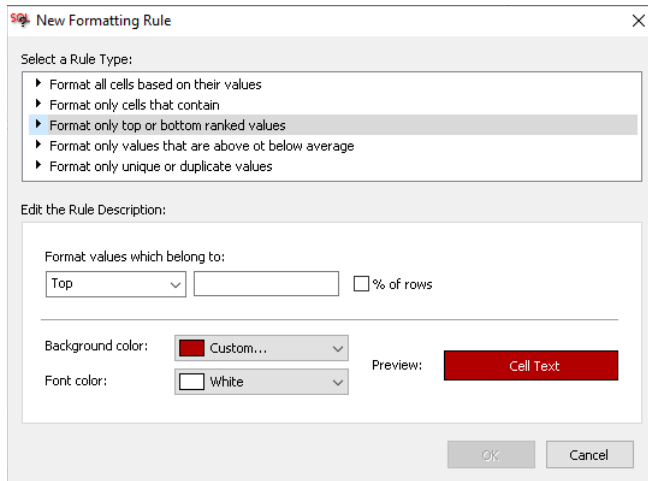
Saving and Reusing Rules

You can save customized rules and rule collections to .CFR files using the Conditional Formatting Rules Manager dialog. The dialog is accessible through the **Conditional Formatting -> Manage Rules…** menu. The 💾 Save and 📂 Load toolbar icon are located in the top-right corner of the dialog.

Data Grid Limitations

The maximum size of a text-based or binary column displayed in the data grid depends on the selected connectivity interface and the underlying database driver. For example, some ODBC drivers limit text values to the first 255 characters. For any column that exceeds this limit, some values might appear truncated.

The nature of the data stored in BLOB and similar binary columns is unknown to SQL Assistant. The values stored in such columns are displayed as hex encoded strings. The values in CLOB, JSON, XML columns are displayed as a normal human readable text.

The amount of data that can be loaded using the **Retrieve All** command in the right-click menu is limited by the amount of free memory available to your computer at the time of code execution.

The **Edit** mode does not support all data types. Only the simple data types and data that your database can automatically convert from text values to the table specific data type values can be edited in the data grid.

CHAPTER 12, Working with Workspace Database

Overview

SQL Assistant automatically creates a new workspace database for you when you invoke any related feature very first use. SQL Assistant uses internally integrated SQLite database engine. The Workspace Database provides persistent storage for saving various data objects and data sets, it also provides advanced SQL querying interface for retrieving and working with the saved data. The workspace database is always available, you do not need to start and stop it like other database servers.

Saving Query Results to Workspace Database

You can save data from any data grid populated with results of your database queries to the workspace using the following simple method:

1. Right-click the data grid to display its content menu and then select **Save to Workspace Database…** menu.

2. Click Ok to save it using the automatically suggested name, or change the name to your liking and then click Ok to save the results to a new table in the Workspace Database.

Executing Queries and Reports Against Saved Data

In SQL Assistant Integrated Development Environment select **File -> Open Workspace Database** menu. This will open new SQL editor tab connected to the Workspace Database. Use familiar SQL queries to retrieve and modify the data in the Workspace Database as you would use with any other database server. Use SQLite compatible SQL syntax when working with the Workspace Database,. The SQL Assistant advanced SQL Intellisense and other built-in features will assist you with coding and testing your queries.

In addition to SQLite standard set of functions the integrated engine provides several extensions that enable you to use regular expressions, JSON operators, external file tables. For more information see:

SQLite language expressions - https://www.sqlite.org/lang_expr.html

JSON functions and operators - https://www.sqlite.org/json1.html

Attaching CSV Files as External Tables

You can attach CSV files to the Workspace Database and turn them into external file tables that can be queried using regular SQLite compatible SQL queries. The file tables are read-only tables. Unlike data in regular tables their contents cannot be modified in the attached files through this interface. Use the integrated **Data File Query** utility to manipulate data in external files.

To attach a CSV file as external table:

1. In SQL Assistant Integrated Development Environment open the Workspace Database using **File -> Open Workspace Database** menu.

2. In the Database Explorer right-click any empty area. The context menu will appear. In the context menu select **Register CSV File as External Table** menu. Follow the prompts to select CSV file and file specific options.

3. You can now enter and execute SQL queries to retrieve data from the attached file, including queries that join that external table with other existing regular and external tables in the Workspace Database.

Exporting Data from Workspace Database

The data stored in the Workspace Database can be exported to files and to other databases using a variety of built-in methods. See CHAPTER 13, Scripting, Exporting, and Importing Data for specific details.

Backing Up and Restoring Workspace Database

The Workspace Database is located by default in %APPDATA%\SQL Assistant\Workspace folder on the system where SQL Assistant is installed.

Offline Database Backup

1. To backup the database in offline mode, close all instances of SQL Assistant and other development tools you use SQL Assistant with.

2. Copy all files from the Workspace folder to a network drive or other safe location.

Online Database Backup

1. To backup the database in online mode without closing anything, in SQL Assistant Integrated Development Environment open the Workspace Database using **File -> Open Workspace Database** menu.

2. In the Database Explorer right-click the "**main**" schema. The context menu will appear. In the context menu select Backup and Restore -> Backup … menu. The **SQLite Database Backup** dialog will appear.

3. In the backup dialog choose destination folder and name for the backup database. Click the **Backup** button to start the backup operation.

Database Restore

1. Currently the Workspace Database can be restored in offline mode only. You have to close all instances of SQL Assistant and other development tools you use SQL Assistant with.

2. Copy workspace database files from the backup location to the %APPDATA%\SQL Assistant\Workspace folder replacing the existing files.

CHAPTER 13, Scripting, Exporting, and Importing Data

Overview

Moving data around is a common requirement faced by many database developers and DBAs. Whether you are migrating data to another database server, rebuilding tables, or just creating test data, SQL Assistant provides several useful methods and utilities for helping with various types of data migration. As a true cross-database tool, it also supports copying data between different types of database systems. This can be achieved by exporting data to database type independent flat files, like XML, JSON, TXT, CSV, or XLS/XLSX files, and then importing them into the target database. as well as scripting data using SQL INSERT statements in a database specific format and then executing the resulting scripts in the target database system.

Exporting Table Data to Flat Files, Excel Files, and Apache Parquet Files

SQL Assistant has a data export facility that is available from the right-click menu in the target editor and in the Database Explorer. This method can be used for exporting a single table or view data to TXT, TBL, XML, JSON, XLS, and CSV. To use this facility:

1. Select a table in the Database Navigator or select a table or view name appearing within the code in the target editor.

2. Right click the selected name and in the popup menu select **SQL Assistant → Import / Export** Data menu branch and then select **Export To...** command. The **Save To** file dialog will appear.

3. Choose output file format and name and then click OK. The **Save To** dialog will disappear and the SQL Assistant - Table Data Export dialog will appear.

4. Choose additional export options if required, and then click the **Export** button.

Exporting Multiple Tables in a Schema or Database

This function enables you to export data from multiple tables and/or views at once. It works very similar to the function described in the previous topic.

1. Select a schema or database in the Database Navigator or select a schema or database name appearing within the code in the target editor.

2. Right click the selected name and in the popup menu select **SQL Assistant → Import / Export** Data menu branch and then select **Export To...** command. The **Save To** file dialog will appear.

3. Choose output file format and base file name then click OK. The **Save To** dialog will disappear and the SQL Assistant - Table Data Export dialog will appear.

4. Choose specific tables and views. Note that all tables and views in the chosen schema are selected by default.

5. Choose additional export options if required, and then click the **Export** button.

Note: SQL Assistant will export each table to a separate file with the choose file format and extension. File names will be the same as the table names appending with the base file name you selected in step 3.

Exporting Query Results to Flat Files

SQL Assistant has a data export facility that is available from the right-click menu in SQL Assistant's Table Data Preview pane as well as in the Query Results pane. This method can be used for exporting a single table or view data as well as query results to TXT, TBL, XML, JSON, and CSV. For more information, see the Saving Data to Files topic in CHAPTER 11, "Using Table Data Preview."

Tip: You can customize data formatting before exporting data as described in CHAPTER 11, Table Data Preview and Editing, in Customizing Data Display Formats topic.

Exporting Query Results to Excel

This method can be used for exporting table and query results data into Microsoft Excel files

The data export facility is available via right-click menu in SQL Assistant's Table Data Preview pane and in Query Results pane used for displaying results of SQL code execution. For more information see Saving Content to Files topic in CHAPTER 11, Using Table Data Preview.

1. In the target editor execute query or stored procedure returning one or more result sets using Ctrl+Sfit+F9 hot key or using SQL Assistant's **Execute SQL Code** command in the right-click menu .

2. Right-click the returned data in the Data Preview tab and choose **Open in Excel** command.. SQL Assistant will start Microsoft Excel and copy formatted data from the Preview tab to new Excel worksheet.

3. You can use Save function in Excel to save the data to an Excel native file type or to any other Excel support file type.

Important Note: Microsoft Excel must be installed on your computer in order to use that function

Tip: You can customize data formatting before exporting data as described in CHAPTER 11, Table Data Preview and Editing, in Customizing Data Display Formats topic, or you customize it directly in Excel after the results are copied over to Excel.

Exporting Data to Other Programs Using Clipboard

This method can be used for exporting table and query results into other programs

The data copy facility is available via right-click menu in SQL Assistant's Table Data Preview pane and in Query Results pane used for displaying results of SQL code execution. For more information see Copying Content to Clipboard topic in CHAPTER 11, Using Table Data Preview.

Exporting Tables, Schemas, and Databases to XML and JSON Files

Overview

The XML/JSON Data Export utility can be used for exporting large amounts of data from many tables into hierarchical XML or JSON files. The utility provides simple to use graphical interface for mapping database data to XML or JSON schema elements. The export can be run immediately or scheduled to run unattended later using SQL Assistant's command line utilities.

The XML/JSON Data Export utility can export data in 2 modes: flatted 3-level data schema and hierarchical multi-level data schema. The following topics describe the supported modes.

Export to Flattened 3-Level XML or JSON Data Schema

The flattened data export can be used to export one or more tables to a single XML or JSON file. The tables are exported entirely. With the flattened export the output file will contain only 3 levels with a separate node for each database in the selection containing sub-nodes for each schema which in turn will have sub-node for each table within its schema. In other words, all databases will appear as 1st-level nodes, all schemas as 2nd-level nodes and all tables as 3rd-level nodes. For example:

XML Schema	Sample Content

<pre>Root
    Database 1
        Schema 1.1
            Table 1.1.1
                ... data record ...
            Table 1.1.2
                ... data record ...
            ...
        Schema 1.2
            Table 1.2.3
                ... data record ...
            Table 1.2.4
                ... data record ...
            ...
        ...
    Database 2
        Schema 2.1
            Table 2.1.1
                ... data record ...
            ...
        Schema 2.2.
            ...
        ...
    ...</pre>	

🖈 **Important Note:** In the flattened export mode, tables metadata is saved in the header nodes of the export file. If the generated output file is imported into a different database using the XML/JSON Data Import utility, it can accurately recreate the original tables in the destination database using the saved metadata. This improves cross platform data transporting using industry standard XML and JSON file formats. In the hierarchical export

mode the metadata is not saved. When importing data files without the metadata, SQL Assistant analyzes first 100 records of the actual data values and based on that analysis suggests column data types for the destination tables. Because the suggestions are based on a relatively small data sample, they might be sometimes incorrect. You should review them and correct as required before importing the data.

To generate flat 3-level XML or JSON file:

1. In the target editor open SQL Assistant menu or use the right-click menu for SQL Assistant's system tray icon. Navigate to **SQL Assistant → Import / Export Data** menu branch and then select **Export to XML or JSON Data Schema** menu. **The Export Data to XML or JSON Schema dialog** will appear.

2. In the **Destination File** field enter output file name, or use the Browse **[…]** button next to that field to select the output file using the system's standard file browse dialog.

     Note: The output file format depends on the file extension. If you choose XML, the output data will be written in XML format. If you choose JSON, the output data will be written in JSON format. If you choose anything else, then XML format is assumed.

3. Click the Plus Sign icon in the right top corner (to the right of the **Tables** list) to select tables to export. The Select Objects dialog will appear.

4. Select one or more tables and click the OK button. The **Export Data to XML or JSON Schema** dialog will be populated with the selected objects and the corresponding data file structure.
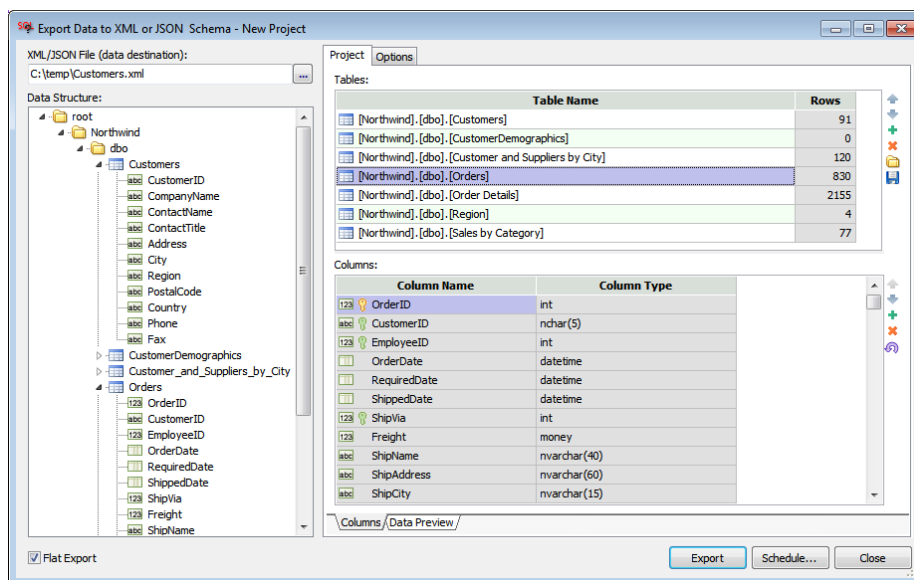
5. Optional step - you can use the Arrow Up and Down icons next to the **Table** list to rearrange table order in the export file.

     You can use the same icons in the **Columns** list to rearrange columns in the output.

     Note: The content of the **Columns** list matches the selected table in the **Tables** list. It is a master-detail view.

6. Optional step - you can use the Delete icons to remove tables or some of their columns from the export

7. Optional step - on the **Options** tab choose data formatting, error handling, and logging options. Refer to Scripting out Table Data topic in this chapter for specific details.

8. Click the Save 💾 icon to save the project. You will be prompted for the project file name.

9. If you wish to run the export immediately, click the **Export** button.

    If you wish to schedule a recurring export process, or a later run, use the **Schedule...** button, or use the **sacmd** command line utility provided with SQL Assistant. The command line for the export is
    sacmd xmlexp:"<project-file>;<data-file>" conn:"connection-name>"
    In the command above replace <project-file> with the full file path name of the project settings file, replace <data-file> with the full path and name of the XML or JSON output file, replace <connection-name> with the name of preconfigured database connection saved in SQL Assistant options.

🐁 **Tip**: Use Open Project 📂 icon to load previously saved XML/JSON Export and XML/JSON Import projects into the dialog.

Export to Multi-Level Hierarchical XML or JSON Data Schema with Nested Tables

This type of data export can be used to export one or more tables to a single XML or JSON file. With this type you can hierarchically nest tables within other tables, as well as you can specify additional data filters. Typically the structure of the generated output files will follow your database referential integrity design. Table referenced by referential constrains will be represented in the output file as sub-nodes of their parent tables within the associated parent records. For example:

XML Schema	Sample Content

<pre>root
    (optional level) Database 1
        (optional level) Schema 1.1
            Table 1.1.1
                ... data record ...
                Nested Table 1.1.1.1
                    ... data record ...
                    Nested Table 1.1.1.1.1
                        ... data record ...
                    Nested Table 1.1.1.1.2
                        ... data record ...
                Nested Table 1.1.1.2
                    ... data record ...
            Table 1.1.2
                ... data record ...
                Nested Table 1.1.2.1
                    ... data record ...
                Nested Table 1.1.2.2
                    ... data record ...
                    ...
        (optional level) Schema 1.2
            Table 1.2.3
                ... data record ...
            Table 1.2.4
                ... data record ...
            ...
        ...
    (optional level) Database 2
        (optional level) Schema 2.1
            Table 2.1.1
                ... data record ...
                Nested Table 2.1.1.1
                        ... data record ...
            ...
        (optional level) Schema 2.2.
            ...
        ...</pre>	

. . .	

Important Note: In the flattened export mode, tables metadata is saved in the header nodes of the export file. If the generated output file is imported into a different database using the XML/JSON Data Import utility, it can accurately recreate the original tables in the destination database using the saved metadata. This improves cross platform data transporting. In the hierarchical export mode the metadata is not saved. When importing data files without the metadata, SQL Assistant analyzes first 100 records of the actual data values and based on that analysis suggests column data types for the destination tables. Because the suggestions are based on a relatively small data sample, they might be sometimes incorrect. You should review them and correct as required before importing the data.

To generate multi-level XML or JSON file:

1. In the target editor open SQL Assistant menu or use the right-click menu for SQL Assistant's system tray icon. Navigate to **SQL Assistant → Import / Export Data** menu branch and then select **Export to XML or JSON Data Schema** menu. The **Export Data to XML or JSON Schema** dialog will appear.

2. In the **XML/JSON File** field enter the required output file name or use the Browse **[…]** button next to that field to select the output file suing the system's standard file browse dialog.

   Note: The output file format depends on the file extension. If you choose XML, the output data will be written in XML format. If you choose JSON, the output data will be written in JSON format. If you choose anything else, then XML format is assumed.

3. Uncheck the **Flat Export** checkbox.

4. Click the Plus Sign icon in the right top corner (to the right of the **Tables** list) to select tables to export. The Select Objects dialog will appear.

5. Select one or more tables and click the OK button. The **Export Data to XML or JSON Schema** dialog will be populated with the selected objects and the corresponding data structure.

6. Edit XML path values for each node in the **Path** column, for example, if you want to export Orders and Order Details tables, you can enter path for the OrderDetails table relative to the Order table
Path for Order table: root/Northwind/dbo/Orders
Path for Order Details table: root/Northwind/dbo/Orders/Order_Details

   Here is a sample of how it may look like in the Export Data to XML or JSON Schema settings
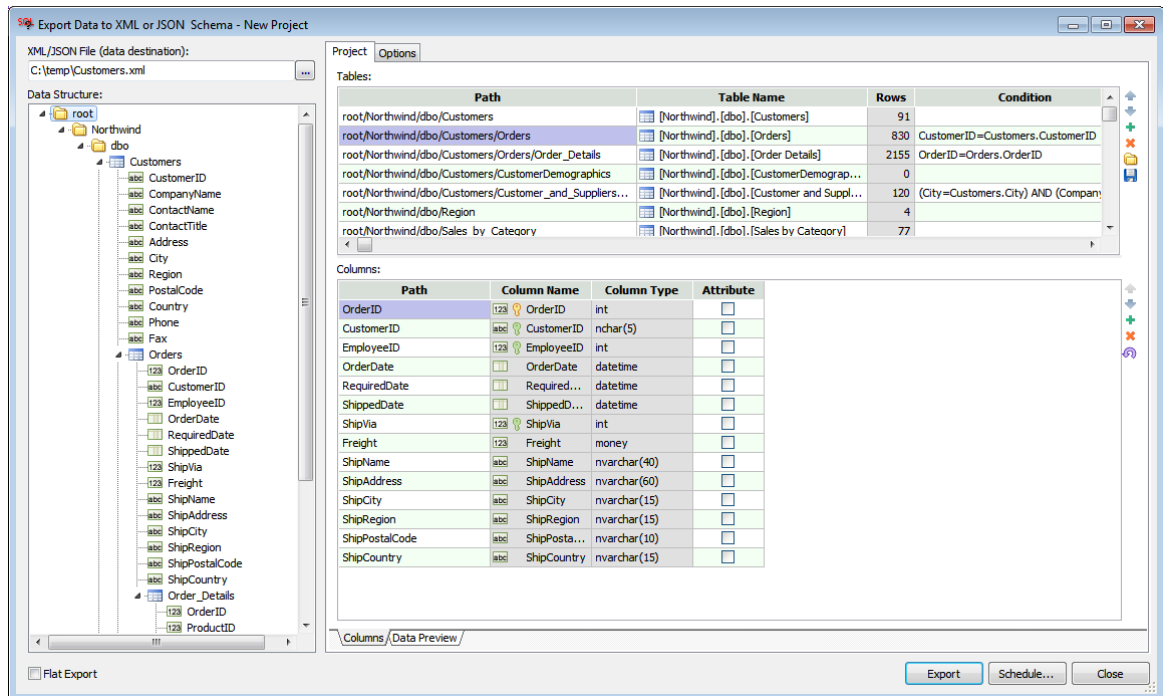
7. In the **Condition** column enter record linking conditions. Typically you would use the existing referential constraints to link records from child tables to parent tables. For example, if you want to export Orders and Order Details tables, you can link records in the Order Detail table as [Order Details].OrderID = [Order].OrderID.

    Important Notes: The record linking conditions are used in WHERE clauses of SQL queries SQL Assistant executes internally for retrieving table data. Failure to provide correct record linking conditions may lead to unpredictable results. It's very important to enter the correct conditions.

    Tip: The condition specified in the base table can be used to filter the export results. For example, in the sample setup pictured above, as a condition for the Customers table you can specify [Customers].City = 'London' to export customers residing in London.

8. Optional step - you can use the Arrow Up and Down icons next to the **Table** list to rearrange table order in the export file. The order of tables in the list impacts tables at the level only, it does not impact tables set as child tables. In the sample setup pictured above, moving Order Details table above Order table makes no difference.

    You can use the same icons in the **Columns** list to rearrange columns in the output.

    Note: The content of the **Columns** list matches the selected table in the **Tables** list. It is a master-detail view.
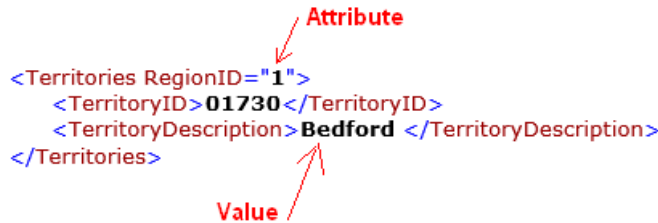
9. Optional step - you can use the Delete icons to remove tables or some of their columns from the export

10. Optional step – In the **Columns** list you can specify how to output column values to XML files. Every XML item can have one or two types of optional elements: attributes and a value. If the column's **Attribute** checkbox is not ticked, which is the default, every column value will be output as an XML value, otherwise as an attribute within XML element for the table row. The following example demonstrates how attributes and value are written to XML files.

The **Attribute** flag is not applicable to JSON export files and so it has no effect on the JSON output data format.

11. Optional step - on the **Options** tab choose data formatting, error handling, and logging options. Refer to XML Import/Export Options topic in this chapter for specific details.

12. Click the Save icon to save the project. You will be prompted for the project file name.

13. If you wish to run the export immediately, click the **Export** button.

If you wish to schedule a recurring export process, or a later run, use the **Schedule...** button, or use the **sacmd** command line utility provided with SQL Assistant. The command line for the export is
sacmd xmlexp:"<project-file>;<data-file>" conn:"connection-name"
In the command above replace <project-file> with the full file path name of the project settings file, replace <data-file> with the full path and name of the XML or JSON output file, replace <connection-name> with the name of preconfigured database connection saved in SQL Assistant options.

Tip: Use Open Project icon to load previously saved XML/JSON Export and XML/JSON Import projects into the dialog.

Scripting out Table Data

To script out INSERT statements for data from a database table or a database view

1. In your SQL script, right-click on the table or view that you want to script out. In the simplest case, use an empty editor and enter the required table or view name then right-click that name.

2. In the context menu, choose SQL Assistant menu branch, then in that branch select **Script Table Data…** command.

3. The **SQL Assistant – Table Data Export** dialog will appear on the screen
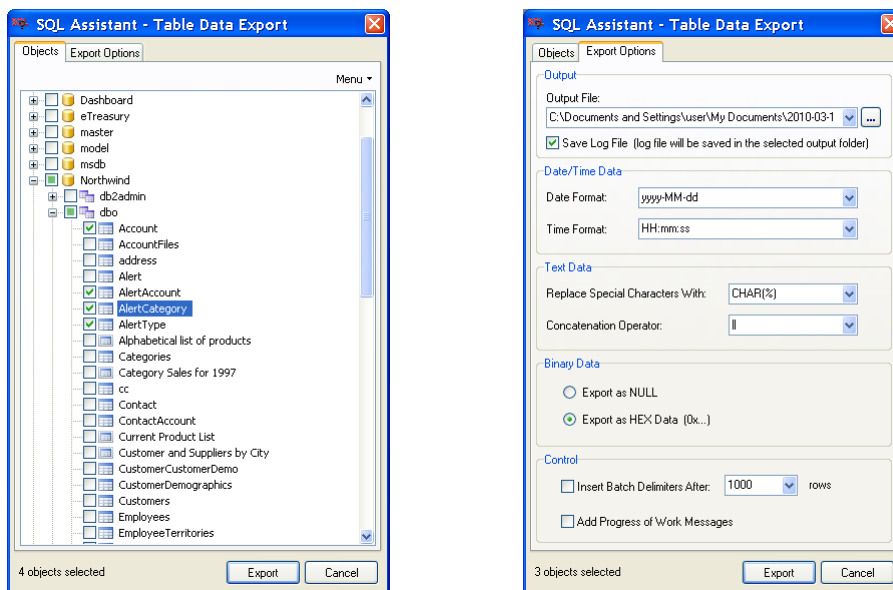
4. On the **Objects** tab, select one or more tables whose data you want to script out to a SQL file.

5. On the **Export Options** tab, specify the required output format properties for date, time, text and binary values.

    Important Note: that format properties are database type dependent and computer locale dependent. For example, the default format for date values in SQL server is locale specific. If a table data is scripted out on a computer located in the USA, using dates in MM/DD/YYYY format, the resulting SQL script cannot be executed on a SQL Server located in a European country where the required date format is DD/MM/YYYY. Another example is cross-platform data export. For example for Oracle databases the default date format is DD-MMM-YYYY. If you are going to export data from a SQL Server database and import it into an Oracle database, for the date format you should pick the format of the target Oracle system, which is DD-MMM-YYYY.

    See the following sections for details on supported scripted data format masks.

6. Click the **Export** button to start the process

Handling Date and Time Values

The **SQL Assistant – Table Data Export** dialog contains several predefined values for date and time format masks which you can choose from the **Date Format** and **Time Format** drop-down lists. If necessary, you can overwrite the predefined values and enter custom format mask values.

In case you need a custom format mask, use the following elements to construct a format mask. Note that if you use spaces to separate the elements in the format mask, these spaces will appear in the same location in the output string for the date value. The letters must be in uppercase or lowercase as shown in the following table (for example, "MM" not "mm"). Characters in the format mask that are enclosed in single quotation marks will appear in the same location and unchanged in the output string.

Value	Description
D	Day of month as digits with no leading zero for single-digit days.
Dd	Day of month as digits with leading zero for single-digit days.
Ddd	Day of week as a three-letter abbreviation. The function uses the LOCALE_SABBREVDAYNAME value associated with the specified locale.
Dddd	Day of week as its full name. The function uses the LOCALE_SDAYNAME value associated with the specified locale.
M	Month as digits with no leading zero for single-digit months.
MM	Month as digits with leading zero for single-digit months.
MMM	Month as a three-letter abbreviation. The function uses the LOCALE_SABBREVMONTHNAME value associated with the specified locale.
MMMM	Month as its full name. The function uses the LOCALE_SMONTHNAME value associated with the specified locale.
y	Year as last two digits, with a leading zero for years less than 10. The same format as "yy."
yy	Year as last two digits, with a leading zero for years less than 10.
yyyy	Year represented by full four digits.
gg	Period/era string. The function uses the CAL_SERASTRING value associated with the specified locale. This element is ignored if the date to be formatted does not have an associated era or period string.

Value	Description
h	Hours with no leading zero for single-digit hours; 12-hour clock
hh	Hours with leading zero for single-digit hours; 12-hour clock
H	Hours with no leading zero for single-digit hours; 24-hour clock
HH	Hours with leading zero for single-digit hours; 24-hour clock
m	Minutes with no leading zero for single-digit minutes
mm	Minutes with leading zero for single-digit minutes
s	Seconds with no leading zero for single-digit seconds
ss	Seconds with leading zero for single-digit seconds
t	One character time marker string, such as A or P
tt	Multicharacter time marker string, such as AM or PM

Note: When choosing the output data and time formats, keep in mind that the formats must be compatible and understood by the target database server, not the source server. In other words, understood by the server against which you are going to run the generated SQL script.

Handling Special Symbols in Text-based Values

In many cases, special symbols contained in the text-based data values cannot be scripted out as is. In such cases, SQL Assistant automatically generates appropriate SQL expressions in the output SQL script. Special symbols include all symbols not represented as digits and letters in the standard ASCII characters table.

You can choose one of the predefined functions CHAR(%) or CHR(%), or you can enter your own. Note that you can also refer to user-defined SQL functions. The % symbols in the format mask represent ASCII code placeholder. This symbol is required.

For example, text-based value 'Peter<end-of-line>Pan' contains end-of-line symbol having ASCII code 10 in the middle of the value. If you choose to use CHAR(%) function and || concatenation operator, the output SQL file will have that value scripted as 'Peter' || CHAR(10) || 'Pen'

Note: The format of text- based expressions is database type dependent. When choosing the output format, keep in mind that the format must be compatible and understood by the target database server, not the source server. In other words, understood by the server against which you are going to run the generated SQL script.

Handling Binary Data Values

Because SQL scripts do not support binary data in row formats, binary data must be converted to some other format or be ignored. SQL Assistant offers the following 2 options for scripting out binary data:

- **Export as NULL** – in other words, ignore binary data values, and in the generated scripts replace them with NULL values.

- **Export as HEX Data (0x**…) – encode binary values as HEX code strings. In this case, the target DMS must be able to understand HEX code values and be able to convert them back to binary values.

Importing Data from Excel and Flat Files

To import data from one or more worksheets from Excel file or any other file that can be opened in Excel.

1. Right-click in the target editor and in the context menu choose **SQL Assistant** menu branch then expand **Import / Export Data** menu branch and select **Import CSV, Excel, Other Files… …** menu command. Alternatively you can invoke the same command from SQL Assistant's system tray icon right-click menu. The **Import Data** project wizard dialog will appear. Click the **Next** button. The standard system Open File dialog will appear on the screen.

2. Browse and select a file that can be opened in Microsoft Excel. You can select XLS, XLSX, CSV, TXT, XML or a file with other extension supported by Excel. As long as the data file can be opened in Excel, its content can be loaded into the database.

3. Wait for a few seconds while Excel is reading the selected file. The larger the file, the longer it may take to load it. Once the content is loaded into Excel, SQL Assistant will popup **Import Excel Data** dialog. See Import Excel Data Dialog sub topic for detailed description of the dialog, its usage and

controls.

4. In the **Tables** table choose worksheets to import and map them to new or existing database tables.

5. In the **Tables** table click the first selected worksheet and in the **Columns** table choose columns to import. Map worksheet columns to database columns and if necessary adjust their data types.

6. Repeat step 5 for other selected worksheets. When done with the table and column mapping, proceed to the next step.

7. To start the data import immediately, click the **Import** button. To postpone the load for later processing or manual execution, click the **Save Script** button to generate SQL script with the required INSERT and other statements.

Important Note: Microsoft Excel must be installed on your computer in order to use that function

Import Excel Data Dialog Usage and Controls

The following sample screenshot demonstrates Import Excel Data dialog..
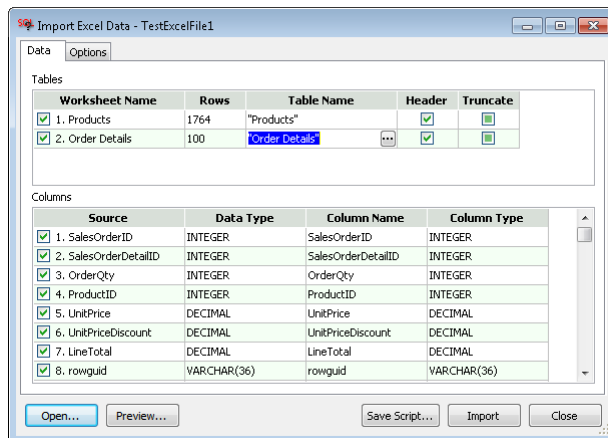
The dialog contains two tabs:

The **Data** tab provides you with a list of loaded Excel worksheets and their columns. It enables you to map worksheets to existing or new database tables, choose worksheets and columns to import into the database.

1. **Tables** table **–** this table contains list of worksheets in the chosen Excel file and their mapping to database tables. To select which worksheets to load, use the left-most column with checkboxes.

    Worksheet Name column **–** this informational column contains Excel worksheet number and label which can be used to visually identify worksheet in the chosen Excel file.

    Rows column –this informational column indicates how many rows are available in the worksheets and can be loaded into the database.

    Table name column **–** the value in this column references target table name in the database. To change the name, click on the relevant table cell, the value in the sell should appear highlighted. You can type or paste table name or you can click the browse **[…]** button displayed next to the table name to select the target table using graphical **Select Table** dialog. Note that the browse **[…]** button appears only when the table name cell is focused. If the required target table does not exist, simply type its name in the column cell. SQL Assistant will create that table automatically.

    Tip: To create a table in non-current schema or database, specify the complete fully qualified

table name.

Header column **–** tick checkbox in the Header column if the worksheet contains a header row with column names and that row should be skipped.

🦎 **Tip**: SQL Assistant recognizes single row headers only. If your Excel file contains multi-row headers or irregularly shaped headers, copy worksheet data into new worksheet and format header rows as required for the import.

Truncate column **–** tick checkbox in the Truncate column if loading data into an existing table and you want the table content to be replaced.

2. **Columns** table **–** this table contains columns found in the selected worksheet and their mapping to database table columns. Note that the content of the **Columns** table matches the selected worksheet in the **Tables** table. It is a master-detail view.

Source column **–** the number and label of the Excel column within the selected worksheet. In case the Excel worksheet has no header row and the **Header** checkbox is not ticked in the **Tables** column, SQL Assistant will use alphabetic column names, for example, A, B, C, D

🦎 **Tip**: You can use the checkbox to the left of the **Source** column to select which worksheet columns to import and which to skip.

Data Type column **–** this informational column indicates column data type in Excel file based on SQL Assistant's analysis of the worksheet data.

Column Name column **–** the names in this column reference columns in the selected target table. In case the selected worksheet is mapped to a non-existing table, you would need to enter new column names or use the suggested names. In case the selected worksheet is mapped to an existing table, a drop-down list will appear when you click the Column Name cell and which you can use to map worksheet column to an existing table columns.

The drop-down list is populated with columns from the selected table
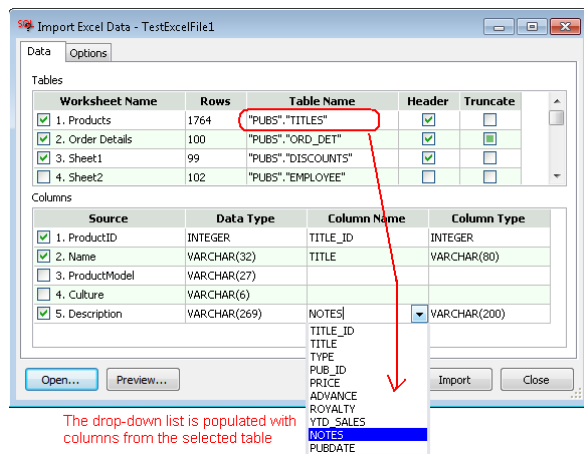
Column Type column **–** the data types in this column reference data types of the matching columns in the selected target table. In case the selected worksheet is mapped to a non-existing table, you would need to enter the required data type or use the suggested data type. In case the selected worksheet is mapped to an existing table, the values in the **Column Type** cannot be changed manually; they are populated with the data types of the mapped table column picked in the **Column Name** cells.

The **Options** tab provides additional data processing, error handling, and logging options.

Save Log File -**–** If this option is checked, SQL Assistant writes processing status messages to the log file specified in the **Log** property.

Log – The name of the output log file. This name must be specified if **Save Log Option** is checked.

Commit Inserted Data After n Rows – This option is effective with database connections that require

explicit commits for the inserted data. The option value controls batch update size. Frequent commits lead to small batches which can potentially affect the data import process performance. In comparison, large batches can potentially cause large database transactions which in turn require more space in database transaction logs and rollback segments.

This option has no impact on database connections with automatic commits after each change. Note that by default, SQL Server, Sybase ASE, Sybase ASA, and MySQL connections have automatic commit enabled, while Oracle, DB2, and PostgreSQL connections do not.

If this option is not checked, SQL Assistant does not generate COMMIT statements, during the processing.

Stop Importing Data After – This option controls what SQL Assistant should do in case an error occurs during the data generation process. The available choices are:

- **Any error** – if any kind of error occurs, the data import project run stops immediately.

- **All rows of a table failed** – if data import fails for one of the tables in the import project, and not a single row could be inserted successfully, the data import project run stops, otherwise it continues with the next table in the project.
  Warning: When this option is selected, SQL Assistant attempts to import all available rows, and it reports an error for every failed row.

- **Stop processing table on error** – if data import fails for a row for one of the tables in the import project, stop processing that table and continue with the next one. This option enables you to skip tables that cannot be loaded correctly, and do not attempt to load them partially.

If **Stop Importing Data After** option is not checked, SQL Assistant ignores all errors and continues running the process after an error.

Disable Triggers – This option instructs SQL Assistant to disable triggers on the destination tables before the import process starts and then enable them after completion. This option can greatly improve data import performance for tables with triggers. However, if triggers carry on any business logic such as cascading table updates of automatic data corrections that will not happen for the imported data rows. This option has no effect on tables without triggers.

Disable Constraints – This option instructs SQL Assistant to disable foreign key constraints on the destination tables before the import process starts and then enable them after completion. This option can greatly improve data import performance for tables with referential constraints. However, it may allow unreferenced data to be inserted into the destination tables. This option has no effect on tables without foreign key constraints.

Date Format – Date value format understood by your database. It is very important that you select correct date format for your database. An incorrect date format may lead to invalid data inserted into the database.

Tip: Note that the date format is used for loading the data into database tables. SQL Assistant uses it internally for formatting date and date-time values in SQL INSERT commands. This option is not used for reading Excel files, which is done by Excel and not controlled by SQL Assistant.

Time-Format – Time value format understood by your database. It is very important that you select correct time format for your database. An incorrect time format may lead to invalid data inserted into the database.

Tip: Note that the time format is used for loading the data into database tables. SQL Assistant uses it internally for formatting date and date-time values in SQL INSERT commands. This option is not used for reading Excel files, which is done by Excel and not controlled by SQL Assistant.

Decimal-Separator – Decimal-separator symbol understood by your database. It is very important that you select correct symbol for your database. An incorrect value may lead to invalid data inserted into the database.

Tip: Note that the symbol is used for loading the data into database tables. SQL Assistant uses it internally for formatting decimal values in SQL INSERT commands. The option is not used for reading Excel files, which is done by Excel and not controlled by SQL Assistant.

See Handling Date and Time Values topic in CHAPTER 13, Scripting, Exporting, and Importing Data, for details on supported date and time value format masks.

To preview the data to be imported you can use the **Preview...** button. This will open the selected file in Microsoft Excel.

To open a different file for data import use the **Open...** button. This will display the standard File Open dialog you can use to select a different source data file.

After you are done with table and table column mappings, click the **Import** button to start the import process.

Importing Data from Apache Parquet Files

To import data from Apache Parquet files (PARQUET):

1. Right-click in the target editor and in the context menu choose **SQL Assistant** menu branch then expand **Import / Export Data** menu branch and select **Import CSV, Excel, Other Files… …** menu command. Alternatively you can invoke the same command from SQL Assistant's system tray icon right-click menu. The **Import Data** project wizard dialog will appear. Click the **Next** button. The standard system Open File dialog will appear on the screen.

2. Browse and select one or more Parquet files.

3. Wait for a few seconds while SQL Assistant to read file headers.

4. In the **Tables** table map the selected files to new or existing database tables.

5. In the **Tables** table click selected a file in the list and in the **Columns** table choose columns to import. Map file columns to database columns and if necessary adjust their data types.

6. Repeat step 5 for other files. When done with the table and column mapping, proceed to the next step.

7. To start the data import immediately, click the **Import** button. To postpone the load for later processing or manual execution, click the **Save Script** button to generate SQL script with the required INSERT and other statements.

Importing Tables, Schemas, and Databases from XML and JSON Files

Overview

XML/JSON Import utility can be used for importing data from hierarchical XML and/or JSON files. The XML/JSON data can be produced by the Export feature described in Exporting Data to XML and JSON Files topic, as well as by other applications. The data import can be run immediately or scheduled to run later using SQL Assistant's command line utilities.

The utility provides simple to use graphical interface for mapping XML schema elements to database tables and data. The data can be loaded into existing tables or new tables that will be created dynamically if they do not yet exist in the database.

Notes:

- The amount of work required to map the data greatly depends on how closely the data in input XML file matches your database schema and tables.

- In cases the data needs to go into existing data tables but the structure of XML file cannot be mapped to the existing database schema, you would need to load that data into intermediate staging tables and then write and execute SQL code to copy the loaded data from staging tables to data tables applying the necessary data transformations as required.

Importing XML Data Schema

Before you can load data from an XML file, you need to map XML file structure to tables and columns in your database. This can be done graphically using the Import XML data utility which allows mapping XML nodes to tables in your database, choosing which XML elements to import and which not, mapping referential constrains and data types.

We will use the following sample XML file to demonstrate the data mapping process.

```xml
<?xml version="1.0" encoding="utf-8"?>
<Root>
  <Customers>
    <Customer CustomerID="GREAL">
      <CompanyName>Great Lakes Food Market</CompanyName>
      <ContactName>Howard Snyder</ContactName>
      <ContactTitle>Marketing Manager</ContactTitle>
      <Phone>(503) 555-7555</Phone>
      <FullAddress>
        <Address>2732 Baker Blvd.</Address>
        <City>Eugene</City>
        <Region>OR</Region>
        <PostalCode>97403</PostalCode>
        <Country>USA</Country>
      </FullAddress>
    </Customer>
    <Customer CustomerID="HUNGC">
      <CompanyName>Hungry Coyote Import Store</CompanyName>
      <ContactName>Yoshi Latimer</ContactName>
      <ContactTitle>Sales Representative</ContactTitle>
      <Phone>(503) 555-6874</Phone>
      <Fax>(503) 555-2376</Fax>
      <FullAddress>
        <Address>City Center Plaza 516 Main St.</Address>
        <City>Elgin</City>
        <Region>OR</Region>
        <PostalCode>97827</PostalCode>
```

```
            <Country>USA</Country>
          </FullAddress>
        </Customer>
    </Customers>
    <Orders>
      <Order>
        <CustomerID>GREAL</CustomerID>
        <EmployeeID>6</EmployeeID>
        <OrderDate>1997-05-06T00:00:00</OrderDate>
        <RequiredDate>1997-05-20T00:00:00</RequiredDate>
        <ShipInfo ShippedDate="1997-05-09T00:00:00">
          <ShipVia>2</ShipVia>
          <Freight>3.35</Freight>
          <ShipName>Great Lakes Food Market</ShipName>
          <ShipAddress>2732 Baker Blvd.</ShipAddress>
          <ShipCity>Eugene</ShipCity>
          <ShipRegion>OR</ShipRegion>
          <ShipPostalCode>97403</ShipPostalCode>
          <ShipCountry>USA</ShipCountry>
        </ShipInfo>
      </Order>
      <Order>
        <CustomerID>GREAL</CustomerID>
        <EmployeeID>8</EmployeeID>
        <OrderDate>1997-07-04T00:00:00</OrderDate>
        <RequiredDate>1997-08-01T00:00:00</RequiredDate>
        <ShipInfo ShippedDate="1997-07-14T00:00:00">
          <ShipVia>2</ShipVia>
          <Freight>4.42</Freight>
          <ShipName>Great Lakes Food Market</ShipName>
          <ShipAddress>2732 Baker Blvd.</ShipAddress>
          <ShipCity>Eugene</ShipCity>
          <ShipRegion>OR</ShipRegion>
          <ShipPostalCode>97403</ShipPostalCode>
          <ShipCountry>USA</ShipCountry>
        </ShipInfo>
      </Order>
      <Order>
        <CustomerID>HUNGC</CustomerID>
        <EmployeeID>4</EmployeeID>
        <OrderDate>1997-07-16T00:00:00</OrderDate>
        <RequiredDate>1997-08-13T00:00:00</RequiredDate>
        <ShipInfo ShippedDate="1997-07-21T00:00:00">
          <ShipVia>1</ShipVia>
          <Freight>45.13</Freight>
          <ShipName>Hungry Coyote Import Store</ShipName>
          <ShipAddress>City Center Plaza 516 Main St.</ShipAddress>
          <ShipCity>Elgin</ShipCity>
          <ShipRegion>OR</ShipRegion>
          <ShipPostalCode>97827</ShipPostalCode>
          <ShipCountry>USA</ShipCountry>
        </ShipInfo>
      </Order>
    </Orders>
</Root>
```

Here are the steps:

1. In the target editor open SQL Assistant menu or use the right-click menu for SQL Assistant's system tray icon. Navigate to SQL Assistant → Import / Export Data menu branch and then select Import XML Data Schema menu. The **Import XML Data** dialog will appear.
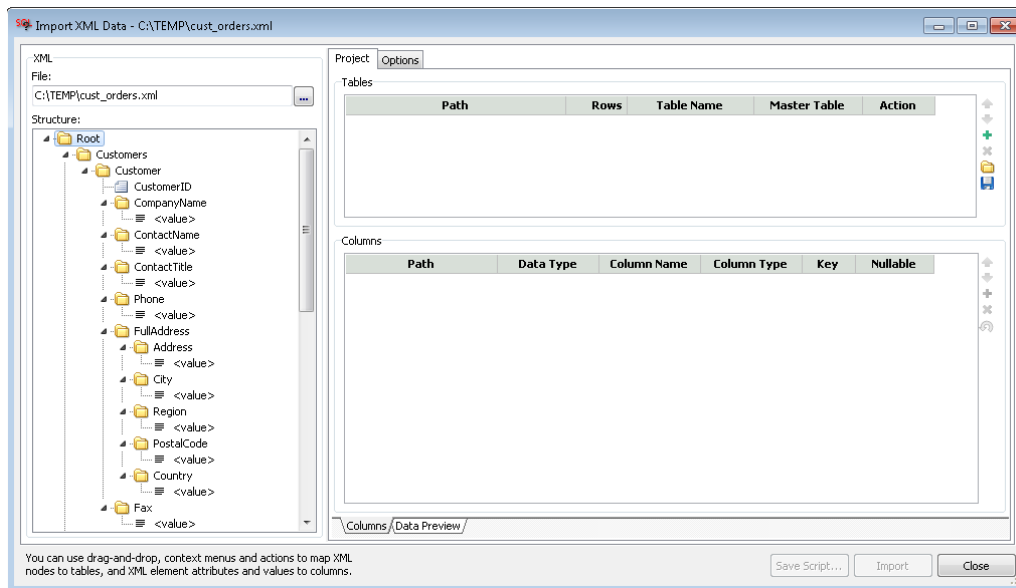
2. In the File field enter input file name or you can use the Browse **[…]** button next to that field to select the input file.

3. To map the Customer data, drag-and-drop the "Customer' node from the XML **Structure** tree to the Tables list, you can also select the node and click the Plus Sign icon in the right top corner (to the right of the Tables list) to add "Customer" table to the list. Note that the appearance of the **Structure** tree will change and the mapping of the "Customer" node and its elements will be displayed as tables and columns. SQL Assistant will analyze the data and detect columns data types.

4. Optional step: Click "Customer" value in the **Table Name** column in the **Tables** list. The table cell will become editable. You can now enter fully qualified table name in the database to map that XML data to an existing database table, or you can use the Browse button displayed in the focused cell **[…]** to graphically browse and select the required target table.
    Note If you do not map the selected node to an existing table, the import is run, SQL Assistant will automatically create new table with the specified name. If the table name does not contain schema and database name, it will be created in the current schema derived from the database connection context.

5. Use the **Columns** list to map XML elements of the selected node and table to actual columns in the database table.
    Notes:
    *   Element names in the XML can different from column names in the table.

    *   Columns that cannot be mapped should be deleted from the **Columns** list.

    *   In case the mapping is for a new table to be created by the import utility, you can click on column names in the **Columns** list and rename them as required. You can also change their data types if that is required.

    *   for Order need contains Customers and Orders nodes which are at the same level. To maintain the internal referential integrity, you would need to add CustomerID

6. Repeat step 3, 4, and 5 for FullAddress, Order, and ShipInfo nodes.

7. Note that in certain cases you may need to add additional columns to Columns list. For example, in the sample XML file described here the Order node does not contain any unique identifier. To correctly link ShippingInfo data to Orders, you would to add "OrderID" column to the Order table column list and set the value of **Key** column cell to "Primary." You would also need to add "OrderID" column to ShippingInfo column list and set the value of **Key** column cell to "Foreign,"

8. Use the **Master Table** column in the Tables list to set the relations between child and parent tables. The resulting settings should look similar to what is shown on the next screenshot.
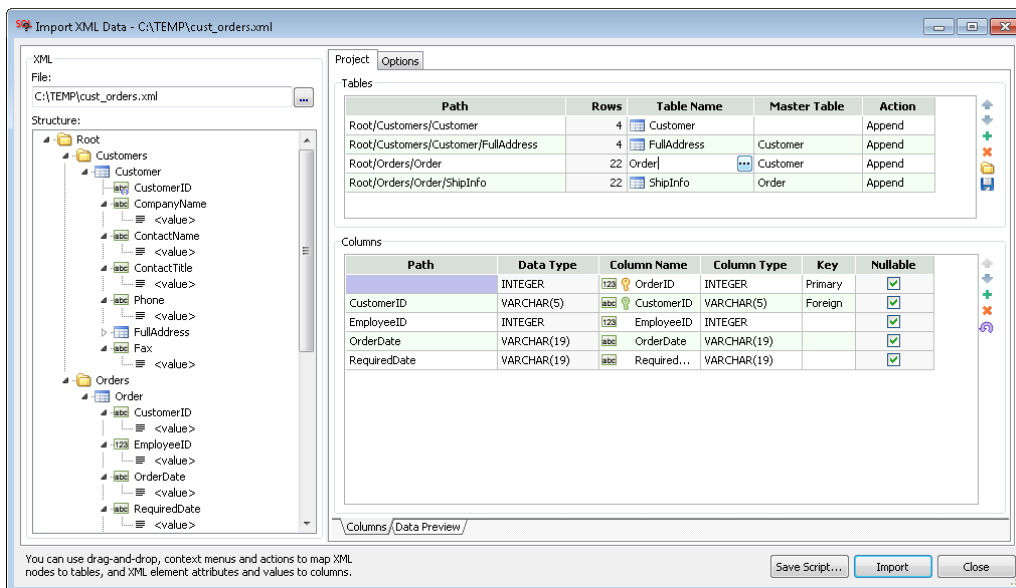
9. In case the data is being imported into existing tables the **Action** column in the Tables list is used to instruct the import utility what to do with the existing data. Use the *Append* action to append data to the table, *Truncate* – to truncate it before loading new data, or *Skip not Empty* - to skip that table if it is not empty.

10. Optional step - on the **Options** tab choose data formatting, error handling, and logging options. Refer to XML Import/Export Options topic in this chapter for specific details.

11. Click the Save icon to save the project. You will be prompted for the project file name.

12. If you wish to run the import immediately, click the **Import** button.

    If you wish to schedule a recurring import process, or a later run, use the **Schedule** button, or use the **sacmd** command line utility provided with SQL Assistant. The command line for XML Export is
    sacmd xmlimp:"<project-file>;<data-file>" conn:"connection-name"
    In the command above replace <project-file> with the full file path name of the project settings file, replace <data-file> with the full path and name of the XML input file, replace <connection-name> with the name of preconfigured database connection saved in SQL Assistant options.

    You can also use the **Save Script** button to generate SQL script with INSERT statements. However, please be aware that the content of the generated SQL script will be based on the state of the database relative to the point in time the script is generated, not the point in time it is run. The content of database tables may differ and the script results may vary depending on the database state.

Tip: Use Open Project icon to load previously saved XML Export and XML Import projects into the dialog.

XML and JSON Import/Export Options

The **Options** tab on the XML Import and Export dialogs provides the following data processing, error handling, and logging options.

Save Log File –- If this option is checked, SQL Assistant writes processing status messages to the log file specified in the **Log** property.

Log – The name of the output log file. This name must be specified if **Save Log Option** is checked.

Commit Inserted Data After n Rows – This option is effective with database connections that require explicit commits for the inserted data. The option value controls batch update size. Frequent commits lead to small batches which can potentially affect the data import process performance. In comparison, large batches can potentially cause large database transactions which in turn require more space in database transaction logs and rollback segments.

This option has no impact on database connections with automatic commits after each change. Note that by default, SQL Server, Sybase ASE, Sybase ASA, and MySQL connections have automatic commit enabled, while Oracle, DB2, and PostgreSQL connections do not.

If this option is not checked, SQL Assistant does not generate COMMIT statements, during the processing.

Stop Processing Data After – This option controls what SQL Assistant should do in case an error occurs during the data import process. The available choices are:

- **Any error** – if any kind of error occurs, the data import project run stops immediately.

- **All rows of a table failed** – if data import fails for a table, and not a single row could be inserted successfully, the data import project run stops, otherwise it continues with the next table in the project.

If this option is not checked, SQL Assistant ignores all errors and continues running the process after an error.

Create Tables – This option is available for data import operations only. The available choices are:

- **Never** – Do not check for existence of target tables and do not execute CREATE TABLE commands for target tables. If a target table does not exist, the data import operations for that table fail. This option ensures that new tables are not created during the data import operations.

- **Always** – Do not check for existence of target tables and always execute CREATE TABLE commands. If a target table already exists, the CREATE TABLE command for that table fails, but the data import operations continue running. This option ensures that all target tables are created during the data import operations.

- **If not Exists** – Check for existence of target tables and execute CREATE TABLE commands for tables that do not exist. If a target table already exists, the CREATE TABLE command for that table fails, but the data import operations continue running. This option ensures that new tables are not created during the data import operations.

    Important Notes: If not Exists option can be used with SQL Server, Sybase ASE, Sybase ASA, MySQL, Oracle, and PostgreSQL version 9.1 or later database servers.
    It is not supported for DB2 (all versions), Microsoft Access, and PostgreSQL version before 9.1.

Date Format – Date value format understood by your database. It is very important that you select correct date format for your database. An incorrect date format may lead to invalid data inserted into the

database.

![Tip icon] **Tip**: Note that the date format is used for loading the data into database tables. SQL Assistant uses it internally for formatting date and date-time values in SQL INSERT commands.

Time-Format – Time value format understood by your database. It is very important that you select correct time format for your database. An incorrect time format may lead to invalid data inserted into the database.

![Tip icon] **Tip**: Note that the time format is used for loading the data into database tables. SQL Assistant uses it internally for formatting date and date-time values in SQL INSERT commands.

Decimal-Separator – Decimal-separator symbol understood by your database. It is very important that you select correct symbol for your database. An incorrect value may lead to invalid data inserted into the database.

![Tip icon] **Tip**: Note that the symbol is used for loading the data into database tables. SQL Assistant uses it internally for formatting decimal values in SQL INSERT commands.

See Handling Date and Time Values topic in CHAPTER 13, Scripting, Exporting, and Importing Data, for details on supported date and time value format masks.

Bulk Loading Data into Schemas and Databases

Overview

The Data Factory utility provides simple and intuitive ETL-like interface for bulk loading entire schemas and databases, right from your favorite database development tool, making data integration easier and faster. It does not require installations of additional ETL servers and other middleware, it runs directly on your computer. The Data Factory is an automation utility that automates and orchestrates data processing tools supported by SQL Assistant

The Data Factory enables you to load data into database tables using several different methods listed below::

- Copying data from other databases and database servers - see Using Data Transfer Utility topic for specific details on the supported data formats and options.

- Copying data from flat file – see Importing Data from Excel and Flat Files topic for more details on the supported file formats and options.

- Using SQL scripting to create new data sets using from other schemas and tables – see CHAPTER 14, Executing SQL Scripts for additional details.

- Using Test Data Generator to create data samples for application development and testing – see CHAPTER 18, Generating Test Data for additional details.

The Data Factory supports automatic chained execution of Data Sanitizer projects, which can be used in situations when production data is loaded into test and development environments and it needs to be sanitized for non production use.. For more details about data sanitization and obfuscation see CHAPTER 40, Sanitizing and Obfuscating Database Data.

To launch the Data Factory Wizard,

1. In your SQL Editor select SQL Assistant's menu, and then select **Import/Export Data -> Data Factory...** menu. The **Data Factory Project Wizard** dialog will appear.

2. Choose **New Project - Guide me Step by Step** option and follow the instructions.

Copying Data Between Database Servers

Overview

Several methods are supported by SQL Assistant for copying data between databases and database servers. All supported methods enable you to copy data between database servers of the same or different types. Each method has its own unique advantages. Different methods can be used in different situations. The following topics describe the supported methods and their differences.

Method 1 – Using Data Transfer Utility

There are several advantages of using the Data Transfer Utility method over other methods described in this chapter.

- The Data Transfer utility can automatically create destination tables if they do not already exist.

- The Data Transfer project can be configured to clear destination tables before the new data inserted, or SIMPLY append the new data. It can also be configured to skip all tables in the destination database if they already exist and non-empty.

- The Data Transfer project can be configured to copy entire tables or small data samples only.

- This method allows copying data from selective columns only.

- This method allows copying data to system generated columns such as IDENTY, SERIAL, AUTO_INCEREMENT and preserving the original values from the data source. This is very important when copying data from / to tables with existing referential integrity constraints.

- For heterogeneous data transfers between different types of database servers, the Data Transfer utility automatically maps source data types to compatible target data types and performs required data conversions during the data transfer process.

- Both tables and views can be selected as the data sources and the data destination objects.

- Graphical and command line interfaces are available for running the data transfer.

To copy data into a different database using the Data Transfer utility:

1. In your SQL Editor select SQL Assistant's menu, and then select **Import/Export Data -> Transfer Data...** menu. The **Transfer Data** dialog will appear.
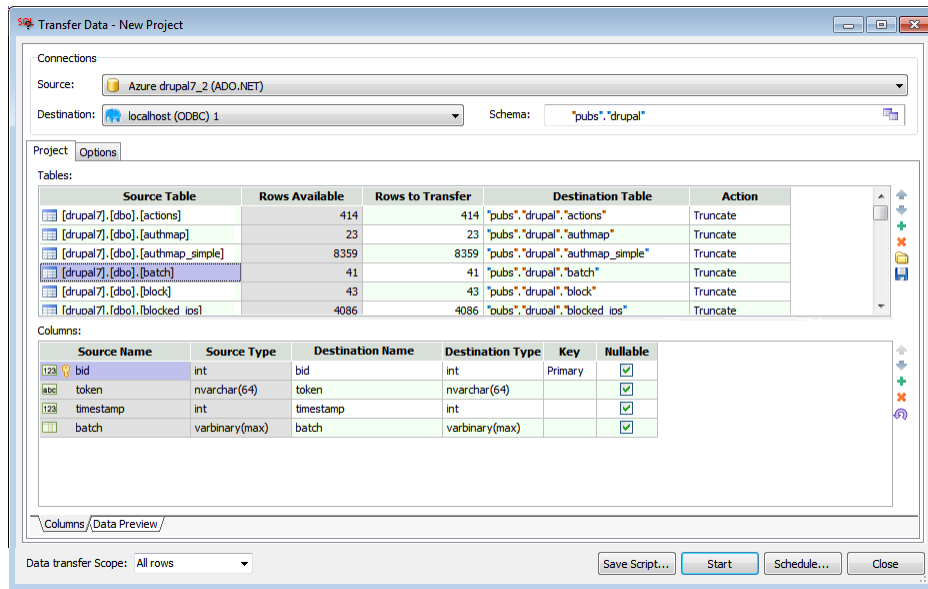
2. In the **Connections** box, in the **Source** drop-down list select the data source database server connection. The source connection must be either a current connection in your SQL editor or one of the previously configured and saved connection profiles. In the **Destination** drop-down list select the destination database server connection. The destination connection must be either a current connection in your SQL editor or one of the previously configured and saved connection profiles. In the **Schema** control click the icon. The schema selection dialog will appear. Select the destination database and schema for the data transfer operations.

   Note that the definitions of database connections available for use with the **Data Transfer** utility are shared with connections associated with target editors. Just like other database connections, they can be managed using SQL Assistant's main Options dialog. See the Managing Database Connections topic in CHAPTER 48 for more information.

3. In the **Project** box click the green plus sign ✚ icon on the toolbar displayed to the right of the **Tables** grid. The Select Objects dialog will appear. Select data sources tables and/or views whose data you want to copy to the other database.

4. By default the **Tables** grid is populated with destination table mappings having destination table names the same as in the source database. If you want to change the mappings, click twice the destination table name. The grid cell with the table name switches to edit mode. You can now edit the table name, or you can use the browse [...] button available at the right edge of the field to graphically browse the destination database and select an existing table or view.

   Note: If you pick a view for the data destination, make sure that view is updatable.
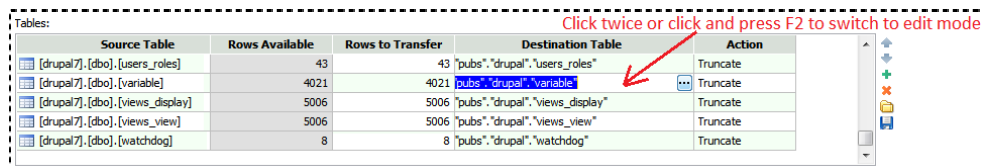
   Use the Arrow Up and Down icons on the toolbar displayed to the right of the **Tables** grid to arrange the order of tables in the list. The order is important if you need to populate existing tables with referential constraints. Tables with primary key records should be populated before tables with foreign key records, for example, Sales Orders table should be listed above the Order Details table.

5. If you only want to copy only a subset of rows from the source tables, edit values in the **Rows to Transfer** column as required. The values in the **Rows to Transfer** column must be less or equal the values in the **Rows Available** column.

    Tip: To quickly set values in the **Rows to Transfer** for all tables to the same value, use the **Data-transfer Scope** combo control available at the bottom of the **Transfer Data** dialog. The **Data-transfer Scope** is populated with several common values. You can enter a custom numeric value if required. Note that the actual value set is a lesser of the selected data sample or the value in the **Rows Available** column.

6. Select the action for the data transfer.

    - Select **Truncate** action for destination tables you want to clear before the new data is inserted.
    - Change this action to **Drop and Create** for tables which you want to recreate. This action should be used for tables having different structure or incompatible data types.
    - Change this action to **Append** for tables in which you want to preserve the existing data rows.
    - To ignore already populated tables, select **Skip non Empty** action.

7. To verify and update mappings for individual columns, click each table in the **Tables** list and update destination columns and data types as required. To edit a particular cell in the **Columns** grid, double-click that cell to switch to the edit mode. To add and delete columns, use the green plus sign icon and the red X icons on the toolbar displayed to the right of the **Columns** grid. The order of columns in the column mapping is not important for existing destination tables. However, if the selected destination table does not exist, it will be created having columns ordered as they are specified in the column mapping.

    Important Note: Selected data types of the destination columns must be compatible with the destination database type. If you are copying data between different database types, make sure you enter correct data types for the destination columns or the data transfer operation would fail. Entering correct data types in the destination table is very important for both new and existing tables.

    Note: The Data Transfer utility automatically maps source data types to compatible destination data types taking into account the original data type, column length, scale, and precision. In certain cases, if the source data type length or precision are not supported by the destination database type, it may pick a more generic data type with more capacity. For example, if the source data type in your SQL Server database is VARCHAR(5000) and the target database type is Oracle, which is supporting only up to 4000 characters in VARCHAR2 data type columns, then CLOB data type is used for the destination column. For all data types that do not have a compatible equivalent in the destination database, the Data Transfer utility uses by default VARCHAR(255) data type for the target column. For example, when copying values from an ARRAY type column in PostgreSQL database to SQL Server, it will attempt to convert values in the array column to a value-delimited string and save the resulting string in the destination column. You should always carefully review the **Columns** mapping before saving the project and running the process.

8. Switch to the **Options** tab and complete other data transfer project settings as required. The supported options and their usage are described in the following section.

9. After you complete table and column mapping and complete other options it is recommended that you click the Save icon on the toolbar displayed to the right of the **Tables** grid to save the data transfer project settings.

10. Click the **Start** button to run the data transfer project immediately or click the **Schedule...** button to schedule a later run. You can also click the **Save Script...** button to have the Data Transfer utility generate a SQL script based on your selection, which you can review and execute later.

The **Options** tab provides additional data processing, error handling, and logging options.

Save Log File -– If this option is checked, SQL Assistant writes processing status messages to the log file specified in the **Log** property.

Log – The name of the output log file. This name must be specified if **Save Log Option** is checked.

Commit Inserted Data After n Rows – This option is effective with database connections that require explicit commits for the inserted data. The option value controls batch update size. Frequent commits lead to small batches which can potentially affect the data import process performance. In comparison, large batches can potentially cause large database transactions which in turn require more space in database transaction logs and rollback segments.

This option has no impact on database connections with automatic commits after each change. Note that by default, SQL Server, Sybase ASE, Sybase ASA, and MySQL connections have automatic commit enabled, while Oracle, DB2, and PostgreSQL connections do not.

If this option is not checked, SQL Assistant does not generate COMMIT statements, during the processing.

Stop Processing Data After – This option controls what SQL Assistant should do in case an error occurs during the data transfer process. The available choices are:

- **Any error** – if any kind of error occurs, the data transfer project run stops immediately.

- **All rows of a table failed** – if data transfer fails for a table, and not a single row could be inserted successfully, the data transfer project run stops, otherwise it continues with the next table in the project.
  Warning: When this option is selected, SQL Assistant attempts to copy all available rows, and it reports an error for every failed row.

- **Stop processing table on error** – if data transfer fails for a row for one of the tables in the data transfer project, stop processing that table and continue with the next one. This option enables you to skip tables that cannot be loaded correctly, and do not attempt to load them partially.

If this option is not checked, SQL Assistant ignores all errors and continues running the process after an error.

Disable Triggers – This option instructs SQL Assistant to disable triggers on the destination tables before the data transfer process starts and then enable them after completion. This option can greatly improve data transfer performance for tables with triggers. However, if triggers carry on any business logic such as cascading table updates of automatic data corrections that will not happen for the copied data rows. This option has no effect on tables without triggers.

Disable Constraints – This option instructs SQL Assistant to disable foreign key constraints on the destination tables before the data transfer process starts and then enable them after completion. This option can greatly improve data transfer performance for tables with referential constraints. However, it may allow unreferenced data to be inserted into the destination tables. This option has no effect on tables without foreign key constraints.

Date Format – Date value format understood by your database. It is very important that you select correct date format for your database. An incorrect date format may lead to invalid data inserted into the database.

Tip: Note that the date format is used for loading the data into database tables. SQL Assistant uses it internally for formatting date and date-time values in SQL INSERT commands.

Time-Format – Time value format understood by your database. It is very important that you select correct time format for your database. An incorrect time format may lead to invalid data inserted into the database.

Tip: Note that the time format is used for loading the data into database tables. SQL Assistant uses it internally for formatting date and date-time values in SQL INSERT commands.

Decimal-Separator – Decimal-separator symbol understood by your database. It is very important that you select correct symbol for your database. An incorrect value may lead to invalid data inserted into the database.

Tip: Note that the symbol is used for loading the data into database tables. SQL Assistant uses it internally for formatting decimal values in SQL INSERT commands.

See Handling Date and Time Values topic in CHAPTER 13, Scripting, Exporting, and Importing Data, for details on supported date and time value format masks.

Method 2 – Using Data Factory

To copy data between different servers, database, and files you can use the Data Factory utility. Conceptually it's similar to the Data Transfer utility. However it enables you to transfer data from multiple data sources in a single project. Data Factory provides more advanced ETL capabilities for transferring data across different data sources and destinations. To use the Data Factory

1. In your SQL Editor select SQL Assistant's menu, and then select **Import/Export Data -> Data Factory...** menu. The **Data Factory Project Wizard** dialog will appear.

2. Choose **New Project - Guide me Step by Step** option and follow the instructions.

Method 3 – Using Data Scripting

To copy data into a different database using SQL scripts:

1. Connect your SQL editor to the data source database server.

2. Script out table data using the method described in Scripting out Table Data topic in this chapter.

3. Connect your SQL editor to the target database server.

4. Open the SQL script generated in step 2 in your SQL editor and execute it. You can use either your SQL editor's native functions for SQL code execution, if such are supported by the editor, or you can use the SQL Assistant provided code execution functions. See CHAPTER 14, Executing SQL Scripts for more information on executing SQL scripts.

Note: This method requires presence of data destination tables in the destination database. The generated scripts contain INSERT, COMMIT, and "print progress" commands only.

Method 3 – Using Data Export and Import Utilities

To copy table data into a different database using XML or JSON files:

1. Connect your SQL editor to the data source database server.

2. Script out table data using the method described in the Exporting Tables, Schemas, and Databases to XML and JSON Files topic earlier in this chapter.

3. Connect your SQL editor to the target database server.

4. Load the previously generated XML or JSON files into the target database.

Note: This method does NOT require presence of destination tables in the destination database. The destination database may or may not have the specified tables. If the tables already exist, SQL Assistant will attempt to load them from XML/JSON data. If the tables do not exist, and you used the "Flat export" mode to generate export files, SQL Assistant will create them after analyzing data in the exported XML files. However, columns in the created tables may have different data types and data type precisions as compared to the source table columns. Also important to note that the created tables will not feature the same indexes and referential constraints as the original tables

Loading Data Concurrently into Multiple Database Servers

Method 1 – Using the ETL Orchestrator

The ETL Orchestrator support running parallel operations and can be used for concurrently loading data into multiple server. That utility is described in detail in CHAPTER 46, Automating ETL Operations and Other Processes

Method 2 – Using Scripted Datasets

You can use scripted data to load it into other database servers. The required procedure is the same as the procedure described in the topic Method 2 – Using Data Scripting, except that for the code execution step you would use SQL Assistant's multi-server parallel script execution utility. That utility is described in detail in CHAPTER 15, Executing SQL Scripts on Multiple Servers.

Method 3 – Using Command Line Interface

You can use command line interface for the Data Transfer utility to run concurrently multiple data transfer operations. First create a data transfer project as described in the topic Method 1 – Using Data Transfer Utility. Open as many command windows as many concurrent loads you want to run. In each command window execute sacmd.exe utility with appropriate data source and data destination connection parameters.

Scheduling Automated Data Import, Export, and Transfer Operations

Starting with SQL Assistant version 12.2 use of the ETL Orchestrator utility is recommended method for automating repetitive ETL operations. See CHAPTER 46, Automating ETL Operations and Other Processes for detailed information about the ETL Orchestrator utility.

The older methods are supported as well, however they provide less options and less flexible.

The SQL Assistant's command line sacmd.exe utility can be used to run unattended SQL scripts, data and XML import and export operations, and data transfer operations. If you have previously scripted the data required for data import to a SQL script file, you can use Windows Task Scheduler to schedule the following command to execute the script.

```
sacmd ex:"<sql-file>" conn:"connection-name>"
```

In the command above replace <sql-file> with fully qualified file name for the script you previously generated using the Data Scripting feature, replace <connection-name> with the name of preconfigured database connection saved in SQL Assistant options.

See Scripting out Table Data topic for more details on scripting table and view data

If you have previously exported data to XML files and saved the export project settings, you can use Windows Task Scheduler to schedule the following command following command to run the XML import.

```
sacmd xmlimp:"<project-file>;<data-file>" conn:"connection-name>"
```

In the command above replace <project-file> with fully qualified file name for the xml import project you previously generated using the XML Import feature, replace <data-file> with the full path and name of the XML data file, replace <connection-name> with the name of preconfigured database connection saved in SQL Assistant options.

See Exporting Data to XML and JSON Files topic for more details on exporting data to XML and JSON files.

See Importing Tables, Schemas, and Databases from XML Files topic for more details on importing data from XML files.

If you have previously created a data transfer project, you can use Windows Task Scheduler to schedule the following command following command to run the XML import.

```
sacmd td:"<project-file> srcconn:"connection-name>" dstconn:"connection-name>"
```

In the command above replace <project-file> with fully qualified file name for the data transfer project you previously created using the Data Transfer utility, replace first <connection-name> with the name of preconfigured source database connection saved in SQL Assistant options, replace first <connection-name> with the name of preconfigured destination database connection saved in SQL Assistant options

See Method 1 – Using Data Transfer Utility topic for more details on creating data transfer projects.

See your Windows documentation for instructions on how to use Windows Task Scheduler utility.

Tip: For your convenience the Import and Export dialogs, as well as the Data Transfer and the Data Factory dialogs provide the **Schedule** button that allows you to schedule jobs using graphical tools. The Schedule button opens the **Schedule** dialog. The options and controls are the same for all scheduled operations. See CHAPTER 16, Scheduling SQL Script Execution chapter for more information.

Managing Scheduled Tasks

Managing scheduled data import / export tasks is no different than managing other scheduled tasks.

See Managing Scheduled Tasks topic in CHAPTER 16, Scheduling SQL Script Execution chapter for more information

CHAPTER 14, Executing SQL Scripts

Overview

SQL Assistant's code execution facility provides several important features that may not be present in the target editor. Examples include the following:

- Execution of SQL scripts on multiple servers, even supporting concurrent execution of scripts on database servers of different types

- Display of result-sets returned by the database engine during code execution

- Support for batch SQL scripts and multiple result-sets returned from a single batch

- Automatic stripping of comments for database servers that do not support comments in individual SQL commands

- Display of code execution timing statistics

- Separate display of code execution statistics and result-sets

- Display of result-sets on separate tab pages that can be retained for later use. In other words, result windows do not need to be closed or refreshed after each code execution.

- Ability to export returned data to external files, including Excel

- Ability to retain returned results on screen or in files

- Database code execution in target editors like Notepad, Notepad++, UltraEdit that do not support database connectivity or code execution functions

SQL Assistant supports batch code execution. It automatically parses the SQL script submitted for execution and breaks it into individual SQL batches or SQL statements, depending on the type of database it is connected to.

Handling of Batch Delimiters

SQL Assistant is capable of executing individual SQL statements as well as batch SQL scripts consisting of multiple SQL statements.

For T-SQL compatible databases like SQL Server, Sybase ASA, and Sybase ASE; SQL Assistant uses the **GO** keyword as the default batch delimiter. The GO delimiter must appear on a new line. If GO delimiters are not found in the script, SQL Assistant executes the entire script as a single batch. Note that use of the "go" is not case sensitive. You can also specify a custom batch delimiter in SQL Assistant Options.

For Oracle, SQL Assistant uses forward slash symbol as the default batch delimiter when it appears as the first character on a new line. You can also define a custom batch delimiter in SQL Assistant Options.

For MySQL, the batch delimiter is typically specified directly in the SQL code using the **DELIMITER** directive. If the directive is not found in the script, SQL Assistant uses the $$$ string as the default batch delimiter. A different default MySQL batch delimiter can be specified in SQL Assistant options.

For DB2, SQL Assistant uses the delimiter specified directly in the SQL code using the **--#SET TERMINATOR** directive. . If the directive is not found in the script, SQL Assistant uses the $$$ string as the default batch delimiter. A different default DB2 batch delimiter can be specified in SQL Assistant options.

Important Note: Batch script delimiters can be customized in SQL Assistant options. On the **DB Options** tab, select the type of SQL Assistance whose options you want to customize then enter the desired batch delimiter into Batch Delimiter field. Local batch delimiter directives specified directly in the scripts override default values specified in SQL Assistant options. If you intend to execute only the highlighted portion of a script in the editor, make sure that all relevant directives are also highlighted with that portion.

Working with the SQL Code Execution Interface

Invoking the SQL Code Execution Function for the Current Connection

As with all other SQL Assistant functions, the **SQL Code Execute** function can be applied either to the entire file in the editor or to the highlighted text only. If no text is highlighted, SQL Assistant executes the entire file.

You can use any of the following methods to execute SQL code:

- Use the default Ctrl+ Shift + F9 hot key or a custom hot key if you changed the default key

- Use SQL Assistant's system tray icon menu (see Using System Tray Icon Menu topic for details) and choose the **Target / Execute SQL Code** command.

- If the menu integration option is enabled (see Using Context and Top-level Menus topic for details), use the target editor's context or top-level menus and select the **SQL Assistant / Execute SQL Code** command.

The results of the code execution are displayed using one or multiple tab pages in the horizontally docked SQL Assistant window. Each returned result set is displayed on a separate tab pages. The tab page names begin with the definition of the result-set query or source table name. Tab pages remain open until you close them.

Reading and Understanding Code Execution Output

Messages

Errors and other messages returned by the database server during code execution are written to the **Messages** tab page along with SQL Assistant's status messages. Normal database messages are displayed as dark blue text, error messages as dark red text, and SQL Assistant's status messages as black text.
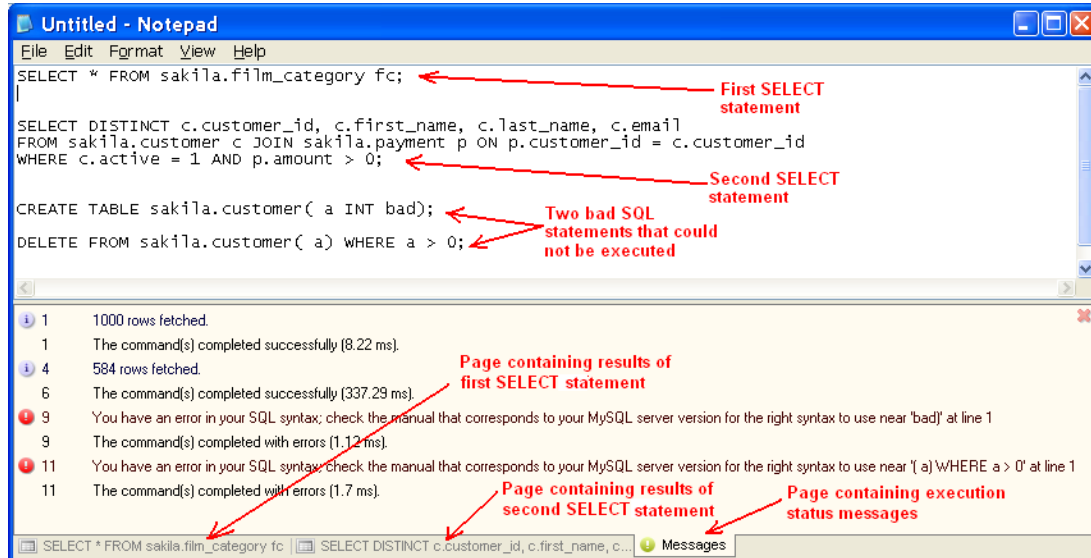
Line numbers and message type icons on the **Messages** tab page indicate the type of each message and the line in the original text where the error occurs. Note that line numbers are absolute and are counted from the first line in the target editor. If you highlight and execute a block of code in the middle of the script, the report lines will show the line offset from the beginning of the script, not from the beginning of the highlighted text.

Tip: By default the Messages tab is reused after each code execution. The previous content is erased and replaced with the output of the latest execution. If you need to compare results and you want to save the previous content, rename the already displayed **Messages** tab to something else. To rename the tab, right-click the tab page handle, and choose the **Rename** command in the context menu.

Query Results

In the case where executed code returns one or more result-sets, each result-set returned is displayed as a separate tab page. The tab page name begins with the definition of the result-set query. The result-set data is displayed in a grid control whose appearance and behavior are identical to the appearance and behavior of the grid control used for the Table Data Preview feature.

Tip: By default the fist 30 characters of the query text are used to label the tab with the query results. To rename the tab to something else, right-click the tab page handle, and choose the **Rename** command in the context menu.

Working with Query Results

See CHAPTER 11, Using Table Data Preview, Working with Table Data Preview Interface for specific instructions on working with the returned query data, editing data, copying and exporting data, and performing other data related operations.

Enabling and Reading Oracle's DBMS_OUTPUT Output

The DBMS_OUTPUT package is typically used for returning messages and reports from stored program code such as stored procedures, packages, triggers, etc… SQL Assistant supports polling of the DBMS_OUTPUT server buffer and printing its messages to the **Messages** tab page.

To enable or disable DBMS_OUTPUT polling:

1. Open SQL Assistant's **Options** dialog.

2. Select the **DB Options** tab.

3. On the left, in the **SQL Assistance** group of options, select **Oracle**.

4. On the right, set the DBMS_OUTPUT Polling option to "Yes" to enable polling and printing of DBMS_OUTPUT messages or set it to "No" to disable it.

To verify that DBMS_OUTPUT output is working, use the SQL Assistant code execution facility to execute any stored procedure that prints messages to the DBMS_OUTPUT. Check that the printed messages appear on the **Messages** tab. For ad-hoc testing without an existing stored procedure, you can try executing a simple SQL command like the following:

```
BEGIN
    DBMS_OUTPUT.PUT_LINE('test message');
END;
/
```

Important Note: To be able to read and display messages printed to DBMS_OUTPUT, you must have permissions to execute the DBMS_OUTPUT, DBMS_CONTEXT, and DBMS_SESSION packages.

Scrolling Content

Use the standard scroll bars in the **Messages** and **Result-set** windows to scroll the content or use the keyboard navigation keys and mouse wheel control if such is available.

Locating Errors

Another helpful feature of the **Messages** window is dynamic code highlighting. When you move the mouse pointer over messages displayed in the **Messages** window, SQL Assistant automatically highlights referenced SQL statements in the target editor, if is the statements are visible in the editor window. If referenced SQL statements are not visible on the screen, click the error message and SQL Assistant will automatically scroll text in the editor to display the statements on the screen. The cursor is automatically set to the beginning of the SQL statement.

Resizing Content

To resize the docked window containing output messages and the returned result-sets window, drag the top edge of the window up or down. Note that when you place mouse pointer over the top edge of the window the cursor shape changes to resize shape as on the following screenshot.

Make sure the cursor takes the right shape before dragging the window edge.

For instructions on how to resize columns in grid controls see Resizing Content topic in CHAPTER 11, Data Display and Editing.

Limitations

The maximum size of a text-based or binary column displayed in a Result-set Preview grid is limited to 32KB. If a column value exceeds this limit, the column text is truncated.

The amount of data that can be retrieved from the database and displayed in the Result-set Preview grid is limited by the amount of free memory available on your computer at the time of code execution.

Using Code Execution History

Overview

SQL Assistant supports two types of code history. Code entry history is stored in the SQL **History Cache**. That cache is used for code entry auto-completion. For more information, see the Using Context-based Suggestions Based on Historical Coding Patterns topic in CHAPTER 3, Using Code Assistant.

A separate **SQL Execution History** view is provided for recently executed SQL statements. You can easily recall SQL statements from the SQL Execution History buffer using the SQL Execution History menu. To access the SQL Execution History menu in the target editor top level or context menu, select these menu options:

SQL Assistant ➔ Execute / Schedule SQL - > Show SQL History.

This will open the Execution History pane. As you execute code in the editor, that code is added to the SQL Execution History buffer with some additional informational messages and query execution statistics.
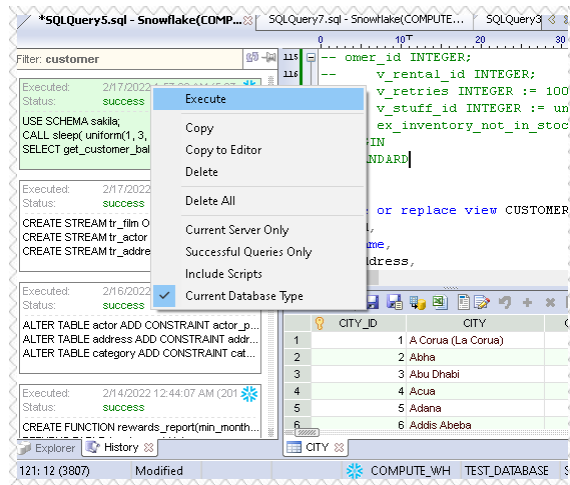
Using the right-click menu in the **SQL Execution History** pane, you can:

- Rerun the SQL query without reentering SQL query text into the editor

- Copy text of the SQL query to the clipboard

- Copy text of the SQL query to the editor window

- Delete individual SQL statements or all cached SQL statements from **SQL Execution History** buffer

The SQL History pane lists queries executed using SQL Assistant's code execution facility, typically executed using SQL Assistant's menus or using Ctrl+Shift+F9 key or whatever you selected for the code execution.

The SQL History pane also lists queries executed in the editor using the editor's code execution facilities; for example, using the F5 key in SQL Server Management Studio. TheF5 key press is trapped, and code is collected from the editor's window.

If your editor uses a different key for code execution, you can change the **Execute SQL Shortcut** option in SQL Assistant settings to match your editor's key. This option is available in the Options dialog on the **Targets** tab, in the **Advanced…** group of options (note that each registered target has its own Advanced group..

When executing queries, SQL Assistant collects several statistics that are displayed in the SQL History pane along with the query reference. Not all of these statistics are available when the code executes using the editor's code execution facility. See the next section for more details.

Filtering SQL Execution History

Several filtering options can be used. To filter historical queries by their text, type the filter criteria in the **Filter** box at the top of SQL History pane and then press the Enter key.

To display queries for the current database server the tab is connected to, right-click the SQL History pane and select **Current Server Only** menu.

To exclude invalid queries and show only queries that executed successfully, right-click the SQL History pane and select **Successful Queries Only** menu.

By default the SQL History pane shows historically executed individual SQL statements and small scripts with multiple statements having less than 2 Kbytes in the size. When you execute larger scripts, they are saved in

the history too, but not shown in the SQL History pane. The make SQL History show all saved history, right-click the SQL History pane and select **Include Scripts** menu.

By default the SQL History pane shows historically executed queries for the type of the database server the tab is connected too. If you work with heterogeneous environments and multiple types of database servers and you want to reuse some of queries, right-click the SQL History pane and unselect **Current Database Type** menu to show the history for all databases.

SQL History Retention Period and Other Options

SQL History feature provides a number of configurable options you can use to control and customize its behavior. You can find them in the main **Options** dialog,

1. In the dialog select the **Targets** tab.

2. Select the **Common** section on the left.

3. On the right click the [+] icon in front of the **History** group of options to expand the group.

4. Customize the options to your liking. For example, if you experience application slowness when using the SQL History feature, you can reduce the amount of data cached and change the history retention period from the default 30 days to 5 days, or make SQL Assistant save the history only when you choose so and not always automatically.

5. Click the **Apply** button to save your changes in the **Options** dialog.

Query Execution Statistics

For your convenience several SQL query statistics are displayed in the SQL history pane.

The first line of each row displays the query start time, query execution duration shown in brackets, and then query size in number of lines. The duration is displayed only for queries executed using SQL Assistant's code execution facility.

In the second line, you can see query execution status, which could be one of the following:

- **Success** – this is reported for successfully executed queries.

- **Failure** – this is reported for queries that resulted in query execution errors.

- **Not Executed** - this status may be reported in case of a transaction rollback or some similar operation. It is also returned for queries executed using the target application code execution facility, since SQL Assistant has no knowledge of code execution status and has no access to the results.

The database icon shown in the right top corner makes it easier for you to find queries for the type of the database you are currently working with.

Using SQL Preprocessor for Advanced Code Execution

SQL Assistant installation includes SQL Preprocessor plugin providing advanced options for SQL code execution. SQL preprocessing simplifies coding of long and complex scripts. It allows you

- To specify code execution directives for conditional code execution.

- To reference external SQL files in your SQL scripts.

- To define constant values that can be used through a script. For example, you can define them in the beginning of the script and quickly change the values if there is a need for that.

Preprocessor directives must be entered inside comments and begin with # sign, for example:

```
--#define __USERNAME__ 'PETER'
/*#define __SOME_TEXT__ 'line1
line2'*/
```

The following directives are supported

Directive	Description

#define
#def
	This directive defines new preprocessor variable or macro. Variable name must be a valid identifier containing alphanumeric characters and underscore symbols only. The value of the variable is the text following space after the variable name and ending at the logical end comment line or block. Using block /* */ comments you can define multiline values. The value of the variable is used in the following part of your SQL script as a substitute for the preprocessor variable name. For example:
	`--#define _COUNT_ 5`
	`PRINT 'This will print 5'`
	`PRINT 'And this will print ' + CAST(_COUNT_ AS VARCHAR(3)) + ' too'`
	
	IMPORTANT: You should choose variable names carefully to ensure they are very unique and do not match other unrelated text in the SQL script
	For macros its argument names must be added in brackets after macro name, just like a function name.
	One macro or variable could be used inside another macro,
	Example:
	
	`/*#define IIF(cond, true, false) CASE WHEN cond THEN true ELSE false END*/`
	`--#define ABS(x) IIF((x) < 0, -(x), x)`
	
	Note that #def is simply a short version of #define.

#undefine **#undef**	This directive undefines a previously defined variable or macro. After this directive the matching preprocessor variable names will not be substituted in the code below it. IMPORTANT: When undefining a macro, only its name without arguments must be specified, For example: `--#undefine IIF`
#include **'filename'**	This directive loads the specified file and inserts it into the script prepared in memory for the execution. The inserted script is also processed by the SQL preprocessor using variables and macros defined above the #include line. Variables and macros defined in the included file cover only inserted script and the script that follows the #include directive. For example `--#include 'C:\my scripts\SQLScript1.sql'`
#if	This directive evaluates specified conditional expression following the directive. If the result of the expression is TRUE, the SQL code lines between this directive and the next #else, #elseif or #endif directive are executed. Otherwise they are ignored and not executed. Boolean operators NOT, AND, OR could be used to define complex expression. Brackets can be used to group boolean parts. The expression must be a valid boolean expression containing data value comparison operations and/or predefined boolean functions. The comparison operations may use the following standard operators: =, <>, >, <, >=, <=. Operators = and <> can be used with string and integer operands, while other compare operators can be used with integer operands only. Examples:

<table>
<tr><td></td><td>

```
--#define __MY_VAR__ 10
--#if (__MY_VAR __ >= 70) AND (__MY_VAR __ =< 90)
    PRINT 'We are here because __MY_VAR__ is between 70 and 90';
--#else
    PRINT 'Do something else in this case';
--#endif
```

Supported predefined functions:

1. **dbver('version')** - checks whether the database server version is greater or equal to specified value, which must be entered in 2-parts 'major_version.minor_version' format. For example:

```
--#if not dbver('10.1')
    PRINT 'CRITICAL: Your database version is less than 10.1.'
    PRINT '           We don't support it. Aborting the script.'
    RETURN
--#endif
```

2. **defined(var_name)** - checks whether the specified preprocessor variable is defined. For example:

```
--#if defined(__DEBUG__) OR defined(__VERBOSE__)
    PRINT 'If variable DEBUG is defined, printing additional'
    PRINT 'diagnostic message here'
--#endif
```
</td></tr>
<tr><td>**#ifdef**</td><td>

This directive checks whether the preprocessor variable specified after the directive is defined. This directive is a short version #if defined(var_name), which can be used when only a single preprocessor variable needs to be checked. For example:

```
--#ifdef __DEBUG__
    PRINT 'If variables DEBUG or VERBOSE are defined, let''s print'
    PRINT 'additional diagnostic message'
--#endif
```
</td></tr>
<tr><td>**#ifndef**</td><td>

This directive is an opposite of #ifdef. It checks whether the preprocessor variable specified after the directive is not defined.. For example:

```
--#ifndef DEBUG
    GOTO some_label
--#endif
```
</td></tr>
<tr><td>**#elseif**</td><td>This directive is similar to #if. It combines two directives #else and #if. It must logically follow #if or another #elseif directive.</td></tr>
<tr><td>**#else**</td><td>

This directive can be used together with #if, #ifdef, #ifndef or #elseif. It begins the SQL code that would be executed if the conditional expression in the preceding #if, #ifdef, #ifndef or #elseif condition evaluates to FALSE.

Examples:

```
--#if (__MY_VAR__ >= 70) AND (__MY_VAR__ =< 90)
    PRINT 'Because __MY_VAR__ value is between 70 and 90…';
--#else
    PRINT 'Do something else in this case';
--#endif
```
</td></tr>
<tr><td>**#endif**</td><td>This directive logically closes the preprocessor block started with #if, #ifdef, #ifndef or #elseif.</td></tr>
</table>

🐾 **Tip**: You can also use graphical SQL Preprocessor interface to manage global preprocessor variables that apply to all script. The simplest method for opening the plugin's graphical interface:

1.   In the editor press Ctrl+Shift+Space to open the Command Selector

2.   Type *Preprocessor*

3.   Press Enter key it

Tip: If you use the SQL Preprocessor often, you can customize SQL Assistant's menu as described here, and add it to the menu.

CHAPTER 15, Executing SQL Scripts on Multiple Servers

Overview

SQL Assistant utilizes the concept of Connection Groups for executing code on multiple database servers. You can define as many groups as you want, and you can associate the same connection to any number of groups. Database server connections and groups can be defined directly in the **SQL Assistant – Execute Code on Multiple Servers** dialog. All available connections always appear in the special **All Connections** group. See the Managing Connection Groups and Connection Settings topic in this chapter for more information.

Note that the definitions of database connections available for use with the **Execute on Multiple Servers** function are shared with connections associated with target editors. Just like other database connections, they can be managed using SQL Assistant's main Options dialog. See the Managing Database Connections topic in CHAPTER 48 for more information.

Running Scripts on Multiple Servers

As with most other functions, the **Execute SQL On…** function can be applied either to an entire file in the editor or to the highlighted text only. If no text is highlighted, SQL Assistant will execute the entire file.

1. You can use either of the following methods to execute SQL code:

    *   If the menu integration option is enabled (see the Using Context and Top-level Menus topic for details), you can use the target editor's context or top-level menu. In that menu select the **SQL Assistant / Execute SQL On…** command.
    *   Use SQL Assistant's system tray icon menu (see the Using System Tray Icon Menu topic for details) and choose **Target / Execute SQL On…** command.

    Either action will bring up the **SQL Assistant – Execute Code on Multiple Servers** dialog.

2. In the top-left list box, select the Connection Group you want to use for the code execution.
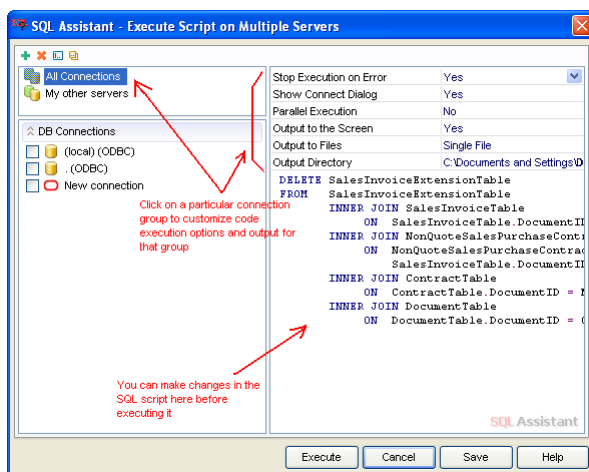
3. In the selected Connection Group, choose database connections for the script execution.

    Tip: To quickly select or deselect all connections, right-click in the "DB Connections" pane and click the **Select All** or **Unselect All** command available in the right-click menu for the connections list.

4. If necessary, you can modify the properties of the selected connections and script execution options.

5. If necessary, edit the script loaded into the embedded SQL editor window.

    Tips:

    -   The **SQL Assistant – Execute Code on Multiple Servers** dialog contains a full featured SQL editor with SQL Intellisense and all other features. For more information on the supported features, see CHAPTER 3, Code Assistants and SQL Intellisense.

    -   If you need to edit the code before its execution, resize the dialog by dragging any of its edges to allow room for the SQL editor box.

6. Optionally, click the **Save** button to save chosen options and connection settings so they can be reused later.

7. Click the **Execute** button to start code execution.

Code Execution and Output Options

The following code execution options can be customized:

Stop Execution on Error – use this option to control error handling. If this option is set to **Yes** and the **Parallel Execution** option is set to **No,** an error during script execution causes execution to abort immediately. SQL Assistant stops all further activities related to the script being executed. If the **Parallel Execution** option is set to **Yes**, SQL Assistant makes an attempt to cancel script execution for all other servers associated with the selected connection group. However the exact reaction to the error may vary. It depends on the type of performed activities and on certain connection parameters. Certain types of connections do not support canceling of already running scripts. If the **Stop Execution on Error** option is set to **No**, the error is ignored and the error message is logged. Script execution continues as if no error occurred.

Show Connect Dialog – use this option to control connection handling. If this option is set when an automatic connection fails or when the connection settings do not contain the required parameters, such as a missing or incorrect password, the Connect dialog is displayed so you can correct the problem and continue running the script. If a connection fails and the **Show Connect Dialog** option is set to **No**, this condition is treated as an error and handled accordingly to the **Stop Execution on Error** settings.

Parallel Execution – use this option to control parallel script execution. If set to **No**, SQL Assistant executes the script sequentially on each database server associated with the selected connection group. If set to **Yes**, SQL Assistant establishes parallel connections to each database server associated with the selected connection group and runs the script concurrently on each server.

Output to the Screen– use this option to control script output. If set to **Yes**, all error messages and script output; such as results of SELECT, RAISERROR or PRINT statements; are displayed on the screen in real time. Note that this option is independent of the **Output to Files** option, and any combination of values can be entered for these two options.

Output to Files – use this option to control script output. If set to **Single File**, all error messages and script output from each server in the connection group; such as results of SELECT, RAISERROR or PRINT statements; are written to a single log file. If set to **Separate Files**, a separate file is written for each server connection. If set to **No**, no output is logged to files. Note that this option is independent of the **Output to**

the Screen option, and any combination of values can be selected for these two options.

Notes: If the **Single File** option is selected, the name of the log files matches the name of the selected connection group followed by the execution date-time suffix and ending with .TXT extension. If the **Separate Files** option is selected, multiple log files are recorded. Each file name matches the name of the associated connection followed by the execution date-time suffix and ending with .TXT extension.

Output Directory – use this option to select the output folder for log files. This option is not used if the value of **Output to Files** option is set to **No**.

Managing Connection Groups and Connection Settings

All available connections always appear in the special **All Connections** group. Additional connection groups can be setup as needed. A single connection can be associated with any number of different connection groups. Different groups can be used for different tasks.

To associate existing connections with a connection group:

1. Click the group name you want to use.

2. Select checkboxes next to any connection names you want to add to the group.

3. Click the **Save** button.
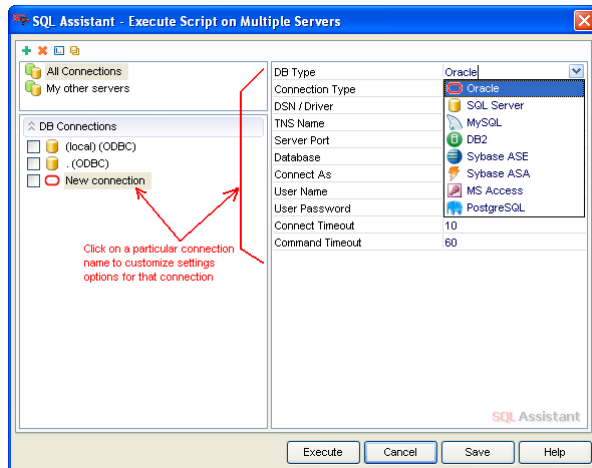
To remove a connection / group association, repeat the same steps and simply deselect the checkbox next to connection name. Do not delete the connection because it will delete that connection from all groups and also from SQL Assistant's general settings.

Tips:

• Use the connection management icons available in the left-top corner of the **Execute Code on Multiple Servers** dialog to create new connection groups or to rename, duplicate or delete existing connection groups and database connections .

Note that the icon functions are sensitive to the location of the focus in the dialog. For example, if the focus is set to the left-top list box containing connection group names when you click the X button, the selected connection group will be deleted entirely, including all associated connections (but not

connection definitions). However, if the focus is set to the left-bottom list box containing connection names and a connection name is highlighted, clicking the same X button deletes the selected connection.

- The content of the right side of the dialog is also context sensitive. If a group is selected on the left, it will display the code execution settings. If a connection name is selected, it will display properties of the selected connection.

- Use the right-click menus in the connection groups and connection lists to quickly associative / de-associate multiple connections and to manage selection specific settings.

- The special group **All Connections** cannot be deleted. This group is used as a placeholder for all configured connections.

- By default, connection names are constructed using the database server name, user name and connection method. However, such names can be long and truncated, making it difficult to recognize different connections. It is recommended that you rename them and give them friendly descriptive names. To rename a particular connection use the Rename icon 🖿 or simply double-click the connection name and type a new name.

To modify connection settings

1. In the top-left window, select the name of the connection you want to modify. The connection properties will appear on the right side of the dialog.

2. Modify connection type and settings as required. See CHAPTER 2, Connecting to Your Database. for information on supported connection types and their properties.

3. Click the **Save** button.

CHAPTER 16, Scheduling SQL Script Execution

Overview

SQL Assistant's code execution facility supports scheduled automated execution of SQL scripts on your database servers. You can use this facility to schedule one-time or recurring processes that run in unattended mode.

Scheduled scripts are stored in files in the %APPDATA%\SQL Assistant\Schedule folder on the system where SQL Assistant is installed. Also stored in this folder are configuration files that contain database connection information for scheduled scripts. Each scheduled script and the associated configuration files must be linked to a Windows scheduler task. The task definition contains references to the SQL Assistant code execution utility, as well as the names of the SQL script and configuration files.

Scheduled tasks can be managed using the standard Windows Task Scheduler user interface. For example, you can use the Task Scheduler to modify an existing task or to schedule an additional task after it has been set up in the SQL Assistant. If you need to modify the task's script, you can edit the associated script file using any SQL or text editor. If you have forgotten the script name, simply open the task properties and check the parameters in the task command line. Refer to your Windows documentation for more details.

Important Notes:

- The SQL Assistant's code execution facility is initiated by the Windows Task Scheduler. It is important that the Task Scheduler service be running at the time the task is scheduled SQL to run. If the service is stopped, no tasks will run. Similarly the computer running the Task Scheduler must be powered on at the time of the scheduled task run. It is good practice to use an "always on" server-based computer for scheduling and running SQL tasks.

- The database connection specified in a scheduled script can point to any database server that is accessible over the network.

- A single SQL Script can be associated with multiple database connections pointing to different database servers. SQL Assistant's code execution facility supports multi-server code execution.

- If a task needs to run on a remote database server, the task must be run under a domain user account. The account must have sufficient privileges to connect to the server. LocalSystem and other system accounts confined to the local system cannot be used for running tasks requiring remote database server connections.

Scheduling SQL Scripts

You can schedule an entire script to run or you can schedule any part of a script. For example, you can schedule a single SQL statement included within a larger script.

To schedule unattended execution of a SQL script:

1. Start your SQL Editor. Enter a new script or open an existing script.

   IMPORTANT: You can schedule the entire script or any part of it. For example, you can schedule a single SQL statement which is part of a larger script.

2. If you want to schedule only part of the script, highlight the relevant part of the script.

3.   Right-click the text in the editor. From the context menu, choose the **SQL Assistant** →
     Execute/Schedule SQL → **Schedule SQL...** menu command. Alternatively, if full menu integration is
     enabled, you can select the same command from the top level menu. In either case, the **SQL**
     Assistant – Schedule SQL Script dialog will appear. An example of this dialog is shown in the
     screenshot below.
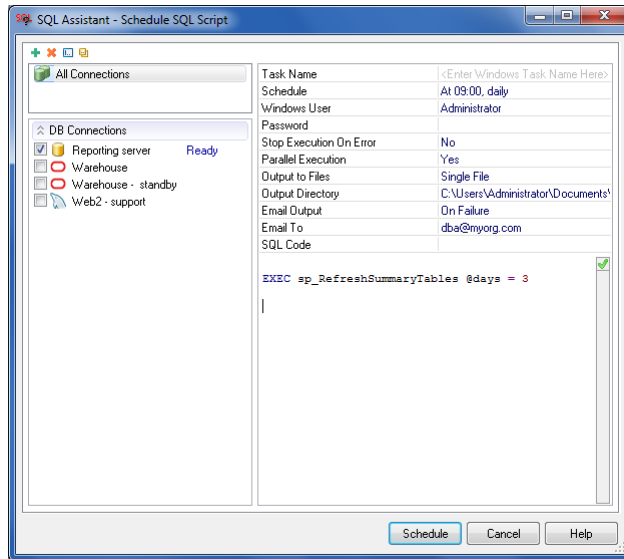
4.   Choose one or more database connections for the scheduled task. If you do not see the required
     connections, you can add them to the list using the green plus sign icon on the toolbar above the
     Connection Groups box.

     For more information on how to add, modify, or delete connections, see CHAPTER 15, topic Managing
     Connection Groups and Connection Settings.

     Tip: The content on the right side of the **SQL Assistant – Schedule SQL Script** dialog is context
     driven. Scheduled task properties are displayed on the right side of the dialog when the focus is set on
     the Connection Groups box. Connection properties are displayed on the right side of the dialog when
     the focus is set on a connection name in the Connections box. To return to task properties after
     selecting or modifying a connection, click the associated connection group in the Connection Groups
     box.

5.   Edit scheduled task properties and schedule type as necessary
     Tip: If you cannot find the required schedule type, pick any available type. You can customize the
     scheduler properties later using the Windows Task Scheduler user interface. You can also use the
     Windows Task Scheduler interface to specify multiple schedules.

     See the Scheduled Task Properties topic for details on the supported scheduled task properties and
     script execution modes.

6.   If necessary, edit the SQL script in the embedded SQL Editor.

     Tips:

     - The **SQL Assistant – Schedule SQL Scripts** dialog contains a full featured SQL editor with
       SQL Intellisense and all other features. For more information on the supported features, see
       CHAPTER 3, Code Assistants and SQL Intellisense.

     - If you need to edit the code, you can resize the dialog by dragging any of its edges to allow
       room for the SQL editor box.

7.   Click the **Schedule** button.
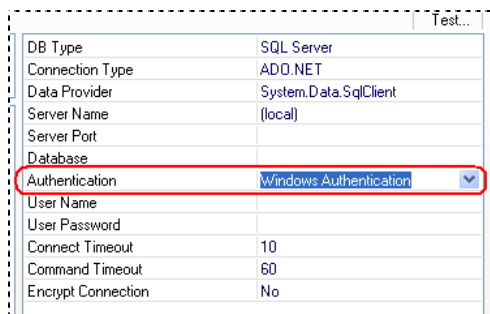
Scheduled Task Properties

General Properties

Task Name – name of the scheduled task. This is a required property. The name should be a friendly name that briefly describes the purpose of the task. The name cannot contain special characters not allowed for use in file names.

Windows User – name of the Windows user account under which the task will run. To specify a local user name, enter the name in `.\user` format or `machine\user` format. To specify a domain user name, enter the user name in `domain\user` format.

📌 **Important Notes:** Do not confuse the Windows user account under which the task will run with the database connection user account. The Windows user account is used to schedule and start the task. The database connection user account is used to connect to the database server. The database connection user is specified in the connection properties of the connection associated with the task. It may be different from the Windows user account name. However, if the connection properties are set to use the **"Windows Authentication"** method, the connection will be attempted within the security context of the Windows user account specified for the task.

		Test...
DB Type	SQL Server	
Connection Type	ADO.NET	
Data Provider	System.Data.SqlClient	
Server Name	(local)	
Server Port		
Database		
Authentication	Windows Authentication	⌄
User Name		
User Password		
Connect Timeout	10	
Command Timeout	60	
Encrypt Connection	No	

See CHAPTER 15, topic Managing Connection Groups and Connection Settings for more information on how to add, modify, and delete connections. See CHAPTER 2, Connecting to Your Database for more information on supported connection methods and their properties.

Password - password of the Windows user account under which the task will run.

SQL Code – the SQL code that will be executed by the scheduled task. The code could be any valid SQL statement or a SQL script containing multiple SQL statements.

📌 **Tip:** The SQL edit box on the **SQL Assistant – Schedule SQL Script** dialog is a full featured embedded SQL Editor. You can use it to edit the code as required. The syntax bar on the right of the SQL edit box indicates the code state. If a green checkmark appears at the top of the syntax bar, the code is valid and ready for execution. For more information, see CHAPTER 20, SQL Syntax Checker

Schedule

The scheduling options available on the **SQL Assistant – Schedule SQL Script** dialog allow you to quickly set up a new scheduled task. You can use these options to schedule the task to run once or to run periodically on a daily, weekly, or monthly schedule. The time you set is relative to the time zone set for the

computer that runs the task.

Tip: If you cannot find the required schedule type, pick any available type. You can customize the schedule type later using Windows Task Scheduler user interface. You can also use Windows Task Scheduler interface to specify multiple schedules.

To enter a new schedule, click the Schedule option on the **SQL Assistant – Schedule SQL Script** dialog, then click the ⋯ button that will appear to the right of the Schedule field. The **SQL Assistant – Task Schedule** dialog will appear on the screen.
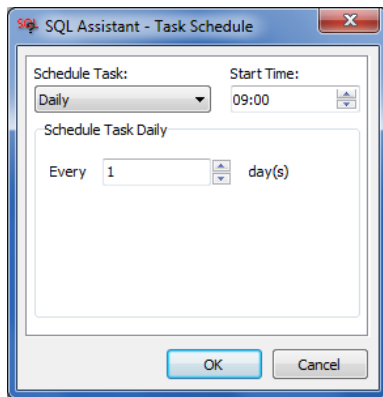
To schedule one-time execution of the SQL script, select **Once** in the **Schedule Task** drop-down list, then enter a date and time to start the task.

If you select the **Daily** option, you can enter the recurrence interval for the task as well as the date and time to start the task. An interval of 1 produces a daily schedule and an interval of 2 produces an every other day schedule. The task will start at the specified time each day.

If you select the **Weekly** option, you can enter the recurrence interval for the task, the date and time to start the task, and the days of the week in which to start the task. An interval of 1 produces a weekly schedule and an interval of 2 produces an every other week schedule. The task will start at the specified time on each of the specified days.

If you select the **Monthly** option, you can enter the months in which you want to start the task and the weeks and days of the month in which you want to start the task. You can also specify that you want to start a task on the last day of each selected month.

Script Execution Mode – Multi-server Support, Output, and Error Handling

Stop Execution on Error – use this option to control error handling. If this option is set to Yes and the Parallel Execution option is set to No, an error during script execution causes execution to abort immediately. SQL Assistant stops all further activities related to the script being executed. If the Parallel Execution option is set to Yes, SQL Assistant makes an attempt to cancel script execution for all other servers associated with the selected connection group. However the exact reaction to the error may vary. It depends on the type of performed activities and on certain connection parameters. Certain types of connections do not support canceling of already running scripts. If the Stop Execution on Error option is set to No, the error is ignored and the error message is logged. Script execution continues as if no error occurred.

Parallel Execution – use this option to control parallel script execution. If set to No, SQL Assistant executes the script sequentially on each database server associated with the selected connection group. If set to Yes, SQL Assistant establishes parallel connections to each database server associated with the selected connection group and runs the script concurrently on each server.

Output to Files – use this option to control script output. If set to **Single File**, all error messages and script output from each database server associated with the scheduled task (such as results of SELECT statements, RAISERROR and PRINT statements) will be written to a single log file. If the option value is set to **Separate Files**, a separate file will be written for each associated connection. If the option value is set to **No**, no output will be logged.

Notes: When the **Single File** option is specified, the name of the log files matches the name of the selected connection group followed by the execution date time suffix. and ending with the .TXT extension. When the **Separate Files** option is specified, multiple log files are recorded. Each file has its name matching name of the associated connection followed by the execution date time suffix and ending with .TXT extension.
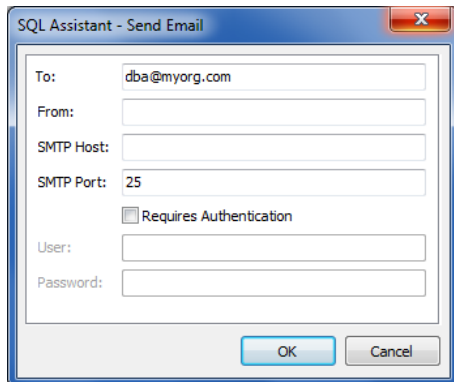
Output Directory – use this option to select the output folder for log files. This option is not used when the **Output to Files** option is set to **No**.

Email Output– this option controls if and when SQL Assistant emails script output results. The following values are supported:

- **Always** –results are automatically emailed after each scheduled script run.

- **On Failure** –results are emailed only if script execution fails for some reason.

Email To - this group of options is used in conjunction with the Email Output option described above. It allows you to automatically email script output. Click the ⋯ button that will appear to the right of the **Email To** field. The **SQL Assistant – Send Email** dialog displays on the screen.

The following email options are supported:

To – semicolon-separated email addresses of the mail recipients

From – the email address of the message sender.

SMTP Host – server name or IP address of your SMTP email server

SMTP Port – SMTP server port used by your server. By default, SMTP port 25 is used.

Authentication is Required – enable this option if your email server requires user authentication.

User – if your server requires user authentication and the **Authentication is Required** option is checked, enter your user name for the email server.

Password – if your server requires user authentication and the **Requires Authentication** option is checked, enter your password for the email server.

Modifying Scheduled SQL Scripts

To modify the SQL script of an existing scheduled task:

1. Locate the SQL script file in the %APPDATA%\SQL Assistant\Schedule folder. Note that %APPDATA% is a system environment variable. If you do not know the value of this variable, you can display it by opening the DOS command window, and executing the command echo %APPDATA%. The script name is the same as the task name ending with .SQL extension.

2. Edit the SQL script in the SQL Editor of your choice and save your changes.

To modify scheduling details for a scheduled task on a Windows 2000, Windows XP, or Windows 2003 system:

1. Open Windows Control Panel.

2. Open the Scheduled Tasks folder.

3. Double-click the task you want modify.

4. Click the Schedule tab page.

5. Modify task schedule details as required, then save your changes.

To modify scheduling details for scheduled task on a Windows Vista, 7, 8, 8.1, 2008, 2010, and 2012 system:

1. Open Windows Control Panel.

2. Open the Administrative Tools folder.

3. Open the Task Scheduler application.

4. Click the Triggers tab page.

5. Select the existing schedule.

6. Click the Edit button.

7. Modify task schedule details as required, then save your changes.

To delete a scheduled task on a Windows 2000, Windows XP, or Windows 2003 system:

1. Open Windows Control Panel.

2. Open the Scheduled Tasks folder.

3. Select the task you want to delete and press the Delete button. If you get a prompt to confirm the action, choose Yes.

To delete a scheduled task on a Windows Vista, 7, 8, 8.1, 2008, 2010, and 2012 system:

1. Open Windows Control Panel.

2. Open the Administrative Tools folder.

3. Open the Task Scheduler application.

4. Select the task you want to delete and press the Delete button. If you get a prompt to confirm the

action, choose Yes.

CHAPTER 17, Generating SQL Procedures and Automating Database Management Operations

Overview

The SQL Code Generator is an advanced code generation utility. It provides flexible, customizable interface supporting code snippets, macros, and SQL language statements for creating code generator templates. The default options include sample CRUD (Create, Read, Update, Delete) and other database administration templates like granting or revoking permissions to set of objects, batch renaming operations, and so on…

Code generator templates are created in SQL Assistant settings. You can create as many templates as you need for different types of bulk code generation tasks. For convenience and manageability, code templates are organized in logical groups. For example, multiple templates for generating CRUD procedures are organized in groups by their purpose and database type.

The SQL Code Generator utility can be invoked from the target editor's menu or from SQL Assistant's system tray context menu. Based on the template definition, the utility will allow you to select multiple database objects of various types, as well as templates to apply to each selected object. Code generated from templates can be executed immediately or it can be saved to SQL script files for a later execution.
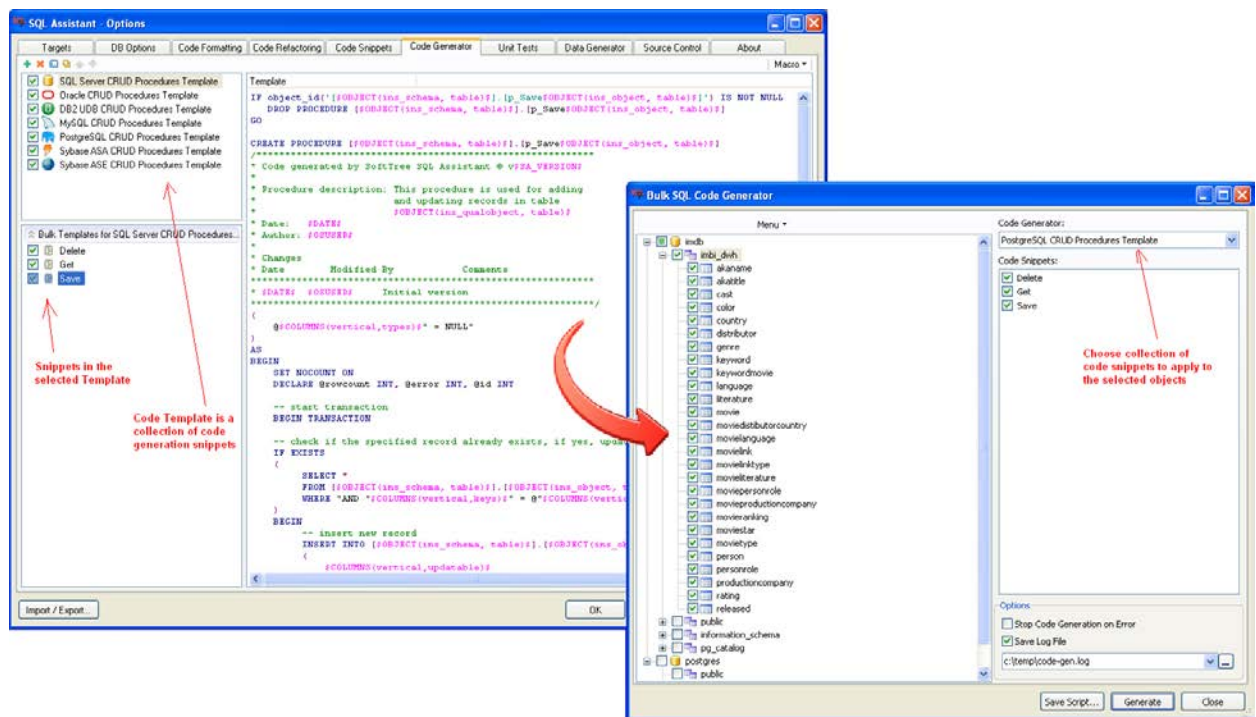
Within the SQL Code Generator templates you can use various macro-variables supported by SQL Assistant to automate the code generation. Macro variables allow you to generate context-driven dynamic code, for example, table specific CRUD (Create, Read, Update, Delete) procedures as in the sample templates installed with SQL Assistant default settings.

Code Generator Macros

Code generator macros are special macros intended for use in code generation templates that can be used along with other supported macros described in CHAPTER 7, [Macro-variables and Dynamic Code Generation](#) topic to customize behavior of code generation methods.

Variable	Meaning
$REFACTORING_DATABASE$	The name of the database containing selected object, or the selected database, depending on templates objective.
$REFACTORING_SCHEMA$	The name of the database schema containing selected object, or the selected schema, depending on templates objective.
$REFACTORING_OBJECT$	The name of the selected schema object.
$REFACTORING_PROTO$	The argument signature for the selected database function or procedure. The actual presentation is database type specific. For example, for SQL Server and Sybase, the signature includes function argument names starting with @ sign. The signature is extracted from the function definition. Example SQL Server: @cust_id INT, @eff_date DATETIME, @ret INT OUT MySQL: p_customer_id INT, p_effective_date DATETIME Together with $REFACTORING_OBJECT$ it can be used to generate code for a new function definition or in other operations where full function signature is required.
$REFACTORING_OBJECT_ID$	The database id of the selected object. Please note not all database types support public references to internal object identifiers. This macro can return empty value in case object id is not available.
$REFACTORING_OBJECT_TYPE$	The type of the selected object. For example: TABLE, PROCEDURE, MATERIALISEZ VIEW, and so on…
$REFACTORING_DDL$	The DDL code of the selected object.

Creating and Customizing Code Templates and Template Groups

Use the SQL Assistant's Options dialog to customize pre-configured code generator templates or to create new templates or template groups.

1. On the Options screen, select the **Code Generator** tab.

2. To modify an existing template, select the appropriate code template group from the Template Groups box in the upper left portion of the dialog (see the screenshot below). The list of templates included in the group displays in the Templates list in the lower left window. Select the template you want to modify and edit the template properties displayed in the window on the right.

    To create a new template group, click the green plus sign icon on the toolbar above the Template Groups box.

3. On the right side of the dialog, specify template group settings. The example screenshot below illustrates the template group options.
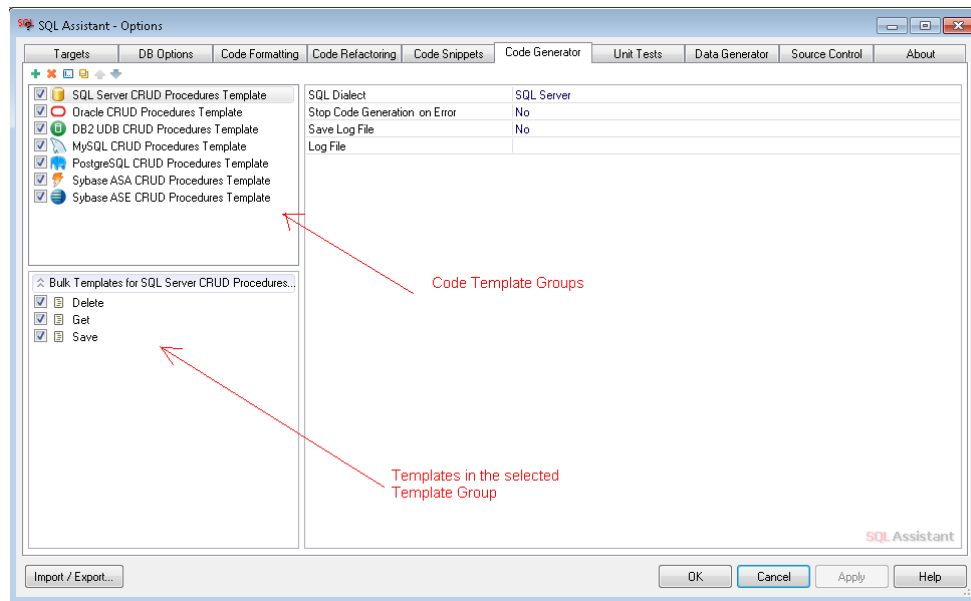
The following options can be specified at the template group level:

SQL Dialect – Specifies the database type for this template group.

Abort Code Generation on Error – Specifies whether SQL Assistant should stop generating code in case of error or to continue with the next template and object in the code generator's object list. The default is not to abort but to continue processing following errors.

Save Log File - Specifies whether SQL Assistant should write work progress messages and run-time errors to a log file. By default, messages and errors are output to the screen only, and errors are written to SQL Assistant's main log file, SQLAssist.log. This log file is located in %APPDATA%\SQL Assistant\[version] folder. Note that %APPDATA% is a system environment variable. If you do not know the value of this variable, you can display it by opening the DOS command window, and executing the command echo %APPDATA%.

Log File – Specifies an optional custom log file in which the SQL Code Generator utility writes progress-of-work messages and run-time errors for the selected template group.

4. Select a template from the list box on the bottom left.

5. In the right window, modify template code as required.

Adding, Deleting and Disabling Templates

To add a new template to a template group:

1. Select a template group from the list box on the top left.

2. Right-click in the Templates list in the lower left window, and select the Add command.

3. In the right window, define appropriate properties for the new template.

To delete template from a template group:

1. Select the template group from the list box on the top left.

2. Select the template from the list box on the lower left and press the Delete button.

To disable a template:

1. Select the template group from the list box on the top left.

2. In the lower left window deselect the checkbox next to the template name.

To enable a template:

1. Select one of the previously disabled template groups from the list box on the top left.

2. In the lower left window select the checkbox next to the template name.

Note:

If a template is disabled, its definition remains in the SQL Assistant options, but the template is not active and cannot be used. Disabled templates do not appear in SQL Assistant's Bulk Code Generator dialog. For more information on how to use and change SQL Assistant's options, see CHAPTER 48.

Additional Tips:

*   Use the code template management icons available in the left-top corner of the Options dialog to create a new template group or to rename, duplicate or delete code template groups and individual templates.

    Note that the icon functions are sensitive to the location of the focus in the Code Generator tab. For example, if you click the X button when a template group is selected in the Template Groups window on the upper left, the selected template group is deleted entirely, including all code templates included within the group. However, if the focus is set to the left-bottom list box containing template names and a template name is highlighted, clicking the X button deletes only the selected template.

*   The content of the right side of the Code Generator tab is also context sensitive. If a template group is selected, the group properties are displayed in the right window. If a template name is selected in the Templates window on the lower right, the template code is displayed in the right window.

*   You can drag-and-drop template group names in the left-top list to rearrange their order. You can use this method to push most commonly used groups to the top of the list and minimize the amount of scrolling and clicking required for customizing code templates.

- To insert a macro-variable into the template code, you can type its name, including the $ sign delimiters, into the code, or you can use the **Macro** button available in the top-right corner of the tab to insert the macro-variable name at the current cursor location in the code. Inserting macro-variables using the Macro button also provides interactive assistance for macro-variables that support multiple options, such as for example, the $OBJECT$ macro variable.
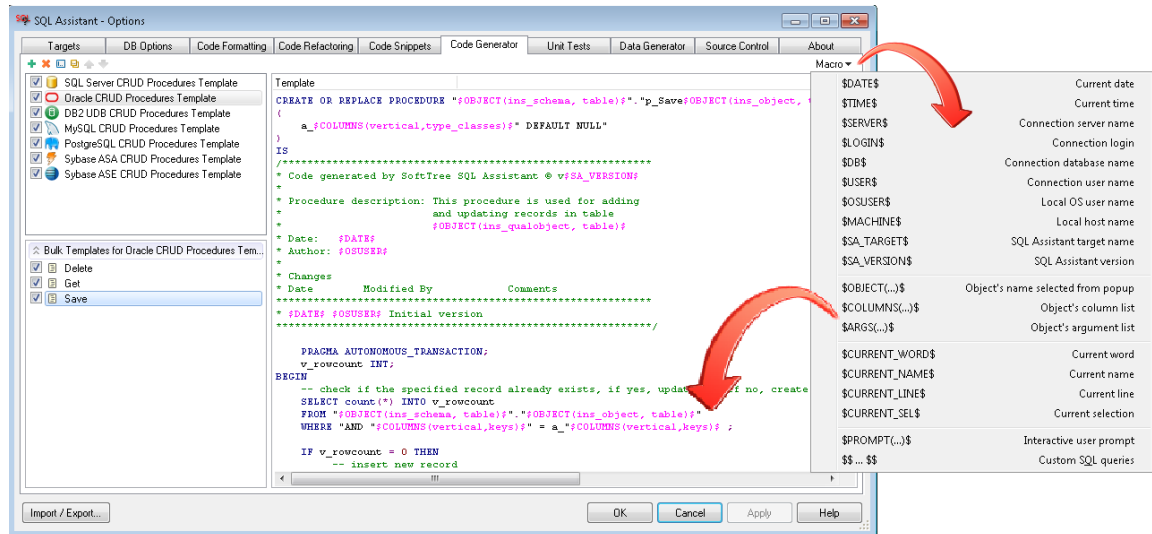
Macro-parameters and their usage are described in detail in the Macro-variables and Dynamic Code Generation topic in CHAPTER 7, Code Entry Automation using Code Snippets.

- If you need to include literal $ symbols in the generated code, for example to include $body$ tags in PostgreSQL functions, escape $ symbols with ^ suffix. For more details see Escaping $ Symbols in Snippet Codes topic.

Practical Example – Creating a New Template for Data Retrieval with Paging

This section presents a generalized example of a SQL Server template created using the bulk code generator. The purpose of this example is to demonstrate how you can easily build your own templates. You can use the same procedure for creating templates in any of the other supported database systems, substituting database type specific code for the SQL Server code in step 6.

This example assumes that the data will be retrieved from tables with a primary key. The primary key may be a singleton or a composite key. The retrieving procedure will allow specifying any combination of columns in the selected table which to be used in the WHERE clause in the query . It will allow omitting not required columns when calling the resulting procedure.

Complete the following steps:

1. Double-click the SQL Assistant's system tray icon to open the **Options** dialog.

2. Select the **Code Generator** tab.

3. Click the Plus Sign icon ➕ in the top left corner of the tab to create new code template group. Name the group "Paged Data Retrieval".

4. In the right window, select the "SQL Server" option for **SQL**.

5. Right-click the templates list box in the bottom left corner and select the **Add** command from the context menu. Name the new template "Retrieve".

6. In the code box on the right enter the following code:

```
/*
    Sample template for retrieving paged data from SQL Server
    Requires SQL Server 2005 or later
*/
CREATE PROCEDURE [dbo].[prGet_$OBJECT(table, view, tblFunc, synonym)$_ByPage]
    @$COLUMNS(vertical,types)$"=NULL"
    ,@PageSize int
    ,@PageNumber int
AS
WITH RowEnumeratedData AS
(
    SELECT RowNumber = ROW_NUMBER() OVER ( ORDER BY $COLUMNS(keys)$ ),
           $COLUMNS(vertical)$
    FROM $OBJECT(ins_qualname, table, view, tblFunc, synonym)$
    WHERE "AND (@"$COLUMNS(vertical)$" IS NULL OR "
$COLUMNS(vertical)$" = @"$COLUMNS(vertical)$)")"
)
SELECT *
FROM RowEnumeratedData
WHERE RowNumber BETWEEN (@PageNumber - 1) * @PageSize) + 1
      AND (@PageNumber - 1) * @PageSize + @PageSize
ORDER BY $COLUMNS(keys)$
```

7. Click the **Apply** button to save changes.

   The result is demonstrated on the following screenshot
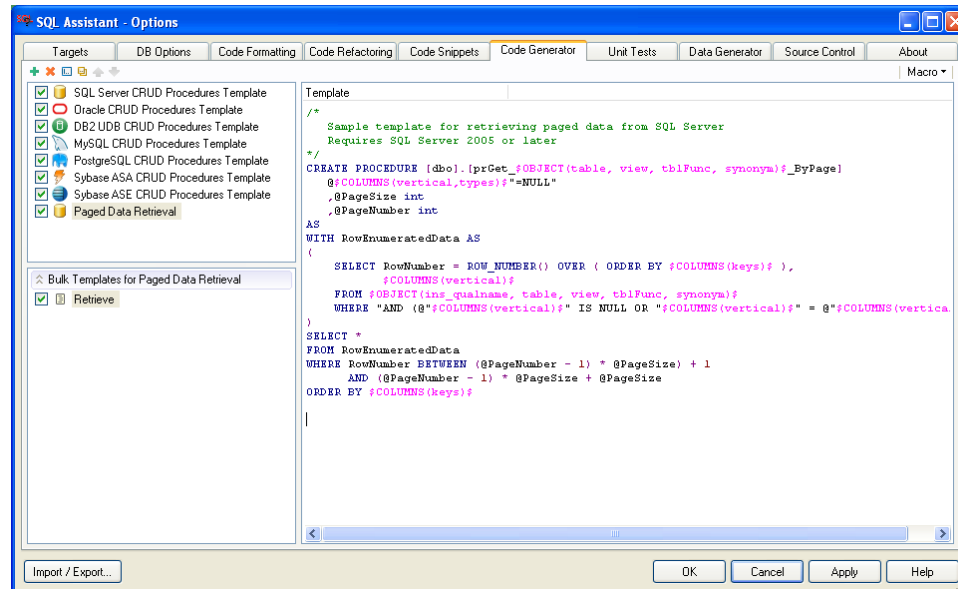
You can now test the new code generator template:

1. From your editor, select the SQL Assistant menu.

2. Choose the Bulk Code generator command

3. Choose the "Paged Data Retrieval" template and click the Generate button.

4. When prompted to select database objects, choose one or more tables for which you want to

create procedures with data paging.

5. Click the Ok button to generate the code.

Advanced Methods for Programming Code Generator Templates

You can use the full power of SQL language supported by your database to generate dynamic SQL code. If the basic macro-parameters supported by SQL Assistant, such as $OBJECT(…)$, $COLUMNS(…)$, and others, do not satisfy your requirements, you can use the **$$..$$** macro to embed dynamically executed SQL code. Within the code of **$$..$$** macro, you can implement the required logic. Code one or more SELECT statements that return the required SQL code as their result set, or call your custom database stored procedures and user-defined functions, outputting the required SQL code.

Use of the $$...$$ macro is described in detail in the $$…$$ Macro topic in CHAPTER 7. Using and Creating Code Snippets for Fast Code Entry.

Following is a simple example of using the $$..$$ macro in a code generator template.

```
$$
SELECT DISTINCT 'DROP PROCEDURE "$OBJECT(ins_schema, table)$"."p_Delete$OBJECT(ins_object,
table)$";' || CHR(10) || CHR(13) || CHR(36) || CHR(36) || CHR(36)
FROM   syscat.routines
WHERE  routineschema = '$OBJECT(ins_schema, table)$'
       AND routinename = 'p_Delete$OBJECT(ins_object, table)$'
       AND routinetype = 'P'
$$
```

In this example, the *'DROP PROCEDURE…'* command will be inserted into the generated SQL script only if the actual procedure exists in the database at the time the SQL Code Generator runs that template. Note that the name of the procedure is dynamic as well. The SQL Code Generator constructs the procedure name from the $OBJECT(…)$ macro that returns schema and object names selected by the user.

Generating SQL Code

1. To open the Bulk Code Generation Utility, use the right-click menu, the target editor's menu, or SQL Assistant's system tray context menu. Navigate to the **SQL Assistant** menu branch and then select the **Code and Data Generators ⬝ SQL Code Generator** command. The SQL Code Generator dialog will appear on the screen.

2. In the database tree on the left of the dialog, select the objects you want to include in the code generation. For example, when using CRUD code generators, you can select one or more tables in one or more database schemas.

3. On the right of the dialog, select a bulk code generator template group and template to apply to the objects selected in the database tree.

4. Set the custom error handling and logging options as needed.

5. Click the **Save** Script button if you want to save the generated script to a file without executing the generated code. You will be prompted for the file name.

   Alternatively, you can click the **Execute** button to have SQL Assistant execute the generated SQL code immediately; or click the **Open in Editor** button to have SQL Assistant create a temporary file

with the generated script and immediately open it in the editor.

CHAPTER 18, Generating Test Data

Overview

You can use the Test Data Generator tool for creating large volumes of meaningful, realistic data for application quality assurance testing, usability analysis, application performance testing, and many other purposes.

The Test Data Generator is able to automatically detect primary and foreign keys and, without user intervention, generate test data that match all referential integrity constraints rules.

The Test Data Generator can generate test data for all standard data types. You can customize the default data generation rules on per-table-column basis to fit your specific business requirements.

The Test Data Generator comes with a large set of pre-built data libraries of the most commonly used data values, including meaningful addresses, names, phone numbers, company names, industries, financial terms, healthcare data, drugs, auto-parts and more. The Test Data Generator also gives you the flexibility to easily create and modify your own data libraries, create custom data generation rules, and specify existing database tables with real business information as "lookup" data values for the newly generated data. See the Managing Data Library topic for more information.

The Test Data Generator intelligently recognizes commonly used column names and presets their value data sources to pre-installed and pre-configured test data libraries.

To launch the Test Data Generator, use right-click menu in the SQL Editor. Alternatively, you can right-click the SQL Assistant icon in the Window system tray and, in the right-click menu, navigate to the SQL Assistant submenu. Choose **Code and Data Generators ☐ Test Data Generator** menu command.
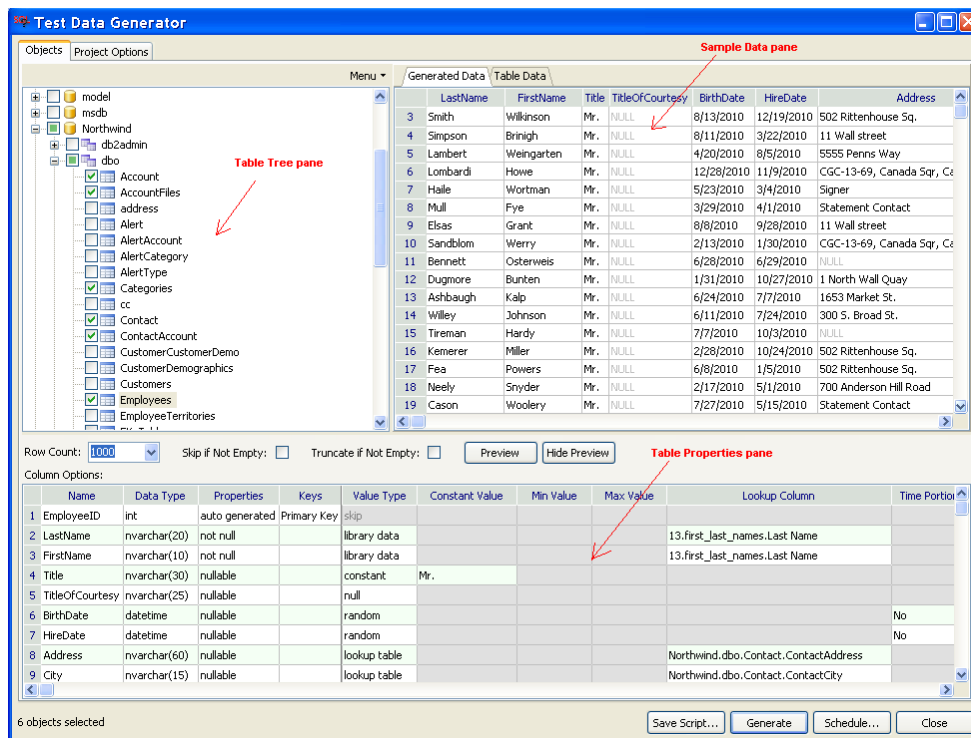
Tips:

- The SQL Assistant's Test Data Generator can be integrated with third-party testing frameworks and applications. To invoke it from other applications, use the command line interface described in the Command Line Interface topic.

- Using the Schedule... button on the Test Data Generator dialog you can schedule running of Test Data Generator projects during low database use times. Alternatively you can schedule Test Data Generator running from the Windows Scheduled Tasks or other scheduling utilities using its command line interface described in the Command Line Interface topic.

Working With Test Data Generator

Common Concepts

The Test Data Generator can be used to generate test data for a single table or for multiple tables in a single run. To simplify data generation management for multi-table environments, the Test Data Generator supports data generation projects. Data generation projects are useful for:

- Grouping collections of related tables together under a single project name

- Persistent storage of project configuration parameters and data generation rules within a set of project configuration files

- Invoking test data generation from Unit Testing projects and consequently using the resulting test data for code unit testing, see Modifying Test Cases and Units in CHAPTER 19, Unit-testing Database Code

Opening and Saving Projects

To open an existing test data generation project to modify project details or to execute a test data generation run:

1. Use right-click menu in the SQL Editor to launch the Test Data Generator,. Alternatively, you can right-click the SQL Assistant icon in the Window system tray and, in the right-click menu, navigate to the SQL Assistant submenu. Choose **Code and Data Generators □ Test Data Generator** menu command.

2. Click the **Menu** button displayed in the top right corner of the Test Data Generator dialog.

3. Choose the **Open Project…** menu command. The Open Test Data Generator Project dialog opens.

4. Select the project file you want to open, then click the **Open** button. The Table Data Generator dialog will be populated with project tables and their individual settings.

To save selected tables and their data generation properties in a project file.

1. Click the **Menu** button in the top right corner of the Test Data Generator dialog.

2. Choose the **Save Project…** menu command. The Save Test Data Generator Project dialog opens.

3. Select the project file to which you want to save project settings and click the **Save** button. To distinguish project files from other XML files, it is a good idea to use "project" or a similar prefix or suffix when naming project files .

Adding Tables to a Project

To add new tables to a test data generator project:

1. In the Test Data Generator window, expand the databases and schemas containing tables that you want to populate with the test data.

2. Select the checkbox in front of the tables that you want to add. To select all tables in a particular schema or database, select the higher level checkbox for the schema or database name.

3. Click each selected table and choose specific data generation settings. When you are done with the settings, save your changes in a project file as described in the "Opening and Saving Projects" topic.

Removing Tables from a Project

To remove tables from a test data generator project:

1. On the Test Data Generator window expand the databases and schemas containing tables that you do not want to populate with the sample data.

2. Deselect the checkbox in front of the tables you want to remove from the project.

3. Save your project changes as described in the "Opening and Saving Projects" topic.

Modifying Table Data Generation Options

Each table selected in the project has its own separate set of properties for the test data generation. To modify the properties.

1. In the tables tree pane, click the table name of the table whose data generation properties your want to modify. Make sure the table name is selected and the checkbox in front of the table name renames selected too. The table properties pane will be displayed at the bottom of the Test Data Generator window.

    Notes: The set of modifiable properties and options depends on the choice of the data

generation method. Different options are available for different data generation methods. Different methods can be chosen for different columns in a table. Data generation properties that cannot be modified for a particular method are displayed with a gray background and are disabled for editing.

2. Choose the appropriate data generation method for each column in the selected table, then modify method specific data generation properties. For specific details on available data generation methods and options, see the topic Data Generation Options in this chapter.

3. Repeat steps 2 and 3 for other tables in the active data generation project

Scrolling Content

Use the scroll bars in the **Test Data Generator** window to scroll the panes. Alternatively, you can use the keyboard navigation keys or mouse wheel.

Note that the table data generation properties pane scrolling is a bit different from other panes. This pane has two parts: fixed columns, which are not scrollable, and data generation properties columns, which are scrollable.

Resizing Content

To resize the **Test Data Generator** window, drag the top edge of the window up or down, left or right.

To adjust sizes of top and bottom panes of the Test Data Generator window, place mouse pointer over the top edge of the table properties pane. The cursor shape changes to resize shape as on the following screenshot.

Drag the edge to adjust pane size. Make sure the cursor takes the right shape before dragging the pane edge.

To adjust sizes of left and right pane, in case data preview pane is visible, place mouse pointer over the right edge of the table tree pane. The cursor shape changes to resize shape as on the following screenshot.

Drag the edge to adjust pane size. Make sure the cursor takes the right shape before dragging the pane edge.

To resize individual columns in the table properties pane, drag the right-edge of the column header left or right. Note that when you place mouse pointer over the right edge of a column header the cursor shape changes to resize shape as on the following screenshot.

Make sure the cursor takes the right shape before dragging the column edge.

Note that the fixed columns displayed on the left side in the data generation properties pane cannot be resized.

Tip: When column width is too narrow to fit the content, 3 dots (also called ellipses) are drawn in each cell with non-fitting data to indicate the data overflow effect. To quickly resize a column so it fits the entire content in all cells, double-click click on the right-edge of the column header. SQL Assistant will calculate the required width and resize this column as needed.

Populating Tables with Test Data

To generate test data and load it into project tables:

1. Start the Test Data Generator. See the Overview topic for more details.

2. Create new or open an existing project. See the Opening and Saving Projects topic for more details.

3. If required, add or remove tables to the project. Choose project-scope, table scope, and column-scope options for tables in the project. as described in the previous topics.

4. Click the **Generate** button to start the process.

You can also use the command line interface to run test data generator projects either from command window or from other applications. See the Command Line Interface topic for more details.

Previewing and Comparing Results

The **Preview** button in the table properties pane can be used to preview how the generated data will look. It can also be used to visually compare the generated data against the actual data currently available in the table. For example, you can use it to adjust various column-scope generation options. See the Column Scope Options topic in this chapter for more details.

Important Notes:

- The generated data sample size is controlled by the **Row Count** table-scope option. Be careful when using the data preview mode. You may need to adjust the Row Count value for the preview only if you intend to generate a lot of data for the selected table. Large values lead to higher memory requirements and longer data sample generation times. The recommended range to achieve the best preview results is 1 to 10000. You can change that value back to the required number when you are satisfied with the quality of the generated data. Refer to the screenshot below as an example.

- The set of columns displayed on the **Generated Data** and **Table Data** tabs may differ. The **Generated Data** tab displays only columns and values that will be generated if you run the project. All auto-generated and auto-calculated columns are omitted from the display since their values are unknown at preview time. Also omitted are all columns marked as "skip" in the Value Type option. On the other hand, the **Table Data** tab displays all actual columns and values from the first 100 records of the table available at preview time.
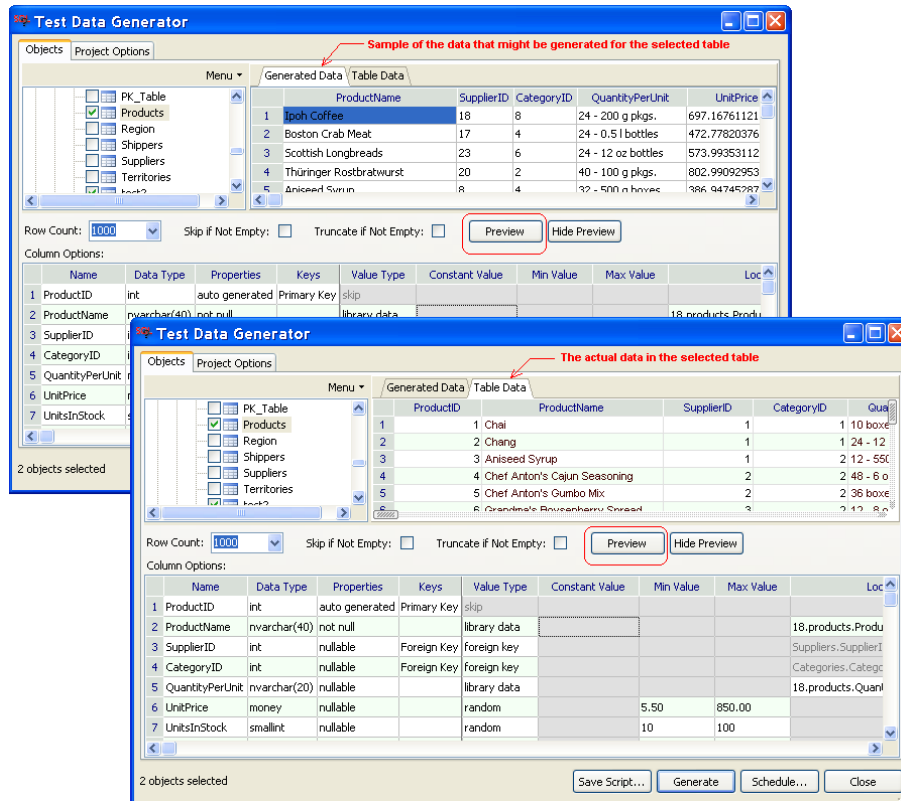
![Tips icon] **Tips:**

- The **Preview** button is visible when a table is added to the project and selected in the project tree pane. To hide the Preview pane, click the **Hide Preview** button.

- Clicking the **Preview** button repeatedly causes the Test Data Generator to generate a new test data sample and refresh the data on the Generated Data tab using this new sample. Use it at your convenient after changing column-scope data generation options.

Creating Seeded Test Data

Seeded data generation is used to load the same set of data every time you run your unit tests. The Test Data Generator allows you to generate a SQL script for inserting the seed data. The script can be saved to a SQL file and then re-run later as many times as needed, always producing the same results. See the Data Generation Options topic in this chapter for more information on the available table data generation options.

The same basic steps are required for setting up test data generation projects and for choosing data generation options. But there are several important things that you should take into considerations when generating seeded test data. They are described in the following notes.

![Important icon] **Important Notes:**

- When generating data for tables that are dependent on lookup tables and primary key tables, make sure the lookup and primary key tables are not empty during the project **Save Script** run. The generation operation will fail if the Test Data Generator cannot retrieve usable data from lookup tables. This is different from the normal data generation run for physically populating table data. In the normal run, the Test Data Generator project can be set up to populate lookup tables and primary key

tables first. This makes the lookup data physically available at the time of data generation for dependent tables so the Test Data Generator can use that data to generate correct lookup values and foreign key in the dependent tables.

- You can set up your data generation project to append new data to lookup and primary key tables, but not to replace it. Do not use the **Truncate** table option table-scope option for tables containing the lookup data. Using this option will cause the lookup data to be replaced in the project run-time. As a result, the rest of the data generation script would become invalid if it contains references to the already deleted lookup data.

- It is recommended that you choose the **Skip if Not Empty** option for lookup and primary key tables to ensure you get consistent results.

- You can tweak the generated scripts after you save them to SQL files. Additional SQL commands can be added to scripts to update or delete the data, to aggregate data in summary tables, and to perform other required operations.

Loading Data Samples from Another Database Server or Database

If you want to use data samples from your production database server or another server, you can use the Data Transfer utility. For more information read Method 1 – Using Data Transfer Utility topic in CHAPTER 13, Scripting, Exporting, Importing, and Copying Data.

Data Generation Options

Project Scope Options

Save Log File -– If this option is checked, SQL Assistant writes processing status messages to the log file specified in the **Log** property.

Log – The name of the output log file. This name must be specified if **Save Log Option** is checked.

Commit Inserted Data After n Rows – This option is effective with database connections that require explicit commits for inserted data. The option value controls the batch update size. Frequent commits lead to small batches which can potentially affect the data generation process performance. In comparison, large batches can potentially result in large database transactions which, in turn, require more space in database transaction logs and rollback segments.

This option has no impact on database connections with automatic commits after each change. Note that by default, SQL Server, Sybase ASE, Sybase ASA, and MySQL connections have automatic commit enabled, while Oracle, DB2, and PostgreSQL connections do not.

If this option is not checked, SQL Assistant does not generate COMMIT statements, during processing.

Stop Generating Data After – This option controls what SQL Assistant should do if an error occurs during the data generation process. The available choices are:

- **Any error** – if any kind of error occurs, the data generation stops immediately.

- **All rows of a table failed** – data generation continues as long as at least one row is successfully inserted into the table. If no rows are successfully inserted, data generation stops, otherwise it continues with the next table in the project.

If this option is not selected, SQL Assistant ignores all errors and continues running the process after an error.

Date Format – The date value format understood by your database. It is very important that you select the correct date format for your data-generation project. An incorrect date format may lead to invalid data inserted into the database.

Time-Format – Time value format understood by your database. It is very important that you select the correct time format for your data-generation project. An incorrect time format may lead to invalid data inserted into the database.

Decimal-Separator – Decimal-separator symbol understood by your database. It is very important that you select the correct symbol for your data-generation project. An incorrect value may lead to invalid data inserted into the database.

Order – The order of tables in your data generation project./ This is the order in which table data will be generated. It is important in a project with referential integrity and parent/child tables that parent tables are populated first, before child tables. Use arrows up and down ⬆ ⬇ to adjust table load order in a multi-table project

Table Scope Options

Row Count – The number of test records to be generated for the selected table**.**

Skip If Not Empty – Instructs the Test Data Generator to skip the table if it is already populated with data.

Truncate Table – Instructs the Test Data Generator to empty the table data before inserting new test data.

Tip: Note that the **Skip If Not Empty** and **Truncate Table** options are mutually exclusive options

Column Scope Options

Available column scope options vary for different column data types, column attributes, and data generation methods. For example, columns with the NULLABLE attribute allow filling in **% of NULLs** data generation properly, while columns with the NOT NULLABLE attribute do not. Another example is a column for which **Constant** is selected as a choice for the **Value Type**. For this value type, you can fill in **Constant Value** column property while for other value types, you cannot.

To make setting up table column data generation properties easier, SQL Assistant uses a color theme to indicate which properties are available and which are not. Note that properties with a gray background are not available and are not editable.

	Name	Data Type	Properties	Keys	Value Type	Constant Va...	Min Value	Max Value	Lookup Column	Time Portion	Text Case	P
1	EmployeeID	int	auto generated	Primary Key								
2	LastName	nvarchar(20)	not null		random						mixed	
3	FirstName	nvarchar(10)	not null		random						mixed	
4	Title	nvarchar(30)	nullable		constant	Mr.						
5	TitleOfCourtesy	nvarchar(25)	nullable		random						upper	
6	BirthDate	datetime	nullable		random					No		
7	HireDate	datetime	nullable		random					No		
8	Address	nvarchar(60)	nullable		library data				16.customers.Address			
9	City	nvarchar(15)	nullable		library data				16.customers.City			
10	Region	nvarchar(15)	nullable		lookup table				Regions.LVCode			
11	PostalCode	nvarchar(10)	nullable		random						mixed	
12	Country	nvarchar(15)	nullable		constant	France						
13	HomePhone	nvarchar(24)	nullable		library data				19.suppliers.Phone			

The following list describes the supported column-scope data generation options:

Value Type – The data value generation method. The following types are available:

- **Skip –** Do not reference the selected column in the generated INSERT statements.

- **Random –** Generates random values. This value generation method can be used with most column data-types. The **Min Value** and **Max Value** properties can be used to specify min and max range for numeric values or length for text values.

- **Sequence** – Generates sequential numeric values. This value generation method can be used with most numeric, date, date-time, and character-based column data -types.

- **Cycle** –Generates sequential numeric values in the specified value range. After reaching the specified maximum value, restart the sequence from the specified minimum value. This value generation method can be used with most numeric and character-based column data- types.

- **Constant** – Uses a constant value for the selected column in all generated records. The actual constant can be specified in the **Constant Value** property.

- **Null** – Uses the NULL value for the selected column in all generated records.

- **Foreign key** – Uses values from the primary key column of a parent table. SQL Assistant automatically recognizes foreign keys and finds their parent key tables. The name of the table and column with the parent keys is entered automatically into the **Lookup Column** property.

- **Lookup Table** – Uses values from a column in an existing table. The table name and column name must be specified in the **Lookup Column** property using dot notation, for example myschema.mytable.mycolumn. The specified lookup table must contain at least one record.

- **Library Data** – Uses values from a column in an existing data library file. This is similar to the Lookup Table option except that values are obtained from a flat text data file rather than from a database table. See the Data Library topic for more information about available data libraries and how to manage and customize them.

- **Regular Expr -** Generates random values using a regular expression specified in the **Constant Value** property. For more details see the following Using Regular Expressions topic.

Constant Value – This value has different meanings for different generation methods

- Used to specify contact values for columns with **Constant** value type.

- Used to specify regular expression for columns with **Regular Expr** value type

Min Value – the minimum generated value or the minimum length of generated values. This value has different meanings for different column data types and generation methods:

- For all numeric data types with the **Random** value generation method, the Minimum value represents the smallest possible generated data value.

- For all string data types with the **Random** value generation method, the Minimum value represents the minimal length of randomly generated strings.

- For all date and date-time data types with the **Random** value generation method, the Minimum value represents the minimal possible value for the date range.

- For all numeric and string data types with the **Sequence** value generation method, the Minimum value represents the starting number for the sequence.

- For all numeric and string data types with the **Cycle** value generation method, the Minimum value represents the starting number for the sequence. The Maximum value represents the maximum number after which the sequence is restarted from the Minimum value.

Max Value – the maximum generated value or the maximum length of generated values. This value has different meaning for different column data types and generation methods:

- For all numeric data types with the **Random** value generation method, the Maximum value represents the largest possible generated data value.

- For all string data types with the **Random** value generation method, the Minimum value represents the largest possible length of randomly generated strings.

- For all date and date-time data types with the **Random** value generation method, the Maximum value represents the maximum possible value for the date range.

- For all numeric and string data types with the **Cycle** value generation method, the Maximum value represents the ending number for the sequence after which the sequence is restarted from the Minimum value.

Lookup Column – the lookup column in a source table or file used as the seed value for data generation. This value could have 3 different types:

- For columns with the **Foreign Key** type, this is the name of a column in a parent table. The column must be either a primary key column or a unique key column. The Test Data Generator finds the parent table and its key automatically. The Test Data Generator uses that column as a data source for the column data values.

- For columns with the **Lookup Table** type, this is the name of a column in a lookup table in the database. It uses that column as a data source for the column data values.

- For columns with the **Library Data** value type, this is the name of a column in a data library file. It uses that column as a data source for the column data values.

Time Portion – the Boolean flag indicating whether to generate time portions of date-time values generated for columns with a "date/time" or compatible data type.

Text Case – the letter case used for text values generated using the **Random** method. This option is only applicable to columns with string based data types. The available options are:

- **Mixed** – both lower case and upper case characters used in the generated data values

- **Upper** – only upper case characters are used in the generated data values

- **Lower** – only lower case characters are used in the generated data values

Prefix – the fixed prefix used in text values generated using the **Random** method. This option is only applicable to columns with string-based data types.

% of NULLs – the percent of nulls generated. This option is only applicable to columns that have the nullable attribute. It cannot be used with the **Sequence** or **Constant** value types. NULL values are generated only if the **% of NULLs** option is set to a non-zero positive value.

Specifying Lookup Table for Column Data Source Values

For columns with **Lookup Table** data source, you must specify from which lookup table the Test Data Generator will load the required data values.

To specify the data source lookup table:

1. In the row with the Lookup Table type, click the **Lookup Column** field. The browse button appears at the left end of the field.

2. Click the browse button. The **Select Lookup Column** dialog will appear.
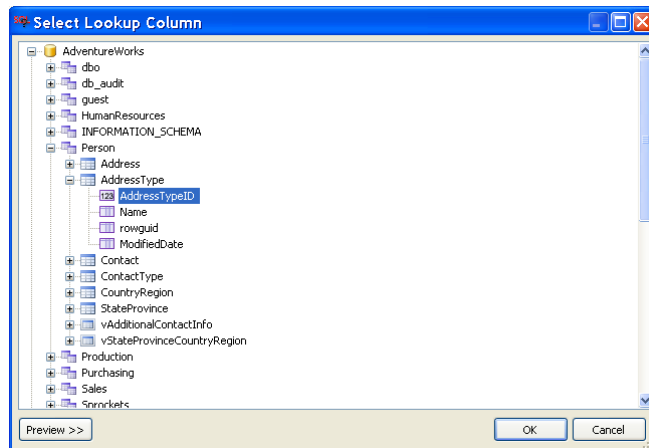
3. In the **Select Lookup Column** dialog, expand the database level containing the required lookup table. Expand the table schema level, and then expand the lookup table level. To preview the data in the selected lookup table, click the **Preview >>** button.

4. Select the column you want to use as a data source.

5. Click the **OK** button.

Specifying the Data Library File for Column Data Source Values

For columns with **Library Data** source, you must specify from which data library file the Test Data Generator will load the data values.

To enter the data source:

1. In the row with Library Data value type, click the **Lookup Column** field. The browse button appears at the left end of the field.

2. Click the browse button. The **Select Library Data Column** dialog will appear.
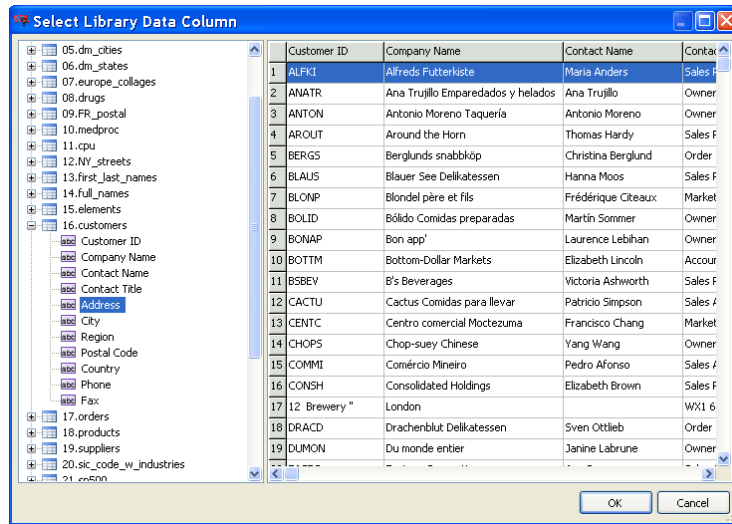
3. In the **Select Library Data Column** dialog, select and expand the level of the data file containing the lookup values.

4. Select the column you want to use as a data source.

5. Click the **OK** button.

Handling Date and Time Values

To generate correctly formatted date and time values, the Test Data Generator needs to know the formats compatible with your database server. Use the Project Scope Options to specify data and time value formats. See the Handling Date and Time Values topic in CHAPTER 13, Scripting, Exporting, and Importing Data, for details on supported date and time value format masks.

Handling Numeric Values

The Test Data Generator is capable of generating 3 types of numbers:

- Integer whole numbers – used for all integer data types, including INT, BIGINT, BIT, SMALLINT, and similar.

- Float numbers with two decimal places - used for MONEY and SMALLMONEY data types.

- Double precision numbers with up to eight decimal places – used for all other numeric data types. Generated double precision numbers may have more digits after the decimal point than allowed by the column data type. This should not be a problem. Your database server should automatically round the values to match the data type specification when the data is inserted into the tables.

Handling Binary Values

There are many types of binary values supported by different database systems, including but not limited to BINARY, VARBINARY, IMAGE, RAW, LONG RAW, BLOB, LONGBLOB, BYTEA. The Test Data Generator supports most of the common binary types, it can generate binary values represented in HEX format. Most database system can recognize HEX values and automatically convert them to the binary storage format. If your database system supports that, you can use the Test Data Generator to populate binary columns with binary values. Two options are available for populating binary values. You can select "constant" in the **Value Type** field, and enter the desired binary value in hex format into the **Constant** field, for example, 0x1A2F10. Do not enter quotes or any other symbols. Or you can use "lookup table" in the **Value Type** field to copy binary values from a table already populated with binary values.

Small Binary Values

In a simple case of small binary values you can specify them directly in the **Constant** field as demonstrated on the following screenshot. The values must be entered in HEX format starting with 0x prefix.

Images and Other Large Binary Object Values

For generating large binary data values select "lookup table" in the **Value Type** field. Use the browse [...] button in the **Lookup Column** field to locate a table containing sample binary values, such as images, audio files, and other types of documents, and then select the binary values source column. See example on the following screenshot.
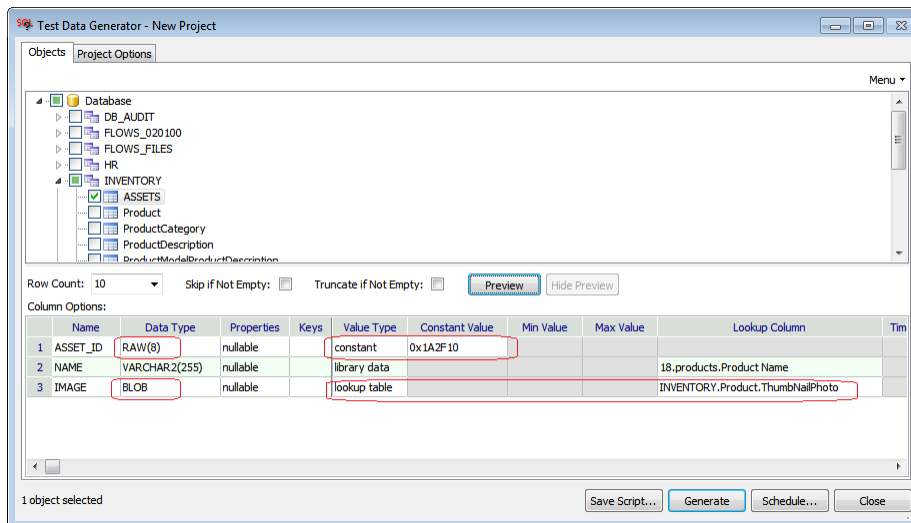
Using Regular Expressions

SQL Assistant supports PCRE like regex engine, which provides the similar syntax and functions as Perl

regular expressions engine. Please see http://perldoc.perl.org/perlre.html for the completed syntax description. It provides some extensions that enable in the Test Data Generator to create random values with predefined value formatting, prefixes, and suffixes, as well as value ranges. In the Application Data Lineage Analyzers it uses extended morde optimized for stremable parsers for processing very large log files. The following examples demonstrate a small subset of the possibilities:

Generating date values with a limited date range within single year:
2019-0[1-9]-[0-2][2-7]

Generating decimal numbers up to 9999 with 2 digits after the decimal point:
\d{2,4}.\d{2}

Generating product code beginning with DS- prefix followed by 3 digit code:
DS-\d{3}

Generating phone numbers for US and Canada
(\d{3})\d{3}-\d{3}-\d{4}

Generating code values beginning with digit 1, 2, or 3, followed by –WORK- suffix ,and then followed by either a,b,c,d, e, or f character
[123]-WORK-[abcdef]

For additional information about regular expressions syntax, see Using Regular Expressions topic in CHAPTER 34, Integrated SQL Editors.

Data Libraries

Data libraries are predefined datasets that can be used as sources of seed data. SQL Assistant supports two types of data libraries:

- **Data Library files** - Data Library files are table-like files containing sets of commonly used values you can use as data sources during the test data generation process.

- **Database Library queries** - Data Library queries enable you to define generic SQL queries that can return data from various databases, and just like the Data Library files provide sets of commonly used values you can use as data sources during the test data generation process.

The Test Data Generator allows you to choose data column from any data library file or data library query as a data source for values in any column of a target database table. Because you choose data source for each table column independently, different files and file columns can be chosen as data sources for different table columns, as well as a single column from a single file can be chosen as a data source for multiple columns in a target database table. In other words, the set of columns in a library file does not need to match the set of columns in a target table.

Theoretically, a single data library containing 4 columns and 200 unique records can be used to produce 1552438800 unique data records. If you also select columns with random or sequential values, you can have the Test Data Generator to produce a virtually infinite number of unique records in a table. The only limitations are data type size constraints and business rules enforced by relational table constraints. For example, a column having the data type NUMBER(3) with a unique constraint, index, or primary key allows only 999 unique rows that can be inserted into the table.

Predefined Data Library Files

The following Data Library files are included in the software distribution package:

1. **Airport Data (total 267 unique records):**
   - Airport Name
   - Airport Code
   - Country Name
   - Country Code

2. **Auto parts (total 171 unique records)**
   - Part Name

3. **Books – Oracle related (total 157 unique records)**
   - Author Name(s)
   - Book Name
   - Cover Type
   - ISBN Number

4. **Books by William Shakespeare (total 199 unique records)**
   - Author Name(s)
   - Book Name
   - Cover Type
   - ISBN Number

5. **Fortune 500 Company Names (total 500 unique records)**
   - Company Name

6. **Computer Science Magazines (total 154 unique records)**
   - Publication Name

7. **Countries (total 240 unique records)**
   - Two-character Country Code
   - Country Name

8. **Denmark Cities (total 82 unique records)**
   - City Name

9. **Denmark States (total 16 unique records)**
    - State Name

10. **European Colleges (total 1283 unique records)**
    - College Name

11. **FDA Drugs and Components (total 3000 unique records)**
    - Application #
    - Product #
    - Form
    - Dosage
    - Product Marketing Status
    - TE Code
    - Reference Drug
    - Drug Name
    - Active Ingredient

12. **French Postal Codes (total 44 unique records)**
    - City Name
    - Postal Code

13. **Medical Procedures (total 248 unique records)**
    - Procedure Code
    - Procedure Description

14. **Microprocessors (total 567 unique records)**
    - Processor Name

15. **New York Streets (total 794 unique records)**
    - Street Name

16. **People's Names - First and Last Name separately (total 500 unique records)**
    - First Name
    - Last Name

17. **People's Names – Full Name (total 500 unique records)**
    - Full Name

18. **Periodic Table of Chemical Elements (total 109 unique records)**
    - Symbol
    - Element Name

19. **Sample Customers (total 99 unique records)**
    - Customer ID
    - Company Name
    - Contact Name
    - Contact Title
    - Address
    - City
    - Region
    - Postal Code
    - Country
    - Phone
    - Fax

20. **Sample Orders (total 878 unique records)**
    - Order ID
    - Customer
    - Order Date
    - Required Date
    - Shipped Date
    - Ship Via
    - Freight
    - Ship Name
    - Ship Address
    - Ship City
    - Ship Region
    - Ship Postal Code
    - Ship Country

21. **Sample Products (total 109 unique records)**
    - Product ID
    - Product Name
    - Supplier
    - Category
    - Quantity Per Unit
    - Unit Price
    - Units In Stock
    - Units On Order
    - Reorder Level
    - Discontinued

22. **Sample Suppliers (total 38 unique records)**
    - Company Name
    - Contact Name

- Contact Title
- Address
- City
- Region
- Postal Code
- Country
- Phone
- Fax
- Home Page

23. **SIC Codes and Industry Names (total 443 unique records)**
    - SIC
    - Industry Name

24. **Stock Quotes – S&P 500 (total 500 unique records)**
    - Symbol
    - Company
    - Country
    - GICS
    - Sector
    - Price

25. **United Kingdom Cities (total 66 unique records)**
    - City Name

26. **USA Counties (total 3092 unique records)**
    - County Name
    - State Name

27. **USA Holidays by Year (total 78 unique records)**
    - Date
    - Holiday

28. **USA Social Security Numbers (total 500 unique records, not in use SSN)**
    - SSN

29. **USA States (total 500 unique records)**
    - 2-character State Code
    - State Name

30. **USA Zip Codes (total 3000 unique records)**
    - Zip
    - State Code
    - City Name

31. **World Currencies (total 3007 unique records)**
    - 3-character Currency Code
    - Country Name
    - Currency Name

32. **Emails (total 100 unique records)**
    - Sample Email Addresses

33. **IP v4 Addresses (total 510 unique records)**
    - IP Address

34. **Websites (total 100 unique records)**
    - Popular Websites

35. **Extended Company Data (total 485 unique records)**
    - Company Name
    - Company URL
    - Category/Industry
    - Business Description

36. **NAICS Industry Classification (total 1065 unique records)**
    - Industry Code
    - Industry Name

37. **Robotic Parts (total 59 unique records)**
    - Part Name
    - Description
    - Price
    - Where to Buy

38. **Ore Metals Annual Production by Country (total 258 unique records)**
    - Country Name
    - Country ISO Code
    - 2000 Production
    - 2001 Production
    - 2002 Production
    - 2003 Production
    - 2004 Production
    - 2005 Production

- 2006 Production
- 2007 Production
- 2008 Production
- 2009 Production
- 2010 Production
- 2011 Production
- 2012 Production

39. **Wine List (total 2710 unique records)**
    - Code
    - Brand
    - Size
    - Age
    - Proof
    - Price

40. **Food Nutrients (total 2710 unique records)**
    - Nutrients Category
    - Component

41. **Business Departments (total 19 unique records)**
    - Department

42. **Music (total 19 unique records)**
    - Song ID
    - Song Name
    - Artist
    - Album
    - Genre
    - Time
    - Track Number
    - Track Year

43. **Exchange rates for various countries and dates (total 804 records)**
    - Date
    - Country
    - Exchange Rate

44. **IBRD income statements (total 2704 records)**
    - Classification
    - Grouping
    - Category
    - Year
    - Amount (millions)

45. **Oil price (in USD) (total 393 records)**

- Date
- Price

46. **Equity stock market data (total 505 records)**
    - Symbol
    - Company Name
    - Sector
    - Share Price
    - Price / Earnings Ratio
    - Dividend Yield
    - Earnings per Share
    - 52 Week Low Price
    - 52 Week High Price
    - Market Capitalization
    - EBITDA
    - Price / Sales Ratio
    - Price / Book Price Ratio

47. **ISO shipping container codes (total 716 records)**
    - Code
    - Description
    - Length
    - Height
    - Group

48. **English football league playoffs (total 343 records)**
    - Date
    - Season
    - Home
    - Visitor
    - FT
    - Hgoal
    - Vgoal
    - Division
    - Round
    - Tie
    - Htier
    - Vtier
    - Aet
    - Pen
    - Pens
    - Venue
    - Attendance
    - Neutral

Tip: You can easily add your own data sets to the library. Read the following topic for more details.

Managing Data Library Files

Test Data Generator's Data Library files are tab-separated files storing collections of common business data. The first line in each data file contains column headers. The data files are located in the **datagen** subfolder of the SQL Assistant home folder. You can modify the existing files as well as add your own data files to the Data Library.

To modify or rename an existing data file: locate the file you want to modify, and open it in Windows Notepad or any other text editor. Make the desired changes and save them. Your changes are effective immediately. If you simply want to rename a file, you can also use the Rename button 🔳 in the top-left corner, click that button, enter new file name and press Enter key. Your changes are effective immediately.

To add a new data file: Create a new tab-separated data file using any editor of your choice. Make sure the first line of the file contains column headers and that the file has the .TXT extension. Copy the file to the **datagen** subfolder. The new file is available immediately.

To delete an existing data file: Use any available file management tool to delete the file from the **datagen** subfolder.

To create new data file or refresh existing file using database SQL query: In the Options dialog select the data library you want to update, click the **SQL Query** link in the top left corner.
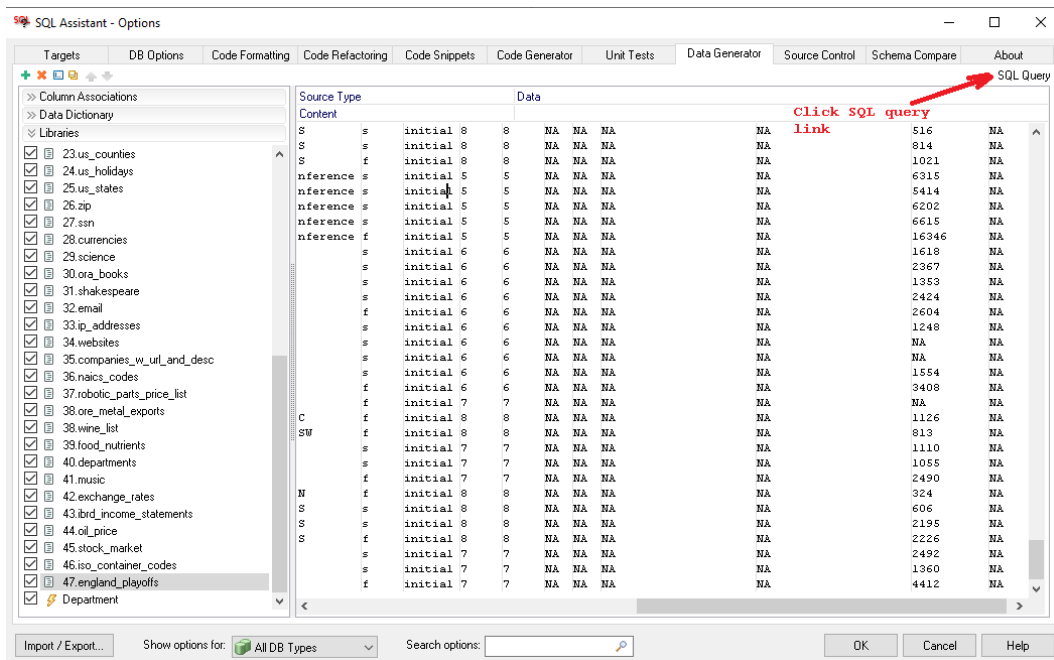
You will be prompted to choose a database connection and then in the next prompt to enter a database query that can be used as a data source for the library file.

Managing Data Library Queries

Unlike the static Data Library file based data sources, the query based data sources provide so called "live" data. They return the current data available in the source database at the time of the data generator project run.

📌 Important Notes:

1. During the data generator project run all referenced library data sources are cached n memory, It's very important that the queries do not return more data than you computer can handle for a reasonably fast test data generation process, and your computer does not run out of memory while caching the source data. General recommendation for the data library queries is to return no more than a few thousands records. The cached values are used for generating randomized test datasets. The raw source data is not copied directly to the target table, so that there is no need for the queries to return a lot of data.

2. Use the Data Transfer utility .in case you want to populate the target table with test data already available in tables in the same or other database server. See Copying Data Between Database Servers topic in CHAPTER 13, Scripting, Exporting, Importing, and Copying Data for additional details.

3. Use the File Import utility .in case you want to populate the target table with test data already available in CSV, Excel, or other flat files. See Importing Data from Excel and Flat Files topic in CHAPTER 13, Scripting, Exporting, Importing, and Copying Data for additional details.

To add a new query to the collection:

1. In the Options dialog activate the Data Generator tab and then expand the Libraries section on the left side of the dialog.

2. Click the Plus button ➕ in the top-left corner of the dialog to add a new data library.
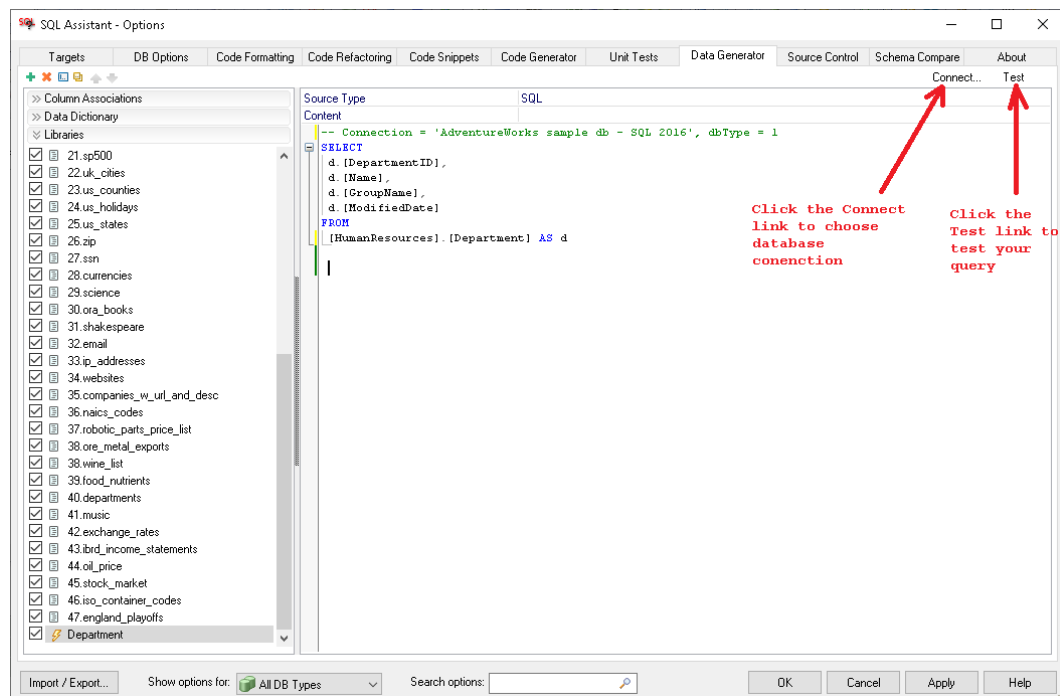
3. On the right side of the dialog, in the **Source Type** field select **SQL**.

4. Click the **Connect** link in the right-top corner of the dialog to choose database connection for the data source. This will add one green line to the query. Do not modify that line.

5. Below that line paste the query you want to use as the data source. You can develop the required query in SQL Assistant's SQL Editor or in any other tool of your choice.

6. Click the **Test** link in the top-right corner to test the query is valid and can be executed.

Important Notes: You can edit the query in the Options dialog as required, but you should not change or remove the top commented line containing the connection details. If you remove that line the Test Data Generator will be unable to use this query as its seed data source.

To rename an existing query source:

1. Locate the query you want to rename

2. Use the Rename button 🔲 in the top-left corner to rename the query.

3. Enter new query name and press Enter key. Your changes are effective immediately.

Command Line Interface

To run a test data generator project from a DOS command line window, use the following command:

```
sacmd dg:"path-to-project-file " sas:"path-to-sa-settings-file" conn:"myserver (userid)"
```

Substitute values in the command as follows:

path-to-project-file	The full file name of the data generator project file
path-to-sa-settings-file	The full file name of the SQL Assistant settings file containing the required database connection parameters. This is an optional parameter. If not specified, the default path for the current user account is used.
myserver (userid)	The database connection name

Example:

```
cd "C:\Program Files (x86)\SQL Assistant 12"

sacmd dg:"C:\Projects\Test Data\dev_pos.datagen" sas:"%APPDATA%\SQL
Assistant\13.0\sqlassist.sas" conn:"DEV001 (sa)"
```

Important Notes:

- Data generator project files are XML files you save using the Test Data Generator graphical interface.

- The SQL Assistant settings file location is version and user profile specific. See the Notes in the Overview topic in CHAPTER 51 for details on how to find out the location of that file.

- You can find out the connection name in the DB Connections group of settings on the DB Options tab page in SQL Assistant Options. If a connection requires a user id and password, make sure that both are saved in the settings. The command line interface does not display interactive prompts and is

unable to prompt for credentials during command processing. For more information about storing and managing database connections, see the [Managing Database Connections](#) topic in CHAPTER 48.

CHAPTER 19, Unit-testing Database Code

Overview

Database unit testing is a validation process for testing database to ensure that objects and units of code are fit for use. Unit testing is very important in application Quality Assurance (QA). It can also help with validating database code after changes are made to the database, including mass changes applied to the database during code refactoring and detecting side effects and other anomalies resulting from previous changes. To fully realize the value of unit testing, you can setup database unit tests to run periodically in automated unattended mode (robo-testing) with the test failure results delivered automatically to your email inbox.

SQL Assistant provides a complete database code-testing framework. The framework allows quick setup of unit tests for multiple database tables, views, and procedures, as well as test cases for any kind of database processing or business scenario. The test case generation interface is highly customizable. User customizable test case templates are used to quickly generate multiple test cases. For example, with just 3 clicks you can generate multiple types of test cases for each table in a selected database schema or entire database. See the Customizing Test Case Templates topic in this chapter for more information about the templates.

To launch the Unit Testing Framework for setup or manual run, use the right-click menu in the SQL Editor. Alternatively, you can right-click the SQL Assistant icon in the Window system tray. In the right-click menu, navigate to SQL Assistant submenu and choose the **Code and Data Generators □ Unit Tests** menu command.
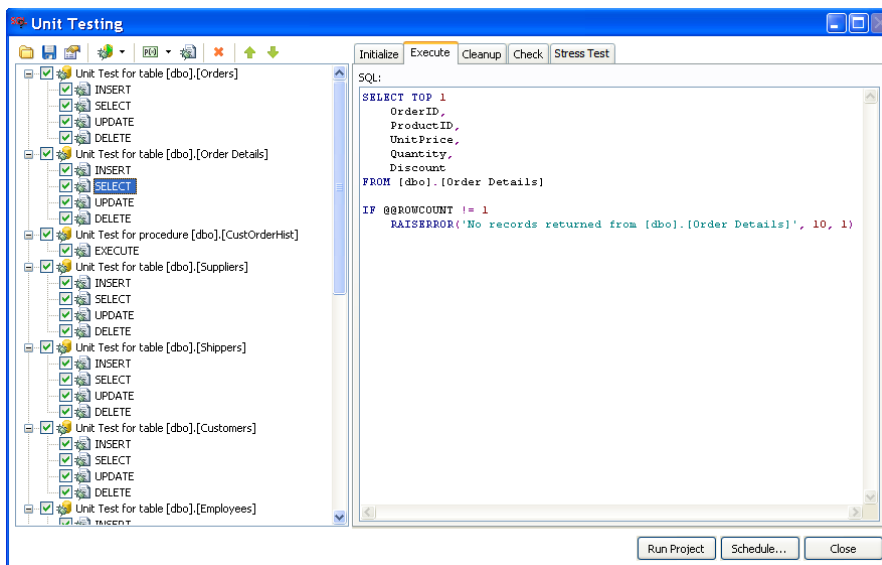
The SQL Assistant's Unit Testing Framework can be integrated with other testing frameworks and applications. To start the Unit Testing Framework from other applications, use the command line interface described later in this chapter.

Additional Tips:

- SQL Assistant's default setup provides 3 predefined unit test types:
  Table-based Unit Test, **View-based Unit Test** and **Procedure-based Unit Test**, each of them

providing several pre-defined standard test cases for their respective object types.

You can use the SQL Assistant's main **Options** dialog to create new unit test types, or to rename, duplicate or delete existing unit test types . You can pre-configure unit test types that better align with your project requirements and then use them to quickly create unit tests for your project with minimal required input from developers. Use the unit test type management icons in the top left corner of. the **Unit Test** tab on the **Options** dialog to manage unit test types and their properties.

Note that the icon functions are sensitive to the location of the focus in the Unit Tests tab. For example, a unit test is selected in the top left list box and you click the X button, the unit test type will be deleted entirely, including all associated test case templates. However, if test case template is selected in the bottom left list box, clicking the same button will delete the selected template.

- The content of the right side of the Unit Tests tab is also context sensitive. If a unit test type is selected, it will display that type definition. If a test case template name is selected, it will display properties of that template and its code.

- Drag-and-drop unit test type names in the left-top list to rearrange their order. You can use this method to push most commonly used types to the top of the list and minimize the amount of scrolling and clicking required to customize test case templates.

- See Using Custom Templates for Generation of Test Cases and the following topics for more details on usage and management of custom test case types.

Working With the Unit Testing Framework

Common Concepts

The Unit Testing Framework can be used to setup and run test cases for a group of database objects or for a business scenario. Related test cases can be grouped together as test units. A group of related test cases is represented by a unit test. Along with test cases, unit tests provide granular control of error handling and test workflow.

To simplify the management of unit tests and test cases, the framework supports unit testing projects. Projects allow persistent storing of project configuration in project configuration files.

The order of units within a test project controls their execution sequence when the project is run. The order of test cases within a unit controls their run sequence within that unit. In certain situations, following a specific run sequence might be very important. For example, if you set up a unit test for a database table and you want to test data deletion from that table, you want the data to be available in that table at the time of the deletion. In this situation, the data insertion test case must be positioned before the data deletion test case so that it can be run first.

Test case execution consists of 4 phases:

- **Initialization phase** – during this phase, SQL Assistant executes the test case initialization instructions specified on the Initialize tab in the test case properties. The instructions are optional and can be left blank. Typically, this phase is used for populating tables with the test data used by the test case code and, optionally, by the following test cases. During this phase, SQL Assistant can automatically invoke test data generation projects created using the Test Data Generator tool.

- **Execution phase** – during this phase, SQL Assistant executes the main SQL code associated with

the test case. The code is specified on the Execute tab in the test case properties. The code is optional. If it is not specified, the test case execution is always successful. If it is specified, SQL Assistant executes the code and checks its success and performance metrics as specified on the Check tab in the test case properties.

Note: During the execution phase the specified SQL code can be executed once per test case, or it can be executed multiple times in stress test mode using multiple concurrent database connections.

- **Status and performance check phase** – during this phase, SQL Assistant analyzes execution results and performance metrics of the main SQL code specified on the Execute tab.

- **Cleanup phase** – during this phase, SQL Assistant executes any cleanup instructions specified on the Cleanup tab in the test case properties. Cleanup instructions are optional and can be omitted. Typically, this phase is used for deleting test data inserted during the initialization phase and for undoing database changes made in the execution phase.

Units and test cases can be enabled and disabled as required. SQL Assistant runs only the enabled units and cases and skips all disabled unit and cases. If a unit is disabled, SQL Assistant skips all cases within the unit regardless of their individual status.

Opening and Saving Projects

To open an existing unit testing project for modification or execution:

1. Click the **Open Project…** icon on the Unit Testing window toolbar. The **Open Unit Testing Project** dialog will appear.

2. Select the project file you want to open, then click the **Open** button. The Unit Testing dialog will be populated with project units, cases, and project settings.

To save all project settings in a project file:

1. Click the **Save Project…** icon on the Unit Testing window toolbar. The **Save Unit Testing Project** dialog will appear.

2. Select the project file you want to save project settings to, then click the **Save** button. It is a good idea to use "project" or similar prefix or suffix when naming project files in order to distinguish them from other XML files.

Adding New Unit Tests

To add new units and cases to a test project:

1. In the Unit Testing window, click the arrow next to the **New Unit...** icon on the Unit Testing window toolbar.

2. Choose type of the unit test to add.
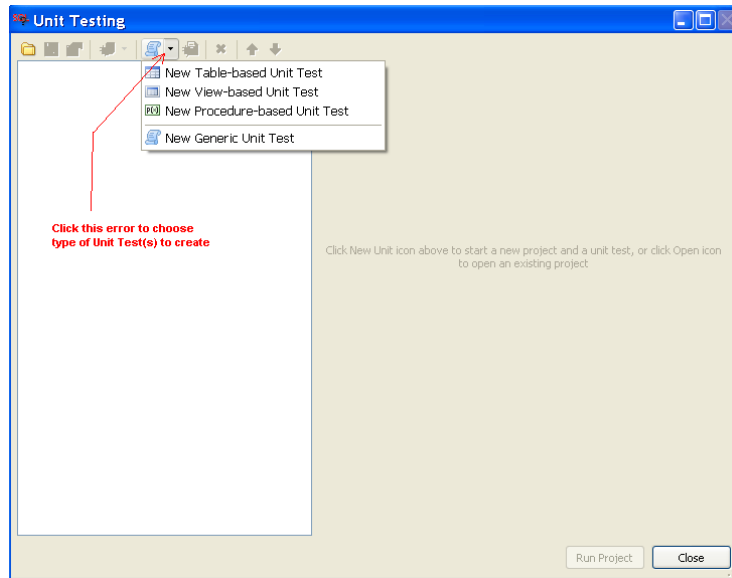
3. If you select a non-generic type, you will be presented with a dialog for selecting database objects for which new test cases will be generated. In the **Select Objects** dialog, expand the databases and schemas in the object tree to select specific objects.

4. Select the check box in front of the object names you want to add. To select all objects in a particular schema or database, select the higher level checkbox for the schema or database name.

5. Click the OK button. SQL Assistant will generate test units and test cases for each selected object. The test code will be generated according to test templates configured in the SQL Assistant settings. See the Customizing Test Case Templates topic in this chapter for more information about the templates.

6. In the main Unit Testing window, click on each added test case and, if necessary, complete the test case properties and code. See the Test Unit Scope Options and Test Case Scope Options topics in this chapter for more information about supported test unit and test case properties.

   It is recommended that you rename the new unit tests and test cases to make their purpose self-explanatory and easier to understand by other developers in your team.

7. You can re-order cases and test units as required, by using the right-click menu in the project tree, by using the up and down arrow icons ⬆⬇ on the Unit Testing window toolbar, or by using keyboard shortcuts.

Adding Generic Test Cases to Units

To add a new test case to an existing unit:

1.   In the project tree select the unit test that you want to modify.

2.   Click the ⬛ **New Case** icon on the Unit Testing window toolbar.

3.   Complete the test case properties and code. See Test Case Scope Options topic in this chapter for more information about supported test case properties.

     It is recommended that you rename the case to make its purpose self-explanatory and easier to understand by other developers in your team.

4.   To change the order of the added case within its unit, either use right-click menu in the project tree, or use ⬆⬇ icons on the Unit Testing window toolbar, or use the available keyboard shortcuts.

Adding New Test Case and Unit Versions

To quickly create a new version of an existing test case or unit:

1.   In the project tree, select the case or unit you want to remove.

2.   Right-click the selected item and choose **Duplicate** command in the context menu. A copy of the selected item will be created.

3.   Rename and modify the copy as required. For details on how to modify and rename items see Modifying Test Cases and Units topic later in this chapter

Removing Test Cases and Units

To remove an existing test case or unit:

1.   In the project tree, select the case or unit you want to remove.

2.   Press the Ctrl+Del hot key or use the ✖ **Delete** icon on the Unit Testing window to remove the selected item.

Disabling and Enabling Test Cases and Units

To disable or enable a specific test case:

1.   In the project tree, select the case you want to modify.

2.   Deselect the check-box in front of the case name to disable a case. Select the check-box to enable a selected case.

To disable or enable all test cases in a specific unit at the unit-level:

1. In the project tree, select the unit that you want to modify.

2. Deselect the check-box in front of the unit name to disable the selected unit. Select the check-box to enable the selected unit.

To disable or enable all test cases in a specific unit at the case level:

1. In the project tree, select the unit you want to modify.

2. Right-click on any case in the selected unit and select the **Disable Test Cases** command to disable all cases in the selected unit. Select the **Enable Test Cases** command to enable all cases in the selected unit..

To disable or enable all test cases in a project at the unit level:

1. Right-click anywhere in the project tree.

2. Select the **Disable All Unit Tests** command in the right-click menu to disable all cases in the project. Select **Enable All Unit Tests** in the right-click menu to enable all cases in the project.

Modifying Test Cases and Units

To remove an existing test case or unit:

1. In the project tree, select the case or unit you want to modify. The controls for the properties of the selected item will appear on the right side of the Unit Testing window.

2. Modify the properties as required. See the Test Unit Scope Options and Test Case Scope Options topics in this chapter for more information about supported test unit and test case properties.

To rename an existing test case or unit:

1. In the project tree, select the case or unit you want to rename.

2. Press F2 key, or alternatively right-click the item and select **Rename** command in the context menu.

3. Type the new name and then press Enter key.

Modifying Project Properties

To add new unit to a test project:

1. In the Unit Testing window, click the **Project Properties…** icon on the Unit Testing window toolbar. The controls for project properties will appear on the right side of the Unit Testing window.

2. Customize the project properties as required and save project changes. See the Test Project Scope Options topic in this chapter for information on the supported project properties

Testing Individual Test Cases and Units

To test working of a specific test case:

1. In the project tree select the case you want to test.

2. On Unit Testing window toolbar, click the arrow next to the **Run Test** icon and choose **Run Selected Test Case** from the drop-down menu.

To test run all cases in a specific unit:

1. In the project tree select the unit you want to test.

2. On Unit Testing window toolbar, click the arrow next to the **Run Test** icon and choose **Run Selected Unit Test** from the drop-down menu.

Running Unit Test Projects

To run a test project in interactive graphical mode:

1. Open the project you want to run. See the Opening and Saving Projects topic for details.

2. Click the **Run Project** button in the right bottom corner of the Unit Testing window.

See the Command Line Interface topic for details on how to run unit test projects from the command line in non-interactive mode or from other applications.

Scheduling Unit Test Project Runs

To schedule an unit test project run in unattended mode:

1. Open the project you want to schedule for unattended execution. See the Opening and Saving Projects topic for details.

2. Click the **Schedule...** button in the right bottom corner of the Unit Testing window. The **SQL Assistant – Task Schedule** dialog will appear.

3. Enter the task schedule properties.

Task Schedule

To schedule a one-time run, in the Schedule Task drop-down select **Once**. Enter a date and time to start the task.

If you select the **Daily** option, you can enter the recurrence interval for the task and the date and time to start the task. An interval of 1 produces a daily schedule, and an interval of 2 produces an every

other day schedule. The task will start at the specified time each day.

If you select the **Weekly** option, you can enter the recurrence interval for the task, the date and time to start the task, and the days of the week in which to start the task. An interval of 1 produces a weekly schedule, and an interval of 2 produces an every other week schedule. The task will start at the specified time on each of the specified days.

If you select the **Monthly** option, you can enter the months in which you want to start the task and the weeks and days of the month in which you want to start the task. You can also specify that you want to start a task on the last day of each selected month.

Task Properties

Task Name – the name of the scheduled task. This is a required property. By default, the Unit Testing project name is used for the task name. The name cannot contain special characters not allowed for use in file names.

User Name – the name of the Windows user account that will run the task. To specify a local user name, enter the name in `.\user` format or `machine\user` format. To specify domain user name, enter the user name in `domain\user` format.

Important Notes: Do not confuse the Windows user account that will run the task with the database connection user account. The Windows user account is used to schedule and start the process. The database connection user account is used to connect to the database server. The database connection user is specified in the connection properties of the connection associated with the task. It may be different from the Windows user account. However, if the connection properties are set to use **"Windows Authentication"** method, the database connection will be attempted within the security context of the Windows user account specified for the task.

See CHAPTER 2, Connecting to Your Database for more information on supported connection methods and their properties.

Password – the password of the Windows user account that will run the task.

Connection – the database connection the task will use for the project run.

See the CHAPTER 15, topic Managing Connection Groups and Connection Settings for more information on how to add, modify, and delete connections.

4. Click the OK button to create the scheduled task and to close the **SQL Assistant – Task Schedule** dialog

Testing in Stress-test Mode

The stress-test mode is useful for testing different kinds of concurrent access issues, for testing locking and dead-locking, as well as for testing for performance and scalability issues in heavy workload conditions.

Use the options available on the **Stress Test** tab in test case definitions to specify stress test parameters:

- The **Concurrent Instances** option specifies number of independent threads to run the Test Case. If you choose 10 instances, during the unit processing the test framework will create and start 10 threads each continuously executing the SQL code specified in the text case properties.

  Note: If the **Concurrent Instances** value is not specified, regular test case executing mode is used.

- The **Total Stress-test Duration (ms)** option specifies in milliseconds how long to run the test case. If a thread completes its execution within the specified duration and the specified number of executions isn't reached, it starts running the test case again, until it riches the specified **Total Stress-test Duration** or riches the specified **Number of Executions per Instance**, whichever is reached first.

  Note: If the **Total Stress-test Duration** value is not specified, each concurrent instance executes its test case as many times as specified by the **Number of Executions per Instance** for as long as it takes.

- The **Number of Executions per Instance** option specifies how many times each thread needs to run the unit test. In case the **Total Stress-test Duration** is specified too, and total execution duration reaches that value, the test is stopped.

  Note: If the **Number of Executions per Instance** value is not specified, each concurrent instance executes its test case continuously until the total duration reaches the **Total Stress-test Duration** value.

Scrolling Content

Use the scroll bars in the **Unit Testing** window to scroll the project tree. Alternatively, you can use regular keyboard navigation keys or the mouse wheel.

Resizing Content

To resize the **Unit Testing** window, drag the top edge of the window up or down, left or right.

To adjust sizes of left and right panes of the window, place mouse pointer over the right edge of the project tree pane. The cursor shape changes to resize shape as on the following screenshot.

Drag the edge to adjust pane size. Make sure the cursor takes the right shape before dragging the pane edge.

Project Scope Options

Unit testing projects typically group unit tests by application or business function. They have the following properties:

Log

> **Save Log File** – this option specifies whether a log file is written for the project run. The log file is used for troubleshooting problems that might occur during the project run. The log file is written in plain text format and contains progress of work and other diagnostic messages.

> **Log File** – specifies the log file name. A log file is only created only if the **Save Log File** option is enabled.

Output Results

> **Save Unit Test Results** – this option specifies whether an output file is written for the project run. The output file records the results of the unit test in the selected output format. The output file is used for analyzing test case results.

> The output file is written at the end of the project run. The output file format depends on the selected output file extension. The following formats are supported:

> - **XML** – output results are recorded in XML format. Each node in the output file contains 6 named properties:

unit	name of the unit test
name	name of the test case
active	test case state, 1 means enabled, 0 means disabled
start	start time of the test case run
finish	end time of the test case run
status	test case completion status, SUCCESS or FAILURE

> Following is an example of XML output for a single test case project:

```
<utres><testcase unit="Sample Unit Test" name="Sample Test Case"
active="1" start="4/1/2010 2:00:46 PM" final="4/1/2010 2:00:46 PM"
status="SUCCESS"></testcase></utres>
```

> - **CSV –** output results are saved in a comma-separated value file. The values are the same as in the XML output format. See XML format description above for more details.

> - **TXT** – output results are saved in a tab-separated value file. The values are the same as in the XML output format. See XML format description above for more details.

> **Important Notes:**
> SUCCESS or FAILURE values in the output file in the status property indicate test case completion status based on the test case definition, not the code execution status. Positive test cases testing for successful code execution have a completion status of SUCCESS when no errors occur during code run. Negative test cases testing for valid errors also have completion status of SUCCESS if the expected errors occur during code run.

Add Time to File Name – this option, if enabled, causes SQL Assistant to add a date and time suffix to the output file name. This ensures that every project run generates unique output file name and does not override previous files.

Send Results

This group of options is used in conjunction with the Output Results group of project scope options. It allows you to automatically email project execution output results. The results can be optionally emailed after each project run or can be emailed only in case of unit test execution failures. It is intended for use in automated unit test and code deployment systems invoking SQL Assistant's command line interface for running database side unit test projects. See the Command Line Interface topic in this chapter for more details.

The following options are supported:

To – semicolon-separated email addresses of the mail recipients

From – the email address of the message sender

SMTP Host – server name or IP address of your SMTP email server

SMTP Port – SMTP server port used by your server. The default SMTP port is 25.

Requires Authentication – specifies whether your email server requires user authentication

User – specifies a valid user name for your email server logon. This option must be specified if your server requires user authentication and the **Requires Authentication** option is selected.

Password – specifies a valid password for the user name specified on the **User** option. This option must be specified if your server requires user authentication and the **Requires Authentication** option is selected.

Email Unit Test Project Results– specifies whether SQL Assistant should attach the project output file to the email message. This option is ignored if the **Save Unit Test Results** option is not enabled or if an output file name is not specified on the **Output Results** tab page.

Send Results – specifies when SQL Assistant sends project output email. The following values are supported:

- **Always** – project results are automatically emailed after each project run
- **On Failure** – project results are emailed if at least one test case fails.

Unit Test Scope Options

Unit tests group individual cases by purpose or by target database object. Unit tests have two modifiable properties:

Case Ordering - see the Adding New Unit Tests topic for details on how to reorder cases within units.

Error Handling Action – controls case processing in the event of a case failure. The following options are available:

- **Always process all test cases** – if selected, all cases within the selected test unit are processed whether they complete successfully or not

- **Stop on the first failed test case and go to the next unit** – if selected, the failure of any test case causes all remaining cases in the selected unit to be skipped and the next unit test to be run. If the selected unit is the last in the project, the project run will end.

- **Abort project on first failed test case** – if selected, project execution ends immediately on failure of any test case

Test Case Scope Options

Initialization

Test case initialization instructions are entered on the **Initialize** tab page. See the Common Concepts topic in this chapter for more information on the purpose and intended usage of these instructions. The following options are supported:

Test Data Generator Project – specifies SQL Assistant's Test Data Generator project file that you want to run before the test case main code is run.

SQL – specifies the initialization SQL to be run before the test case main code. This option is provided to allow the logical separation of test case initialization, main body, and cleanup code. You can enter any SQL batch code understood by your database server. In some database servers, batch SQL code is called as a compound SQL statement. For example, if you want to execute multiple PL/SQL statements in an Oracle database, enclose them in a standard BEGIN…END block.

Important Notes:

- Not all database servers support batch and standalone compound SQL statements. For example, you cannot use multiple SQL statements in unit tests for Microsoft Access, MySQL, and certain versions of DB2 database servers.

- SQL Assistant does not process output results of the executed initialization code. It only checks for successful completion status. If database errors occur during the initialization code run, test case execution fails. The error handling action set at the unit test level controls what happens in case of a test case failure. See the Unit Test Scope Options topic for more details.

Execution

Test case main execution instructions are entered on the **Execute** tab page. See the Common Concepts topic in this chapter for more information on the purpose and intended usage of these instructions.

This is the main test case instruction. It has only one option:

SQL – specifies the test case main SQL code to be run. The value specified must be either a single SQL

statement or a valid SQL batch file.

Important Notes:

- Not all database servers support batch and standalone compound SQL statements. For example, you cannot use multiple SQL statements in unit tests for Microsoft Access, MySQL, SQLite, and certain versions of DB2 database servers.

- Not all types of database servers support returning result sets from batch SQL statements. SQL Assistant supports only directly output result sets. Result sets returned via reference cursors, output parameters, and other indirect interfaces are not supported.

- The error handling action set at the unit test level controls what happens in case of a test case failure. See the Unit Test Scope Options topic for more details.

- Success or failure of the executed SQL code is evaluated in the scope of the test case overall definition, not the code execution status. Positive test cases testing for successful code execution have successful completion status if no errors occur during the main SQL code run and the code completes within the specific performance parameters. Negative test cases testing for valid errors also have successful completion status if the expected errors occur during the main code run. You specify test case evaluation type and metrics on the **Check** tab page.

Tip: Enable logging options at the project level to log error messages returned by the database server and test case timing statistics. The log file is helpful for troubleshooting execution errors after project run, especially when the unit testing project run is part of a larger automated application build process.

Cleanup

Test case cleanup execution instructions are entered on the **Cleanup** tab page. See the Common Concepts topic in this chapter for more information for the purpose and intended usage of these instructions.

The following options are supported:

SQL – specifies the cleanup SQL code to be run after execution of test case main code. This option is provided to allow the logical separation of test case initialization, main body, and cleanup code. You can enter any SQL batch code understood by your database server. In some database servers, batch SQL code is called as compound SQL statement. For example, if you want to execute multiple PL/SQL statements in an Oracle database, enclose them into standard BEGIN…END block.

Important Notes:

- Not all database servers support batch and standalone compound SQL statements. For example, you cannot use multiple SQL statements in unit tests for Microsoft Access, MySQL, SQLite, and certain versions of DB2 database servers.

- SQL Assistant does not process output results of the executed cleanup code. It only checks for successful completion status. If database errors occur during the cleanup code run, the test case execution fails. The error handling action set at the unit test level controls what happens in case of a test case failure. See the Unit Test Scope Options topic for more details.

Status and Performance Checking

Test case execution evaluation instructions are entered on the **Check** tab page. See the Common Concepts topic in this chapter for more information on the purpose and intended usage of these instructions.

The following options are supported:

Maximum Execution Time (ms) – the maximum allowed execution time, in milliseconds, of the main test case SQL code. Enable this option to test that the test case completes within allowed time span. Following are some helpful time conversions: 1000 ms = 1 second, 60000 ms = 1 minute; 300000 = 5 minutes.

Affected Row Count – specifies the number of affected rows. The significance of this value and the resulting behavior differs for different server types. For example, MySQL servers return the number of affected rows on a successful SQL query execution and a -1 value if a SQL query failed. Microsoft SQL Servers return the number of affected rows for the last SQL statement executed within a successfully executed SQL batch, and 0 value if a SQL batch fails. This is the same as the value of the @@ROWCOUNT variable. For Oracle servers, the affected row is the same as the value of SQL%ROWCOUNT for the last executed SQL statement. Consult your database server documentation for additional information.

Validate result – specify the method for result validation. The following methods are available:

- **Do not check results, only check for errors** – the values returned are not validated, the test case checks for execution errors only.

- **Check first value returned, fail test if it doesn't match the condition** – the value returned is compared to the expected result. The value is obtained from the first column of the first row of the first result set returned by the database server after executing the main SQL code. This method is most appropriate for testing queries expected to return single value, such as record count, return code, a value expected after some data update.

- **Check full result returned, fail test if it doesn't match the condition –** the results are converted to a text table with TAB characters as value separators and compared with the saved results. In order to save the expected results, you will need to run the test case successfully and then copy the logged results to the text case expected value field so that it can be compared to in future test case executions.

- **Validate using SQL script, fail test if exception raised** – the test case will you user defined SQL script to validate the values. This method is usable if the code executed makes changes in the database or if the project is configured to save all test case results including all result sets returned to a database table. You can enter one or more SQL queries to check the results. Your queries will be executed as a single batch. If providing more than one SQL statement for the validation be sure to enter an block of valid SQL code that your database can execute.

  See Custom Validation Using SQL Code topic for an example demonstrating how to validate results using SQL code.

- **Four different options for validating results using custom Pascal, Visual Basic, C++, or JavaScript**. – the test case will execute your script. The test case fails if the script raises exceptions.

  See Custom Validation Using Scripting Interfaces topic for examples demonstrating how to validate results using SQL code.

Important Notes for "Check first value returned" method:

- Do not confuse the **Result** option with return codes of stored procedures or procedure output parameters. Typically, this option is used in test cases that execute SELECT operations or stored procedures that execute nested SELECT operations. Following is an example of a SQL Server-specific test case that checks that all required components have been included when assembling a business product:

```
SELECT count(DISTINCT ComponentID) AS major_components_count
FROM AdventureWorks.Production.BillOfMaterials
WHERE BOMLevel = 1
```

  If there are 55 major components that must be used in the product and you expect a return value of 55 in the `major_components_count` column, then specify 55 as the test value for the Result option.

  The code example in the following **Error Processing** section demonstrates how to check the return codes of stored procedures and output parameters.

- If the data type of the first column of the first row of the first result set is not of type "string", SQL Assistant converts the returned value to a string value and then compares it to the value in the **Result** option. String conversion rules for numeric and date time values depend on the regional settings of the computer running the unit test. These could be different from those specified in the database server settings.

Error Processing – The unit test error evaluation type. The following values are supported:

- **Fail test on error** – specifies that a unit test will be considered to have failed under any of the following conditions:

    o The unit test completes with a SQL error

    o Test execution does not meet the specified performance characteristics

    o Test execution does not return the expected return value

  This is a **positive** type of test case. A good example of a positive test case is a test to determine that a table UPDATE operation using valid parameters succeeds without database errors, or that a stored procedure EXECUTE operation using valid parameters succeeds without database errors.

- **Fail test on success** – specifies that a unit test will be considered to have failed under any of the following conditions:

    o The unit test completes successfully without a SQL error

    o Test execution exceeds the specified performance characteristics

    o The unit test fails

  This is a **negative** type of test case. A good example of a negative test case is a test to verify that a table UPDATE operation using invalid parameters fails with an error, or that a stored procedure EXECUTE operation using invalid parameters fails with an error.

Tip: When executing stored procedures, SQL Assistant checks for database errors, not the stored procedure return code. SQL Assistant has no knowledge of meaning of the return code values. If you need to evaluate the return code, add a return code check to the main test case SQL code and have it raise an

error if the returned code is not what you expect. Here are examples for SQL Server specific test cases:

Check for stored procedure return code:

```
DECLARE @ret_value INT
EXEC @ret_value = my_procedure @param1 = 1, @param2 = 'ABC'
IF @ret_value <> 1 RAISERROR('Unexpected return value', 10, 1)
```

Check for stored procedure output parameter value:

```
DECLARE @out_value INT
EXEC my_procedure @param1 = 55, @param2 = @out_value OUTPUT
IF @out_value <> 1 RAISERROR('Unexpected output value', 10, 1)
```

Similar methods can be used for other database servers that support batch and standalone compound SQL statements. Consult your database server documentation for more details.

Custom Validation Using SQL Code

The validation implementation totally depends on your business case. For example, if you are testing table updates for a table with triggers writing log records to a different log table, your SQL code validation can check records in the log table and it finds missing records, raise an exception. For example if you need to check the actual results returned by the stored procedure you are testing, you can configure your unit testing project to save test case results to a table in your database, and then in your SQL code check what is saved to that table, and if meets the requirements. If your database supports running anonymous SQL blocks, you can use multiple SQL statements in the validation. Else create a validation procedure or function in your database that you can invoke using SQL syntax compatible with your database type. Here are examples for both described scenarios.

Example 1: use SQL code to check records in the application log table. This example is for SQL Server

```
IF NOT EXISTS (
        SELECT *
        FROM   dbo.app_log
        WHERE  module = 'SALES'
               AND dt > CURRENT_TIMESTAMP - 1
               AND line_txt LIKE '% orders updated %' )
    RAISERROR('Test case failed, SALES ORDERS log record not found.', 16, 1)
```

Example 2: use SQL code to check records returned by the function. This example is for SQL Server. In this example the project is configured to save test case results to QA.SA_UNITTEST_LOG table. The custom check is for test case named "Test for missing results" which is in unit test named "Supplier batch test"

Test case **Initialization** section

```
EXEC sp_prepare_supplier_data
```

Test case **Execute** section

```
DECLARE @today DATE = CAST(GetDate() AS DATE)
DECLARE @last_batch_id INT
SELECT @last_batch_id = MAX(batch_id) FROM supplier_batch WHERE batch_date = @today
EXEC sp_run_batch @date = @today, @batch_id = @last_batch_id
```

Test case **Check** section

```
-- here we verify batch processing results returned from sp_run_batch
-- and make sure it processed all suppliers and returned a value for each of them

DECLARE @misisng_results INT

SELECT @misisng_results = count(*)
FROM lookup_tables.suppliers AS s
    LEFT JOIN qa.sa_unittest_log AS sa
        ON s.SupplierID = sa.Results.value('results/table[@id=1]/row/cell[@Name=SupplierID]',
INT)
        AND sa.unit_test = 'Supplier batch test'
        AND sa.test_case = 'Test for missing results'
        -- AND if sa_unittest_log wasn't truncated earlier, add subquery for last run
WHERE sa.Results.value('results/table[@id=1]/row/cell[@Name=SupplierID]', INT) IS NULL


IF @misisng_results > 0
    RAISERROR ('sp_run_batch procedure failed to return results for all suppliers', 16, 1);
```

Custom Validation Using Scripting Interfaces

In the validation scripts you can use the complete set of supporting scripting functions and extensions, and you have full access to the results returned by the result set. The results are stored in the data tables accessible through the **Connection** reference. For more information on the supported scripting interfaces, their capabilities and functions, refer to the PluginDeveloperGuide.pdf document.

Example 1: use Pascal scripting to check number of results returned. If now equal to two, fail test case:

Test case **Check** section

```
begin
  if (Connection.DataTableCount <> 2) then
      RaiseException('My validation error 123. Incorrect number of results returned.');
end.
```

Example 2: use Pascal scripting to check number the value in the second row, second column is 55.

Test case **Check** section

```
begin
  // note, in Pascal all array references are zero based

  if (Connection.DataTableCount <> 1) then
      RaiseException('My validation error 210. Incorrect number of results returned.');
  if (Connection.DataTables[0].RowCount < 2) then
      RaiseException('My validation error 211. Incorrect number of records returned.');
  if (Connection.DataTables[0].Cells[1, 1] <> '55') then  // column #2, row #2
      RaiseException('My validation error 212. Incorrect value returned.');
end.
```

# Using Custom Templates for Generation of Test Cases

Custom templates allow you to quickly generate application specific test cases. The process involves two steps:

- One-time setup of custom template groups and associated test case templates or, alternatively, customization of default template groups pre-installed with SQL Assistant

- Application of templates to objects in the database or business scenarios.

This graphical example demonstrates how to set up custom templates. See the following topics for specific instructions on creating and modifying templates.
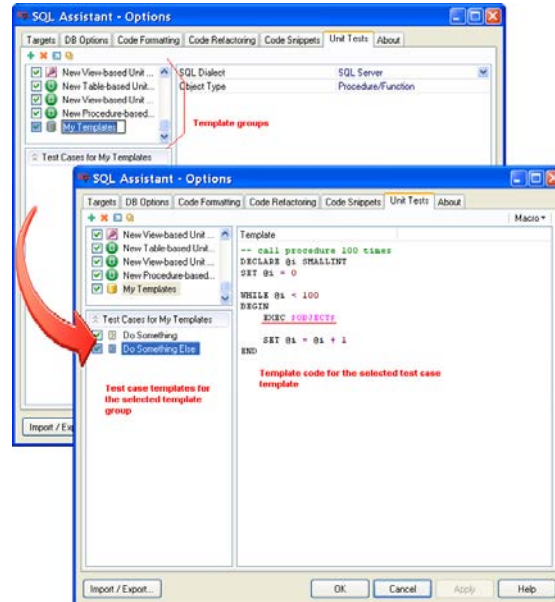
This graphical example demonstrates how to use previously set up custom templates to use the bulk test case generation process for creating unit tests and their test cases
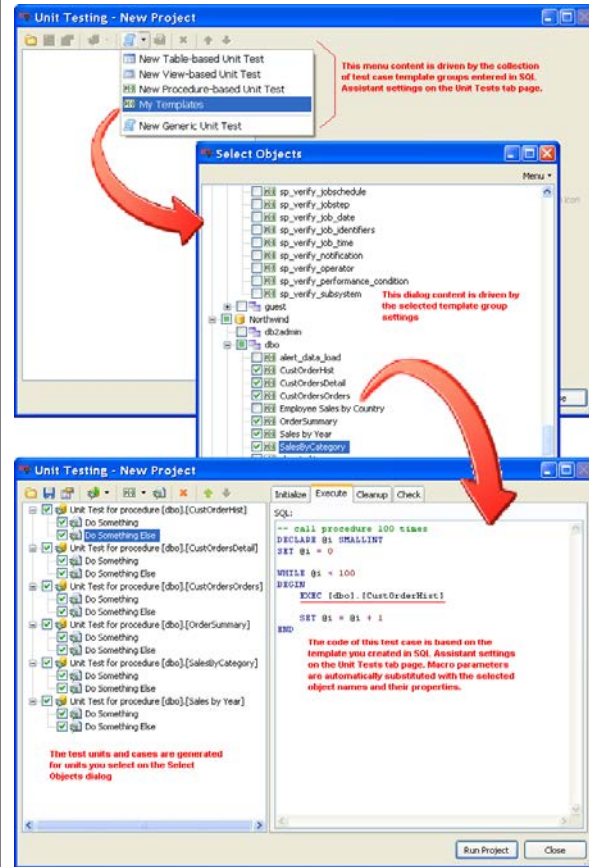

Customizing Test Case Templates

Adding and Removing Unit Test Types

To add a new custom unit test type:

1. Double-click the SQL Assistant icon in the Windows system tray. The Options dialog will appear.

2. Click **Unit Tests** tab page.

3. Click any unit test type name in the list in the top left corner of the dialog.

4. Click the Plus Sign icon in the top left corner of the **Unit Tests** tab page to add a new type.

5. Enter a descriptive name for the new unit test type. The new name will be also used for the new menu item that SQL Assistant will create for you in the Units Tests main window. Note that the trailing text in brackets is automatically omitted in the menu. For example, if you enter the name "My Templates (SQL Server)" for the new type, "My Templates" will be displayed in the main window menu.
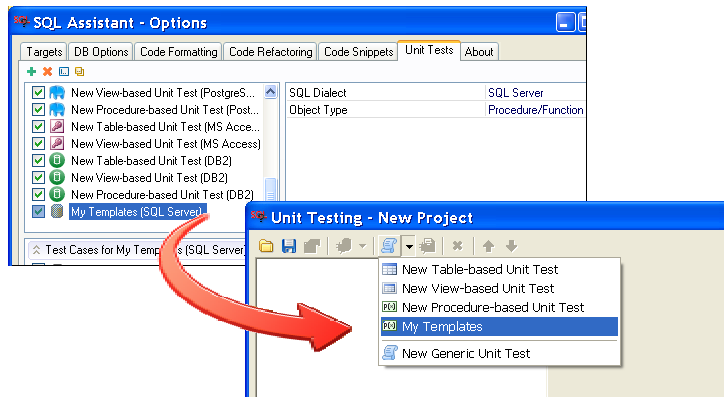
Tip: You can use DB, $SCHEMA$, and $OBJECT$ macro-variables in test case and unit test names to generate object specific names. See the Macro-variables and Dynamic Code Generation topic in CHAPTER 7 for more information about supported code macros and their usage.

6. On the right side of the Options dialog, use the SQL dialect property to choose the correct database server type to be associated with the new template type. It is important that you select the correct database type. SQL Assistant uses different code execution and validation methods for different database types.

7. On the right side of the Options dialog, choose the database object type to be associated with the test cases in the new unit test type. This selection controls the kind of object selection dialog displayed on the screen for test cases in the new unit test type. For more information, see the graphical examples in the previous topic Using Custom Templates for Generation of Test Cases.

8. Add new test case templates as instructed in the Adding and Removing Test Case Templates topic.

9. Click the OK button to save all changes and close the Options dialog.

Tip: Drag-and-drop unit test type names in the left-top list to rearrange their order. You can use this method to push most commonly used types to the top of the list and minimize the amount of scrolling and clicking required to customize test case templates.

To delete an existing unit test type and all associated test case templates:

Follow steps 1 to 2 as described above. In place of step 3, select the unit test type you want to delete. Click the Delete Sign icon in the top left corner of the **Unit Tests** tab page.

Tip: To temporarily disable any unit test type without deleting it, deselect the checkbox in front of the type name. This will remove the disabled unit test type name from SQL Assistant menus. The unit test type can be re-enabled at a later time if you want to use it again.

Adding and Removing Test Case Templates

To add new custom test case template:

1. Double-click the SQL Assistant icon in the Windows system tray. The Options dialog will appear.

2. Click the **Unit Tests** tab page.

3. In the top left corner of the Options dialog, click the name of the unit test type to which you want to add a new test case template.

4. Click the Test Cases box below the types list.

5. Click the Plus Sign icon ➕ in the top left corner of the **Unit Tests** tab page.

6. Enter a descriptive name for the new test case template.

7. On the right side of the Options dialog, enter the SQL code for the new test case template.

   Tip: The template code may contain references to macro-variables supported by SQL Assistant. You can use the drop-down menu in the top right corner of the Options dialog to paste macro names into the template code. See the Macro-variables and Dynamic Code Generation topic in CHAPTER 7 for more information about supported code macros and their usage.

8. Click the OK button to save all changes and close the Options dialog.

To delete an existing test case template:

Follow steps 1 to 3 as described above. In place of step 4, select the test case template you want to delete from the Test Cases box. Click the Delete icon ❌ in the top left corner of the **Unit Tests** tab page.

Tip: To temporarily disable any test case template type without deleting it, deselect the checkbox in front of the template name. SQL Assistant will ignore the disabled template when generating test cases using the unit test type of the selected template. The template can be re-enabled at a later time if you want to use it again.

Modifying Templates

To modify an existing test case template:

1. Double-click the SQL Assistant icon in the Windows system tray. The Options dialog will appear.

2. Click the **Unit Tests** tab page.

3. In the unit test type list in the top left corner of the Options dialog, click the name of the unit test type containing the template you want to modify.

4. In the Test Cases box, click the name of the test case template you want to modify.

5. On the right side of the Options dialog, modify the SQL code for the test case template as needed.

   Tip: The template code may contain references to macro-variables supported by SQL Assistant. You can use the drop-down menu in the top right corner of the Options dialog to paste macro names into the template code. See the Macro-variables and Dynamic Code Generation topic in CHAPTER 7

for more information about supported code macros and their usage.

6. Click the OK button to save all changes and close the Options dialog.

Command Line Interface

To run a unit testing project from a DOS command line window, use the following command:

```
sacmd ut:"path-to-project-file " sas:"path-to-sa-settings-file" conn:"myserver (userid)"
```

Substitute values in the command as follows:

path-to-project-file	The full file name of the unit testing project file
path-to-sa-settings-file	The full file name of the SQL Assistant settings file containing the required database connection parameters. This is an optional parameter. If not specified, the default path for the current user account is used.
myserver (userid)	The database connection name

Example:

cd "C:\Program Files (x86)\SQL Assistant 12"
sacmd ut:"C:\Projects\ Testing\dev_ut_pos.xml" sas:"%APPDATA%\SQL Assistant\13.0\sqlassist.sas" conn:"DEV001 (sa)"

 Important Notes:

- Unit testing project files are XML files that you save using the Unit Testing graphical interface

- SQL Assistant settings file location is version and user profile specific. See notes in Overview topic in CHAPTER 51 for details on how to find out the location of that file.

- You can find out the connection name in DB Connections group of settings on DB Options tab page in SQL Assistant Options. In case the connection requires user id and password, make sure that both are saved in the settings. The command line interface does not display any interactive prompts and is unable to prompt for credentials during command processing. For more information about storing and managing database connections, see Managing Database Connections topic in CHAPTER 48.

CHAPTER 20, SQL Syntax Checker

Overview

SQL Assistant's **SQL Syntax Checker** enables you to check for SQL syntax for validity without actually having to execute the code. Using this tool you can find and correct any errors, ensuring that your code will work correctly when executed.

SQL Assistant supports two modes for the SQL Syntax Checker: **Automatic** (near-real time) mode and **Manual** mode.

Automatic Mode

In **Automatic** mode, SQL Assistant constantly checks the syntax of SQL code entered into the editor and displays warnings whenever an error is encountered. Warnings are displayed as colored stripes on the Syntax Indicator Bar displayed on the right side of the editor window. Red stripes indicate syntax errors. Yellow stripes indicate syntax warnings. The position of a stripe on the Syntax Indicator Bar indicates the relative position of the error or warning in the code.
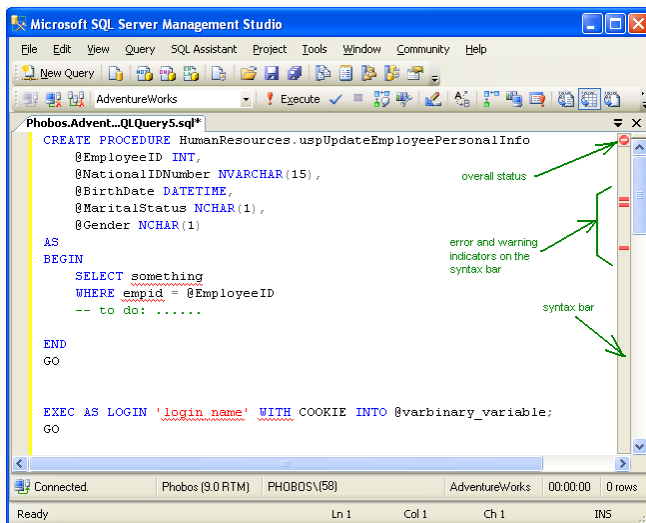
In **Automatic** mode, SQL Assistant also uses red, wavy lines to underline syntax errors in the code. To see the actual error or warning message, rest the mouse over the indicators on the Syntax Indicator Bar or over the underlined text in the editor.

To quickly navigate to lines of code containing syntax errors, click on the corresponding error indicators displayed on the Syntax Indicator Bar.

Tip: The small box at the top of the Status Indicator Bar is an overall syntax status indicator for the current file. A green checkmark in this box indicates there are no errors or warnings in the current file. A red stop sign indicates the presence of possible errors during code execution.

Manual Mode

In **Manual** mode, the syntax checker outputs the results of a syntax check to a separate window named **Syntax Check**. Each message in the Syntax Check window begins with the line number of the code line where a syntax error has been found. The line number is followed by the text of the message. In certain cases, several messages might be reported for the same SQL statement. In this case, several lines with the same line number will appear in the Syntax check window.
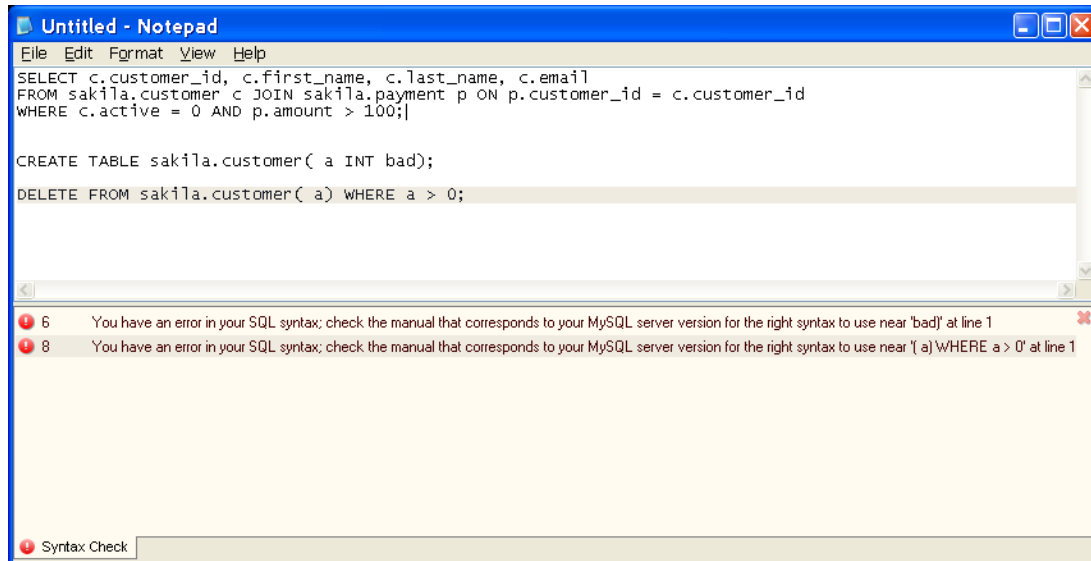
Important Notes:

- The **SQL Syntax Checker** uses whatever syntax checking method is provided by your database server software. It is your database software, not SQL Assistant, that validates the code and reports on any syntax errors encountered.

- Because database behavior and available syntax checking methods differ between database servers, the results of a syntax check may depending on the environment.

- All error messages returned by the database are displayed in the **Syntax Check** tab. If no syntax errors are found, a simple **Syntax Check Completed** message is displayed.

Important Notes for Oracle Targets: Oracle's methods for parsing and checking syntax of DDL statements causes these statements to be executed immediately. You are advised to use SQL Syntax check selectively and apply it to DML statements only.

Special Considerations

The syntax checker internally invokes the appropriate database server APIs for performing SQL syntax checks. Errors and warnings that appear in the syntax checking pane or on the syntax bar are errors and warnings reported by your database server.

Syntax checking results can be affected by the database connectivity layer. Certain types of database drivers perform pre-parsing of SQL commands on the client side and perform internal command conversion before commands reach the database server. Because different drivers can use different conversion rules, syntax checking results may vary in different environments with different connectivity methods and driver versions.

Important Notes for Microsoft Access:

The behavior of the default ODBC driver installed on most Windows systems may significantly affect syntax checking and query execution results for Microsoft Access databases. Certain types of queries that can be successfully executed in Microsoft Access fail to execute when submitted via an ODBC connection. These queries also fail to produce valid syntax check results. Keep in mind that SQL Assistant connects to Microsoft Access targets using an ODBC driver and is therefore affected by driver behavior. For example, queries that use double quotes as string delimiters will make SQL Assistant and the driver report various syntax errors that are difficult to understand. This problem occurs because, for compliance with ANSI standards, the driver expects double quotes only as table and column name delimiters. You can avoid this problem by using only single quotes as string delimiters.

Example: The following query will execute successfully in the Microsoft Access IDE, but will fail to execute and fail syntax check in SQL Assistant:

```
SELECT * FROM Cities WHERE name = "Tokyo";
```

The following query will successfully execute and pass syntax check:

```
SELECT * FROM Cities WHERE name = 'Tokyo';
```

Important Notes for Oracle systems:

Syntax checking of DDL commands is not currently supported, because this feature it is not supported by Oracle database servers.

Important Notes for MySQL systems:

Syntax checking may work sporadically or may not work at all if you use a MySQL ODBC driver with a version number less than 5.1.4.

Syntax checking in a MySQL server version prior to version 5.0.34 may cause MySQL server to crash. This is caused by a known MySQL bug. It is recommended that you upgrade to MySQL server version 5.0.34 or later to resolve this issue.

Important Notes for all database systems:

SQL Assistant takes into account SQL batch delimiters when parsing and syntax checking SQL code . Each SQL batch is evaluated independently. As a result, any variables and temporary objects declared in one batch are not visible to other batches, and references to such objects will be reported as invalid.

Example: The following SQL code will fail a syntax check:

```
SELECT * INTO #temp_table FROM Cities WHERE name = 'Tokyo'
INSERT INTO #temp_table SELECT * FROM Cities WHERE name = 'Montreal'
go

SELECT * FROM #temp_table
go
```

In this example, the last SELECT statement will report syntax errors indicating that the temporary table #temp_table doesn't exist. This is because the second batch is checked independently and, in that context, the referenced table is not valid. If the entire script is executed, however, and the first batch completes successfully, the second batch will execute successfully as well.

Working with SQL Syntax Checker Automatic Interface

Starting the SQL Syntax Checker

By default, the SQL Assistant's Syntax Checker is set to operate in automatic mode. You do not need to do anything special. The Syntax Checker activates automatically after a few seconds of inactivity after it senses code changes in the current file.

Controlling Syntax Check Frequency

You can use either of the following methods to adjust SQL syntax checker frequency:

- Use SQL Assistant's system tray icon menu (see the Using System Tray Icon Menu topic for details) and choose the **Target / Check SQL Syntax** / **Auto Check** submenu. In the submenu select the desired frequency.
- Use target editor's context or top-level menus, if menu integration option is enabled (see the Using Context and Top-level Menus topic for details). In that menu select the **SQL Assistant / Check SQL Syntax / Auto Check** submenu. In the submenu select the desired frequency

Tip: If you are working with a slow database connection or a busy database server, it is recommended that you choose a low frequency value. If you are working with a fast connection, it is recommended that you choose a higher frequency value so the error and warning indicators appear more quickly.

Turning off Automatic Syntax Checking Mode

You can use either of the following methods to turn of automatic SQL syntax checking mode.

- Use SQL Assistant's system tray icon menu (see the Using System Tray Icon Menu topic for details) and choose the **Target / Check SQL Syntax** / **Auto Check** / **Off** command.
- Use target editor's context or top-level menus, if menu integration option is enabled (see the Using Context and Top-level Menus topic for details). In that menu, select the **SQL Assistant / Check SQL Syntax / Auto Check** / **Off** command

SQL Assistant remembers your settings for the target and will not use automatic mode until you set the syntax check frequency value.

Working with SQL Syntax Checker Manual Interface

Starting the SQL Syntax Checker

You can use either of the following methods to manually start the SQL syntax checker.

- Use the default Ctrl+Shift+F9 hot key or a custom hot key (if you changed the default key).
- Use SQL Assistant's system tray icon menu (see the Using System Tray Icon Menu topic for details) and choose **Target / Check SQL Syntax** command.
- Use target editor's context or top-level menus, if menu integration option is enabled (see the Using Context and Top-level Menus topic for details). In that menu, select **SQL Assistant / Check SQL Syntax** command.

Tip: As with all other SQL Assistant functions, the **SQL Syntax Check** function can be applied either to the entire file in the editor or just to the highlighted text. If no text is highlighted, SQL Assistant validates the entire file.

The results of the syntax checking are displayed in the **Syntax Check Output** window

Scrolling Syntax Check Content

Use the scroll bars available in the **Syntax Check** window to scroll the content. Alternatively, you can use the keyboard navigation keys or the mouse wheel.

Locating Syntax Errors

Another helpful feature of the **Syntax Check Output** window is dynamic code highlighting. When you move mouse pointer over an error message displayed in the **Syntax Check Output** window, SQL Assistant highlights the referenced SQL statement in the target editor if the statement is visible in the editor window. If the referenced statement is not visible, click on the error message and SQL Assistant will automatically scroll to that position in the file and set the cursor to the beginning of the referenced SQL statement.

Resizing Content

To resize the **Syntax Check** window, drag the top edge of the window up or down. Note that when you place mouse pointer over the top edge of the **Syntax Check** window the cursor shape changes to resize shape as on the following screenshot.

Make sure the cursor takes the right shape before dragging the window edge.

CHAPTER 21, SQL Performance Monitoring and Tuning

Overview of Performance Tools

SQL Assistant provides a number of tools for analyzing and tuning database performance including the following

- Performance dashboards, see Performance Dashboards

- Background SQL code performance analyzer, see Overview of SQL Performance Analyzer and Execution Plans

- Graphical query execution plans, see SQL Execution Plans and Query Tuning

- SQL Profiler, see
  SQL Server SQL Profiler
  Oracle SQL Profiles
  PostgreSQL SQL Profiler

- "Top SQL" monitors for identifying long running recurring SQL queries, as well as top most resource intensive SQL queries and procedures.

- Database load testing tools, see CHAPTER 36, Testing Database Performance Under Heavy Load

- Index and Table defraggers

- Historical SQL query performance and execution plans
  Query Store Explorer and reports for SQL Server
  Active Session History Explorer and reports for Oracle (ASH Explorer)
  Automatic Workload Repository Explorer and reports for Oracle (AWR Explorer)
  See Reviewing and Comparing Historical Execution Plans

- Updating schema object statistics, see Updating Schema Object Statistics and also Database and Schema Scope Aggregated Statistics

Authorization Requirements

You generally need DBA level authorization for using Performance Dashboards and Top SQL monitors.

There are no special authorization requirements for using background SQL code performance analyzers and query execution plans. However when working with Oracle and DB2 servers you would need to have access to the existing EXPLAIN PLAN tables or have sufficient permissions to create new plan tables.

The authorization requirements for using the Database Benchmark utility depend on your custom test cases and not specific to the internal working of that utility.

Performance Dashboards

The database performance dashboards available in SQL Assistant SQL Editor enable you to monitor your database server performance across key dimensions, including but not limited to resource utilization, CPU and memory load, buffer hit ratios, active queries, wait times, applications and users connected to the database server, and other key metrics. The enable you to quickly identify root causes of slow database performance, database server usage patterns, and help you with figuring out what to fix to improve the performance.

As of SQL Assistant version 13.0 performance dashboards are supported for the following database servers:

- SQL Server v2008 and later, all editions

- Azure SQL Database and Azure SQL Managed Instances

- Oracle 11.1 and later, all editions

- DB2 for Linux, UNIX and Windows v10.1 and later; all editions

- PostgreSQL v9.4 and later

- MariaDB v10.0 and later

- MySQL v5.6 and later
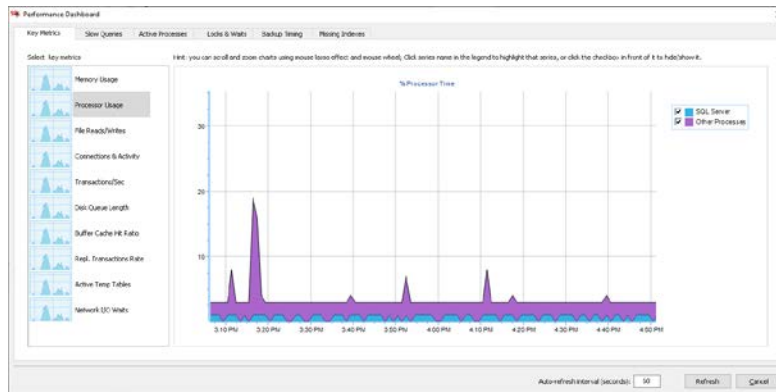
Requirements

All performance dashboards enable to monitor your database servers remotely from your workstation. The dashboards use agent-less architecture, and do not consume any server resources when the dashboard is not in use.

There is a special requirement though for MySQL and MariaDB servers. If you choose to collect operation system performance metrics including CPU and memory usage and also to monitor the disk input/output subsystem, you would need to deploy open source SQL Assistant provided performance data collection library. The direct onscreen instructions for downloading and installing that library are provided within the performance dashboard screen.

In addition, for MySQL and MariaDB you will need to enable their **performance_schema** schema and stats collection, and for analysis slow queries and their bottlenecks, you would need to enable the Slow Query Log feature.

For PostgreSQL servers you will need to enable the pg_stats_statement extension if you choose to utilize the Slow Queries monitor.

# Overview of SQL Performance Analyzer and Execution Plans

SQL Assistant automatically checks your SQL code for potential performance issues while you make changes in the editor. It analyzes the semantic structure of the code and referenced objects, and highlights problematic places while offering suggestions for performance improvements. Problem sections in the code are underlined with wavy, orange lines. In addition orange stripes are displayed on the Syntax Indicator Bar to the right of the editor window enabling at a glance visualization and quick code navigation. The orange stripes indicate locations of performance warnings. The position of a stripe on the Syntax Bar marks the relative position of the corresponding warning in the code.

Performance analysis options and behavior can be customized in SQL Assistant's Options. See CHAPTER 48, Customizing Performance Analysis Options topic for more details.

Performance Evaluation Rules

SQL Assistant uses numerous performance evaluation rules to check your code for potential performance issues. Some rules are static and based on best database coding practices such as improper references to indexed or non-indexed columns. For example, SQL Assistant checks your code for masked indexed columns in expressions within WHERE and JOIN clauses. Other rules are dynamic and based on evaluation results of estimated code execution plans. For example, SQL Assistant looks for various expensive operations, such as full table scans in tables containing lots of records. Note that estimated execution plans are provided by your database server. SQL Assistant relies on the database engine optimizer suggestions for estimated code execution plans.

The following screenshot shows a couple of performance warnings related to incorrect use of indexes.
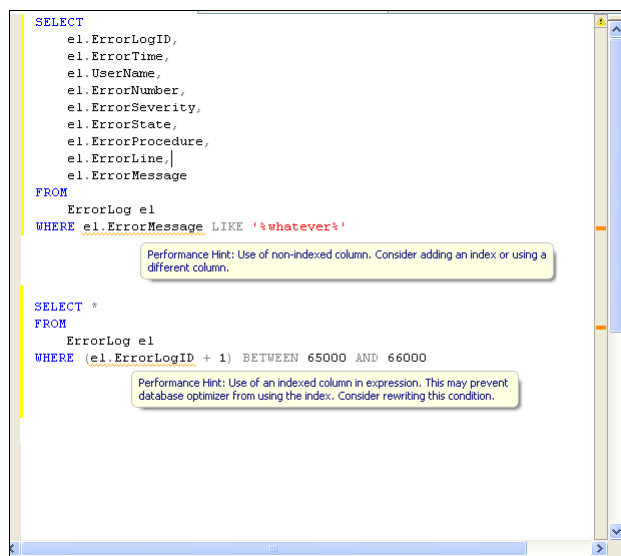
Tips:

- To see a count of the number of records in a database table, rest the pointer over the table name for a couple seconds. A yellow hint window will appear with the row count, table column definitions, column indexes, and other useful information.

- Rest the cursor over a column name highlighted in the performance warning to see a performance improvement suggestion.

- Click the "Info" link in the hint to see table performance analysis statistics (if this information is provided by your database). In SQL Assistant options, you can customize row-count thresholds and other parameters controlling SQL Assistant's performance analysis behavior.

SQL Execution Plans and Query Tuning

Overview of Query Tuning

A SQL query execution plan is a series of steps chosen or estimated by the database server to execute your query. It is important to generate and review execution plans for your application queries. Knowledge of their execution plans is necessary for query performance tuning. Analyzing and tuning of SQL queries is tedious yet indispensable. It involves several basic steps:

1. Identifying poorly performing application processes and the associated database queries.
2. Analyzing query execution plans and identifying the overly expensive operations that consume large amounts of the system resources.
3. Implementing corrective actions to resolve problems identified in steps 1 and 2 above.

Query Execution Plans

SQL Assistant provides you with an advanced graphical **Execution Plan** tool for visualizing and analyzing SQL query plans. With the help of the graphical plan, you can perform checks such as the following:

- Identify places in the code where you should add indexes to tables to improve performance of data lookup operations.

- Check whether the database query optimizer joins tables in an optimal order.

- Evaluate how resources are spent on each internal operation.

- Compare relative cost of different query steps and query versions, rewrite the query using different types of SQL constructs or operators returning the same results. For example, compare query versions using IN (list) operator vs. EXISTS (subquery), CTE query version vs. regular query version, and so on.

To generate a query execution plan:

1. In your SQL Editor, highlight the query you want to analyze.
2. Right-click the selection to display the context menu. In the **SQL Assistant** submenu, click the **Execute/Schedule SQL** -> **Show Execution Plan** command. The Execution Plan tab will open at the bottom of the editor window.

SQL Assistant presents execution plans in several formats that can be accessing differtent tabs in the Explain

Plan pane:

1. **Tree** - An expandable graphical tree view grouping related and nested operations within the plan.

The names of individual operands and operations within the plan differ for different types of database servers. Consult your database server documentation for details on supported operation names, their meaning and usage.

Plan subtrees can be collapsed to hide reduce the clutter and to focus on the costliest steps

Tip: To quickly collapse and expand multiple levels in a long, multi-page plan, use right-click context menu in the **Execution Plan** tab. Choose either a specific level to collapse or expand all tree nodes to; or choose Collapse All context menu command to collapse the tree to its root level.

2. **Diagram** - A graphical diagram with expandable nodes. A diagram appearance can be customized using its toolbar providing controls for diagram orientations, block alignment, zooming, choice of performace metrics shown in the diagram. Mouse-over hints provide additional information for various operations.

The names of individual operands and operations within the diagram differ for different types of database servers. To see details for a specific step, mouse-over the step and rest cursor for two seconds.

Plan subtrees can be collapsed to reduce the clutter. To collapse or expand a subtree, click the [-] connector port icon near the first subtree block. The subtree will hide and the block's connector port icon will change to [+] to indicate the block is now collapsed.

The **Bird's Eye View** control supports quick diagram navigation for complex diagrams stretching across multiple pages. In the control, drag the red rectangle to scroll the diagram and move the viewport.

For your convenience the diagram can be oriented in four different ways with the final step appearing in the beginning or end of the plan.
Use the ⬚ toolbar menu button to choose the preferred orientation. The following options are supported:

- Left to Right
- Top to Bottom
- Right to Left
- Bottom to Top

To switch between compact display format and wide display format with step alignment, use the ⬚ toolbar button.

🔖 **Tip:** To display more diagram elements, you can use the **Zoom** control available in the Explain Plan toolbar. You can also free more space by hiding the **Bird's Eye View** control shown to the left of the diagram. Right-click anywhere in an empty part of the diagram, and click then **Bird's Eye View** command in the context menu. To restore the control, repeat the above operation.

3. **Chart** – An interactive chart for comparing relative performance cost and other metrics of different query steps. Use the **Highlight** drop-down list on the Execution Plan toolbar to choose a different performance metric for steps comparison. The screenshot below demonstrates sample query plan with a steps comparison by estimated number of rows read.
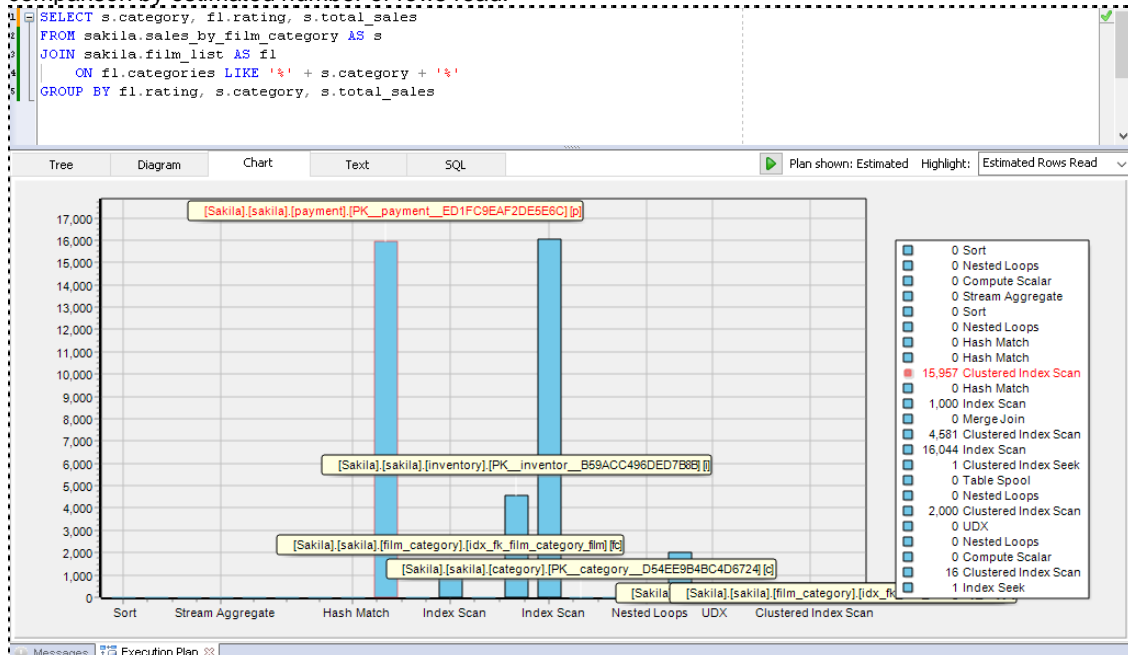
Mouse-over different names in the chart legend or the chart bars to have them highlighted in red in both legend and in the chart area as demonstrated on the screenshot.

4. **Text** - A raw plan returned by the database server. The actual presentation and formatting of the raw plan varies for different database server types. This is the data that SQL Assistant uses and parses internally to render the tree-view based plan and diagram based plan, as well as the data for the steps comparison

chart. The screenshot below demonstrates the raw plan in XML-format returned by SQL Server. Other supported raw format presentations are plain text, JSON, and tabular text.

Additional raw format specific commands and options are available via right-click menu in the **Text** tab.

5. **SQL** – this tab is used to show the SQL statement for which the plan is generated. It might be usefull if you choose to generate plans for a script with multiple statements, and to compare plans for different variants of the same query. For each statement a separate Explain Plan pane will be opened, and the statement text will be copied to its **SQL** tab.

Reading and Understanding Execution Plans

The names of operands and operations shown in the execution plan differ for different types of database servers. The names of individual operands and operations within the plan differ for different types of database servers. Consult your database server documentation for details on supported operation names, their meaning and usage. The following links are provided for your convenience:

DBMS	Explain Plan documentation
Microsoft SQL Server	http://msdn.microsoft.com/library/ms191158.aspx
DB2	ftp://ftp.software.ibm.com/ps/products/db2/info/vr9/pdf/en_US/db2tve90.pdf
Oracle	https://docs.oracle.com/cd/B19306_01/server.102/b14211/ex_plan.htm#i23461
MySQL	http://dev.mysql.com/doc/refman/5.7/en/explain-output.html
MariaDB	https://mariadb.com/kb/en/explainhttps://mariadb.com/kb/en/explain
Sybase ASE	https://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.infocenter.dc00743.1570/html/queryprocessing/queryprocessing44.htm
Sybase ASA	https://infocenter.sybase.com/help/topic/com.sybase.help.sqlanywhere.12.0.1/dbusage/queryopt-s-5395615.html
PostgreSQL and Greenplum	http://wiki.postgresql.org/wiki/Using_EXPLAIN

Amazon Redshift	https://docs.aws.amazon.com/redshift/latest/dg/c-the-query-plan.html
Teradata	https://docs.teradata.com/r/Teradata-VantageTM-SQL-Data-Definition-Language-Detailed-Topics/June-2020/BEGIN-QUERY-CAPTURE-COMMENT/BEGIN-QUERY-LOGGING/Logging-Options-for-BEGIN-QUERY-LOGGING
Snowflake	https://docs.snowflake.com/en/sql-reference/sql/explain
SQLite	https://www.sqlite.org/eqp.html **Important note:** SQLite's EXPLAIN QUERY PLAN command output provides only plan steps and predicates. No performance information like operation cost, CPU and I/O metrics, or estimated row counts is returned by the command. The plan output can be used identify which indexes are being used or not used, and use that to tune SQLite queries.

Query Plan Types, Estimated vs. Actual

SQL Assistant supports two types of query execution plans: Estimated and Actual.

The Estimated plans are generated without actually executing the query. The database query optimizer (also commonly known as query planner) creates query execution plan after considering available table and index statistics. The query optimizer may consider a number different execution plans and choose one that it estimates is going to be most efficient and return results faster.

The Actual query plans are created after the query is sent to the database for processing, and the plan contains the steps that were performed by the database and the actual metrics and details about the query steps, their execution time, the actual number of rows read or updated in the database.

Important notes: Collecting the Actual query execution plan requires executing queries in the database. Generating plans for UPDATE, DELETE, and INSERT queries may result in the data changes.

Not all databases provide actual query execution plans readily available. SQL Assistant uses various techniques to collect run-time metrics and details of actual query steps, which are broadly described below:

SQL Server - STATISTICS XML parameter is enabled during the query execution to make SQL Server collect and report query execution metrics.

DB2 UDB: - A combination of event monitors and SYSPROC.EXPLAIN_FROM_ACTIVITY procedure is used to gather execution step details and timing. In order to use the Actual plans, you must have CREATE EVENT MONITOR privileges and permissions to execute the SYSPROC.EXPLAIN_FROM_ACTIVITY procedure.

PostgreSQL and Greenplum– EXPLAIN ANALYZE is used to retrieve execution steps metrics.

Oracle - DBMS_XPLAN package is used with the query handle to retrieve run-time query plan and metrics after the query execution. In order to use the Actual plans, you must have execute privileges for the DBMS_XPLAN package.

MySQL 5.7 and later – A combination of EXPLAIN and query optimizer traces is used to retrieve additional execution details. Table aliases referenced in the plan are substituted with the actual table names to make the plan more useful, also generic subquery references in the plan like SELECT #N are replaced with the actual subquery text to make query plans easier to read.

Important Note: Be sure to use LibMySQL version 8.0 or later for your database connections. Earlier MySQL connector versions don't return the complete optimizer trace data and that impacts the quality of the constructed Actual execution plans.

Actual query plans are not supports with MySQL versions less than 5.7.

MariaDB 10.4 and later – A combination of EXPLAIN and query optimizer traces is used to retrieve additional execution details.

🖈 **Important Note:** Be sure to use LibMariaDB version 10.4 or later for your database connections. Earlier MariaDB connector versions don't return the complete optimizer trace data and that impacts the quality of the constructed Actual execution plans.

Actual query plans are not supports with MariaDB versions less than 10.4.

Redshift – Additional query statistics are retrieved from SVL_QUERY_REPORT function after the query execution and merged with the estimated query plan.

Snowflake – The query result cashing is disabled during query execution. The query operators statistics are retrieved after query execution using GET_QUERY_OPERATOR_STATS function and merged with the estimated query plan.

Teradata – The query capture facility QCD is used with detailed STATSUSAGE option to trace query execution. . After the query execution its operators statistics are retrieved from TDQCD.XMLQCD. In order to use the Actual plans, you must have INSERT privileges for TDQCD and permissions to execute the SYSPROC.EXPLAIN_FROM_ACTIVITY procedure.

Other databases – Actual query execution plans for the databases not listed above are not currently supported.

Locating Performance Impacting Operations for Query Tuning

SQL Assistant provides an easy one-click method to find the most expensive operations in a query. Click on the column header for a specific parameter to locate and highlight the costliest operations for that parameter.

Tips:

- The highlighted lines indicate "performance-expensive" operations. Red numbers in the highlighted lines indicate specific "performance-expensive" factors. The highlighting can help you with identifying operations in a query where tuning may be useful for improving query performance.

- The comparative analysis and highlighting of "performance-expensive" operations is based on the chosen performance cost criteria, for example, CPU Cost, I/O cost, and so on. Different criteria may produce different analysis results. You can choose the cost criteria metrics in the **Highlight** drop-down list on the Execution Plan toolbar.

- You can also use the **Chart** tab for the comparative analysis. The bars on the chart represent performance cost of different operations making it easy to identify the most expensive operations. You can find an example chart in the Query Execution Plans topic.

Comparing Execution Plans of Different Query Versions

SQL Assistant's multi-tabbed interface enables easy comparison of execution plans for different versions of the same query. A new tab and pane are opened for every query plan generated. Enter multiple query versions of the query you are tunning and generate an execution plan for each of them. With a single click, you can switch between tabs and visually compare the differences between plans.

Additional utilities are provided for Oracle and for SQL Server that enable graphical plan comparison and changes over time. You can use AWR Explorer and Reports available in the **Database -> Performance Monitoring and Logging** menu when connected to an Oracle database server. And Query Store Explorer and Reports available in the same menu when connected to a SQL Server database server. See the next topic for more details

Reviewing and Comparing Historical Execution Plans

SQL Server Query Store Explorer and Reports

The SQL Server Query Store feature available in SQL Server 2016 and later enables capturing a history of executed queries along with their query runtime statistics and execution plans. This historical statistics and execution plans can be browsed and analyzed using the **Query Store Explorer.** The **Query Store Explorer** and related performance reports are accessible via the top level menu **Database -> Performance Monitoring and Logging** menu in SQL Assistant Integrated Development Environments, and also using right-click menu in the Database Explorer component, see the **Performance Monitoring** menu.

The **Query Store Explorer** utility gives a graphical view of the historical query performance data within SQL Server database including historical queries executed, changes in their execution statistics over time, and also enables you to review and compare historical query plans using graphical query plan diagrams.
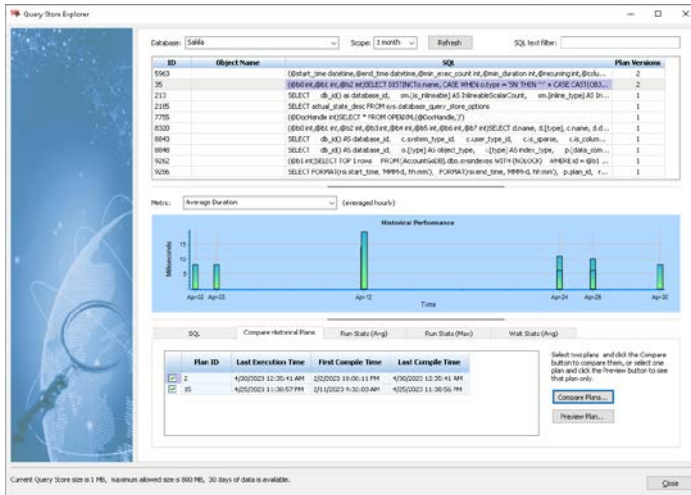
The **Historical SQL Query Performance Reports** utility provides reports for analyzing historical query performance changes, such as identifying queries having poor performance that steadily deteriorated over a certain time period, and identifying recurring queries continuously consuming a lot of database resources. This feature Is accessible using the menus described above. A number of parameters is available to customize the report behavior and filter our unimportant, rare, and other unwanted queries
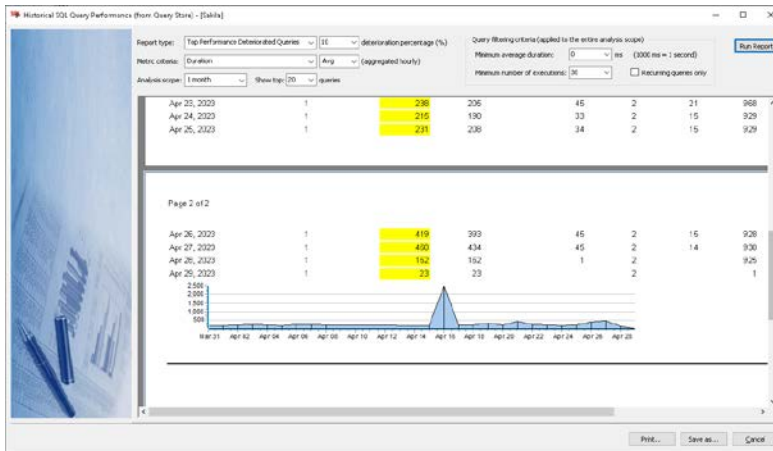
Important note: In SQL Server the Query Store feature used for capturing historical query performance and plans is not enabled by default. You can use the **Database Explorer** to enable that feature.

1. Right-click the database for which you want to enable the Query Store.

2. Select the **Properties…** menu. This will display the **Database Properties** dialog for the selected database.

3. In the Database Properties dialog click the **Query Store** tab. On that table change the **Operation Mode** property to READ_WRITE. Consult SQL Server documentation for description of the supported Query Store parameters, and if you wish to create a custom data capture policy.

4. Click the **Save** button to save changes in the Query Store state.

5. Wait for a few days or weeks to allow SQL Server to capture and record history of the executed queries and their statistics. The more history length is available, the more accurate picture of historical database performance changes can be reported by **Historical SQL Query Performance Reports.**

Oracle Active Session History Explorer

The Active Session History (ASH) feature is available in Oracle 10g and later. It is enabled by default in all editions, however its use requires licensing Oracle Diagnostic and Tuning Pack with the Enterprise Edition of Oracle. ASH samples "Active" sessions into memory buffer every second and in the background saves that data into ASH tables. That data is stored for a period of time, by default for 10 days. The storage policy can be configured in Oracle database settings. The captured statistics can be used to analyze historical application performance, identify periods of bad performance, and diagnose various performance issues. The SQL Assistant's **ASH Explorer** utility can be used to browse historical resource usage statistics by different database users and applications. The **ASH Explorer** is accessible via the top level menu **Database -> Performance Monitoring and Logging** menu in SQL Assistant Integrated Development Environments, and also using right-click menu in the Database Explorer component, see the **Performance Monitoring** menu.

The **ASH Explorer** gives a graphical view of the historical resource usage, including changes over time. Optional filtering is available to ignore system processes, scheduled jobs and other unwanted activities. The resource usage statistics are aggregated by users and/or application names, you can choose the aggregation method in the **Group By** settings.

Oracle Automatic Workload Repository Explorer and Reports

The Automatic Workload Repository (AWR) samples periodic snapshots of database activity between two points in time, and statistics are aggregated in hourly intervals and saved into AWR tables. AWR captures many different statistics, the captured data can be used to compare statistics captured during a period of bad performance to a baseline, and diagnose various performance issues. The captured data includes sampled SQL queries and their execution plans. The SQL Assistant's **AWR Explorer** utility and related AWR based performance reports can be used to browse historical query statistics and plans. The **AWH Explorer** and reports are accessible via the top level menu **Database -> Performance Monitoring and Logging** menu in SQL Assistant Integrated Development Environments, and also using right-click menu in the Database Explorer component, see the **Performance Monitoring** menu.

The **AWR Explorer** utility gives a graphical view of the historical query performance data within Oracle database including historical queries executed, changes in their execution statistics over time, and also enables you to review and compare historical query plans using graphical query plan diagrams.
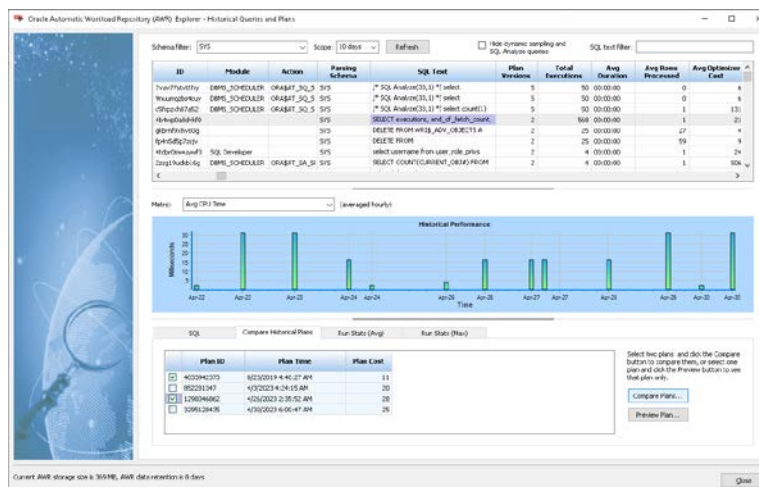
The **Historical SQL Query and Application Performance Reports** utility provides reports for analyzing

historical query and application performance changes, such as identifying queries having poor performance that steadily deteriorated over a certain time period, identifying recurring queries continuously consuming a lot of database resources, and other performance reports. This feature Is accessible using the menus described above. A number of parameters is available to customize the report behavior and filter our unimportant, rare, and other unwanted queries.
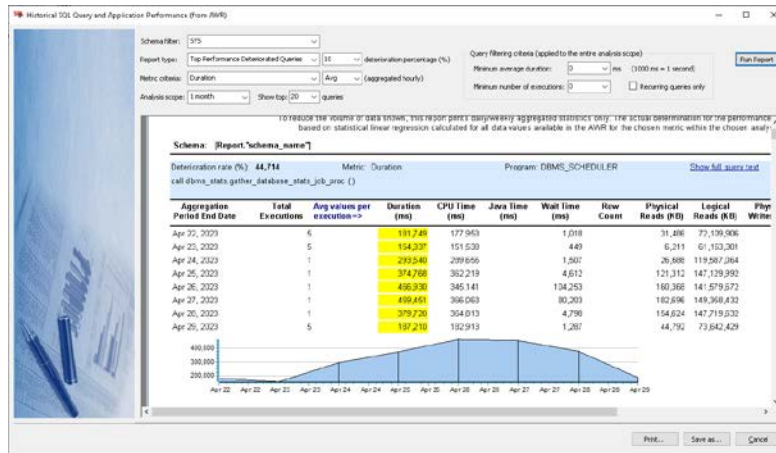
Important note: You will be able to use ASH and AWR Explorers and related performance reports if you have licensing Oracle Diagnostic and Tuning Pack.

SQL Server SQL Profiler

The SQL Profiler provides an easy to use task oriented graphical interface to the SQL Server trace functions which gather events as they occur in your SQL Server instance. The SQL Profiler gives you the ability to monitor everything that is going on inside your SQL Server instance. In addition to supporting user defined traces it provides a number of ready to run templates for the most common tasks including:

- Who is active? What's their performance impact?
- Which queries take most time?
- Which procedures run most frequently?
- What's causing deadlocks?   Long waits because of locks?
- Which procedures are most CPU intensive?
- Which procedures are most I/O intensive?
- Profile stored procedure execution, capture query performance stats
- Profile table usage, who is reading or writing to it?
- Profile application activities, capture executed queries
- Profile user activities, capture executed queries
- Profile session activities, capture executed queries

The SQL Profiler can be started from the main SQL Assistant menu from the Performance Tools menu. It is closely integrated with the SQL Assistant SQL Editor and can be started in a number of ways from within the SQL Editor with context specific preselected settings. For example, it can be started from the Sessions Monitor to profile activities of any given session; from the Database Explorer to monitor table or view usage, or to profile step by step execution of a stored procedure, and so on.

The captured trace events can be monitored in real time while the profiler trace is running. Charts are available in performance related reports. The trace data can be exported to external CSV and spreadsheet files for further analysis using office applications.

Tip: Multiple instances of the SQL Profiler can be started and run concurrently.

Oracle SQL Profiler

The SQL Profiler provides an easy to use task oriented graphical interface for profiling Oracle performance related usage by applications and users. It also includes graphical interface to the Oracle PL/SQL profiler, which can be used to trace and profile code execution of PL/SQL stored procedures, functions, packages, triggers, and other arbitrary PL/SQL code code and capture timing and execution statistics for individual SQL statements within the procedural code. The profiler provides a number of ready to run templates for the most common tasks including:

- Who is active? What's their performance impact?

- How scheduled jobs impact performance?

- Which queries take most time?

- Which PL/SQL procedures run most frequently?

- What's causing deadlocks?

- What's causing long waits because of locks?

- Which PL/SQL procedures are most CPU intensive?

- Which PL/SQL procedures are most I/O intensive?

- Profile PL/SQL code execution, capture query stats

- Profile errors, capture error-stack of faield code

- Profile table usage, who is reading or writing to it?

- Profile application activities, capture executed queries

- Profile user activities, capture executed queries

- Profile session activities, capture executed queries

The SQL Profiler can be started from the main SQL Assistant menu from the Performance Tools menu. It is closely integrated with the SQL Assistant SQL Editor and can be started in a number of ways from within the SQL Editor with context specific preselected settings. For example, it can be started from the Sessions Monitor to profile activities of any given session; from the Database Explorer to monitor table or view usage, or to profile step by step execution of a stored procedure, and so on.

Performance reports and charts are available at the end of the SQL Profiler run. The trace data can be exported to external CSV and spreadsheet files for further analysis using office applications.

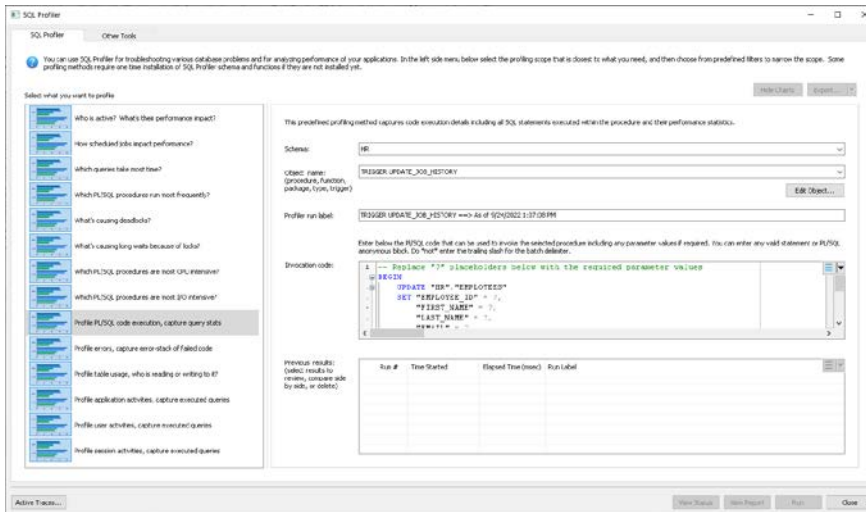**Tip:** Multiple instances of the SQL Profiler can be started and run concurrently.

Important note: Certain tasks supported by the Oracle SQL Profiler may require installation of additional database objects:

1. SA_SQLPROF schema for SQL Profiler internal tables.

2. SQLPROF_TRACE_DIR directory object for accessing trace files.

3. Oracle DBMS_PROFILER package and tables if they are not installed yet.

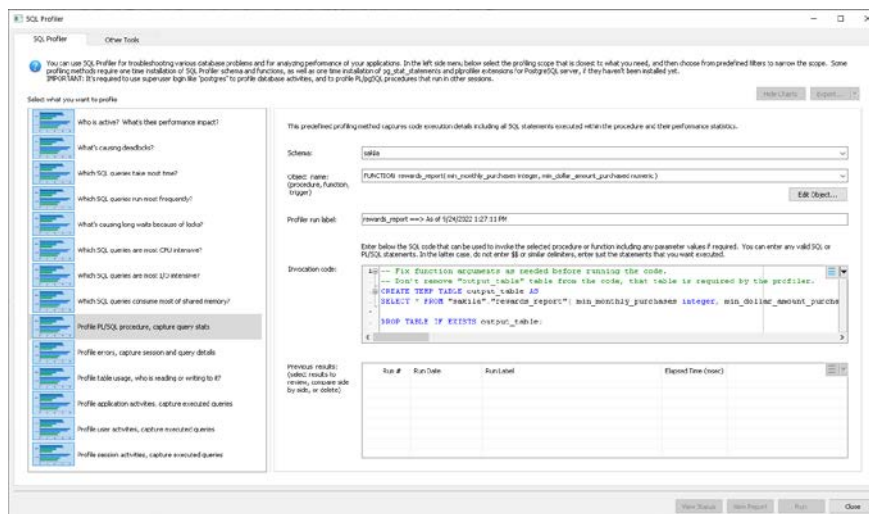If a selected profiler task requires one or more of the above and it's not yet installed in your database, you will be prompted for permissions to install them. If you decline, you cannot use the selected profiler task. The prompt will be shown again next time you select the same profiler task.

PostgreSQL SQL Profiler

The SQL Profiler provides an easy to use task oriented graphical interface for profiling PostgreSQL

performance related usage by applications and users. It also includes graphical interface to the PostgreSQL PL/SQL profiler, which can be used to trace and profile code execution of PL/SQL stored procedures, functions, triggers, and other arbitrary PL/SQL code and capture timing and execution statistics for individual SQL statements within the procedural code. The profiler provides a number of ready to run templates for the most common tasks including:

- Who is active? What's their performance impact?

- What's causing deadlocks?

- Which queries take most time?

- Which SQL queries run most frequently?

- What's causing long waits because of locks?

- Which SQL queries are most CPU intensive?

- Which SQL queries are most I/O intensive?

- Which SQL queries consume most of shared memory?

- Profile PL/SQL procedure, capture query stats

- Profile errors, capture session and query details

- Profile table usage, who is reading or writing to it?

- Profile application activities, capture executed queries

- Profile user activities, capture executed queries

- Profile session activities, capture executed queries

The SQL Profiler can be started from the main SQL Assistant menu from the Performance Tools menu. It is closely integrated with the SQL Assistant SQL Editor and can be started in a number of ways from within the SQL Editor with context specific preselected settings. For example, it can be started from the Sessions Monitor to profile activities of any given session; from the Database Explorer to monitor table or view usage, or to profile step by step execution of a stored procedure, and so on.

Performance reports and charts are available at the end of the SQL Profiler run. The trace data can be exported to external CSV and spreadsheet files for further analysis using office applications.

Tip: Multiple instances of the SQL Profiler can be started and run concurrently.

Important note: Certain tasks supported by the PostgreSQL SQL Profiler may require enablement or installation of additional database features and objects:

1. pg_stat_statements PostgreSQL extension

2. plprofiler PostgreSQL extension

If a selected profiler task requires one or more of the above extensions and they are available, but not yet enabled in your database, you will be prompted for permissions to enable them. If you decline, you cannot use the selected profiler task. The prompt will be shown again next time you select the same profiler task.

Identifying Long Running and Resource Intensive Recurring SQL Queries

Long running and resource intensive queries often cause intermittent problem with a database server. SQL Assistant provides Top SQL monitors for identifying top 20 resource consumers based on a few different metrics. It analyzes performance statistics available in the database dynamic performance views and historical query logs reflecting cached and repeating SQL queries and their performance metrics. The data is digested, aggregated, and then presented in a easy to understand graphical format along with query performance comparison charts and their impact on the server.
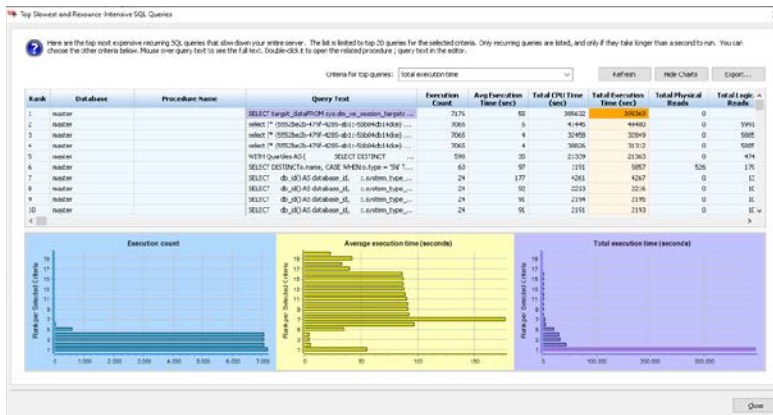
You can modify the default selection criteria by choosing different metrics from the **Criteria for Top Queries** drop-down list.

Mouseover the Query column cells to see the complete query text. Or double-click a sell to open its query text in a new SQL Editor tab. You can then use the SQL Execution Plan to further analyze database strategy for the query execution and identify hot spots and areas for improvements. See the Reading and Understanding Execution Plans topic for more details on execution plans and how to analyze them.

Tip: You can double-click a chart series of interest to quickly jump to the matching row and query text in the data grid above the charts.

CHAPTER 22, Spell Checker

Overview

SQL Assistant's **Spell Checker** seamlessly integrates Microsoft Word's multilingual in-line proofreading functions into your SQL editor. The Spell Checker is an intelligent tool that selectively checks for spelling errors in relevant parts of the SQL code. Because SQL scripts are special documents containing special SQL instructions, object references and object attribute references, as well as other programming constructs, checking the entire content of the script is really meaningless. Rather than checking the entire document, SQL Assistant parses the content first, then applies spell check functions only to comments and to hard-coded string values.

SQL Assistant provides two different ways to check spelling:

- **On-demand Spell Checker** – the interactive spell checker is activated manually using hot keys or context menus. The spell checker scans the highlighted text and provides spelling suggestions as required.

- **Real-time Spell Checker** – this non-interactive spell checker runs in the background at all times. As you type code, the spell checker checks the text and marks possible errors with red wavy underlines. You can then use the context menus to display suggested changed for marked words.

Important Notes: The Spell Checker requires that you have Microsoft Word 97 or later installed on the system. The Spell Checker can only use installed proofreading tools and languages. To check spelling in other languages, use the Microsoft Office installation CD to install additional language support.

Tip: If you wish to check the correctness of your SQL code syntax rather then the spelling, use the **SQL Syntax Checker** function. This function is described in Using SQL Syntax Checker topic in CHAPTER 3.

Using On-demand Spell Checker

The on-demand spell checker provides a straightforward interface that mimics spell check functions in many office and email programs. When it finds a possible spelling error, it displays the **SQL Assistant – Spelling** dialog window which lists suggested corrections and provides controls you can use to navigate, modify and ignore errors.

Like most other SQL Assistant functions, the spell check can be run for the highlighted portion of the text only or, if nothing is highlighted, for the entire content of the target editor. Typically, to run a spell check in SQL code, you should highlight the relevant portion of the text and then activate the spell checker. Then use the **SQL Assistant – Spelling** dialog to make necessary corrections. Note that the dialog disappears automatically when you reach the last error in the text, although you can close it at any time using the Esc key or the Close button.

You can use either of the following methods to invoke the interactive spell checker:

- Use the default Ctrl+Shift+F7 hot key or a custom hot key (if you changed the default key).
- Use SQL Assistant's system tray icon menu (see the Using System Tray Icon Menu topic for details) and choose the **Target / Check Spelling / Check** command.
- Use target editor's context or top-level menus, if menu integration option is enabled (see the Using Context and Top-level Menus topic for details). In that menu select the **SQL Assistant / Check**

Spelling / Check command.

The following example screenshot demonstrates the on-demand spell checker used with "DB Tools for Oracle" target editor.
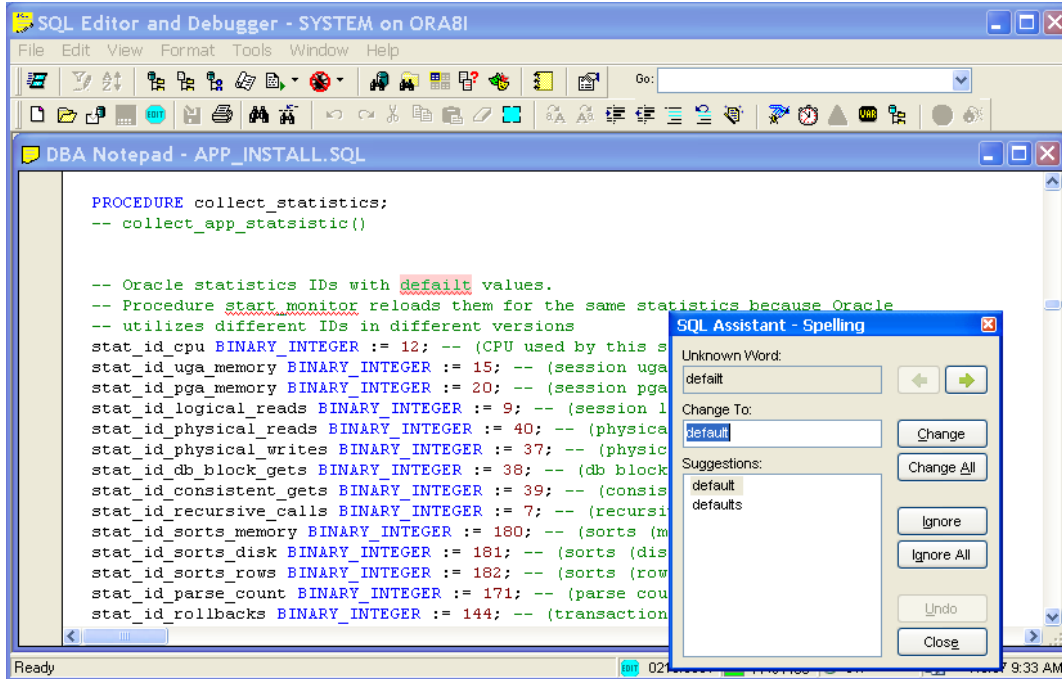
The following functions and controls are available in the spell checker dialog:

Unknown Word – displays a word that the spell checker doesn't recognize

Change To – displays the spell checker's suggested correction, if any. Click the **Change** button to accept the suggestion correction or type over the suggestion to insert a different correction.

Left Arrow and Right Arrow –controls used to navigate between misspelled words

Change – replaces the misspelled word with the text in the **Change To** edit box

Change All – updates all occurrences of the misspelled word in the editor with the text in the **Change To** edit box. You can use this function if the same spelling error repeats multiple-time.

Ignore – instructs the spell checker to leave the highlighted word unchanged and move on to the next error

Ignore All – instructs the spell checker to skip this and all subsequent occurrences of the highlighted word without making changes. Note that this selection is not persistent and applies only to the current spell check session.

Undo – this button undoes the last change. The button can be used repeatable to recursively undo previous changes. Note that only changes made in the current spell check session can be undone. To undo older changes use your editors **Edit / Undo** features.

Close –closes the spell checker dialog and ends the spell check session. It can be used at any time. Alternatively you can use the Esc hot key to dismiss the spell checker dialog

Important Notes:

Only the text in the editor is changed. If that text is associated with a file, the file is not updated until you use your editor's **File Save** feature.

If, for whatever reason, you want to undo the changes, use your editor's Edit | Undo features to undo the changes. If multiple spelling changes have been made, you may need to use the Undo feature multiple times, to reverse each consecutive change.

Using Real-time Spell Checker

The **Real-time Spell Checker** is non-interactive. If activated, the spell checker quietly runs in the background and marks possible errors as you type. Errors are marked with red wavy underlines. The spell checker only checks text in comments and in hard-coded string values; in other words, values enclosed in single quotes.

To correct errors, simply edit the misspelled words directly in the editor. If the new text is correct, the wavy underlines disappear automatically.

To see a list of spelling suggestions for a certain word, highlight the word in text and then use either of the following methods to open the spelling suggestions popup list.

- Use SQL Assistant's system tray icon menu (see the Using System Tray Icon Menu topic for details) and choose **Target / Check Spelling / Suggestions** command.
- Use target editor's context or top-level menus, if menu integration option is enabled (see the Using Context and Top-level Menus topic for details). In that menu, select the **SQL Assistant / Check Spelling / Suggestions** command.

The popup list with suggestions appears below the highlighted word. You can either select the correct word or press the Esc key to dismiss the suggestions list.
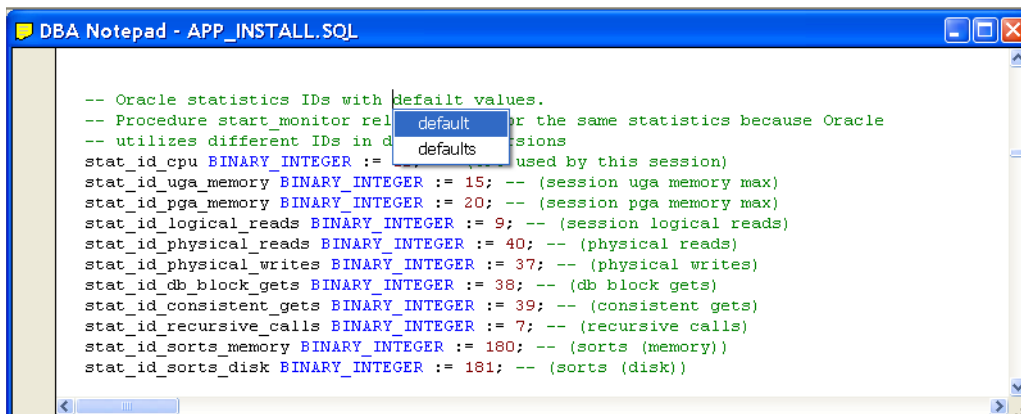
Similarly, if you would like to see a list of synonyms for the highlighted word, you can use the **Synonyms** command from the same menu to display a list of possible synonyms.

To deactivate the spell checker, use the **Deactivate** command available in the SQL Assistant menus, which are described in the previous paragraphs.

Choosing Spell Check Language

By default, the Spell Checker uses the language selected as the default language in the Microsoft Word options. If you need to check spelling in a different language, make sure that language support is installed on your system. If the required language is not available, use the Microsoft Office installation CD to install additional language support.

To pick a non-default language for the spell check, use either of the following options:

- Use SQL Assistant's system tray icon menu (see the Using System Tray Icon Menu topic for details) and choose **Target / Check Spelling / Language** command and then from the next level menu select the required language.
- Use target editor's context or top-level menus, if menu integration option is enabled (see the Using Context and Top-level Menus topic for details). In that menu, select the **SQL Assistant / Check Spelling / Language** command and then from the next level menu select the required language.

Tip: Language selection affects the current target editor session only. If you close the target editor and then reopen it, or simply switch to a different editing window within the same target editor, you would need to repeat the language selection again.

CHAPTER 23, Database Source Code Control Interface

Overview

SQL Assistant provides an advanced multi-platform and multi-database source code system (SCS) interface for team-based database development. It allows the following:

- Several developers can work with the same databases, schemas, and schema objects concurrently making changes in schema objects and application code.

- Store code of database schema objects in SCS repositories.

- Keep track of what was changed, when it was changed, and by whom and assign unique revision numbers to each change.

- Deploy database and schema code stored in a SCS repository to one or more database servers.

- Retreat to older versions stored in SCS repository, if you decide that the current version is broken or not yet fully ready for use.

- Undo accidental database schema object deletes.

- Developers can setup multiple SCS interfaces and concurrently use different SCS repositories with different development tools, projects, and databases.

Tip: SQL Assistant version 13.0 supports Subversion (SVN), Microsoft Team Foundation Server (TFS), Git and GitHub, and Perforce based source code control systems. Future versions may support other types of source code control systems.

Concepts and Source Code System Differences

Prerequisites

Source Control Interfaces

To use the Subversion interface you must have SVN client installed on your system .You can get the latest version here http://subversion.apache.org/packages.html. Subversion client is free.

To use the Team Foundation Server interface you must have TFS client installed on your system .To install TFS client, install Visual Studio 2008, Visual Studio 2010 or later, and make sure to select TFS related options. You can also install the free Team Foundation Server Explorer which will install TFS client.

To use the Git interface you must have Git client installed on your system You can get the latest version here

http://git-scm.com/downloads. Git client is free.

To use the Perforce interface you must have Perforce Helix version client installed. You can get the latest version here http://www.perforce.com/downloads/integrations?sitelink=PerforceClients.

SQL Server 2000, 2005, 2008, 2012, and 2014

Microsoft SQL Server's Server Management Objects (SMO) version 2008 or later must be installed on the computer running SQL Assistant . SMO is preinstalled with Microsoft SQL Server Management Studio (SSMS) and with Microsoft SQL Server Management Studio Express (SSMSE). if you don't have them installed, you can download and install SMO separately as part of Microsoft SQL Server 2008 Service Pack 2 Feature Pack http://www.microsoft.com/en-us/download/details.aspx?id=6375

SQL Azure

Microsoft SQL Server's Server Management Objects (SMO) version 2008 R2 SP2 or later must be installed on the computer running SQL Assistant . SMO is preinstalled with Microsoft SQL Server Management Studio 2008 R2 SP2 (SSMS) and with Microsoft SQL Server Management Studio Express 2008R2 SP2 (SSMSE). if you don't have them installed, you can download and install SMO separately as part of Microsoft SQL Server 2008 Service Pack 2 Feature Pack http://www.microsoft.com/en-us/download/details.aspx?id=6375

Repository

SQL Assistant uses temporary work area called workspace to maintain your client-side copies of database schema objects code and source code repository objects. When you add, edit, drop, rename, or otherwise manage any source-controlled item, your changes are persisted, or marked as pending changes, in the workspace. Pending changes are isolated in your local workspace until you commit them to the repository server and the database server.

Workspace

SQL Assistant uses temporary work area called workspace to maintain your client-side copies of database schema objects code and source code repository objects. When you add, edit, drop, rename, or otherwise manage any source-controlled item, your changes are persisted, or marked as pending changes, in the workspace. Pending changes are isolated in your local workspace until you commit them to the repository server and the database server.

You can synchronize your workspace with the most recent repository versions using the **Get Latest from Repository** command. If you want to get latest copies from the database server, use **Update Workspace from Database** command.

Creating a Workspace

To begin working with the SCS interface control, you need to establish at least one workspace. Creating a new workspace is simple create-folder operations. You can use Windows Explorer or command line tools to create a new workspace folder.

Multiple workspaces

Not all source code control systems allow using multiple workspaces. Changing workspace for a project may require project rebinding to the repository server, as well as it may cause loss of some project specific metadata.

If you need extra copies of project source files on your computer, you can use the Export command in the Repository Browser to copy files from your source code repository to a arbitrary folder on your computer.

Connecting to SVN Repository Server

The following screenshot demonstrates SQL Assistant's SVN repository connection.
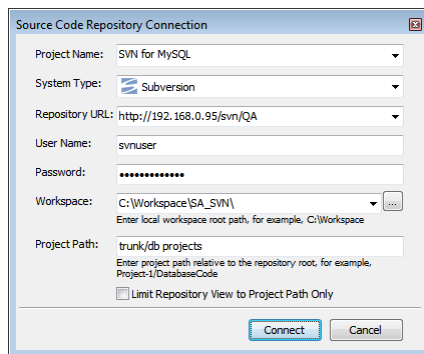
Connection parameters:

Project Name – logical project name grouping together business tasks and/or code releases. You can define multiple projects as needed.

System Type – Type of the source code control system. For SVN based repositories this value must be set to **Subversion.**

Repository URL – Repository server URL, typically HTTP or HTTPS URL pointing to the root folder of the SVN repository. This can be also a path to a local file based SVN repository.

User Name – your user name for the repository server connection. If user authentication is not required, leave this property blank. SQL Assistant will use Anonymous connection type.

Password – your password for the repository server connection. If user authentication is not required, leave this property blank. SQL Assistant will use Anonymous connection type.

Workspace – location of the work-folder containing files checked out from the repository. In SQL Assistant Options you can configure multiple workspaces and use different workspaces with different projects and/or environments. If not specified, SQL Assistant will create and use default workspace located in %APPDATA%\SQL Assistant\Workspace folder. Please note that the default location of the folder varies in different Windows versions. If you do not know the value of %APPDATA%\ environment variable, open DOS command prompt, and execute echo %APPDATA% command.

Project Path – Path to the project root in the repository, relative to the repository root. The combination of **Server URL** + **Project Path** should point to absolute location of project files in the repository. You can enter **Project Path** when entering connection parameter for the repository or you can select **Project Path** later using graphical menus in the Repository Browser dialog. See the Choosing Project Path in the Repository topic in this chapter for more details on how to properly choose Project Path and how it maps to the **Workspace**.

Limit Repository View to Project Path Only – by default the [Repository Browser](#) shows all folders in the repository under the repository root. This enables you to change Project Path on a fly as described in the [Choosing Project Path in Repository](#) topic. In case the repository is very large, it might be undesirable to retrieve and show all folders in the repository. If the **Limit Repository View to Project Path Only** option is enabled, the Repository Browser shows folders in the Project Path only. That is faster and more efficient, but effectively disables dynamic changing of the Project Path because no other folders are visible.

Connecting to TFS Repository Server

The following screenshot demonstrates SQL Assistant's TFS repository connection dialog.
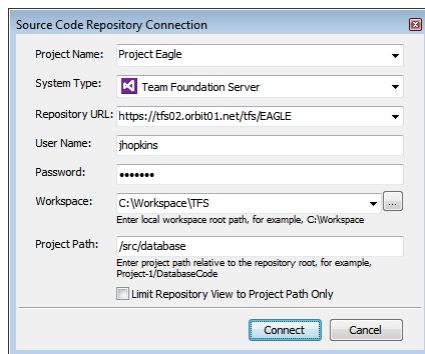
Connection parameters:

Project Name – logical project name grouping together business tasks and/or code releases. You can define multiple projects as needed.

System Type – Type of the source code control system. For TFS based repositories this value must be set to **Team Foundation Server.**

Repository URL – Repository server URL, typically HTTP or HTTPS URL pointing to the root folder of the TFS repository. This can be also a path to a local file based TFS repository.

User Name – your user name for the repository server connection. If user authentication is not required, leave this property blank. SQL Assistant will use Anonymous connection type.

Password – your password for the repository server connection. If user authentication is not required, leave this property blank. SQL Assistant will use Anonymous connection type.

Workspace – location of the work-folder containing files checked out from the repository. In SQL Assistant Options you can configure multiple workspaces and use different workspaces with different projects and/or environments. If not specified, SQL Assistant will create and use default workspace located in %APPDATA%\SQL Assistant\Workspace folder. Please note that the default location of the folder varies in different Windows versions. If you do not know the value of %APPDATA%\ environment variable, open DOS command prompt, and execute echo %APPDATA% command.

Project Path – Path to the project root in the repository, relative to the repository root. The combination of **Server URL** + **Project Path** should point to absolute location of project files in the repository. You can enter **Project Path** when entering connection parameter for the repository or you can select **Project Path** later using graphical menus in the Repository Browser dialog. See the following [Choosing Project Path in the Repository](#) topic in this chapter for more details on how to properly choose Project Path and how it maps to the **Workspace**.

Limit Repository View to Project Path Only – by default the [Repository Browser](#) shows all folders in the

repository under the repository root. This enables you to change Project Path on a fly as described in the Choosing Project Path in Repository topic. In case the repository is very large, it might be undesirable to retrieve and show all folders in the repository. If the **Limit Repository View to Project Path Only** option is enabled, the Repository Browser shows folders in the Project Path only. That is faster and more efficient, but effectively disables dynamic changing of the Project Path because no other folders are visible.

Connecting to Git Repository Server

Whether you use shared Git repository, which is often called "remote" repository, or you use local repository only, you can use the Source Code Repository Connection dialog to configure how SQL Assistant's database source control communicates with the repository. The following screenshot demonstrates SQL Assistant's Git repository connection dialog.
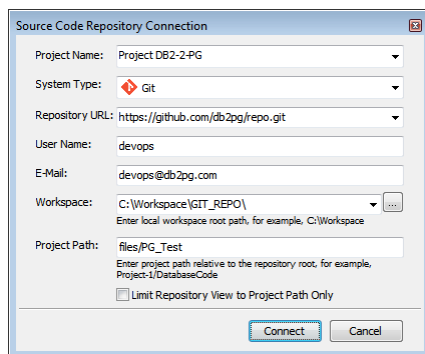
Connection parameters:

Project Name – logical project name grouping together business tasks and/or code releases. You can define multiple projects as needed.

System Type – Type of the source code control system. For Git and GitHub based repositories this value must be set to **Git.**

Repository URL – A remote (central) repository server URL, typically HTTP or HTTPS URL pointing to the root folder of the Git repository. That URL could be your repository on GitHub, or another user's fork, or even on a completely different server. Do not confuse this with the Git's local repository. You do not need to specify location of your local Git repository in SQL Assistant's settings because that is transparent to SQL Assistant's operations. SQL Assistant works in tandem with your Git client for staging, committing, and getting latest files from the local repository.

Note: If you don't use remote repositories, , you can leave this value blank.

User Name – your user name for the repository server connection.

Email – your email for the remote repository server connection. If you never commit your code to the remote repository, you can leave this value blank.

Workspace – location of the work-folder containing files checked out from the repository. In SQL Assistant Options you can configure multiple workspaces and use different workspaces with different projects and/or environments. If not specified, SQL Assistant will create and use default workspace located in %APPDATA%\SQL Assistant\Workspace folder. Please note that the default location of the folder varies in different Windows versions. If you do not know the value of %APPDATA%\ environment variable, open DOS command prompt, and execute echo %APPDATA% command.

Project Path – Path to the project root in the remote repository, relative to the repository root. The combination of **Server URL + Project Path** should point to absolute location of project files in the remote repository. You can enter **Project Path** when entering connection parameter for the repository or you can select **Project Path** later using graphical menus in the Repository Browser dialog. See the Choosing Project Path in the Repository topic in this chapter for more details on how to properly choose Project Path and how it maps to the **Workspace**.

Limit Repository View to Project Path Only – by default the Repository Browser shows all folders in the repository under the repository root. This enables you to change Project Path on a fly as described in the Choosing Project Path in Repository topic. In case the repository is very large, it might be undesirable to retrieve and show all folders in the repository. If the **Limit Repository View to Project Path Only** option is enabled, the Repository Browser shows folders in the Project Path only. That is faster and more efficient, but effectively disables dynamic changing of the Project Path because no other folders are visible.

Connecting to Perforce Repository Server

The following screenshot demonstrates SQL Assistant's Perforce repository connection dialog.
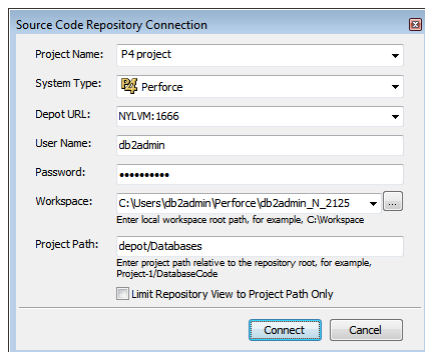
Connection parameters:

Project Name – logical project name grouping together business tasks and/or code releases. You can define multiple projects as needed.

System Type – Type of the source code control system. For Perforce based repositories this value must be set to **Perforce.**

Depot URL – Depot server location, typically in **host:port** reference. **Host** is the name of the machine where the Perforce server is running. **Port** is the port on which the Perforce server listens for requests.

User Name – your user name for the repository server connection. If user authentication is not required, leave this property blank. SQL Assistant will use Anonymous connection type.

Password – your password for the repository server connection. If user authentication is not required, leave this property blank.

Workspace – location of the work-folder containing files checked out from the repository. In SQL Assistant Options you can configure multiple workspaces and use different workspaces with different projects and/or environments. If not specified, SQL Assistant will create and use default workspace located in %APPDATA%\SQL Assistant\Workspace folder. Please note that the default location of the folder varies in different Windows versions. If you do not know the value of %APPDATA%\ environment variable, open DOS command prompt, and execute echo %APPDATA% command.

Project Path – Path to the project root in the depot, relative to the repository root. You can enter **Project Path**

when entering connection parameter for the repository or you can select **Project Path** later using graphical menus in the Repository Browser dialog. See the following Choosing Project Path in the Repository topic in this chapter for more details on how to properly choose Project Path and how it maps to the **Workspace**.

Limit Repository View to Project Path Only – by default the Repository Browser shows all folders in the repository under the repository root. This enables you to change Project Path on a fly as described in the Choosing Project Path in Repository topic. In case the repository is very large, it might be undesirable to retrieve and show all folders in the repository. If the **Limit Repository View to Project Path Only** option is enabled, the Repository Browser shows folders in the Project Path only. That is faster and more efficient, but effectively disables dynamic changing of the Project Path because no other folders are visible.

Basic Database to Workspace to Repository Comparison

The Repository Browser uses internal metadata saved by SQL Assistant and the selected source code control system as well as file and database object most recent modification times to locate and show differences between the database, workspace and repository. Any found differences will be highlighted using different colors and marked using overlay icons as described in the Icon Overlays topic later in this chapter.

The basic metadata based comparison is not always enough to locate all differences. When database objects do not have correct last modification time or that is simply not supported by the database server, such as in the case of PostgreSQL, Redshift and SQLite, the metadata can get outdated if database objects are modified by users directly bypassing the source code control interface. Another common case is direct changes in the database object attributes or related "child" objects. Such changes do not update the last modification date of the main object. Examples include changes in table indexes or table triggers, changes in table permissions, and some others that have a material impact on the table DDL code, but not update table's last modification time. To workaround this issue and to refresh the metadata, use the **Rescan Database** command available in the top-level and the right-click menus in the Repository Browser.

# Advanced 3-Way Code Comparison and Synchronization

The Repository Browser allows you to manage and synchronize content of the database, workspace and repository. You can synchronize the content between all 3 places. Note that any synchronization of the database content to the repository or vice versa explicitly updates your workspace content with the latest versions from the database server or from the repository server depending on the direction of the selected synchronization method.

The Repository Browser reads internal metadata saved by SQL Assistant and the selected source code control system as well as file and database object most recent modification times to show the differences between the database, workspace and repository. Use of any "rescan" or "compare" type of operations implicitly causes it to update the metadata.

Use the top-level or the right-click menus in the Repository Browser to start the **Compare Database to Repository**, **Compare Database to Workspace**, and/or **Compare Repository to Workspace** processes. SQL Assistant will perform scanning of your database schema objects for recent changes, content of your workspace files, and content of your source code repository files and compare their differences. Any found differences will be highlighted using different colors and marked using overlay icons as described in the Icon Overlays topic later in this chapter.

Configuring Multiple Source Code Control Projects

Each SCS project is a collection of settings defining SCS repository server connection and its mapping to the local the workspace. Each project has a name. SQL Assistant' saves SCS project settings in the main configuration file. You can use SQL Assistant Options dialog to customize project settings
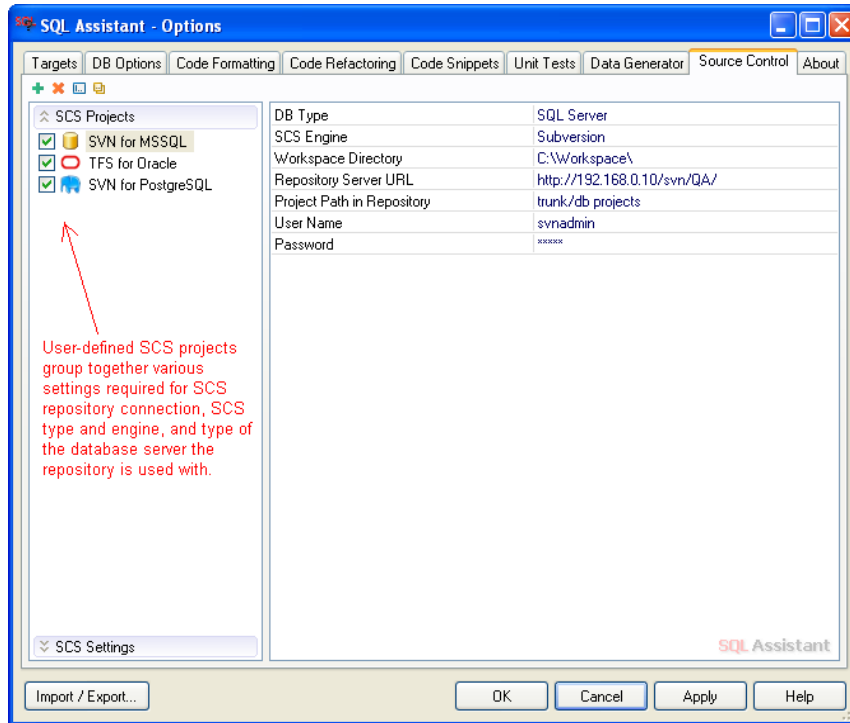
Managing SCS Projects

Use project management icons available in the left-top corner of the Options dialog to create new, rename, duplicate or delete SCS projects.

 Tips:

- To temporarily disable any project without deleting it, un-tick the checkbox displayed in front of the project name. This will make the disabled project disappear from SQL Assistant SCS selection lists, and it can be re-enabled at a later time in case you want to use it again.

- Project deletion simply removes project settings from SQL Assistant's configuration files. It does not make any changes in the database server and in the SCS repository

Repository Browser

The SCS Repository Browser is a convenient way to view all of the content of a repository and also content of the database server

To launch the Repository Browser, right click anywhere in the target editor and in the right-click menu select **SQL Assistant → Source Code Control → Repository Browser** menu. The Repository Browser window will appear. If you are opening Repository Browser first time in the current target, editor type, you will be prompted to enter repository server connection parameters. See Connecting to SVN Repository Server and Connecting to TFS Repository Server topics in this chapter for more information on supported connection parameters.

The Repository Browser provides an in-depth, browsable view of all of the content in a repository. It is able to show content of any folder, but it recognizes several special folder names as folders related to the database source control system:

> **Tables**
> **Views**
> **Procedures**
> **Functions**
> **Triggers**
> **Types**
> **Packages**
> **Sequences**

Each of the special folders is expected to group schema objects of one type only. Mixing objects of different types in one folder is not supported.

The Repository Browser uses so-called icon overlays to show you at a glance which of your database objects have been modified. See Icon Overlays topic to find out what the different overlays represent.
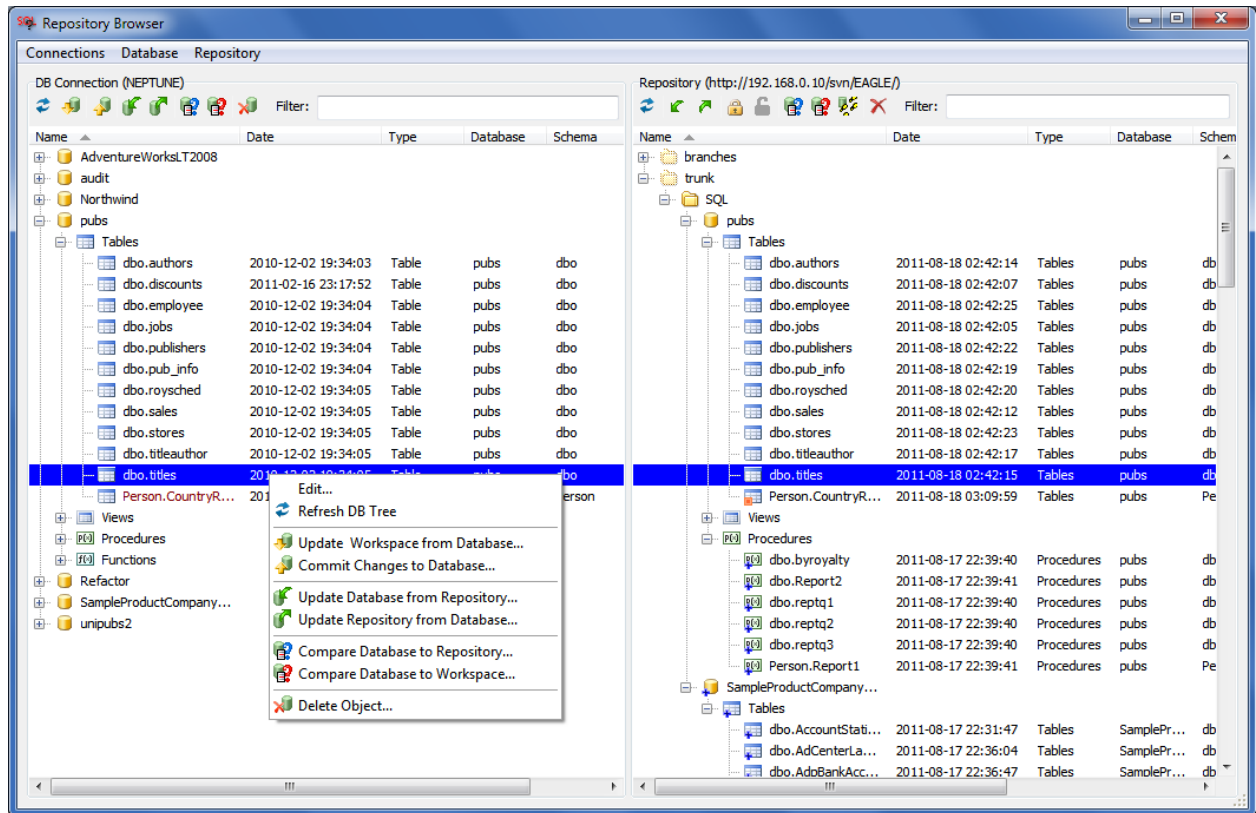
The Repository Browser contains 2 panes:

1. The **database-tree** pane lists all databases, schemas and objects visible to the current database connection

2. The **repository-tree** pane lists all folders and files visible in the repository. Items within special folders are recognized as DDL scripts for database schema objects. Different icons are used for rendering folders and items with different types. the icons match icons used for rendering similar type objects in the database tree

An action can be taken on individual items in both tress as well as multiple items. To action an item, select that item in the database or repository trees and use Repository Browser menus to invoke specific actions. To select multiple items of the same type, use standard Ctrl+click and Shift+click techniques available in Windows for multiple item selection

Repository Browser Menus

Repository Browser features both right-click and top-level menus. Both menus can be used to •Inside the tree, right clicking on items allows getting latest version, check-out, check-in (also called (Commit), lock/unlock, edit, revert, delete, and other operations. Actions taken at the folder or database level affect all items in that folder and/or database.

Tips:

- The menus are context sensitive. The menu context depends on numerous factors including selected item type, item state, SCS repository type, and so on…

- The **database-tree** actions are applied to the database and workspace items. The **repository-tree** actions are applied to the repository and workspace items.

Icon Overlays and Item Colors

When you add, delete and edit database objects you can see them in the Repository Browser with changed icons and colors. The Repository Browser adds a so called overlay icon to each modified object and folder which overlaps the original item icon. Different overlay icons correspond to different item statuses as described in the following bulleted list:

- A modified working item copy has a small orange box as overlay.

- An item that is scheduled to be deleted is shown in bright red color and has a small red X sign as overlay.

- A new item scheduled to be added has a small blue plus sign as overlay.

- An item modified in more than one place and therefore containing conflicting changes is shown in dark red color with a small warning sign overlay.

- An item locked n the repository has a small lock overlay.

- An item modified directly in the database (more recent in the database as compared to your workspace) is shown in dark red color , but without an overlay icon

Content Filtering

The Repository Browser offers super-fast content filtering. Type the substring you want to use as a filter for database objects into the filter box available above the database-tree. Type the substring you want to use as a filter for repository items into the filter box available above the repository-tree.

Target Editor Context Menus

SQL Assistant's menu branch in the editor context menu provides quick access to frequent SCS commands such as

Get Latest from Repository – this command can be used for Getting Source Code from Source Control Repository Server

Open Workspace Version – this command can be used for opening workspace version of object DDL script in the editor in a new tab or window.

Commit Changes to Repository – this command can be used for Submitting Changes to Source Control Repository Server

View Change History – this command can be used for reviewing Database Object Change History

The context commands apply to the single object under the mouse pointer
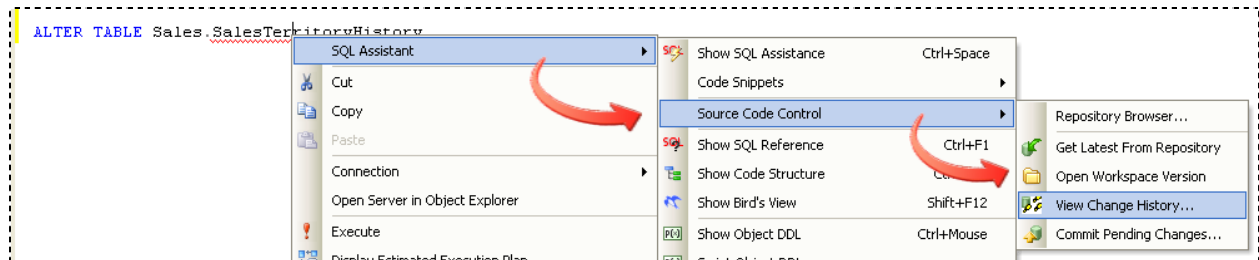
Database Explorer Integration

The Database Explorer context menu provides quick access to most SCS commands just like the rick-click menu in the target editor described in the previous topic.

In addition, you can enable display of SCS state overlay icons directly in the Database Explorer. The icons are the same as in the Repository Browser See Icon Overlays topic for details.

Choosing Project Path in Repository

Before you begin making changes in the repository or to the database, synchronizing their content, and performing some other source control operations, you must choose the root folder in the repository where you will store your database code. The simplest method for selecting such folder is expanding the repository tree, navigating to the target folder and then using **Set Database Project Root** command in the right-click context

menu for the repository-tree. Note that only objects under the selected folder are considered to be part of the SCS project. They are rendering in normal colors. All other items outside of the Database Project path are rendered using dimmed color. In the sample screenshot available in the Repository Browser topic, this command was used for the "SQL" folder. As a result SQL Assistant was able to recognize and automatically match the content of the "pubs" database against the content of the "pubs" folder in the repository which is a subfolder of "SQL" database project root.

Getting Source Code from Source Control Repository Server

Use the **Get Latest from Repository** command available in the repository-tree pane of the Repository Browser to retrieve the latest versions of source code files from the SCS repository server and save them in your local workspace. This command will overwrite old non-modified files and retrieve new files not available in the workspace. .Any files you modified n the workspace will remain as is and won't be overwritten.

When working with TFS and Perforce repositories you can also use the **Checkout** command available in the repository-tree pane of the Repository Browser. This command performs **Get Latest from Repository** and **Lock** in one step.

The scope of these commands depends on the current selection content. If a specific schema object is selected in the tree, only that object is processed. If a folder is selected, all objects in that folder are processed. If a database folder is selected, all objects in all folders of that database are processed.

 Tip: If you work with Git, and you use remote Git repository such as GitHub, and you need to get latest files from the remote repository, you should first use the **Pull Repository from Server** menu command to refresh your local repository from the remote repository, and then use the **Get Latest from Repository** to bring the latest versions to your workspace.

To make a one-time copy of repository files to a place outside of the workspace you can use the **Export** command. The copied files will not be managed by the source code control.

Getting Source Code from Database Server

Use the **Update Workspace from Database** command available in the database-tree pane of the Repository Browser to retrieve the latest versions of schema object DDL scripts from the database server and save them in your local workspace. This command will overwrite old non-modified files and retrieve DDL for new database schema objects not available in the workspace. .Any files you modified n the workspace will remain as is and won't be overwritten.

The scope of the command depends on the current selection content. If a specific schema object is selected in the tree, only that object is processed. If a folder is selected, all objects in that folder are processed. If a database folder is selected, all objects in all folders of that database are processed.

Submitting Changes to Source Control Repository Server

Source control files are checked in to the source control server using the **Commit Changes to Repository** command available in the repository-tree pane of the Repository Browser. Each commit, finds modified objects matches against item selecting in the repository tree and sends them to the repository server. Each commit generates new unique revision number attached to the change. It is highly recommended that when prompted you add comments to each change to simply source code version and change management.

The scope of the command depends on the current selection content. If a specific schema object is selected in the tree, only that object is processed. If a folder is selected, all objects in that folder are processed. If a database folder is selected, all objects in all folders of that database are processed.

Tip: If you work with Git, and you use remote Git repository such as GitHub, and you need to submit your changes to the remote repository, you should first commit changes to the local repository as described above, and then use the **Push Repository to Server** menu command to submit changes staged in your local repository to the remote repository.

Submiting Changes to Database Server

Schema object changes are checked in to the database server using the **Commit Changes to Database** command available in the database-tree pane of the Repository Browser.

The scope of the command depends on the current selection content. If a specific schema object is selected in the tree, only that object is processed. If a folder is selected, all objects in that folder are processed. If a database folder is selected, all objects in all folders of that database are processed.

Important Notes:

- For a modified schema object that already exists in the database SQL Assistant generates DROP and CREATE commands to rebuild that object in the database. If object type object is table, **all data stored in the table is lost during Commit Changes To Database; To avoid data loss, do not commit table changes to database directly!** It is recommended to commit table changes to the repository only. For the database side changes use appropriate ALTER TABLE commands and execute them directly in the target SQL editor.

- The CREATE command and other DDL commands are executed as they are entered in work-files in your local workspace. If you edit the generated DDL code and manually remove GRANT/REVOKE, comments and other by-side object properties, they will be lost during Commit Changes to Database.

- The order of schema object updates is often important. For example, a primary key table must be created before a child table with a foreign key is created. When committing multiple objects in a single batch, use small Arrow Up and Down buttons at the top of the **Commit Database Changes** dialog (see the following picture) or use item drag-and-drop to arrange items in the proper order.
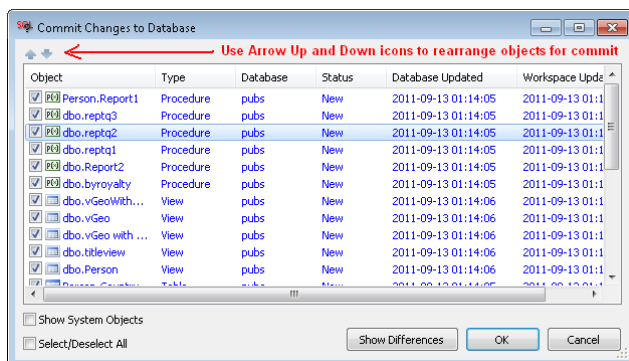
Editing Schema Object Code

1. Open Repository Browser

2. Right-click required schema object in the database-tree or its matching file item in the repository-tree. Select the **Edit** command in the right-click menu. SQL Assistant will open workspace code version of the selected object code in your target SQL editor.

Database Object Change History

1. Open Repository Browser.

2. Right-click required schema object in the repository-tree. Select the **View History** command in the right-click menu. The View History dialog will appear. You can use it to review object's change history.

3. To see specific changes in a specific revision, select that revision and click **View Changes** button at the bottom of the History dialog. This will display SQL Assistant's **Code Compare** utility displaying changes between code revisions. For more information how to use code compare functions, see CHAPTER 25, Code Compare Utility

Tips:

If you prefer to use other code compare and code merge utilities, modify **SCS Settings** options in SQL Assistant's Options. Open the Options dialog. Activate **Source Control** tab. Expand **SCS Settings** option group on the left. On the right modify **External Compare Tool** and **External Merge Tool** options as needed. You will need to specify the complete external tool command line with all required parameters. For example, enter

For example, to use the popular Beyond Compare utility from Scooter Software for the compare tool for 2-way compare operations, enter
`"C:\Program Files\Beyond Compare 3\bcomp.exe" %left %right`
In that case **%left** parameter is used for the workspace file and **%right** parameter is used for whatever the workspace file is compared to (current database code version or current repository version)

Or for 3-way merge operation (database-workspace-repository) enter
`"C:\Program Files\Beyond Compare 3\bcomp.exe" %left %right /savetarget=%out`
In that case the target **%out** parameter is used for the workspace file, **%left** is for the current database version, and %right is for the current repository version.

Additional Workspace and Repository Management Commands

Lock and Unlock

Locking is a mechanism used by SCS repository for mutual exclusion between users to avoid clashing code commits. Use the Lock command in the repository-tree to lock an item and prevent other users of changing it until you unlock that object.

Note, locking is not supported by Git systems.

Cleanup

This command is specific to SVN repositories. It is used to clean up the workspace, removing locks and resuming unfinished operations. If you ever get a "working copy locked" error, run this command to remove stale locks and get your workspace into a usable state again.

Undo

If for whatever reason you want to undo your local changes in the workspace and restore the latest version available in the repository, use the **Undo** command from the Repository side content menu.

Similarly, if for whatever reason you want to undo your local changes in the workspace and restore the latest version available in the database, use the **Undo** command from the Database side content menu.

Create Folder

Using this command you can create new folders in the repository. For example you can use it to create folders for data-seed scripts, for user authorization scripts, for unit test scripts, and so on

Automating Source Control Repository Updates

As an alternative to submitting all changes to the repository by each database developer and DBA you can use SQL Assistant's automatic repository updates features. Basically you can schedule a nightly, daily, or hourly job to automatically push database side schema changes to the source control repository.

To automate repository updates, schedule running of **sacmd** command with parameters using Windows Task Scheduler or any other scheduler utility. The required parameters are:

Parameter	Description
scs:"project-name"	(required parameter) This parameter instructs sacmd to execute source control operation using source control system connection and settings stored in the specified SCS project. Note that the SCS project can be entered and updated on the Source control tab in SQL Assistant's Options. For more details, see Managing SCS Projects topic in this chapter.
sas:"sas-file"	(optional parameter) The SQL Assistant's configuration file storing the required SCS and database connection settings. If this parameter is not specified, the default configuration file is used. For more details, see Overview topic in CHAPTER 51, Backing Up and Sharing SQL Assistant Settings.
conn:"connection-name"	(required parameter)

	Name of the preconfigured database connection saved in the chosen configuration file.. For more details, see <u>Managing Database Connections</u> topic in CHAPTER 48, Customizing SQL Assistant's Behavior.
[operation type]	(optional parameter) The following operation types can be specified: dw:"item-list" - update workspace from database dr:"item-list" - update repository from database wr:"item-list" - commit workspace files to repository wd:"item-list" - commit workspace files to database rd:"item-list" - update database from repository rw:"item-list" - update workspace from repository If operation type is not specified, the **dr** operation is used by default for all databases on the server. The "item-list" contains semicolon separated list of database items. Allowed database items are database, database/type, and database/type/object, for example: dr."Northwind;Adventure Works" – this will compare databases Northwind and Adventure Works to the source control repository and commit found changes to the repository. dr."Northwind/Tables;Northwind/Views" – this will compare tables and views in database Northwind to the source control repository and commit found changes to the repository. dr."Northwind/Tables/dbo.Employee;Northwind/Tables/dbo.Department;" – this will compare dbo.Employee and dbo.Department tables in database Northwind to the source control repository and commit found changes to the repository.

Examples:

```
"C:\Program Files (x86)\SQL Assistant 12\sacmd.exe" scs:"My SVN project" conn:"ORA_SERVER /
system (OCI)"
```

```
"C:\Program Files (x86)\SQL Assistant 12\sacmd.exe" scs:"My SVN project" conn:"QA SQL
Server" rw:"AdventureWorks"
```

 Tips:

- Each command must appear on a single line. The above examples show wrapped text lines as a result of limited page width.

- To execute several different operations or same operation with different settings, write all required sacmd commands to a batch file and then schedule the resulting batch file.

Automating Database Updates from Source Control Repository

The required operations and parameters are the same as in the previous topic, except that you would need to specify **rd** operation type explicitly. For example;

```
"C:\Program Files (x86)\SQL Assistant 12\sacmd.exe" scs:"My SVN project" con:"QA SQL Server"
rd:"AdventureWorks"
```

Scheduling Automated Source Control Operations

As an alternative to using SQL Assistant's interactive Source Control features you can schedule periodic unattended database to source control updates as well as source control to database updates. To streamline the scheduling SQL Assistant provides built in graphical dialogs for scheduling such updates.

To schedule automated source control operations:

1. Open SQL Assistant's Source Control Repository Browser.

2. Navigate to the database or repository folder whose updates you want to automate and right-click the required item. If you want to use the same task for multiple items, select multiple items in the database or repository tree and right-click one of them. . In the right-click content menu choose the **Schedule**… command. The **Schedule Repository Updates** dialog will appear.
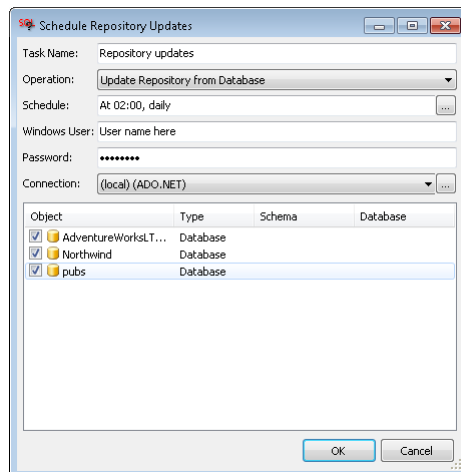
3. Fill in task name, user name and password.

4. Choose required task schedule. To enter new task schedule, click the [...] button to the right of the Schedule field. The **SQL Assistant – Task Schedule** dialog will appear on the screen.
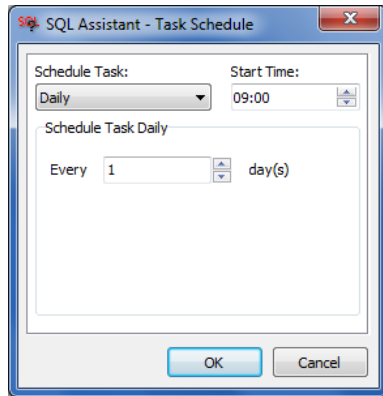
To schedule one time run of the SQL script, in the Schedule Task drop-down select **Once**. Enter a date and time to start the task.

If you select the **Daily** option, you can enter the recurrence interval for the task and the date and time to start the task. An interval of 1 produces a daily schedule and an interval of 2 produces an every other day schedule. The task will start at the specified time each day.

If you select the **Weekly** option, you can enter the recurrence interval for the task, the date and time to start the task, and the days of the week in which to start the task. An interval of 1 produces a weekly schedule and an interval of 2 produces an every other week schedule. The task will start at the specified time on each of the specified days.

If you select the **Monthly** option, you can enter the months in which you want to start the task and the weeks and days of the month in which you want to start the task. You can also specify that you want to start a task on the last day of each selected month.

5.   Choose required database connection. The list of known and already configured connections appears in the **Connection** drop-down. To specify new connection, click the button to the right of the Connection field. The **SQL Assistant – Options** dialog will appear on the screen, which you can use to add new connection. See CHAPTER 15, topic Managing Connection Groups and Connection Settings for information of how to add, modify, and delete connections.

6.   Click the OK button to complete task schedule setup and create the required windows task.

The create Windows task definition contains references to SQL Assistant code execution utility, required parameters describing task and database connection properties.

Tips:

- You can create multiple scheduled tasks for repository updates. For example, you can create a separate task for each database or schema under source control. Or you can select multiple databases and/or schemas at once, and schedule their updates or repository updates using a single task.

- Scheduled tasks can be managed using standard Windows Task Scheduler user interface. For example, you can use Windows Task Scheduler user interface to modify existing or add an additional schedule to a task after it has been setup by SQL Assistant. Refer to your Windows documentation for more details.

Important Notes:

- The SQL Assistant's source control facility is kicked off by the Windows Task Scheduler. It is important that the Task Scheduler service is running at the time of the scheduled update. If the service is stopped, no tasks will run. Similarly the computer running the Task Scheduler must be powered at the time of the scheduled task run. It is a good idea to use an always on server based computer for scheduling and running repository updates.

- The database connection selected for the scheduled script can point to any database server that can be connected over the network.

- If the task needs to run on a remote database server, make sure to select a domain user account to run the scheduled Windows task. The account must have sufficient privileges to connect to the server.

LocalSystem and other system accounts confined to the local system cannot be used for running tasks requiring remote database server connections.

CHAPTER 24, Reporting, Data Pivot and Analytics

Overview

SQL Assistant supports two different methods for creating reports:

- You can create named reports and save them to report definition files with extension .sarpt. You need to use the **Report Designer** to create and modify report definition files. You can run these report files as you would run SQL and other files. You can also add their references to SQL Assistant menus so that reports can be preview with a mouse click without a need to browse and open files. See the following Standalone Reports section for specific details.

- You can create nameless dynamic reports generated from the data retrieved into the data grids. Such reports are very easy to use and create, but their definitions are not saved and they are not reusable. See the following Data Grids Integrated Reporting topic for specific details.

SQL Assistant internally uses FastReport report engine that powers the reports and is also used for developing reports graphically.

Standalone Reports

You can create and modify named standalone reports using the **Report Designer** utility accessible from the top level **Execute** menu in SQL Assistant integrated development environment. Normally you would use the Report Wizard to create a new report and then continue working on the report design in the Report Designer. Here are the basic steps required to create a new report:

1. Select **Execute / Report Designer** … command in the menu. This will open the **Report Wizard** dialog. By the default **New Report – generate report from existing data source** is going to be pre-selected for you. This option enables quick report creating using user-provided SLQ query and one of the selected templates.

2. Click the **Next** button to advance to the next step. Enter **Report Name** and enter main report **SQL Query**. If multiple queries are required for the report, they can be added later.
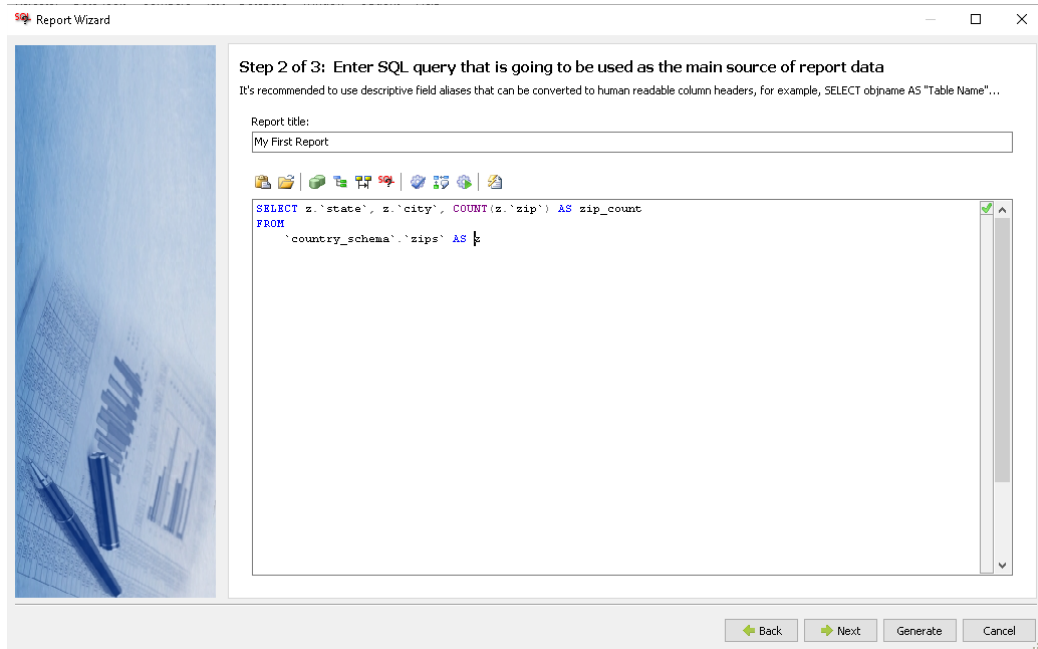
3. Please be sure to verify the query is syntactically valid and returns some data before going to the next step. You can use Syntax Checker, Execute SQL… and other tools accessible using the toolbar above the SQL Query field. Click the **Next** button again when you are done with the query.

4. Select report **Template** and if required adjust column titles and column width. You can do that directly in the wizard. To edit a column title or width, double-click the value shown in the fields list, and then modify it as required. Alternatively you can select the field and click the Edit button on the toolbar to the right of the column names.
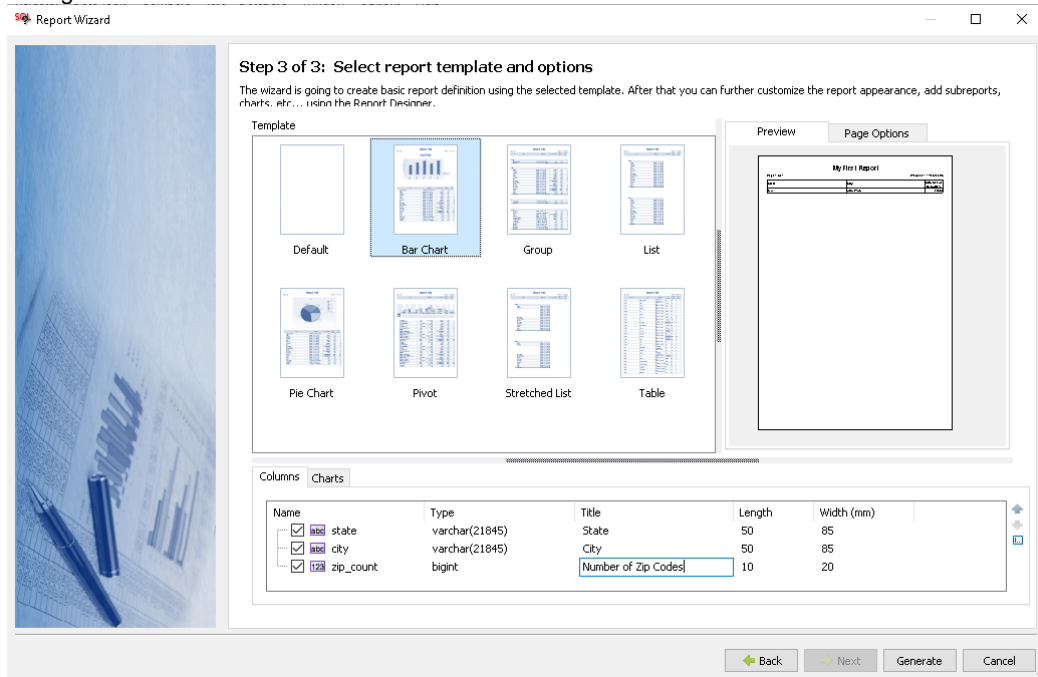
You can also use Arrow Up and Down toolbar buttons to rearrange report columns. The columns do not have to be in the same order as in the query, their order may be different. Click the **Generate** button again when you are done with the basic properties.

5. The report definition will be now generated and opened for modification and finalization in the Report Designer.. You can use to update chart properties, add or remove chart and other components.
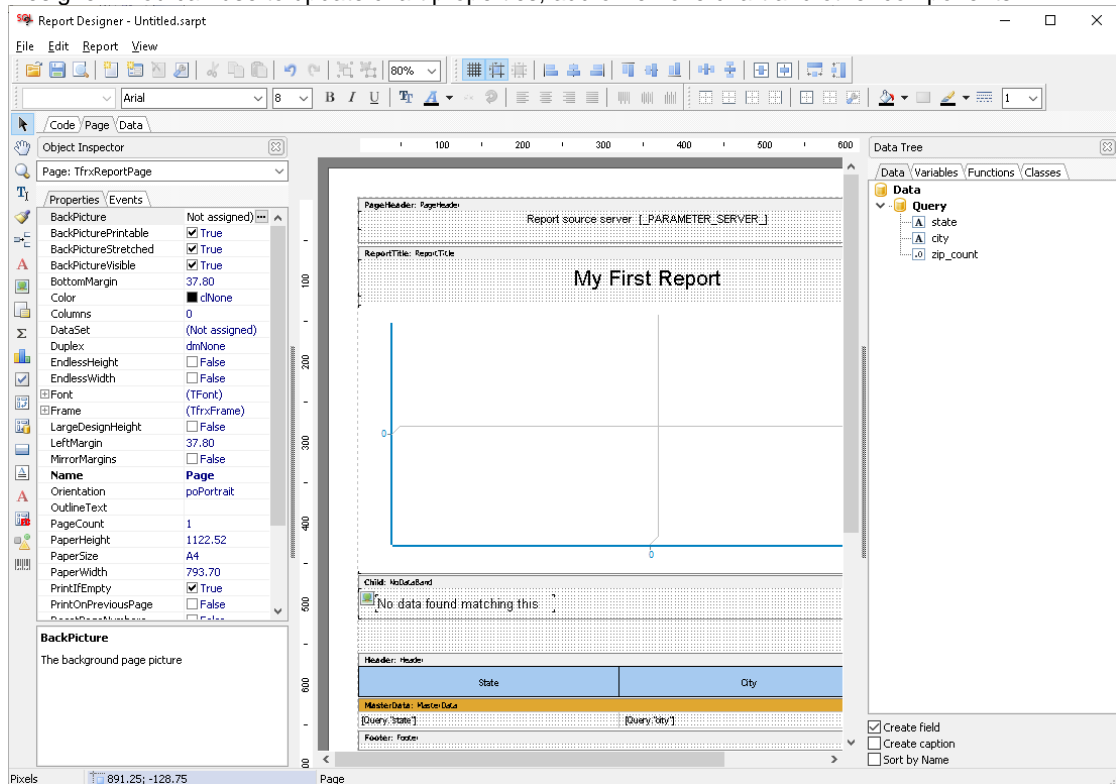
Tip: The report templates available in the Report Wizard provide some common report features like header and footer bands, page enumeration, report title, and so called "no data" band. You can freely move or modify them in the generated report as you see the fit. They are predefined for you to help with easy report creation, but they are not required, and they can be removed if you do not need them.

Predefined Reports

A number of predefined database administration reports are installed with SQL Assistant. The reports are database type specific. They are accessible using right-click menu in the Database Explorer and available for most of the supported database types. If you are working with SQL Assistant Integrated Development Environment, in which the Database Explorer pane is always visible, right-click anywhere within the Database Explorer pane and select **Reports** submenu.

If you are working with some other target type, first you need to open the Database Explorer pane. You can do that using the menus or pressing the hot key assigned in SQL Assistant settings to View Database Explorer function in your target database development tool. You can then follow the instructions provided in the paragraph above.
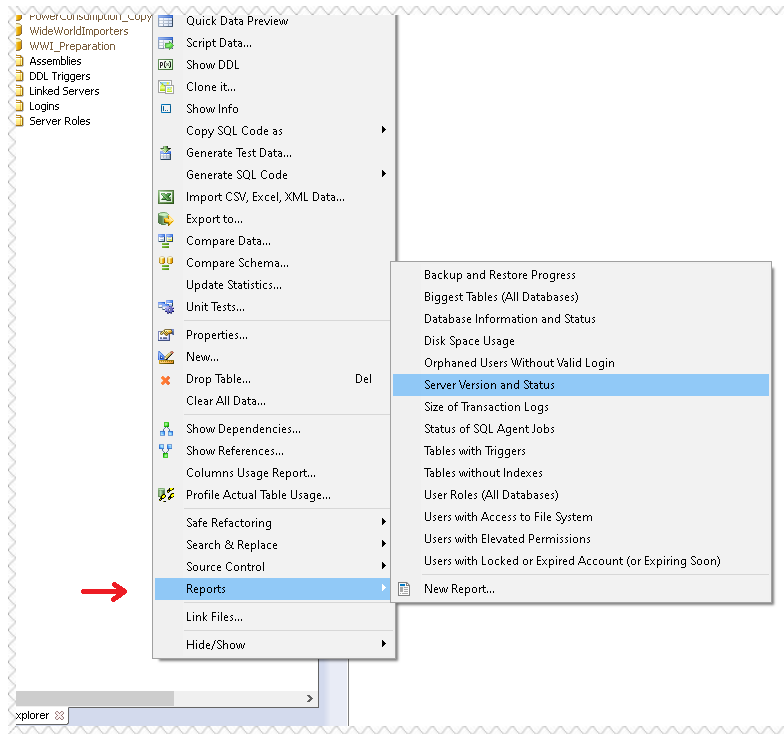
Working with Report Designer

Overview

SQL Assistant provides visual Report Designer tool for creating and modify reports graphically. The **Report Designer** can be opened using **Execute / Report Designer…** menu. Follow the instructions in the Report Wizard dialog to create a new report or open a previously created report.

The Report Designer window is divided into several areas:

- The report design **Workspace** in the center of the window.

- The "**Object Inspector**" pane on the left where you can change properties of the object selected in the Workspace

- The "**Report Object Tree**" pane showing object parent-child relationship within the report, such as which components are located in which bands. You can use drag and drop within the tree to modify their relationship.

- The "**Data tree**" pane on the right with the data sources and their fields. You can drag fields from that tree to the Workspace to quickly add them to the report.

Keyboard and Mouse Controls

Here are important keyboard shortcuts that are often used in the Report Designer

Keys	Description
Ctrl+O	Same as "File > Open…" menu command
Ctrl+S	Same as "File > Save" menu command

Ctrl+P	Same as "File > Preview" menu command
Ctrl+Z	Same as "Edit > Undo" menu command
Ctrl+C	Same as "Edit > Copy" menu command
Ctrl+V	Same as "Edit > Paste" menu command
Ctrl+X	Same as "Edit > Cut" menu command
Ctrl+A	Same as "Edit > Select all" menu command
Arrow, Tab	Moves object selection between objects in the Workspace
Del	Deletes selected object(s) in the Workspace
Enter	Opens object / expression editor for selected object
Shift+arrows	Modifies sizes of selected object(s) Shift + Arrow Right increases their width Shift + Arrow Left decreases their width Shift + Arrow Up increases their height Shift + Arrow Down decreases their height
Ctrl+arrows	Moves selected object(s) in the direction of the arrow.
Alt+arrows	Moves and attaches selected object to adjacent object in the specified direction. Make sure you have single object selected for this operation.

Mouse Operation	Description
Left button click	Multi-functional usage, depending on the context: Selects an object Pastes new object Moves or resizes selected objects Zooms in and out by dragging red square in the bottom left corner of the selected objects' group
Right button click	Opens context menu for the selected objects. The menu commands may different for different types of objects and bands
Double-click	Opens editor for selected object
Mouse wheel	Scrolls the report workspace or pane depending on the input focus
Shift + left button click	Toggles object selection
Ctrl + left button click	Selects multiple objects using mouse-lasso method.
Alt + left button click	For Text objects activates in place text editor

Report Variables

System Variables

The report engine supports a number of system variables such as date, time, page number, and so on… that can be used in report headers and footers, and in other places. They can be accessed through the **Report/Variables** menu in the Report Designer. The system variables have self descriptive names.

Database Context Variables

The following variables can be used in expressions where database context information is required:

_PARAMETER_DB_NAME_ - Current session database name

_PARAMETER_SCHEMA_NAME_ - Current session database schema name

_PARAMETER_SCHEMA_ID_ - Current session database schema ID if supported by the database type.

_PARAMETER_OBJECT_NAME_ - Context database object name if report is invoked for a specific database object

_PARAMETER_OBJECT_ID_ - Context database object ID if supported by the database type and if report is invoked for a specific database object

_PARAMETER_OBJECT_TYPE_ - Context database object type if report is invoked for a specific database object

_PARAMETER_DBMS_TYPE_ - Current connection database system type

_PARAMETER_DBMS_VERSION_ - Current connection database system version

_PARAMETER_SERVER_ - Current connection server name, the server name known to SQL Assistant derived from the session context details or the connection parameters.

_PARAMETER_LOGIN_ - Current connection login

_PARAMETER_DB_ - Current connection database name as defined in the connection settings, this maybe different from the current session database name, which can be changed in the SQL editor before running reports.

_PARAMETER_USER_ - Current connection user name as defined in the connection settings, this maybe different from the current session user name, which can be changed in the SQL editor before running reports.

_PARAMETER_OSUSER_ - Local operating system user name who is running the report.

_PARAMETER_MACHINE_ - Local machine name where the report is running.

_PARAMETER_SA_TARGET_ - SQL Assistant's target environment name

_PARAMETER_SA_VERSION_ - SQL Assistant's version

Report Objects

The Report Designer represents reports as a collection of schematic pages. Objects are placed anywhere on the report pages within report bands and are used to define the report's appearance and to display various information, such as text and graphics. Here are the types of objects available:

- "Band" object : an area on a design page which behaves according to its type (e.g. Header band, Data band)

- "Text" object : displays one or more lines of text within a rectangular area

- "Picture" object : displays a graphic file in "BMP," "JPEG," "ICO," "WMF" or "EMF" format

- "Line" object : displays a horizontal or vertical line

Σ - "System text" object : displays either system information (date, time, page number, etc) or aggregate values

- "Subreport" object : allows insertion of another report design page within the host page

- "Draw" category objects : displays various geometrical shapes

(diagonal line, rectangle, rounded rectangle, ellipse, triangle and diamond)

- "Chart" object : displays data in various chart formats (pie chart, histogram, etc.)

- "RichText" object : displays text in Rich Text Format (RTF)

- "CheckBox" object : displays a checkbox with either a tick or a cross

- "Barcode" object : displays data as one of several barcode types

The basic objects most commonly used are the "Band" and "Text" objects.

Report Bands

Report bands are used for placing the objects they contain at particular locations on the output page.
 When placing an object in the "PageHeader" band we tell the report engine that the given object must be displayed at the top of each page in the finished report. Similarly, objects in the "PageFooter" band are displayed at the bottom of each page. Let's demonstrate this with an example. We'll create a report containing "Hello!" at the top of the page, the current date to the right of it and the page number at the foot of the page on the right hand side.

If you choose to use one of the prepackaged report templates when creating a new report, a number of bands will be inserted automatically including page header and footer, report title, no data band, and some other bands depending on the selected template type.

Among the bands that are typically used in all reports are various bands which are named "...Data". Such bands typically host report field that will be printed on the output page, once for each record in the report data source associated with the report page.

Tip: There is a special band type called the "Child" band. A "Child" band is linked to (and displayed after) its parent band. It enables fine layout of objects within the report page.

Text Objects

The "Text" object has many features. It can display text in a frame and be filled with a color. The text can be displayed using any font of any size and style. All the properties can be set visually with the help of the toolbars.

Here are some examples of text design:
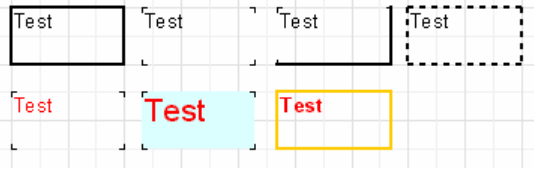
Now let's look at other features of this basic object. As an example, let's create a new "Text" object to display two lines of text:

This is a very, very, very long text line.
And this is another line, the shorter one.

Enable the object frame from the toolbar and re-size the object up to 9x3 cm using the mouse. We see that the object can display not only a single line but also several lines of text. Now reduce the object width to 5 cm. It is obvious that long lines did not fit across the object and were therefore wrapped. This is controlled by the "WordWrap" object property. If wrapping is disabled (either in the object inspector or via the object context menu) any long lines will be simply cut short.

Now let's see how text alignment inside the object works. Alignment buttons are located in the "Text" toolbar and control horizontal or vertical text alignment. Note the "Justify" button which lets you align paragraphs to both object edges - to do this "WordWrap" must be enabled.
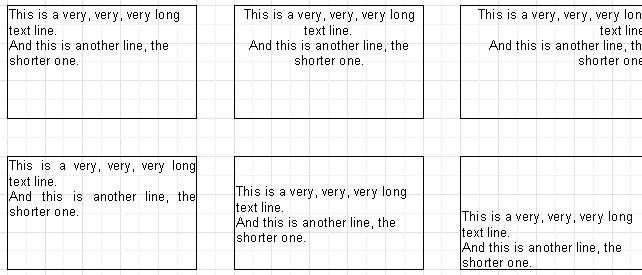
All the text in the object can be rotated to any angle in the range 0..360°. The button in the "Text" toolbar allows you to quickly rotate the text to pre-sets of 0, 45, 90, 180 or 270°. For any other value set the required angle in the "Rotation" property of the object inspector. When rotating text to angles other than 90, 180 or 270° the text may be cut off by the frame of the object, as shown below. To cure this increase the object height a little so that all the text fits within the object.
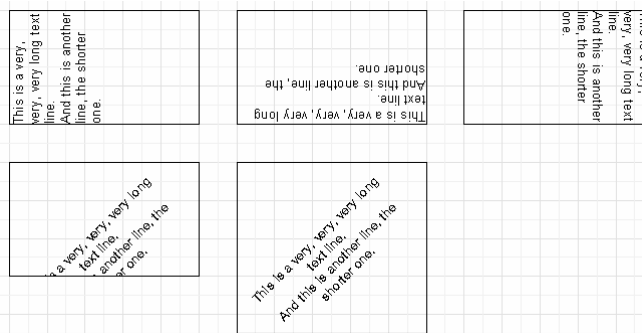
Use of HTML tags

The "Text" object does understand some simple HTML tags. Tags can be located within the text of the object. Tags are disabled by default, but to enable them either select "Allow HTML tags" in the object

context menu or enable the "AllowHTMLTags" property in the object inspector. Here is the list of supported tags:

 : bold text

<i> : italic text

<u> : underlined text

<sub> : subscript

<sup> : superscript

 : font color

<nowrap> : text which is not split when "WordWrap" is enabled, the whole text is shifted to the next line

Note: Please note that only a few tags are supported, but this should be enough for the majority of applications. It is not possible to modify the font size or name by means of HTML tags.

The following examples show how these tags can be used.

text bold text <i>text in italic</i> <i>bold and in italic</i>
E = mc²
A₁ = B²
this is a usual text, and this is a red one
this is a usual text, and this is an orange one
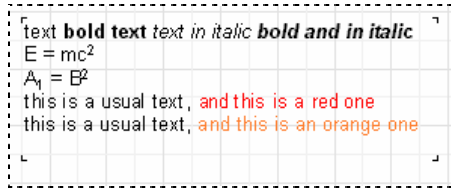
Expressions

One of the most important features of this basic object is its ability to display not only a static text but expressions as well. Expressions can be located within the object mixed in with normal text. Let's see a simple example of how this works. Type the following into the object:

Hello, World! Today is [DATE].

When the report is run we can get something like this:

Hello, World! Today is 01/01/2021.

How does this happen? When the report engine creates the report and encounters an expression enclosed in square brackets the report engine calculates the expression's value and inserts this value into the text in place of the expression. "Text" objects can contain any number of expressions mixed in with the normal text. Complex expressions can contain brackets (for example [1+2*(3+4)]). Constants, variables, functions and fields can all be used in expressions. We will learn more about these later in the chapter.

The report engine automatically recognizes expressions enclosed in square brackets in the text. But what happens if our normal text contains square brackets which we do not want to be considered as expressions? For example, if we need to display the following:

a[1] := 10

The report engine would consider [1] as an expression and display that text as:

a1 := 10

which is not what we want, of course. One way to avoid this happening is to disable expression recognition. Disable the "AllowExpressions" property (or "Allow Expressions" in the context menu) and all expressions in the text will be ignored. In our example, the report engine would then display exactly what we need:

a[1] := 10

But sometimes text is required to contain both an expression and normal text with square brackets, for example:

a[1] := [myVar]

Disabling "AllowExpressions" lets us display square brackets in the required places, but it also disables handling of expression. In this situation the report engine allows you to use an alternative set of symbols to designate expressions. The "ExpressionDelimiters" property, "[,]" by default, is responsible for this. In our example we can use angular brackets for the expressions instead of square ones:

a[1] := <myVar>

The "<,>" value must be set in the "ExpressionDelimiters" property, where the comma is required to separate the opening and closing symbols. Another requirement is that the opening and closing symbols cannot be identical, so "%,%" will not work. Complex symbols can be used, for example "<%,%>". So our example could look like this:

a[1] := <%myVar%>

Groups

The data in reports can be grouped using the Group bands feature which supports multi-level reports as demonstrated in the following example

The "Group header" band is output only when the field to which it is linked changes in value. Otherwise the data band connected to the "Group" dataset is displayed.

Reports having nested groups can be designed in a similar way. The depth of nesting of groups is unlimited. Reports which use groups have some advantages over reports of the master-detail type:

- The whole report needs only one dataset (query)
- The number of data grouping levels is unlimited
- Data sorting becomes possible
- More optimal usage of the database resources (the query returns only one dataset for output, with filtering done by the query).

There is another important group feature. The group header has a property called DrillDown. If you set it to True, the group becomes interactive. This means you can click on the group header in the Report Preview

window and the group will expand (display all records in the group) or collapse (display only the header and, if ShowFooterIfDrillDown is True, then the footer).

You can control whether all groups are collapsed or expanded when the report first runs. By default a group is collapsed but you can set ExpandDrillDown to True if you want it expanded. You can also use the preview's context menu to expand or collapse all groups at once.

Line numbering within groups

To do this, we add a "Text" object containing a system variable [Line] to both of our bands (this is most easily done by dragging & dropping from the "Variables" tab of the "Data Tree" pane).

When previewing the report, we can see that both the data levels now have their own line numbers:

To continuously number the second level data lines, use the [Line#] variable instead of [Line] in the "Text" object on the data band. The result will then look like:

Aggregate Functions

In most cases group reports need to display some summary information (such as: "total of a group", "number of group elements" etc). The aggregate functions can be used for calculating aggregate values over some data span. The aggregate functions are:

SUM　　　　　returns the total of an expression

MIN　　　　　returns the minimal value of an expression

MAX　　　　　returns the maximal value of an expression

AVG   returns the average value of an expression

COUNT   returns the number of lines (rows) in the data span

The syntax of all aggregate functions (except COUNT) is similar to that of the SUM function:

SUM(expression, band, flags)
SUM(expression, band)
SUM(expression)

The parameters are:

expression – the expression to be calculated

band – the name of the data band within which the calculation is performed

flags – a bit field, with values

  1 - include invisible bands in calculation

  2 - accumulate the aggregate as a running total, in other words, do not reset the aggregate when the current data span resets.

  3 - (both of the two previous options)

An expression is the only mandatory parameter, the other two are optional. Nevertheless, to avoid making mistakes it is recommended that band parameters are always given.

The "COUNT" aggregate function has the following syntax:

COUNT(band, flags)
COUNT(band)

where the parameters have the same meaning as above.

There is a general rule for all aggregate functions: an aggregate can only be calculated over a data band and can only be used in that band's footer, which can be one of:

  footer
  page footer
  group footer
  column footer
  report footer (summary band).

How do aggregate functions work?

We will look at this using our group report example. Let's add some new elements to the report:

The Group."ItemsTotal" field in the data band displays the current order total. Place a "Text" object in the group footer containing the aggregate SUM shown above. It will display the total of all orders placed by the given customer. Using a calculator, we can check that the result is correct:

Aggregate functions work like this : before outputting a report, the report engine scans the "Text" object contents to find any aggregate functions. The aggregates found are linked to the data bands in their parameters (in our example "SUM" is linked to the "MasterData1" band). During report output (when the data band is displayed) the values of the aggregates linked to it are calculated. In our case the Group."ItemsTotal" field values are accumulated. Once the group footer displaying the aggregate has been output the aggregate value is reset to zero, and the cycle is repeated for the next group, and so on.

What is the purpose of the optional "Flags" parameter in aggregate functions? Reports may have some, or all, of the data bands hidden. We may, however, need to calculate aggregates over all the data bands, whether visible or not. In our example, set the "Visible" property of the data band to false, so preventing its display. To still have this hidden data band included in the calculations, we have to set the third, optional parameter in the function call to the figure 1, i.e.:

[SUM(<Group."ItemsTotal">,MasterData1,1)]

This produces a report looking like tis:

When the "Flags" parameter value is set to 2, the aggregate value is not reset immediately after it is displayed : the aggregate becomes a "running" calculation for each successive output. Let's modify the function call as shown here:

[SUM(<Group."ItemsTotal">,MasterData1,3)]

The value "3" is a bit combination of "1" and "2", meaning that we need to include the invisible bands, without resetting the total after each group. As a result, we have:

Page and Report Totals

The usage of aggregate functions in page and report totals is exactly the same as in the group totals. The only difference is that you will place them into page and report header and footer bands as demonstrated in the following example
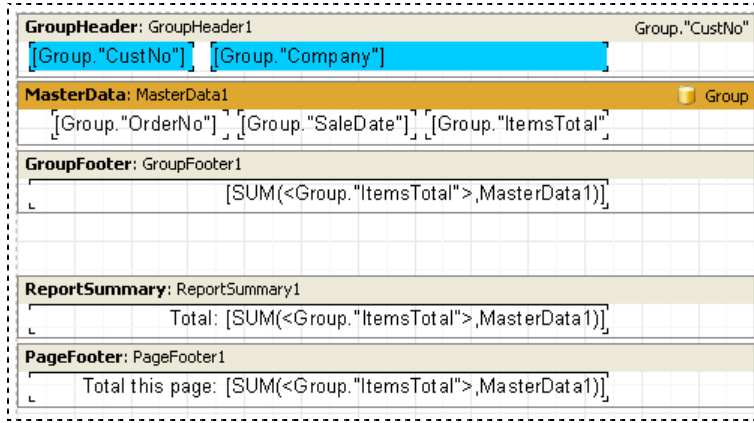
Conditional Highlighting

Please refer to the Conditional and Continuous Data Highlighting topic later in this chapter for specific details on how to use conditional row and value highlighting in reports and data pivot.

Multi-page Reports

The report engine supports reports consisting of several design pages. Multi-page designs allow for the adjustment of properties such as size and orientation of each page, as well as allowing variation in the placement of objects and bands on the pages. When this type of report is output all bands from the first design page are be displayed, then bands from the second page, and so on.

New reports contain one design page by default. You can add a new page by clicking on the 📄 button in the toolbar or by selecting the "File>New page" menu command. Then you would see that a new page tab appears in the designer:
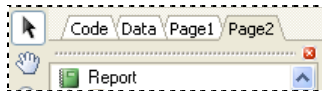
You can easily switch between pages by clicking on the page tabs. Page tabs can be dragged to easily change their printing order. An unnecessary page can be deleted using the 📄 button in the toolbar or by selecting the "Edit>Delete page" menu command. You can also call the context menu by right-clicking on the page tab:
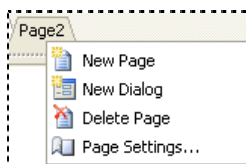
The number of design pages in a report is unlimited. As a rule additional pages are used either for title

pages or, in more complicated reports, for data coming from many data sources.

Here is a simple example of creating a title page. Let's use our previous report having one data level. Add a new page to it, it will be added as a second page. Move it to the front of the report by grabbing the page2 tab with the mouse and dragging it over the first page tab, page1. This changes the page order. Select the new page and place a "Text" object containing "Our report" in the middle of the page. That is all that is needed. The report with a title page is complete:

Tip: If the "PrintOnPreviousPage" property is enabled in the object inspector for the second output page, then the second output page's objects will start printing on the white space of the first output page, and not on the new output page.

Nested Reports (Subreports)

Sometimes very complex report are needed which contain blocks of additional data inserted at particular points in the design. Although many of these reports can be designed using an arrangement of bands, sometimes it just becomes too complicated. In these circumstances it is necessary to use the "Subreport" object .

When a "Subreport" object is inserted into a page, the engine automatically adds a new page, which is connected to the "Subreport". Such a nested report resembles a multi-page report in terms of design structure. The only difference is that the nested report is displayed in a specific location on the basic design page, and not after it. When this report is output, as soon as the "Subreport" object is encountered the report engine outputs the connected subreport page in its entirety. After that output continues with the rest of the basic design page.

Further "Subreport" objects can be inserted into a subreport design page, so increasing the depth of nesting.

The "Subreport" object has a "PrintOnParent" property which can sometimes be useful. This property is False by default. Usually a subreport is output as a set of bands on the basic report page. When this is so, the height of the parent band containing the "Subreport" object is not controlled by the bands in the subreport, i.e. it cannot be stretched. If the subreport "PrintOnParent" property is set to True, either in the object inspector or the context menu, the objects in the subreport are physically printed on the band which contains the "Subreport" object. This band can be made to stretch and can have stretched objects placed on it:

Tips:

- Subreports enable deep data nesting. In comparison the number of data levels is limited to six when the Detail data bands are used instead of the "Subreport" objects.

- Two or more "Subreport" objects can be placed side-by-side on the same data band to create side-by-side subreports:

- Since subreports are placed on the basic design page, they cannot contain any of the following bands: "ReportTitle/ReportFooter", "PageTitle/PageFooter/PageBackground" or "ColumnTitle/ColumnFooter". If any of these bands are placed on a nested report page they will not be recognized. For the same reason there is no point in changing any nested report page options, as the options of the basic report page override those of any nested pages.

Report Scripts

A script is a program, which is part of a report. As the report runs, the script runs as well. A script can handle data in ways that are not possible just using the normal operations of the reporting engine; for example, a script can hide redundant data depending on a predefined condition, or show conditionally a no-data band.

A script can be written in one of the languages supported by the script engine:

- PascalScript
- C++Script
- BasicScript
- JScript

The following features are supported by the engine:

- Standard language set : variables, constants, procedures, functions (which may be nested and having variables, constants, default parameters)
- All standard operators (including case, try, finally, except, with), types (integral, fractional, logical, character, line, multi-dimensional arrays, variant), classes (with methods, events, properties, indexes and default properties)
- Type compatibility checking
- Access to any of the report's objects

The Report Designer contains a script editor with syntax highlighting. There is also an embedded debugger which has the following functions: "Step", "Breakpoint", "Run to cursor" and "Evaluate".

Important Notes:

The complete description of the supported scripting functions and interfaces can be found here
https://www.fast-report.com/public_download/html/FR5UserManual-HTML-en/index.html

Running Reports, Printing, Saving to PDF files

To run a predefined report, locate the report in the Database Explorer's right-click menu. The report will run immediately and its preview will appear in the **Report Preview** window.

To run a user-defined report

1. Make sure your current editor is connected to the database in which you want to run your report.

2. Select **Execute / Execute Report…** menu. You will be prompted to select previously saved report definition, a file with **.sarpt** extension. Select your report file and click the **Open** button in the file browse dialog. The selected report will run and its preview will appear in the **Report Preview** window.

Important Notes: The database connection details don't get saved in the report definition. The same report can be run in multiple databases. Yet, it's very important to select the database connection having the same type the report was created with. Failure to select correct connection type may lead to report execution errors.

To send report to a printer for printing, click the Printer icon on the report preview toolbar

To save report to a PDF file, which can be opened later or emailed, click the PDF icon

To save report to a Prepared Report file which can be opened later without re-running the report, click the Save icon . You will be prompted to select file name to save report to. The report will be saved along with the data. To open previously saved report, click the Open Report icon . You will be prompted to select previously saved prepared report file.

Adding Reports to Database Explorer

To have your custom reports appear in the Database Explorer right –click menu, copy or save them to `%APPDATA%\SQL Assistant\Reports\`**[SQL Assistance name]** folder. Note that %APPDATA% is a system environment variable. If you do not know the value of this variable, you can display it by opening the DOS command window, and executing the command echo %APPDATA%. Note that [SQL Assistance name] is a reference to the SQL Assistant type in SQL Assistant settings for the database you are working with. See the Customizing SQL Assistance Types topic for more information. For example if your user name is jack_the_user and the report you want to register is for Oracle and your report definition file name is MyReport.sarpt, the location of the report would be *C:\users\jack_the_user\AppData\Roaming\SQL Assistant\Reports\Oracle\MyReport.sarpt.* For schema object type specific reports place reports into `%APPDATA%\SQL Assistant\Reports\`**[SQL Assistance name]\[Object Type]** subfolder. For example, if you develop custom report named MySpaceUsage to check on table storage allocation, save it to *C:\users\jack_the_user\AppData\Roaming\SQL Assistant\Reports\Oracle\Table\MySpaceUsage.sarpt.*

Tip: If the report is saved to the correct location, it is going to be available in the menu immediately, there is no need to reconnect or restart anything. Simply right-click in the Database Explorer, click Reports submenu and check your report name is showing. Click it to preview and test.

Data Grids Integrated Reporting

The integrated reporting and analytics interfaces are coupled with the data preview and query execution facilities. It is recommended that you read CHAPTER 11, Data Display and Editing and CHAPTER 14, Executing SQL Scripts before you continue. In CHAPTER 11 and 13 you can learn how to execute SQL scripts and different methods for retrieving and viewing the data using built-in data-grid controls.

Using system generated reports:

1. Develop the SQL query to retrieve data required for the report and execute it in SQL Assistant's SQL Editor or in the other target editor. Be sure you use the SQL Assistant's code execution facility (see CHAPTER 14 for details) to run the SQL query.
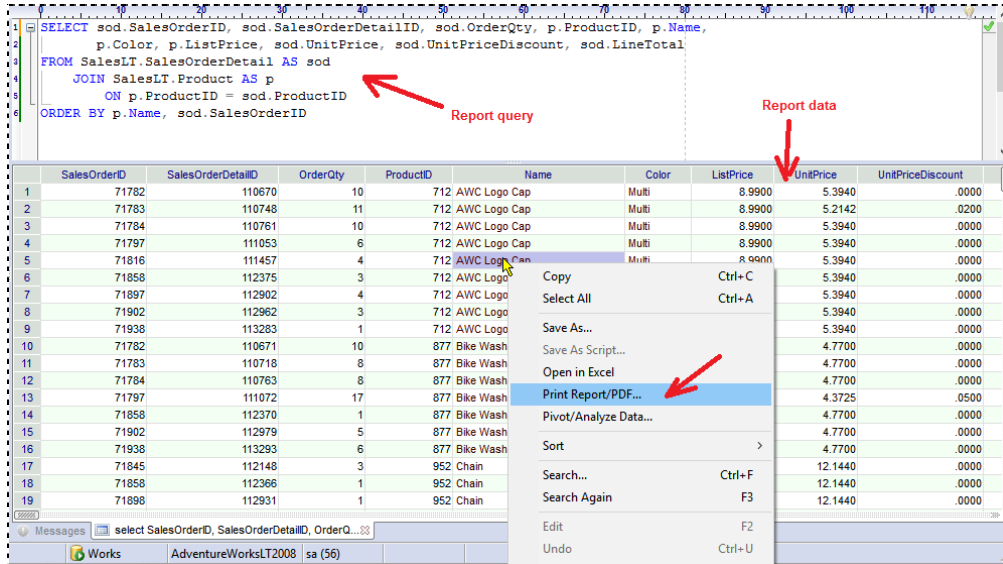
1. Right click the result tab and choose Rename command from the context menu:
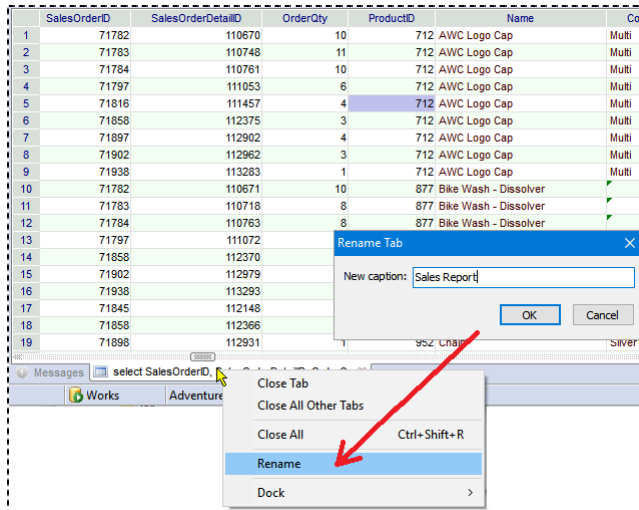
2. Right-click the data grid and choose the **Print Report/PDF…** command in the context menu. This will generate a report and display it in the report Preview dialog as in the following example.

3. Click the **Print** toolbar icon in the report Preview dialog to print the report.

4. Click the **PDF** toolbar icon in the report Preview dialog to save report to a PDF file. If the file already exists, SQL Assistant will prompt to overwrite the file.

5. To close the report, click the **Close** button in the top right corner of the dialog.

Changing Column Sizes

Width of columns in the report matches width of columns in the data grid from which the report was generated. To adjust the width, resize columns in the data-grid and then regenerate the report using **Print Report/PDF…** command in the data-grid menu. If required, repeat adjust column width again until your satisfied with the report column layout.

Data Pivot and Advanced Analytics

Overview

The data-grid integrates directly with pivot-grid based on the FastCube component developed by Fast Reports corporation that can be used for fast data pivoting with advanced analytics functions. It's a full featured OLAP cube enabling instant in-memory data transformation, supporting simple and intuitive drag-and-drop interface, multi-level row and column headers, conditional data formatting, drill down and drill through interfaces, dynamic data sorting, grouping, and filtering, and a number of other advanced features. The headers are filled with the

dimension values. The central part of the grid displays the values of the measures.

Pivot-grid User-Interface

The pivot-grid consists of several interactive regions, identified by the numbers above:

1. **Filter region** - dimensions included in this region can be used for filtering the data

2. **Vertical dimensions** (a.k.a. fields for row headers): - dimensions included in this region form the grid's row headers.

3. **Horizontal dimensions** (a.k.a. fields for column headers): - dimensions included in this region form the grid's column headers.

4. **Row headers –** filled with values from vertical dimensions. You define the dimensions for this region by dragging field names from the "List of fields" drop-down control.

5. **Column headers –** filled with values from horizontal dimensions. You define the dimensions for this region by dragging field names from the "List of fields" drop-down control.

6. **List of fields available** in the pivot cube – you can drag them to row and column headers, filters and the main data region.

7. **Data region** – contains the computed values using the selected fields. You fill this region by dragging field names from the "List of fields" drop-down control, and then optionally choose the kind of values or aggregates to compute.

8. **Quick aggregates** – this region shows aggregates for the selected cells. The choice of aggregates can be customized through the region's context menu.

9. **List of Top-N filters** – the Top-N filters can be modified through the context menu for this region.

10. **Scale control –** the scale factor for the pivot-grid display. You can change the value by clicking the scale value.

11. **Toolbar** – provides quick access to common pivot-grid functions, including saving the cube definition to a disk file and reopening previously saved cubes.

12. **Navigation links** – links for switching back to the simple data grid, printing reports, saving results to PDF files.

Quick Tutorial

The following tutorial demonstrates how to use the data pivot features. The tutorial analyses flight cancelations by airports and carriers using flight cancelations and delays database populated with January 2017 data for North America.

Step 1: Develop a query to retrieve raw data for pivoting and advanced data summarization. Typically you should use a fairly simply query returning all required values. A query may join several tables, including lookup tables returning all descriptive values that you need to use as dimensions and labels on the report. Execute the query to retrieve result set into SQL Assistant's regular data-grid.

Step 2 (optional): Rename the result tab if you are planning on printing or saving results and want to use some meaningful name in the report title. If you don't rename it, your raw query text will be usedi n the title. Right-click the tab and choose the **Rename** command as shown on the following screenshot.

Step 3: Right-click the data-grid and select the **Pivot/Analyze Data** command from the context menu. The data-grid will convert to data-pivot interface.

Step 4: Click the arrow to the right of the Fields drop-down to expand the field list and then drag-fields from the list to row and column header regions. In this tutorial we have airports shown in columns and carriers shown in rows. We also make cancelation reasons available as a dynamic filter. For more details see read annotations on the following screenshot.
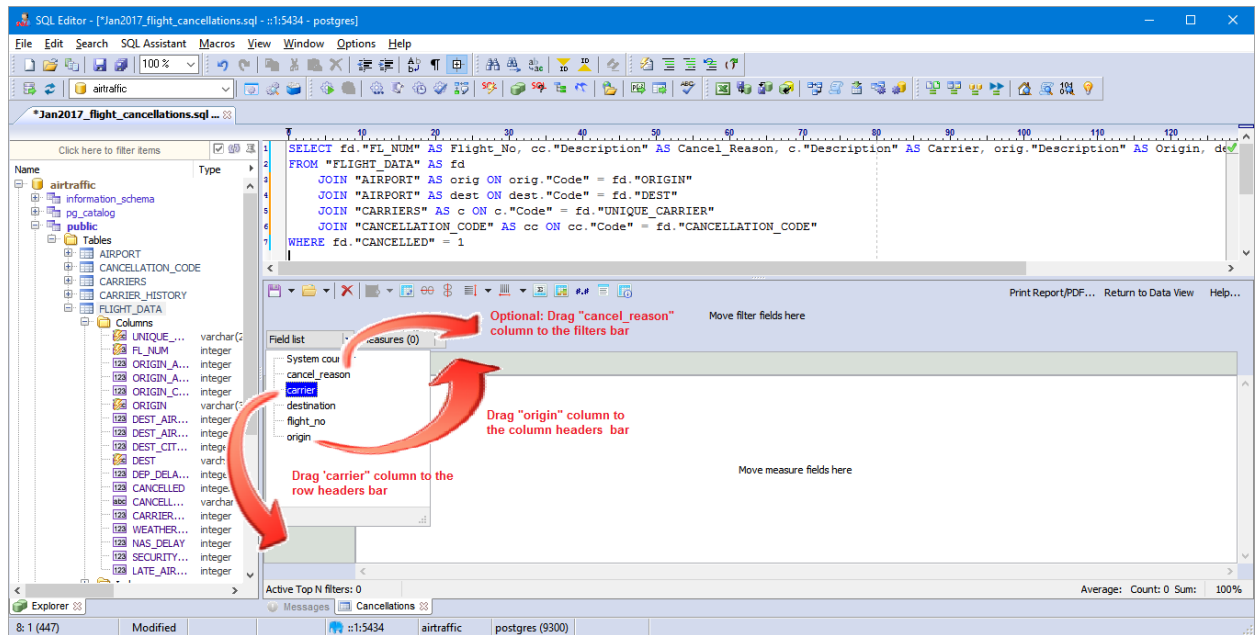
Finally drag the main measures column to the data pivot data region. In this tutorial we use the system generated "System counter" column that calculates the number of occurrences for the each combination of carrier and airport. The resulting grid is shown below.

Multi- dimensions: You can drag multiple fields to the row and column headers to create grouped dimensions with granular field scope numbers break down. In the next step we will add the cancel_reason field to the row headers immediately after the airline name. Note that now the cancelation counts for each airline are broken down by cancelation reason with automatic totals calculated for each airline.

Dimensions and Measures

The pivot-grid can be considered a multi-dimensional generalization of a two- or three-dimensional spreadsheet. For example, you might wish to summarize financial data by product, by time-period, and by city to compare actual and budget expenses. Product, time, city and scenario (actual and budget) are the data's dimensions. Each cell of the pivot-grid holds a number that represents some measure of the business, such as sales, profits, expenses, budget and forecast. In the prior tutorial we summarized the North America's flight data by carrier and by airport to compare their average departure time delays. Carrier and airport were chosen as the data's dimensions. Delay time was chosen as the measure.

The simplest method for setting up dimensions and measures is dragging fields from the **Fields** drop-down list or the **Fields list** dialog to the three pivot-grid regions: filter region, row dimensions region, and column dimensions region. To open the **Fields list** dialog, click the Fields List button on the pivot-grid toolbar.
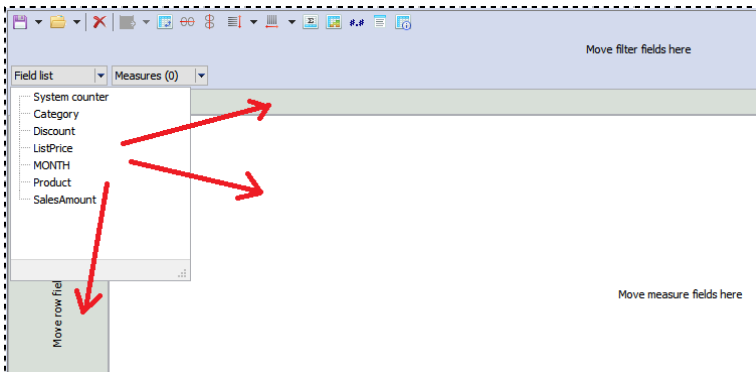
All data pivot interface functions for dimensions and measures can be accessed using right-click menus. Note that various menus are provided for different regions of the pivot grid and for different elements within the regions. For example, to change automatic summaries and counts calculated after the initial pivot-grid is generated to a different set of analytical functions, right-click the required dimension field in the column headers region, and from its context menu choose the kind of function that you want to use for the totals, for example, you can choose to calculate median values instead of averages or sums.
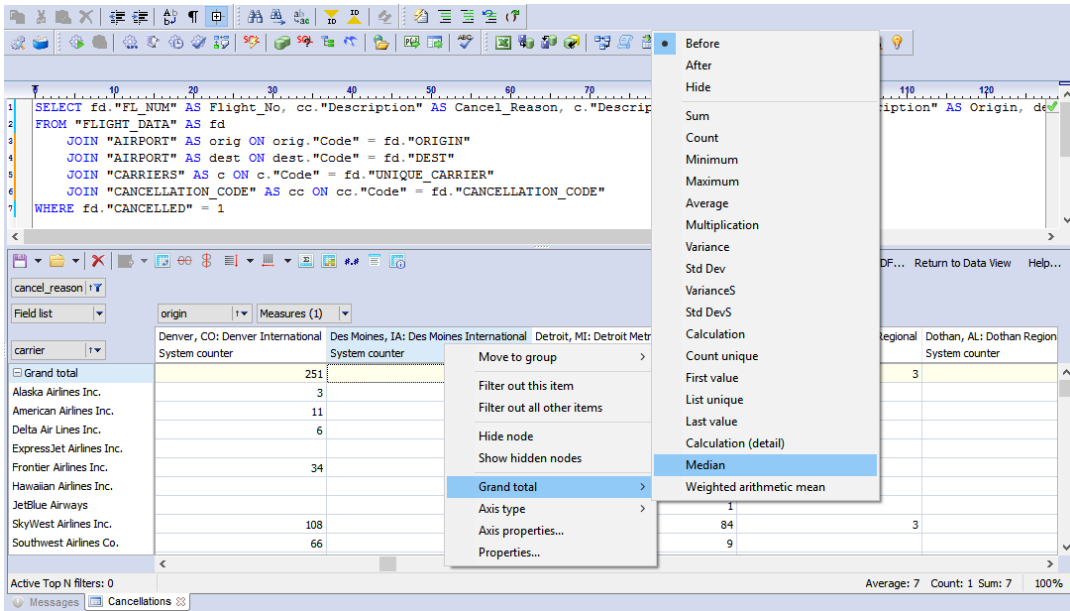
To change the kind of values displayed as measures in the pivot grid, right-click the cell values and then choose value kind
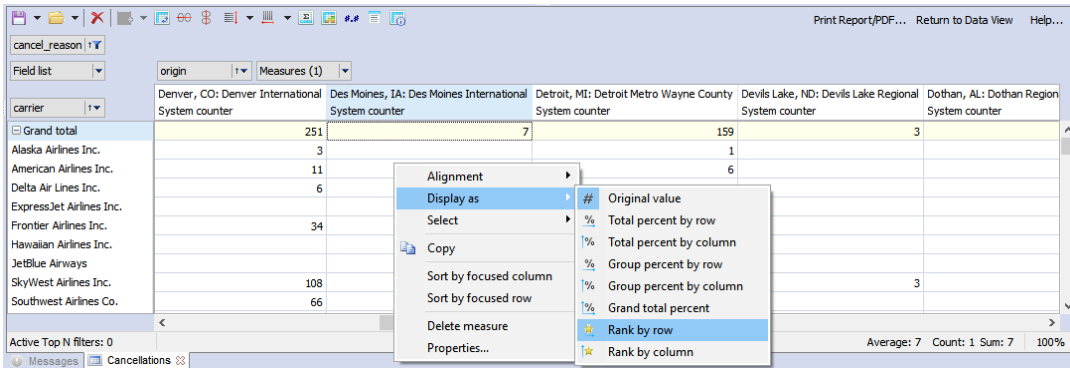
Similarly to change value formatting for different dimensions and regions, right-click the required field and choose the **Properties** command in its context menu, which will open the **Measures Editor** dialog . In the dialog, select the **Display Format** tab and set the required formatting.
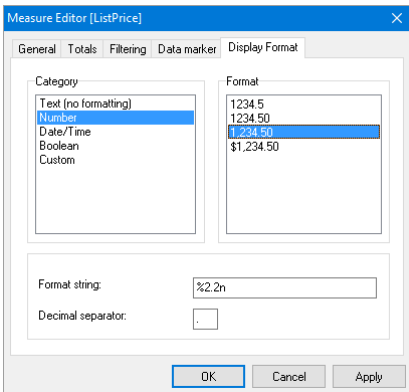
Totals

The pivot-grid supports Total values. Total value is an aggregated measure value over a group of dimension values. A Total value is calculated using the aggregate function selected for the measure. Default aggregate function is sum() for numeric values and count() for non-numeric values. Totals calculated by default for each dimension as well as the grant totals for the entire data set displayed.

You can use the dimension field's context menu to change the position of Totals (before or after), to hide or show Totals, and to change the aggregate function for the dimension Totals. You can also choose different aggregate functions for different dimensions.
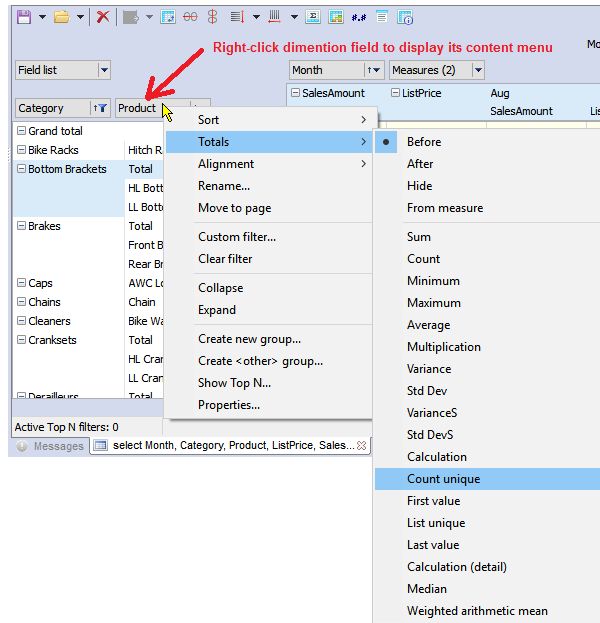
You can also use the **Measures Editor** dialog to select a different analytical function and other related properties. Right-click a cell in the pivot-grid for a measure whose totals calculation you want to change. Select the **Totals** tab and modify it as required.
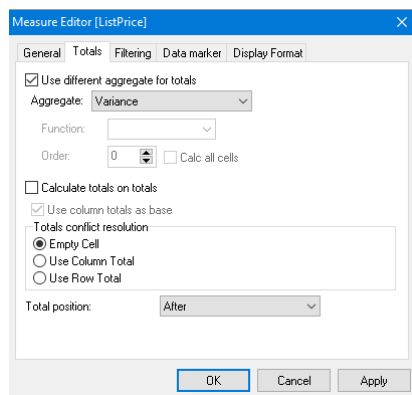
Drill-Down/Up and Drill-Through

The pivot-grid data can be collapsed (drilled up) to exclude the values of the nested dimensions from

processing or can be expanded (drilled down) to include the values of the nested dimensions. These operations are performed using the [-] and [+] buttons located in row and column headers of the pivot-grid.

Drilling up or drilling down changes the structure of the header and the data region. To group the data by the a certain dimension, click the [-] button of the corresponding dimension. And conversely, to refine the data click the [+] button.

Tips:

- Only the main dimension Total is shown when a dimension item is fully collapsed.
- It is possible to collapse the grand totals. The result depends on whether or not the measure fields are placed in the collapsed axis. An axis without measures collapses with the hiding of all the cells except the Grand Total cell. An axis with measures can collapse grand totals for each measure independently. The collapse of measure grand totals hides all the cells of that measure. Collapsing the grand totals of all the measures hides all the cells except the grand total cells for each measure.

Double-clicking a data cell within the Measures regions opens a window showing the **Detail Table** containing the source data rows used for calculation of the selected cell.

Partial Rotation and Full Transposition

The pivot-grid layout can easily be changed on the fly by moving dimensions between the three regions: filter region, row dimensions region, and column dimensions region. In OLAP terms, this operation is called rotation, because it corresponds to rotating a multidimensional data array. Data rotation enables the same information to be analyzed from different perspectives.

To rotate the data drag-and-drop the dimension fields from one region to another.

As the dimension field is dragged over the pivot-grid, a special pointer shows where the dimension would go if the button mouse is released at that moment;

This is the result after the "Item" dimension has been moved from the row dimensions region to the column dimensions region:

Another way to rotate dimensions is to use the **Field list** dialog. Drag fields from the dialog and drop them into filter region, row dimensions region, or column dimensions region. If a field is already in one of the regions, it's automatically moved to the new region.

Full Transposition

To perform the total rotation (pivot-grid transposition), use the Transpose [icon] button on the pivot-grid toolbar. This operation moves all the row dimensions to the column region and all the column dimensions to the row region. Unlike partial rotation after moving a subset of dimensions, the transposition does not require the recalculation of data cells, and as a result, it is instantaneous.

Grouping

The pivot-grid can join several dimensions into a group. Grouping is a two stage process; first you define the group, and then you chose how to populate them with values.

Grouping features:

- Unlimited number of dimension groups
- Empty groups allowed
- Empty groups not shown on axes
- Groups must have unique names inside a dimension
- A dimension value cannot belong to more than one group
- Inclusion of a dimension value in one group automatically excludes it from all other groups
- A dimension value can be excluded from grouping
- A system group "others" can be created, which includes all values not belonging to any other group
- Group deletion automatically excludes all the values it contains
- Group creation, deletion, renaming, value inclusion and exclusion can be performed while the cube is active
- A filter window shows all groups and their members : group filtering state depends on member states
- Moving a dimension from one region to another does not reset groups
- Dimension groups are saved in cube files together with the dimension members

The axis shows dimensions having groups at two levels: group level and member level. The group level contains dimension group names and also dimension values that do not belong to any group. The member level contains the values which belong to the groups. A group can be in a collapsed state, in which case the group members are not shown. If all groups are collapsed then the member level is not shown.

To create a new group, use dimension field's right-click menu.
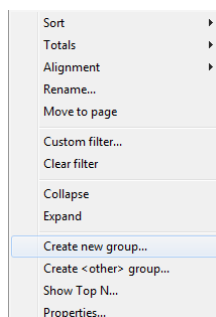
Or use right-click menu for the dimension member
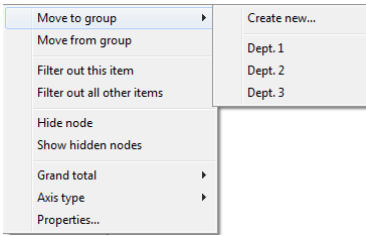
Important Note: Any operation involving a group, including collapse/expansion, causes measures recalculation because of the change to the axis.

Sorting

All data in the pivot-grid is always displayed in certain order, irrespective of the original order in the source data-grid populated from your SQL query. The sort order is either ascending or descending. Numeric values and date/time values are sorted in their natural order, ascending or descending. Strings are sorted alphabetically, ascending or descending. The default sort order is ascending for all dimensions.

Every dimension in the grid can have its own sort order. To change order for a given dimension, right-click it in the dimensions region and then choose Sort command from the context menu as demonstrated o nthe following image.
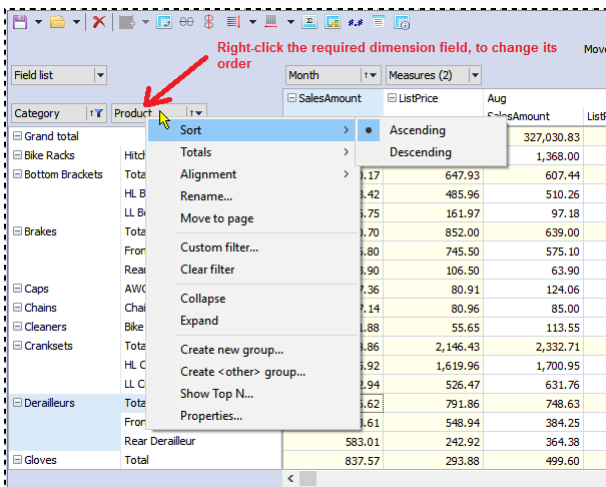
Filtering

In the optional step 4 of the prior tutorial we dragged canel_reason field to the filters bar. As a result, all possible values for that field appeared in the "cancel_reason" drop-down list shown above the grid. To dynamically change the filters, expand the drop-down and select the values that you want to keep or hide as demonstrated on the following screenshot.

Important Note: It does not matter in which region of the pivot-grid the filtered filed is located, it could be in filter region, row dimensions region or column dimensions region. Filtered values are not only hidden in the grid header but are also excluded from measures and Totals calculations.

Values of dimensions can be filtered using the drop-down list which is opened with a click on the dimension filtering button displayed to the right of the dimension field.
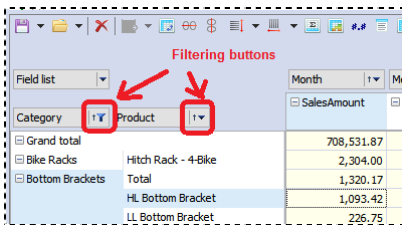
Conditional and Continuous Data Highlighting

The pivot-grid supports conditional cell highlighting. For example, a rule can be setup for displaying numbers inside or outside a particular number range with a different background. The following screenshot demonstrates two rules setup for a pivot grid used for the tutorial in the beginning of this chapter.

To setup a rule for a particular field, right-click that field and choose the **Properties** command in its context

menu, which will open the **Measures Editor** dialog . In the dialog, select the **Data Marker** tab. Click the plus icon ✚ and add the required formatting style and range.
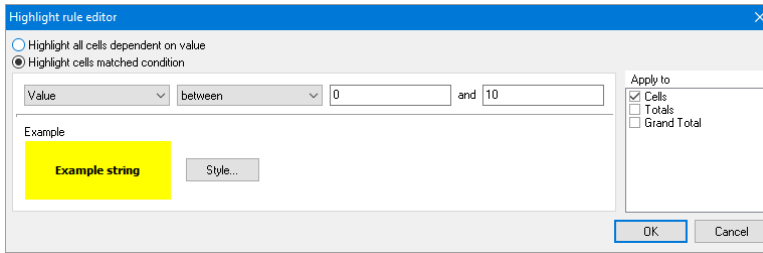
The second rule on the following screenshot illustrates conditional formatting rule.
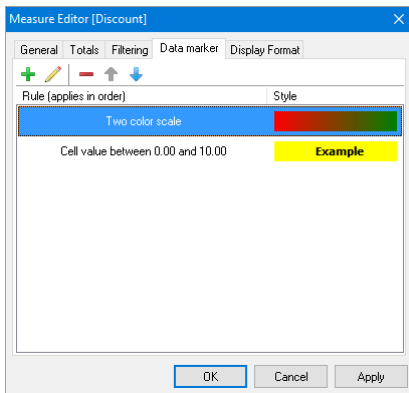
Highlight rules can process measure data as value, text or data, or NULL. Allowable conditions depend on the type of the measure's data. So, for example, value processing allows "greater" and "lower" conditions while text processing allows "contains" and "not contains" conditions, which search for the specified substring within measure's data values.

The display style provides options for choosing cell's background fill style and color, as well as cell's text style and color.

The pivot-grid also supports continuous highlighting, which applies highlighting to all cells dependent on their values. The example rule on image above illustrates continuous highlighting.

Each type of continuous highlighting is described below:

- **Two color scale and three color scale** - Color scale highlighting fills each cell's background with a color calculated from the given color gradient scale. The **Highlighting Rule Editor** dialog sets the values for the extreme points and their corresponding colors, and also for one intermediate point for the three color scale.
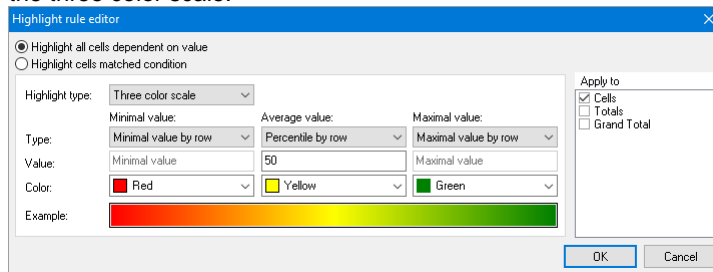
The values for the points can be set either as absolute numbers using the "Number" option, or as one of the following relative options:
Minimum by row/column - minimal measure value in row/column

Percent by row/column - value field sets the percent relative to minimal and maximal measure values in row/column

Percentile by row/column - value field sets the percentile relative to measure values in row/column

- **Bar** - This type of highlighting draws a colored bar within the measure cell. The length of the bar depends on the values set under the Shortest bar and the Longest bar properties. The property values, as previously, can be set with either the absolute or the relative option. The bar can be drawn either in a solid color or with a gradient color to white. The frame color for the bar is set independently. The cell value can also be hidden for this type of highlighting, leaving only the bar visible.
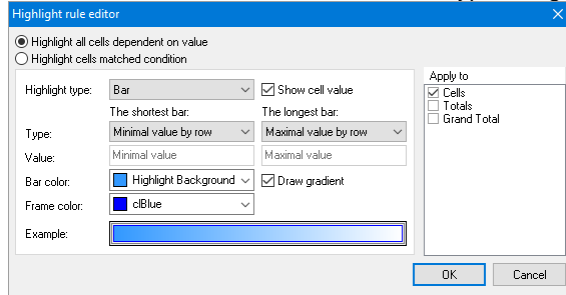
- **Icon set** – This type of highlighting displays an icon related to the value inside the measure cell. Icons are drawn to the left of the cell's value. The **Highlighting Rule Editor** dialog has options for sets of 3, 4 or 5 icons. Each icon requires a value range to be set. The cell value can also be hidden for this type of highlighting, leaving only the icon visible. Here is an example of icon set highlighting with and without visible cell values:
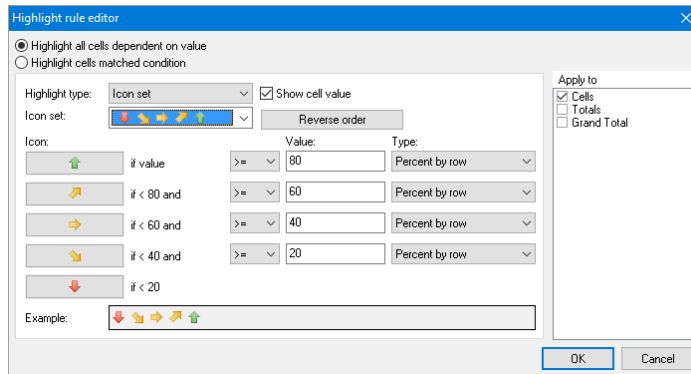
and here is a sample result of using this kind of highlighting style:

Saving Data-Pivot for Continued Data Analysis

Use the Save ⊟ button on the data-pivot toolbar to save the current pivot sate with the data to an .MDC file. You can later open the save file using the Open 🗁 button and continue working with it.

The .MDC file can be also shared with coworkers provided they too have the SQL Assistant software installed on their computers.

Printing Data-Pivot and Saving it

In the right-top corner of the pivot grid click the **Print Report/PDF…** link. The report preview will open.

To send report to a printer for printing, click the Printer icon on the report preview toolbar 🖶

To save report to a PDF file, which can be opened later or emailed, click the PDF icon 📕
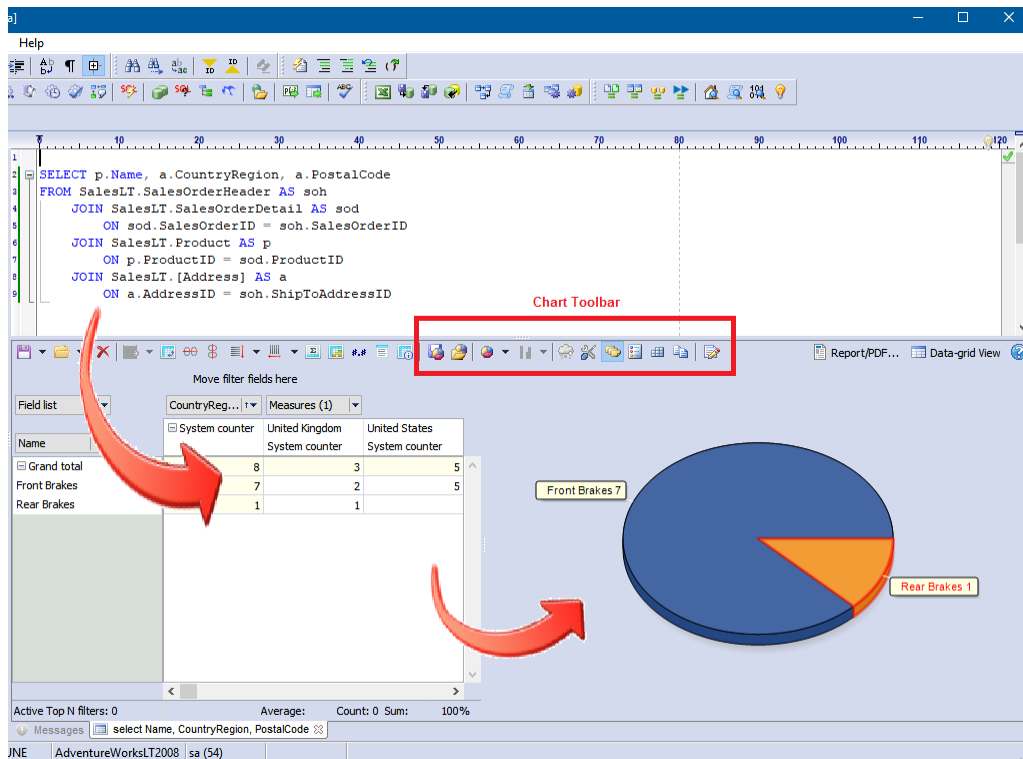
To save report to a Prepared Report file which can be opened later without re-running the report, click the Save icon 💾. You will be prompted to select file name to save report to. The report will be saved along with the data. To open previously saved report, click the Open Report icon 📂. You will be prompted to select previously saved prepared report file.

Charts

Charts are available in Standalone Reports through the charts component, as well as in the data grids through the integrated Data Pivot and Advanced Analytics feature. In the first case, when you design reports you need to add chart component to the report and then add the date series. In the second case you do not need to do anything special to enable them, they are available to you instantly when you switch to the Pivot-grid view as described in the Quick Tutorial topic earlier in this chapter. A variety of chart types are supported, including line, pie, bar, area, and scatter.

As you choose dimensions and measures in the pivot-grid, apply various filters and totals, your chart will automatically update. If you want to disable automatic updates, choose the Freeze Chart toolbar button.

Use the charts toolbar to customize chart behavior and appearance. You can optionally save your preferred chart type and appearance to a template file and later quickly apply it to future charts.

Basic Chart Operations

Save current chart type and selected style to chart template file with .MDT extension.

Open previously saved chart template and apply it to the current chart.

Select chart type using drop-down menu, such as Vertical Bar, Horizontal Bar, Line, Pie, Area, and Point (a.k.a. Scatter chart).

Important Note: The drop-down list contains most common chart types. Changes in the chart type selection apply to all series. To change type of specific series only, and for complete gallery of all supported chart types, open the **Visual Chart Properties** dialog as described in the Customizing Visual Properties topic, select the Series item in the left side navigation menu, and then click the **Change…** button.

Select chart sub-type, such as Stacked, Stacked 100% (Normalized), Self Stack, Side, Side All, and None.

Freeze / un freeze automatic chart updates on pivot-grid properties and data changes.

Open the **Chart Properties** dialog. In the dialog you can choose the chart data source, series and categories.

Show / hide data marks next to data points, or pie chart slices. Data marks show precise value and category labels for the given the data point.

Show / hide chart legend.

Open the **Data Management** dialog, which you can use to drill-down to specific data elements for the calculated totals.

Copy chart to the Clipboard.

Open the chart Visual Properties dialog, which you can use to customize colors for different series, to customize axes lines and other chart elements.

Tips:

- You can copy charts to the Clipboard and then paste them into your documents and presentations, or simply paste into email and send to colleagues.

- To save chart to an image file, open **Chart Visual Properties** dialog. On the left side of the dialog click the **Export** option in navigation tree, this is second last option. Then the **Save…** button.

- To create multiple charts using the same data set, execute the same query as many times as many charts you require. In each data-grid pane switch to the pivot-grid, choose different dimensions and measures for the analytical data aggregation. Customize chart appearance as required.

Data Source Selection

Click 🛠 button in the Chart toolbar to open the **Chart Properties** dialog. Using the dialog select the source of the chart data. For the selected source, customize the source of series and categories. You can also choose how many dimension and measure fields to add to the chart from the selected data source.
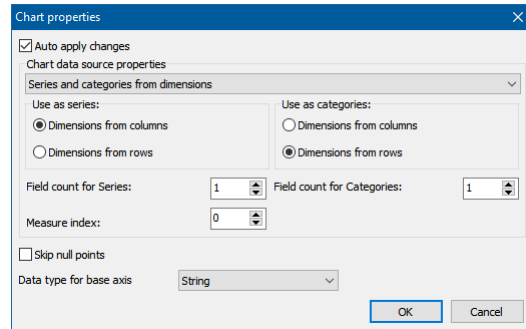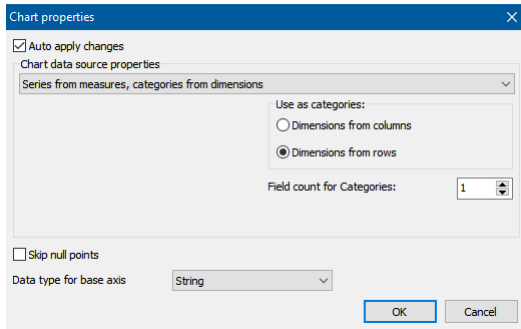
Note that the properties available for customization in the **Chart Properties** dialog depend on the selection of the data source type.

Customizing Visual Properties

To customize chart appearance, double-click anywhere within the chart area. The **Chart Visual Properties** dialog will appear. The dialog allows fully customization of all chart elements, as well as individual data series.
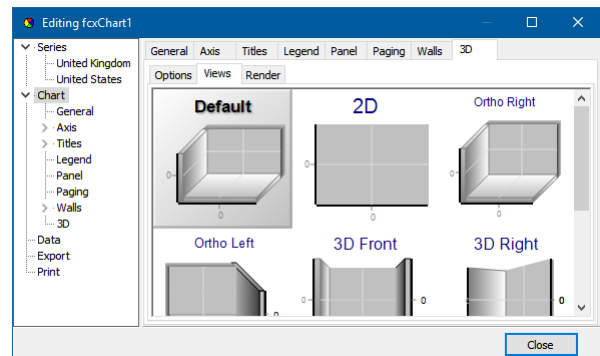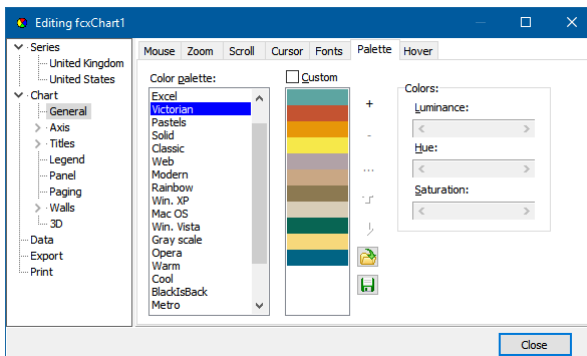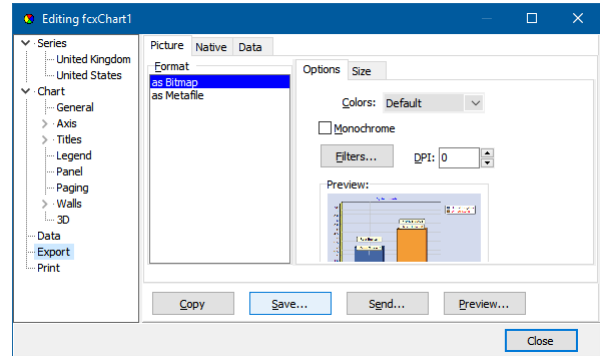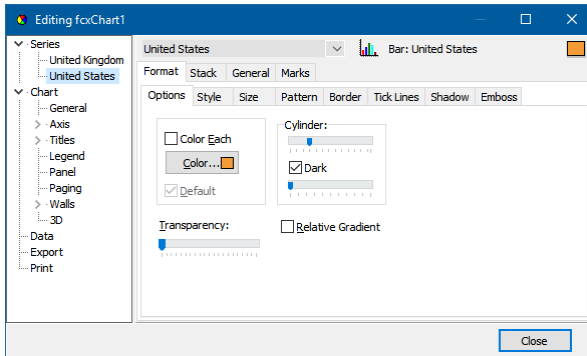
Use the navigation menu on the left side of the dialog to choose specific elements that you want to customize.

The right side of the dialog is filled with tabs and controls specific to the type of the element selected in the navigation menu.

Note: The complete description of all customizable chart elements and their properties is outside of the scope of this manual. Please refer to the on-line documentation for the TeeChart © component by the Steema Software for complete details and tutorials. A copy of the online documentation can be found here http://www.teechart.net/docs/TeeChartGeneralWiki.htm

CHAPTER 25, Code Compare Utility

Overview

SQL Assistant provides an integrated Code Compare utility that can be used to compare changes in code or text data files. This utility offers unique features that allow comparing differences between the following objects:

- Two text files with any file extension.

- Text file with any file extension against text in the target editor.

- Code of a procedural object in the database (such as stored procedure, user-defined function, package or type) against text of in the target editor.

- Code of a procedural object in the database (such as stored procedure, user-defined function, package or type) against text of any text file.

- Text in two edit windows running in the same or different applications both of which are open on the same desktop; for example, text loaded in a Notepad window against text loaded in SQL Server Management Studio window.

The Code Compare utility can be invoked from SQL Assistant's menus. The **Compare Code and Data → Compare Code** command is available from either the right-click context menu in the target editor or the top-level menu (if the target editor top-level menu integration is enabled). This command opens the **SQL Assistant – Text Compare** dialog. In this dialog you can select the target files or edit windows to be compared. The Code Compare utility can be also invoked using special shortcuts in the Code View pane. This pane can be used to compare code of procedural objects from the database against code in the editor window or against code of an external text file. The following screenshot shows where to find these shortcuts.
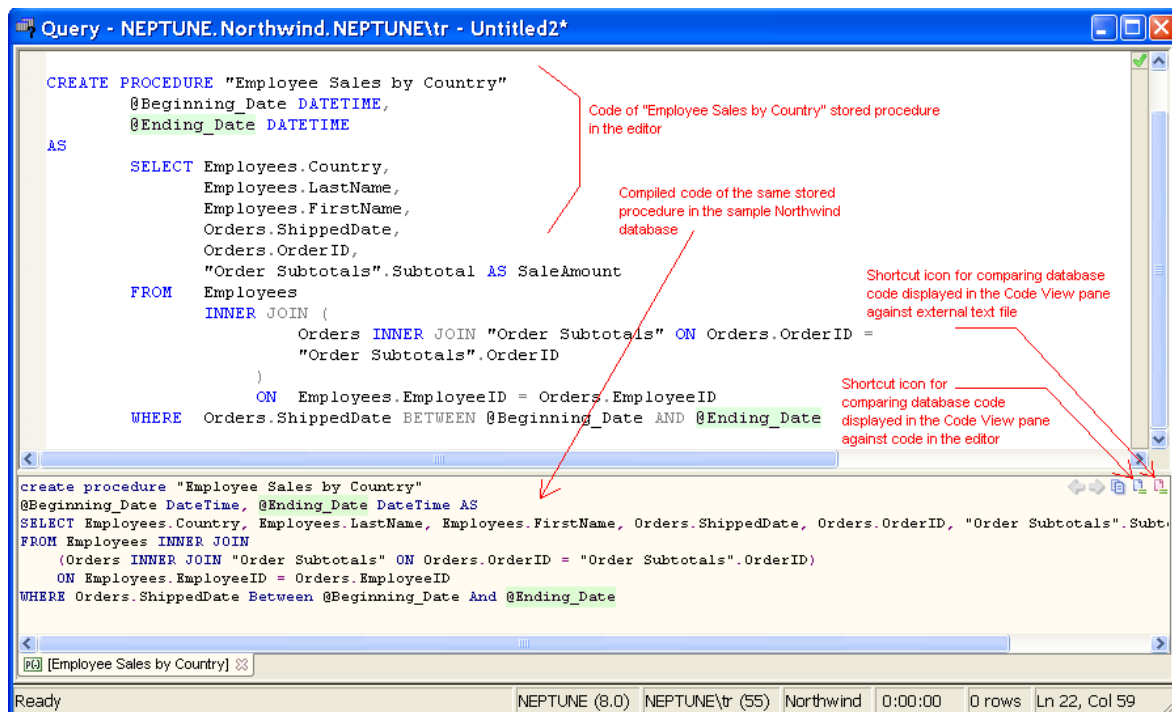
See CHAPTER 10, One-click DDL Code View for information on how to open and display the Code View pane.

Using External File Compare Tools

If you prefer using some other code compare tool, you can configure SQL Assistant to use the external compare tool instead of the built-in tool.

1. In SQL Assistant **Options** dialog, select **Source Control** tab.

2. Click the **SCS Settings** section on the left.

3. In the **External Compare Tool,** choose one of the preconfigured tools or enter a new command line with parameters for an external file compare tool of your choice.

Using Code Compare Dialog

The following screenshot shows an example code comparison dialog.
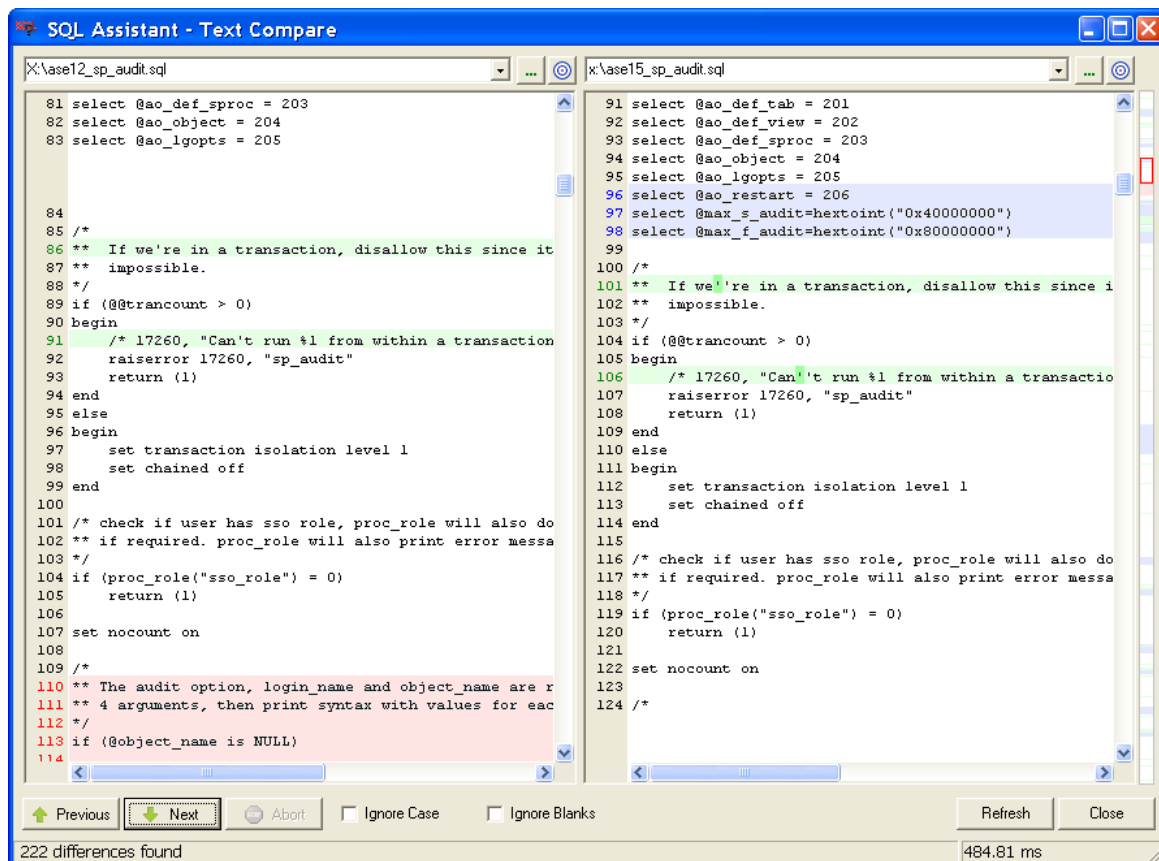
Color Highlighting

The Code Compare function uses different colors to highlight text different differences. Pink indicates deleted lines, light blue indicates new lines, light green indicates modified lines, and bright green within modified lines

indicates actual text differences.

Dialog Controls:

Left-side drop-down list – This list contains recent files selected for comparison. Use this list to quickly pick a file against which you want to compare current text in the editor or code in the database. The content of the selected file appear in the left pane of the dialog. You can also type the desired file name directly in the top part of the control.

Right-side drop-down list – This control operates the same as the left-side drop-down list except that it controls files loaded in the right pane of the dialog.

Left-side File Browse button – Click this button to select a text file on a local disk or network share. The content of the selected file will appear in the left pane of the dialog. See the following topics for more information.

Right-side File Browse button – Click this button to select a text file on a local disk or network share. The content of the selected file will appear in the right pane of the dialog. See the following topics for more information.

Left-side Target Window Finder button – Drag this button to select a target editor window. The content of the selected window will appear in the left pane of the dialog. See the following topics for more information.

Right-side Target Window Finder button – Drag this button to select a target editor window. The content of the selected window will appear in the right pane of the dialog. See the following topics for more information.

Previous button – Click this button to jump to the previous difference.

Next button – Click this button to jump to the next difference.

Abort button – This button is only enabled while the comparison routine is in progress. Click this button to abort the comparison routine.

Ignore Case check box **–** Select this option to specify that comparisons should be case insensitive. In other words, character strings like "CASE", "Case" and "case" should be treated as matching.

Ignore blanks check box – Select this option to specify that code comparisons should ignore the number of spaces between words, as well as the number of leading and trailing spaces on each line.

Refresh button – Reruns the comparison and refreshes the content of files being compared. Use this control if the content of compared files or windows may have changed while the Code Compare dialog was open.

Close button – Closes the Code Compare dialog

Interactive Difference Map bar (displayed on the right side of the Code Compare dialog) – Color marks on the bar indicate the relative locations and types of differences within the compared text. Change indicators on the bar have the same color as the color of the type of the difference within the code. For more details, see the Color Highlighting topic in this chapter. The red rectangle on the Interactive Difference Map indicates the relative position of the section of text displayed in the visible area of the Code Compare dialog.

Selecting Text Files for Comparison

To open a file you have used in a previous comparison, simply select it from the left-side or right-side **drop-down list**. You can also type the desired file name and path directly in the top edit box of the drop-down list.

You can also use the left-side or right-side **File Browse** button to open the standard Select Open dialog. By default, this dialog displays only files with SQL or DDL extensions. You can change this filter to select files with any other extensions.

The comparison is run automatically as soon as two target files or objects are chosen for comparison.

Selecting Window Targets For Comparison

If you have a script open in one of the edit windows and you want to compare it with another script, use the left-side or right-side **Target Window Finder** button. The following describes steps for using the **Target Window Finder** button:

1. Arrange the desired window so that the Compare dialog and the target window are both visible.

2. Drag the **Target Window Finder** button to the desired window. As you drag the button over a window, a green rectangle appears around the window so that you can see which window is the current target. As you drag the button, the SQL Assistant copies details of the highlighted window and its caption to the corresponding drop-down list in the Compare dialog.

3. With the desired target window highlighted, release the left mouse button. SQL Assistant will load the content of the window into the Compare dialog.

Important Notes:

- Applications may be constructed using multiple windows. Only very simple applications like Windows Notepad consist of a single window. More sophisticated applications such as SQL Server Management Studio, Visual Studio, Eclipse, DB Tools for Oracle, Toad, and many others, consist of many windows having smaller specialized windows contained within the main application frame. When dragging the **Finder** control, make sure to drop it on the window within the frame containing the actual text that you want to select for the comparison.

- Only edit windows that respond to the standard EM_GETTEXT message can be used with this method. Some objects on the screen that look like standard edit windows may not be. They are painted on the screen in such a way that they look like windows while in fact they are just dynamically painted pictures. A typical example of a painted window can be found in most Java based applications. Edit "windows" in Java based SQL editors such as Oracle Developer, Sybase ISQL and others, cannot be used with SQL Assistant's Code Comparison utility.

Using Synchronized and Independent Content Scrolling

To scroll both text panes synchronously, drug the scroll bar's thumb control on either left-side or right-side vertical bar of the Compare dialog.

To scroll each pane independently, click the up or down arrow controls at the extremes of the vertical scroll bar in the pane you want to scroll.

To scroll content to the previous or next difference, use the **Previous** and **Next** buttons. Both buttons will perform synchronized scrolling of the content in each pane.

To quickly jump to any place in the text, use the **Interactive Difference Map** bar. The bar performs synchronous scrolling of both panes.
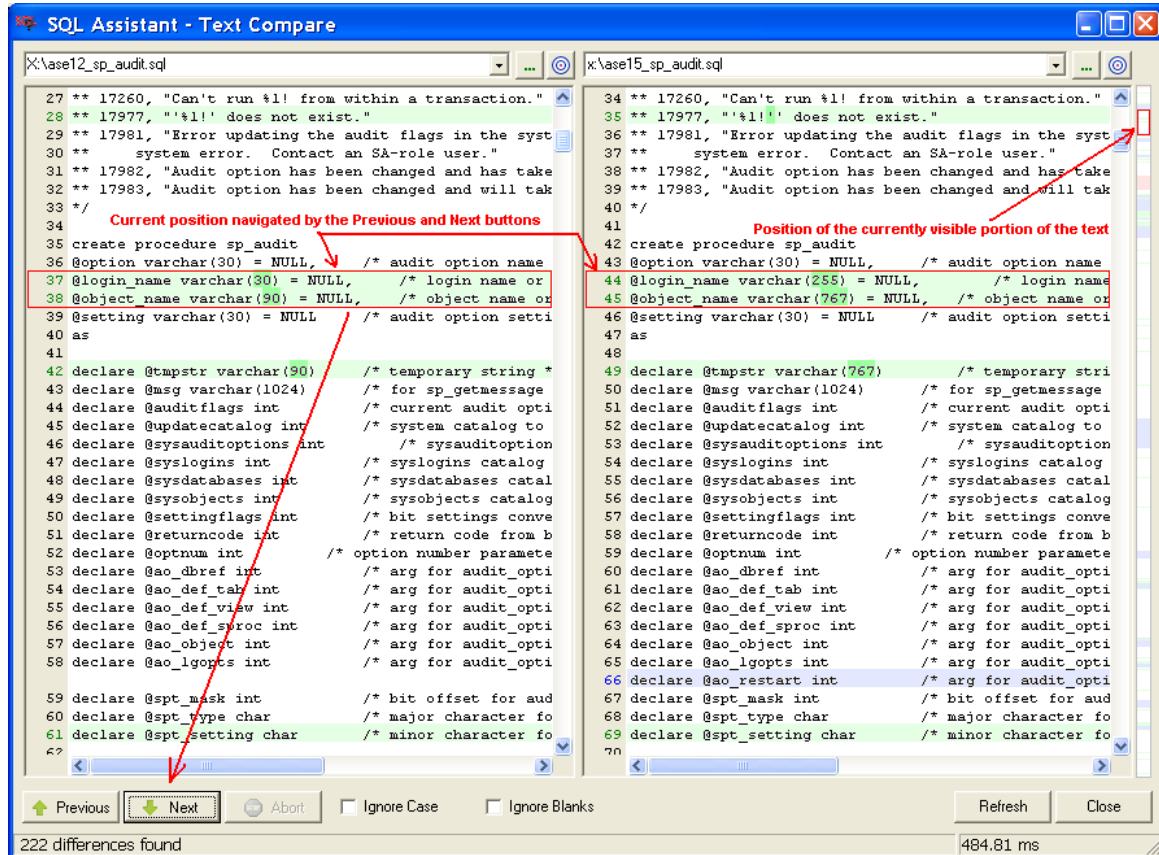
Navigating Content

You can use any standard means for navigating content within the Code Compare dialog, including standard keyboard navigation keys, scrollbars, the **Previous** and **Next** buttons, and the **Interactive Difference Map** bar. The **Interactive Difference Map** bar allows you to quickly jump to any place in the comparison text. When you click on a colored mark displayed on the map, the content is automatically scrolled to that portion of the text visible on the screen. Refer to the previous screenshot for more info.

Resizing Content

To resize the Compare dialog window, drag the resizer handle in the bottom-right corner of the window. See the screenshot at the beginning of the Working with SQL Assistant Popups topic for information on where to locate the resizer handle.

To adjust size of left and right panes within the dialog, drag the vertical split bar separating text panes on the dialog.

Visualizing Code Differences

Since version 10.0 in addition to displaying textual differences SQL Assistant supports visualization of the logical differences and program flow control. The dialog features 3 different tab pages

- Code Differences

- Logical Differences

- Flow Differences

The tab captions are self-descriptive. The following screenshots demonstrate how the visual comparison results are shown. The screenshots utilize default display style for code visualization, the actual appearance may very for various display styles and can be customized in the SQL Code Visualizer utility. See CHAPTER 39, SQL Code Visualizer and Database Documenter for more details on the display styles.

Note that elements with red borders indicate source code changes and removals, while elements with blue borders indicate code additions.
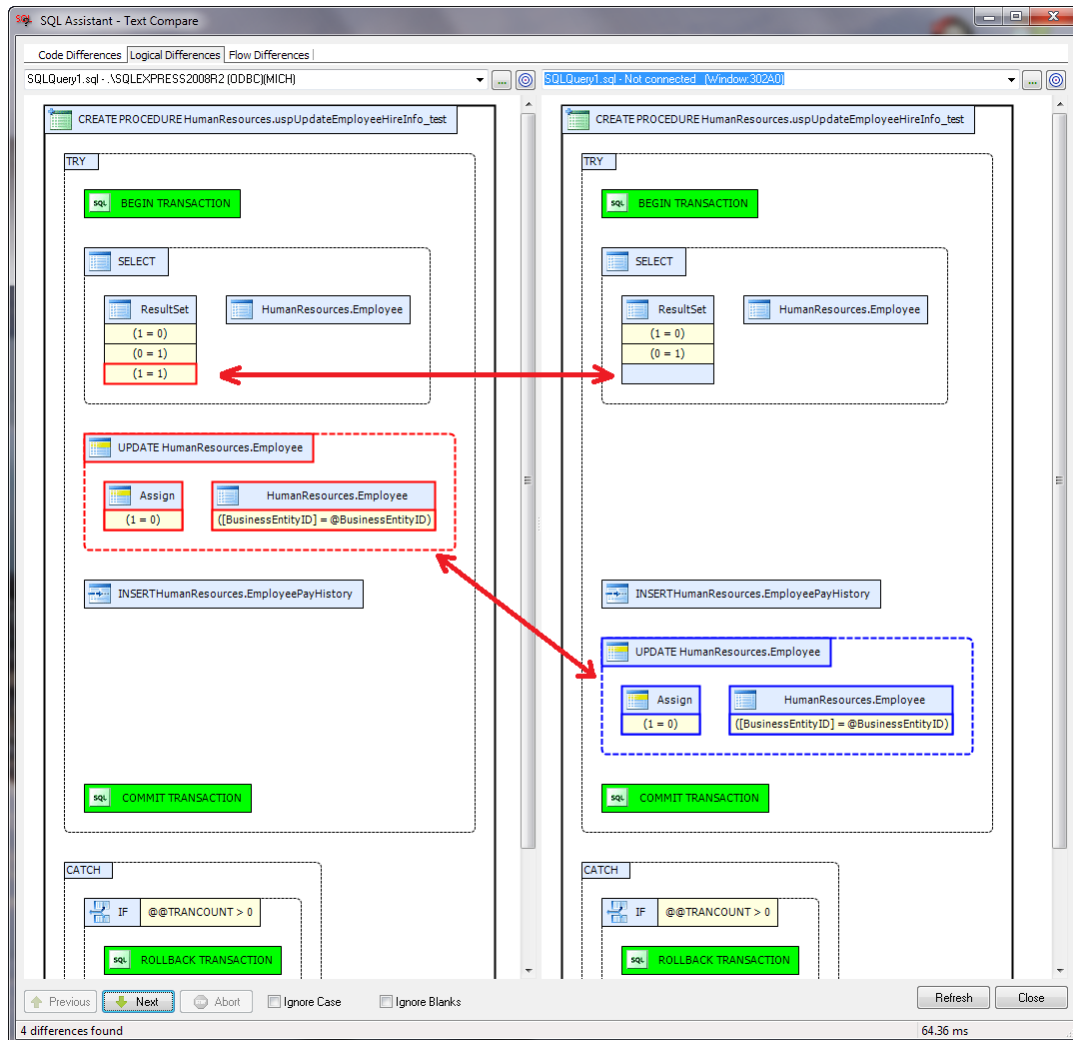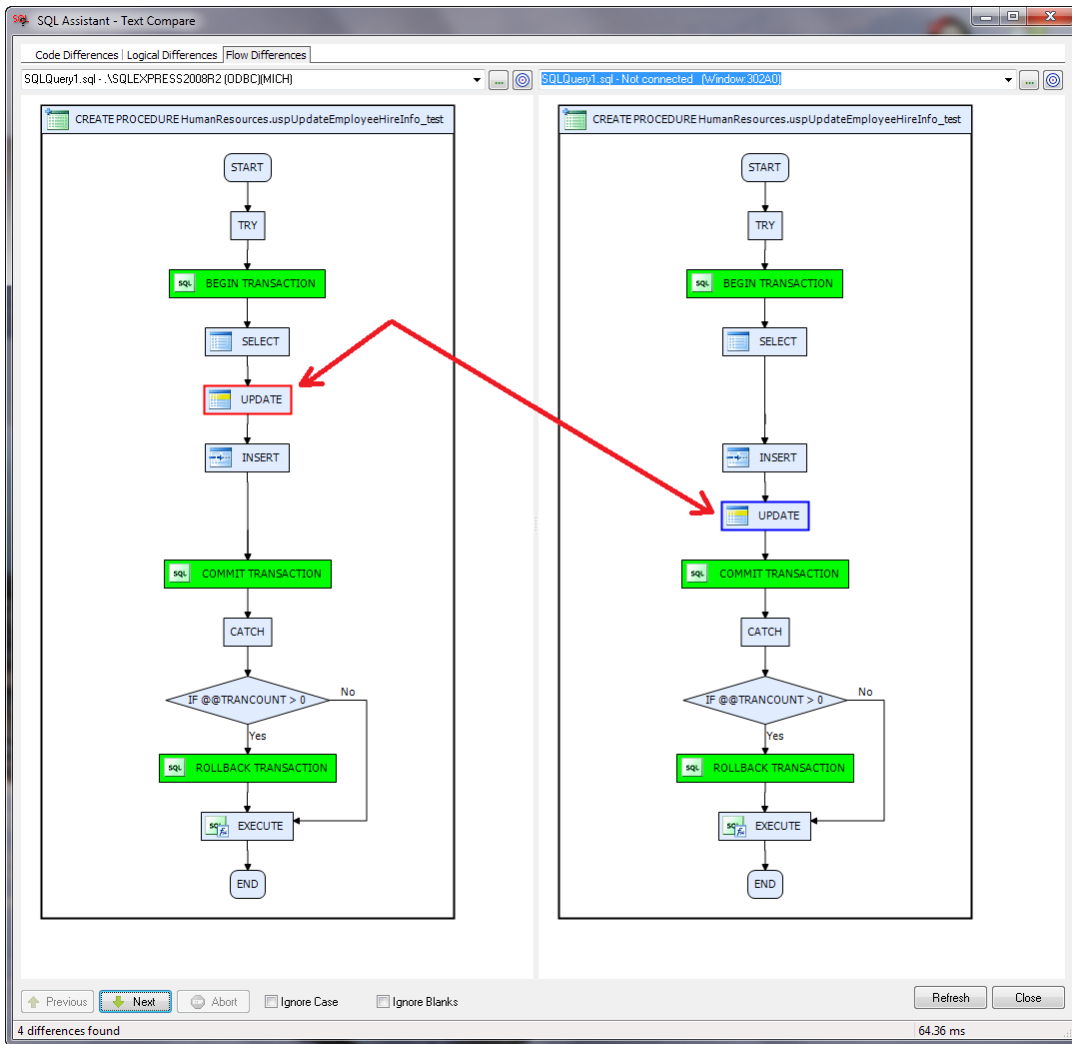
CHAPTER 26, Data Compare Utility

Overview

SQL Assistant provides an integrated Data Compare utility that can be used to compare data difference in database tables and synchronize them too. It supports comparing data differences between the following targets:

- Two tables in the same or different databases.

- All tables in two schemas in the same or different databases.

- All tables in all non-system schemas in two different databases.

Tip: The compared databases, schemas, and tables can reside on the same or different database servers. The compared servers can be of the same or different types. For example, you can compare a schema in SQL Server against a schema in Oracle or MySQL database servers. The Data Compare utility is best used to check for the data differences after database deployments, application conversions and upgrades, and other tasks involving large data migrations and changes.

The Data Compare utility can be invoked from SQL Assistant's menus. The **Compare Code and Data →
Compare Data** command is available from either the right-click context menu in the target editor or the top-level menu (if the target editor top-level menu integration is enabled). This command opens the **Data Compare** dialog. In this dialog you can select the target servers, databases, schemas and tables to compare. The following topics describe how to use the Data Compare utility

The Data Compare utility performs data comparison operations on data in the selected tables only. It first matches tables in the source and destination databases and schemas by their names. It then compares table definitions and columns by column names and data-types. It also retrieves definitions of table primary keys and compares them too. It then matches data rows in the source and destination tables by primary keys. For rows with matching primary keys it compares values in all remaining table columns.

The Data Compare Dialog

The Data Compare dialog guides you through a 5-step data comparison process:

- Step 1: Enter required data comparison options and scope.

- Step 2: Review table and column matches for the selected scope. If required map key columns for tables without primary keys and unique constraints.

- Step 3: Execute the data comparison operation.

- Step 4. Review and optionally save comparison results.

- Step 5 (Optional): Synchronize the data differences.

Comparison Scope and Options

Scope – The scope of the data comparison. One of the following:

- **Single Table** – Select this option to compare data in a single database table against another table residing on the same or different database server.

- **Schema** - Select this option to compare data in all tables in a database schema against another schema residing on the same or different database server.

- **Database** - Select this option to compare data in all tables in a database against another database residing on the same or different database server.
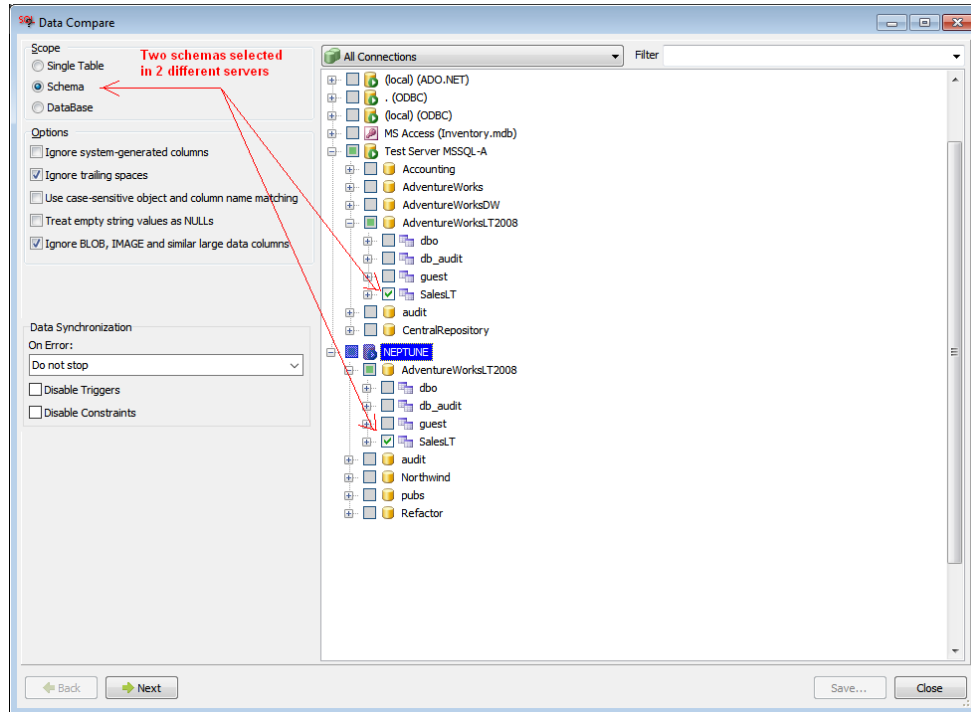
Ignore system generated columns – Instructs SQL Assistant to ignore system generated columns such as TIMESTAMP columns in SQL Server tables, IDENITY and AUTO_INCREMENT columns in SQL Server, MySQL and DB2 databases, which are not part of a table primary index, COMPUTED columns in SQL Server, and GENERATED ALWAYS columns in Oracle in DB2 tables, and so on. If this option is selected, SQL Assistant does not compare data in the system generated columns.

Use case –sensitive column and object name matching – Instructs SQL Assistant to use case-sensitive matching method when selecting tables and columns for the data comparison. If this option is selected, SQL Assistant will treat tables named "Account" and "ACCOUNT" as two different tables.

Treat empty-string values as NULLs - Instructs SQL Assistant to use to teat empty string values '' as NULL values. For example, if a value in the source table is an empty string '' and a value in the destination table is NULL, they are be considered to be equal.

Ignore BLOB, IMAGE, and similar large data columns – Instructs SQL Assistant to not compare values in the binary and other large data columns. This may significantly improve the speed of data comparison operations and significantly lower the memory usage requirements for the data comparison and display of the data comparison results.

Selecting Databases, Schemas, and Tables for Comparison

Connection and Name Filters

Multiple server connections are organized into logical connection groups. You can use the right-click menu to manage connections in place. To manage connection groups use either the SQL Assistant main Options dialog or the Multi-server Code Execution utility. For more information on managing connection groups and connection settings see Managing Connection Groups and Connection Settings topic in CHAPTER 15.

The Filter combo-box offers super-fast content filtering. Type the substring you want to use as a filter for database objects into the Filter box available above the object tree. Previously used filters are available in the drop-down portion of the Filter combo-box.

Tip: You can add new connections directly in the same dialog. Use the right-click menu in the connection tree. The same right-click menu can be used to modify saved connections.

Navigation

All server connections, databases, schemas, and tables are displayed in a single Object Tree. You can select only 2 items of the same type. The type must match the selected comparison **Scope** option.

Object Tree Legend

The following types of checkboxes could be displayed in the object tree.

☐   Item is not selected for data comparison. This item is available for selection.

▨   Item is not available for selection because the item scope does not match the **Scope** option or you already have two items selected for comparison. In the latest case, If you want to change the selection, first unselect one of the selected items and then select another item.

☑   Item is selected for data comparison.

▣   For an expandable schema or database item, this type of checkbox indicates that only a subset of "child" items has been selected in the object tree branch for this item.

Matching Tables and Columns

Step 2 of the Data Compare dialog is where you can review tables and columns matched for the data comparison fine tune the selection criteria. In Step 2, the dialog is split into two parts: the table matches in the top section of the dialog, and the column matches in the bottom part. the bottom part is populated when a table match is selected in the top part. Checkboxes in front of item icons can be used to select and unselect tables and columns you want to compare.
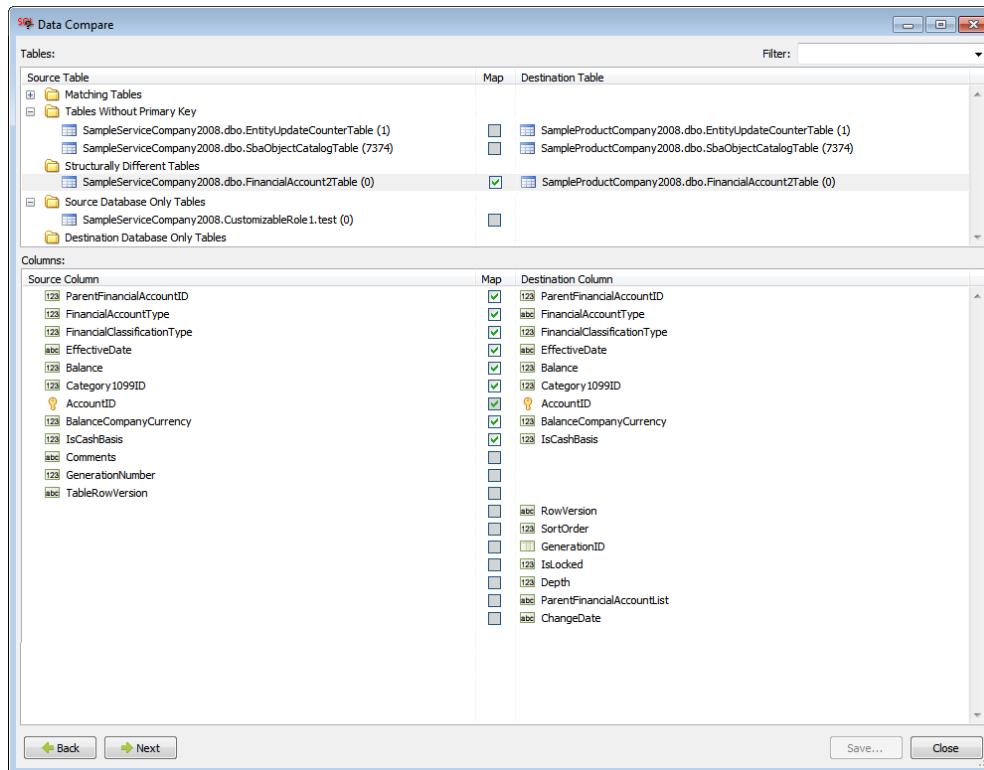
The tables in the top part of the dialog are grouped in several folders:

- **Matching Tables** – Tables referenced in this folder are structurally matching at the column and data type levels in the compared databases and ready for the data comparison. All tables in this folder are automatically preselected for the data comparison operation.

- **Tables Without Matching Primary Key** – Tables referenced in this folder do not have primary keys and unique indexes, and as result, their key columns cannot be matched automatically. If you need to compare the data, please map the key columns for each table in this group that you want to compare.

- **Structurally Different Tables** – Tables referenced in this folder are structurally different at the column or data type levels. However you can still select them if there is a subset of matching columns that you wish to compare.

- **Source Database Only Tables** – Tables referenced in this folder do not exist in the destination database. They can be compared only if you manually select the matching table in the destination database.

- **Destination Database Only Tables** – Tables referenced in this folder do not exist in the source database. They can be compared only if you manually select the matching table in the source database.

Matching Differently Named Tables

1. Select a table in **Source Database Only Tables** or **Destination Database Only Tables** folder.

2. Right-click the selected line and then in the context menu select **Map To** menu branch.
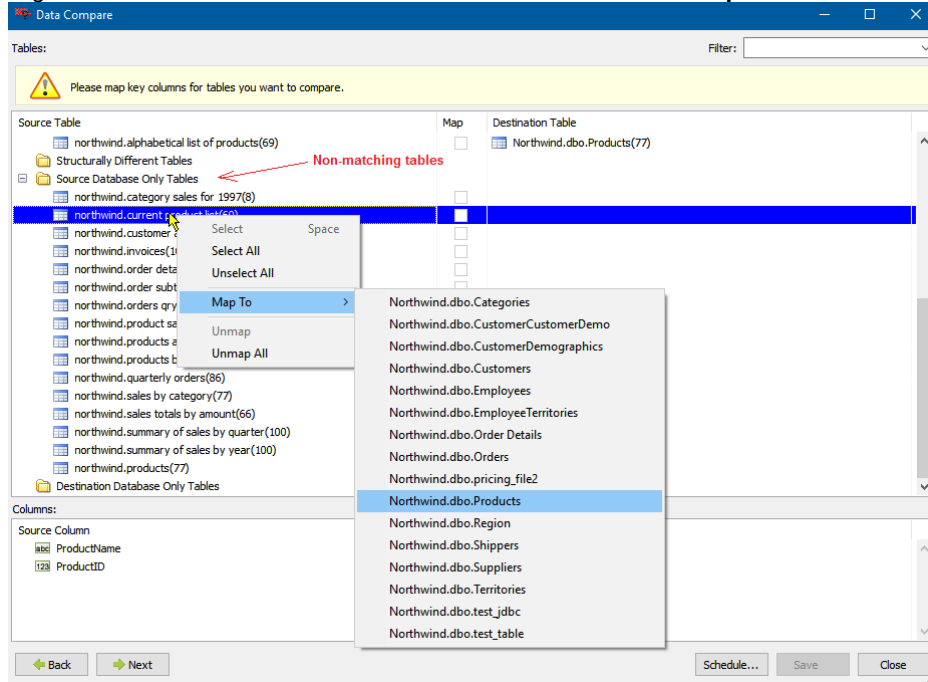
3. Select a table from the other database schema to compare against the selected table.

   Note: Depending on the state of the matched tables 9has primary keys or not, same structure or different), the matched pair is moved to one on of the three folders above for the matched tables.

Defining and Matching Key Columns

Matching of physical or logical unique keys is required for the data comparison operations. Tables having primary or unique keys with the same definitions have their keys matched automatically. All other tables require that you logically define and match their keys.

To define the keys:

1. Select a pair of matched tables. The bottom part of the dialog will show side-by-side view of table columns for the selected tables.

2. Right-click the columns that can be used as unique keys and from the context menu select **Key Columns** menu item. A key column icon will appear in front of the column name indicating its logical key status.

3. Using right-click menu or drag-and-drop method match the source and destination columns that you want compare.

   Note: It's required that all key columns are matched in order for the comparison to proceed further. Matching all other columns is optional.

Data-Types Handling and Compatibility

The Data Compare utility supports comparing data stored in different database systems powered by different types of database engines. It can also compare data stored in columns having different data-types as long as their base data-types are compatible or can be converted to compatible data-types. For example, all numeric data types are compatible for the data comparison, values from an INTEGER column can be compared to values in a FLOAT column. The following rules are used:

- Table columns having numeric data types can be compared to table columns having the same or any other numeric data-type. This includes INTEGER, BIGITEGER, FLOAT, DOUBLE, SMALLINT and other numeric data types.

- Table columns having string data types can be compared to table columns having the same or any other string data-type. Automatic Unicode to ANSI conversion is used internally to compare NVARCHAR to VARCHAR, NCHAR to CHAR and other double-byte data-types. . In addition, certain other not string data-types are treated as compatible after their values are converted to string data-types containing string representations of the original values. This include XML, UNIQUEIDENTIFIER, TEXT and some other types.

- Table columns having date-time and related data types can be compared to table columns having the same or any other date time based data-type. This include DATE, DATEIME, and TIME data-types.

- Table columns having BLOB and IMAGE data-type values can be compared to similar data-types. The comparison is done at the byte level.

Data Comparison Results

The last step in the Data Compare dialog is where all comparison results are shown. The dialog features a classical master-detail display of the results. It is split into 2 parts. The top part lists compared tables. The content of this list can be filtered to reduce the clutter and show only relevant objects. The bottom part shows rows of data from the tables selected in the top part list. The rows list can be filtered too to show non-matching rows only.

The following example screenshot demonstrates how comparison results are shown and provides some helpful tips for how to read them.

![Usage icon] **Usage:**

- The table filter controls and options provided at the top of the top part of the Data Compare dialog can be used to filter the table list. To quickly limit the list to non-matching tables only, select the **Show non-matching tables** option. To show the complete list, select the **Show all** option. To filter the name by names, type the filter string into the **Filter** combo-box, or select one of the previously entered filters from the drop-down list. Note that the filter string is applied on top of the "show…" filters.

- Numbers in parenthesis after table names indicate total number of records in the compared tables.

- The resizer control between the top and bottom parts can be used to adjust the size of each part.

- The table filter controls and options provided at the top of the bottom part of Data Compare dialog can be used to filter the compared rows. To quickly limit the list to non-matching rows only, select the **Show non-matching rows only** option. To show the complete list, select the **Show all rows** option.

- The numbers after the **Show all rows** option indicate number of rows in the compared source and destination tables. The row numbers are separated by **<– >** symbol.

- The numbers after the **Show non-matching rows only** option indicate number of no-matching rows broken by non-match type. the first number indicates number of rows found in the source table only, the second number indicates number of non-matching rows with the same primary key found in both tables, and the third number indicates number of rows found in the destination table only. The 3 row numbers are separated by **<– >** symbols.

- Data from both the source and the destination tables are displayed within the same rows using pairs of columns. Column from the source table are indicated by the "src' suffix in the column headers. Columns from the destination table are indicated by the "dst" suffix in the column headers.

- The **Sync** button can generate and execute the data synchronization SQL script.

- The **Report** button can print the Comparison Report which is a summary report for all found differences. The report can be printed to PDF file or saved to an editable Excel file.

- The **Save** button can save all data comparison results in an XML file, which you can keep for your records and share with other people in your organization. This XML file can be opened in Microsoft Excel and in other programs for further analysis of the comparison results.

- The **Save** button can also save the data difference synchronization SQL script which you can execute later to synchronize the differences

- For your convenience different types of records in the comparison results are grouped into 3 folders:

  ° **Matching Keys** – This folder contains rows with matching primary keys rows

  ° **Source Only** – This folder contains rows with primary keys found in the source table only.

  ° **Destination Only** – This folder contains rows with primary keys found in the source table only.

Color Coding

SQL Assistant uses color coding for visualizing the differences in the data. Color coding provides a convenient way to see the comparison results at a glance. Non-matching values are displayed in pink-colored cells. All other cells have either white or light gray background

Reducing the Clutter, Hiding Columns without Differences

It is very common for compared tables with many columns to have their differences in a small subset of the columns only. In case you are only interested in seeing the columns with differences you can easily hide all other columns.

In the data comparison results right-click anywhere in the comparison results grid.

In the right-click menu choose the **Hide Equal Columns** menu..
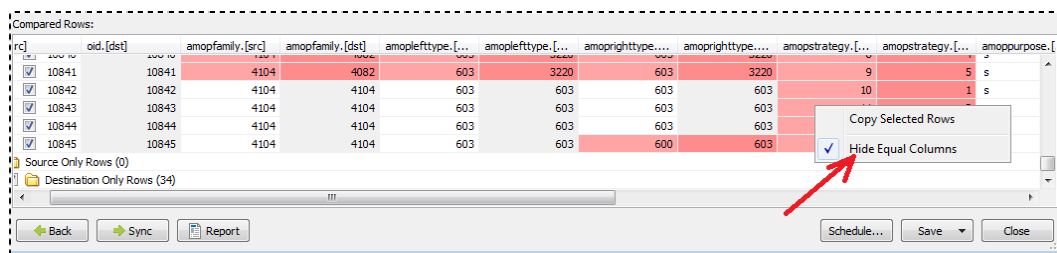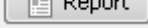
Printing Comparison Results Summary Report and Exporting it to Excel and

PDF

The Summary Report can be used for:

- Printing high level database, schema, and table-level results and/or saving them as PDF versions.

- Documenting database changes.

- Sharing results with coworkers who do not have the SQL Assistant software installed and cannot use the Data Compare tools.

- Using Excel interface applying advanced filtering, sorting, and commenting which are important tools in the decision making process for which changes to promote or ignore.

Printing and saving comparison report is a two-step process. First you have to use the [Report] button shown in the Comparison Results view to display comparison results in a tabular format. After that use the toolbar buttons at the top of the table to generate a printable version of the reports or to save the table to Excel file.

The **Print** button generates printable version of the report that includes report header with the database connection details, selected comparison options, etc.. and sends it to a printer of your choice.

The **Preview** button displays printable version of the report on the screen.

The **Save as PDF** button saves printable version of the report to a PDF file.

The **Save as XLS** button saves tabular version of the report to editable Excel file. Unlike other report actions, this output format is not optimized for printing. The report data is exported to Excel so that you can manipulate it as needed and then save or print only what you want.

To return back to the graphical **Comparison Results** view, click the **Back** button at the bottom of the report table.

To advance to the **Data Synchronization** step, click the **Next** button.

Saving Complete Comparison Results to External Files

You can use this feature to save row-level comparisons results to XML and CSV files. To save the result,

When in results view, click the **Save** button at the bottom of the **Data Compare** dialog. The context menu will appear.

In the context menu choose either **Save as XML..** or **Save as CSV…** command. The file dialog will appear.

Choose the destination file name, and the then click the Save button in the file dialog.
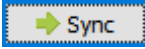
Copying Sample Comparison Results to Clipboard

When investigating data differences it might be useful to copy some data to the SQL editor in use it in your queries.

In the data comparison results select the cells with the data you want to copy.

Right-click the selection and choose **Copy Selected Rows** from the right-click menu.

Synchronizing Data in Destination Tables

1. After you review the reported differences, choose the tables that you want to synchronize.

2. Click the ➡ Sync button to generate and execute the data synchronization SQL script.

Resizing Content

To resize the Data Compare dialog window, drag the resizer handle in the bottom-right corner of the window. See the screenshot at the beginning of the Working with SQL Assistant Popups topic for information on where to locate the resizer handle.

To adjust size of top and bottom parts within the dialog, drag the vertical split bar separating table and row lists.

Opening and Saving Projects

To open an existing data compare project for modification or rerun

1. In the Data Compare dialog, click the **Menu** drop-down in the right-top corner of the dialog
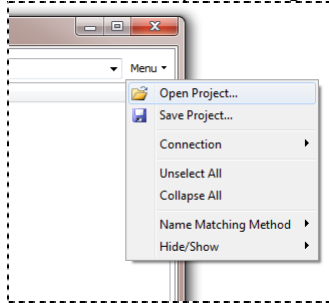
2. Click the 📁 **Open Project…** command. The **Open Data Compare Project** dialog will appear.

3. Select the project file you want to open, then click the **Open** button. The Data Compare dialog will be

populated with previously selected database connections and project settings.

To save initial project settings in a project file:

1. In the Data Compare dialog, click the **Menu** drop-down in the right-top corner of the dialog

2. Click the 💾 **Save Project…** command. The **Save Data Compare Project** dialog will appear.

3. Select the project file you want to save project settings to, then click the **Save** button. It is a good idea to use "project" or similar prefix or suffix when naming project files in order to distinguish them from other XML files.

🦎 **Note:** As you go through the options selection steps, then table matching, and then column matching steps, you can use the **Save** button displayed at the bottom on the Data Compare dialog window to save the current settings.

Scheduling Automated Data Comparisons

The **Schedule** button on the Data Compare dialog enables you to automate periodic data comparison operations or to schedule data comparison runs at night or other quiet times when the database server is not very busy. This opens the Schedule dialog providing graphical interface to the data comparison command line interface described in the next topic. For information on how to manage scheduled tasks in SQL Assistant, see CHAPTER 50, Managing Scheduled Tasks

The scheduled task uses data comparison command line interface described in the next topic to run the comparison at the specific date and time.

Command Line Interface

To run data comparison operations from a DOS command line window, use the following command:

```
sacmd dc srcconn:"src-connection-name" dstconn:"dest-connection-name" srcname:"src-object-
path" dstname:"dest-object-path" outfile:"file-name" dcflags:"flag-chars" sas:"path-to-sa-
settings-file"
```

The above command must be entered as a single line

Substitute values in the command as follows:

src-connection-name	The database connection name for the source database connection
dest-connection-name	The database connection name for the destination database connection
src-object-path	The fully qualified dot separated path to the source table, schema or database. This also defines the scope of the comparison. If you specify just **database** name, then all tables in the database will be compared. If you specify **database.schema** name, then all tables in the specified schema will

be compared. If you specify **database.schema.table** name , then only the specified table will be compared

dest-object-path

The fully qualified dot separated path to the target table, schema or database. This also defines the scope of the comparison. If you specify just **database** name, then all tables in the database will be compared. If you specify **database.schema** name, then all tables in the specified schema will be compared. If you specify **database.schema.table** name , then only the specified table will be compared.

 Important Notes:

- The source and destination path must have the same scope.

- For database servers without database element in the path, such as Oracle, DB2, MySQL, SQLite, and others use **noDB** name in place of the database name

file-name

The output file in which the comparison results will be saved in XML format.

flag-chars

Combination of flags for the data comparison. The following symbols can be used in this parameter:

      s - Ignore system-generated columns
      b - Ignore BLOB, IMAGE and similar large data columns
      c - Use case-sensitive object and column name matching
      t - Ignore trailing spaces
      n - Treat empty string values as NULLs

path-to-sa-settings-file

The full file name of the SQL Assistant settings file containing the required database connection parameters. This is an optional parameter. If not specified, the default path for the current user account is used.

Example:

```
cd "C:\Program Files (x86)\SQL Assistant 12"

sacmd dc srcconn:"DEV001 (sa)" dstconn:"PROD (sa)" srcname:"AdventureWorks.Sales"
dstname:"AdventureWorks.Sales" outfile:"C:\TEMP\DataCompareResults.xml" dcflags:"sbt"
sas:"%APPDATA%\SQL Assistant\13.0\sqlassist.sas"
```

 Important Notes:

- The SQL Assistant settings file location is version and user profile specific. See the Notes in the Overview topic in CHAPTER 51 for details on how to find out the location of that file.

- You can find out the connection name in the DB Connections group of settings on the DB Options tab page in SQL Assistant Options. If a connection requires a user id and password, make sure that both are saved in the settings. The command line interface does not display interactive prompts and is unable to prompt for credentials during command processing. For more information about storing and managing database connections, see the Managing Database Connections topic in CHAPTER 48.

CHAPTER 27, Schema Compare Utility

Overview

SQL Assistant provides advanced fully customizable Schema Compare utility. The Schema Compare utility provides a quick and easy way to compare structures and attributes of databases and database schema objects, identify and display their differences. It highlights objects and attributes that were modified, including exact lines of modified code. It generates the required database and schema synchronization script can be used to propagate schema changes from one environment to another.

The Schema Compare supports comparing schema differences between the following target types:

- Two objects in the same or different schemas residing on the same or different servers.

- All objects in two schemas residing on the same or different servers.

- All objects in all non-system schemas in two different databases residing on the same or different servers.

- All objects in all non-system schemas in all non-system databases residing on the same or different servers.

You can compare schema differences in two databases of the same or different types. For example, you can compare a schema in PostgreSQL database against a schema in MySQL database. It also supports smart cross version comparison, for example, you can compare a schema in SQL Server 2008 against a schema in SQL Server 2012 automatically ignoring version specific differences and comparing only schema objects and attributes supported by both versions. In case of comparing schema differences in two databases of different types, the data types of table columns and procedure/function arguments are compared based on their compatibility rather than names, for example, a column with BYTEA data type in PostgreSQL database is considered compatible with a column having VARBINARY(MAX) data type in SQL Server database, a column with VARCHAR(100) data type in MySQL is considered compatible with a column having VARCHAR2(100) data type in Oracle database, and so on...

The Schema Compare utility supports flexible comparison filters enabling you to choose to include or exclude particular object types, particular object attributes and properties, to choose to ignore minor differences, such as differences in comments, extra spaces, or ignore attributes with database-wide default values that are different in different databases, and so on...

Note: The Schema Compare utility supports scriptable interface comprised of comparison rules, database queries, and templates for schema synchronization. It enables users to extend the existing functionality, adding new features for supporting new object types and properties and altering behavior of pre-configured comparison features. See the Extending and Customizing Schema Compare Functions topic for more details on customizing the Schema Compare.

The Schema Compare utility can be invoked from SQL Assistant's menus. The **Compare Code and Data** ➔ **Compare Schema** command is available from either the right-click context menu in the target editor or the top-level menu (if the target editor top-level menu integration is enabled). This command opens the **Schema Compare** dialog. In this dialog you can select the target servers, databases, schemas and objects to compare. The following topics describe the usage of the Schema Compare utility.

Tip: If you have a need to compare current schema design against a snapshot saved in the source code repository, use the Repository Browser dialog for this purpose. For example, you can use **Compare Database to Repository** command described in the Advanced 3-Way Code Comparison and Synchronization topic in CHAPTER 23, Database Source Code Control Interface.

How Schema Comparison Engine Works

The schema comparison engine implements rule based database catalog data loading and comparison. It uses scriptable templates similar to code snippets for generating schema change synchronization scripts. Users can add and modify rules as needed for handling new comparison widgets and for managing sets of comparable attributes of the existing widgets.

The schema comparison engine is not limited to schema objects only. It can be used to compare database scope and server scope features as well as user-defined widgets.

The engine uses regular SQL queries to retrieve the required database catalog data. The result sets of the queries define comparable sets of objects and their attributes. Each query is associated with a single object type. There should be only one query for each type within a single rule set, with the exception of overloaded queries used for different database server versions. However, the same query can be used to return data for multiple object types. For example, a single query can be used to retrieve table and view properties. The "Target Type" column in the result set can return either TABLE or VIEW value. It is important to note that the query structure is flexible. Different queries can contain different sets of columns depending on the object type and other factors. See The Anatomy of Queries topic in this chapter for more details.

The following image demonstrates how query definitions drive schema comparison behavior and visual results
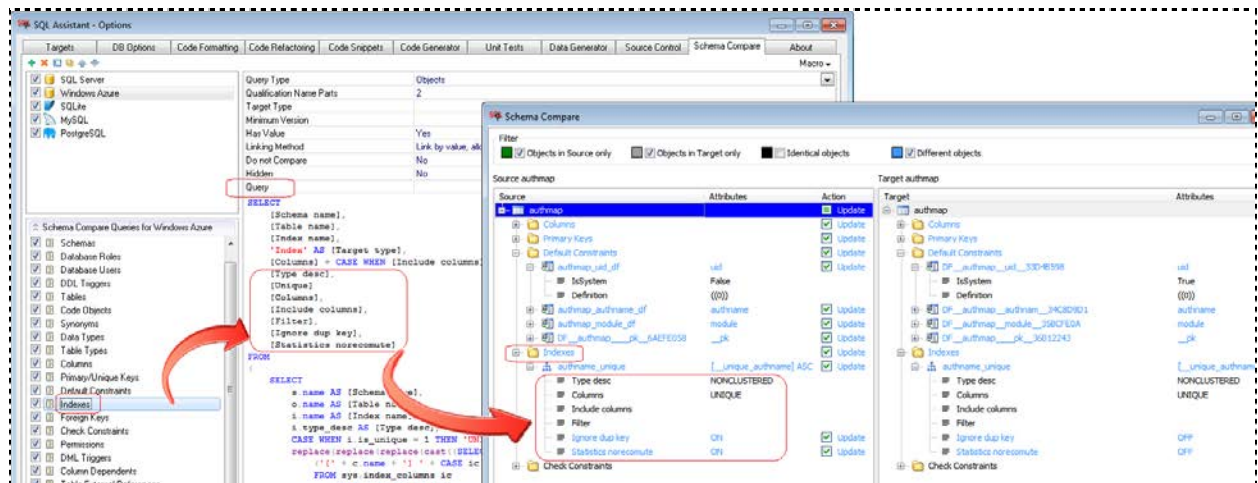
The queries control both the types of objects that can be compared and the comparable object attributes. The schema comparison process is fully dynamic; it can be extended using new user-defined queries.

To support multiple database server versions, the engine allows defining so called overloaded queries. The overloaded queries are associated with the same object type but their code is tweaked for different database server versions, not server types. For example, within the same rule set you can define different queries for SQL Server 2008 and for SQL Server 2012. The "Minimum Version" query property indicates the version of SQL Server the query is compatible with.

The schema engine uses scriptable templates to generate database and schema change synchronization scripts. Different templates are used for different object types. Just like queries templates can be overloaded for different versions of database servers. Several templates can be associated with the same object type to handle different types of changes such as ALTER, RENAME, CREATE, and DROP operations. The type of the template is defined by the Template Type template property. Templates can reference nested templates for handling repetitive elements such as, for example, columns within the CREATE TABLE template are handled by separate template. In this example the template named "Table (CREATE)" is invoked for the CREATE TABLE script generation and that template internally calls the "Column (CREATE)" template for each column defined in the table being processed and then similarly it calls the "Foreign Key (CREATE)" template for each foreign key constraint, and so on. The template code defines the output script syntax and the nested templates that it needs to call during the script generation process. Here is a sample template code for MySQL specific rule set for column change synchronization template named "Column (ALTER)":

```
ALTER TABLE `$SCHEMA_NAME$`.`$OBJECT_NAME$` MODIFY COLUMN $|=COLUMN DEFINITION$;
```

When processing this template code the comparison engine performs the following steps:

3. The simple value macro $SCHEMA_NAME$ is replaced with the target schema name.

4. The simple value macro $OBJECT_NAME$ macro is replaced with the target table name.

5. The last part $|=COLUMN DEFINITION$ is a reference to the nested template named "Column Definition". This part is processed separately and then the output of that nested template is inserted into the output of the "Column (ALTER)".

The referenced nested template "Column Definition" has the following code:

```
`$NAME$` $TYPE$ $CHARACTER SET$ $COLLATE$ $ISNULL$ $DEFAULT$ $AUTO INCREMENT$
$COMMENT$
```

In this template code the macros between $..$ signs are references to attribute values populated in the "Columns" query. In other words, NAME, TYPE, CHARACTER SET, COLLATE, ISNULL, DEFAULT, AUTO INCREMENT, COMMENT are column names defined in the result set of the "Columns" query for MySQL.

The template code also supports basic conditional logic for implementing control of flow code generation. For example, the following sample "Index (CREATE)" template will produce code for synchronizing index allocation, storage, and statistics computation options only if they have been selected for comparison.

```
CREATE $IS_UNIQUE$ $TYPE_DESC$ INDEX [$NAME$] ON [$SCHEMA_NAME$].[$OBJECT_NAME$]
($COLUMNS$)
$|=INDEX INCLUDE$ $FILTER$
$|=INDEX OPTIONS$
```

To support multiple database server versions, the engine allows defining so called overloaded templates. The overloaded templates are associated with the same object type but their code is tweaked for different database server versions, not server types. For example, within the same rule set you can define different templates for SQL Server 2008 and for SQL Server 2012. The "Minimum Version" template property indicates the version of SQL Server the template is compatible with.

For more details on templates read The Anatomy of Templates topic in this chapter.

Read Extending and Customizing Schema Compare Functions topic later in this chapter for more details on schema comparison rules and internal organization, and for specific examples for how to customize the engine working.

The Schema Compare Dialog

Navigation

The Schema Compare dialog is a wizard-like dialog that guides you through a 3-step schema comparison and synchronization process:

- Step 1: Choose comparison scope, database connections, and comparison options.

- Step 2: Run the comparison engine. Review the comparison results. Optionally, filter results in place; Save of print comparison reports to PDF files or editable Excel files.

- Step 3 (Optional): Review, edit, save, and execute the schema synchronization script.

Use the **Next** button at the bottom of the dialog to advance to the next step. Use the **Back** button to go back to the previous step. If you click the **Back** button while there is an active comparison operation in progress, the operation is automatically aborted.

To close the dialog, at any time click the **Close** button in the right bottom corner.

Comparison Scope and Options

Scope – The scope of the schema comparison. One of the following:

- **Object** – Select this option to compare a single schema object against another object of the same type residing on the same or different database server.

- **Schema** – Select this option to compare all objects in a database schema against another schema residing on the same or different database server.

- **Database** – Select this option to compare all schema objects in a database against another database residing on the same or different database server.

- **Server** – Select this option to compare all schema objects in all databases against another database server.
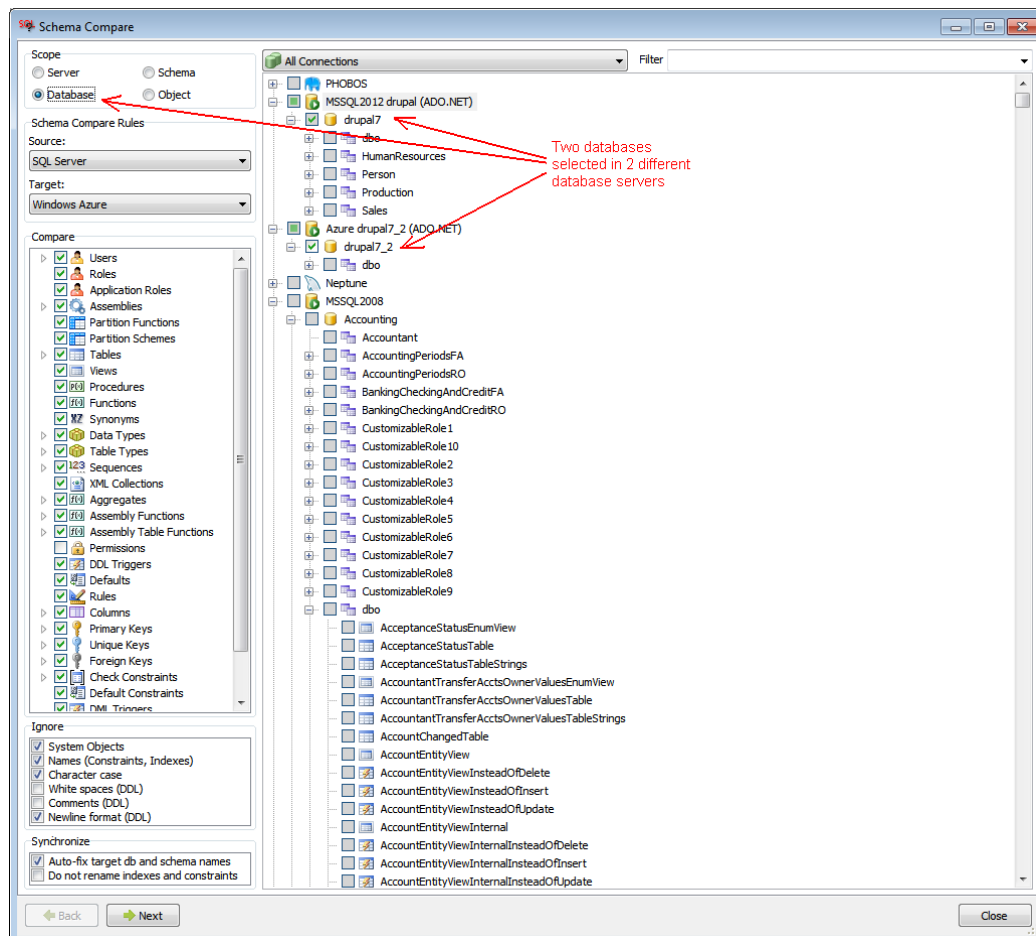
Schema Compare Rules – The set of comparison queries and templates designed for specific database server type and version that will be used for the selected Source and Target items. Typically the rule names are matching database server types but in fact they are just names and can be anything you like. It is important to choose rules that match your database type and version in order to produce correct comparison results and generate valid schema synchronization scripts.
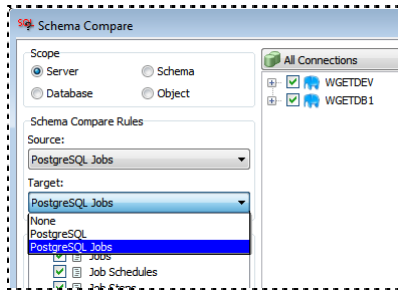
Compare options tree –The collection of database and schema object types and their attributes available for comparison. In the options tree you can choose which types of schema objects and their attributes you want to compare. If you want to ignore certain object types and/or some of their attributes, for example, fill factor for indexes and primary keys, unselect them in the **Compare** list as demonstrated on the following screenshot.
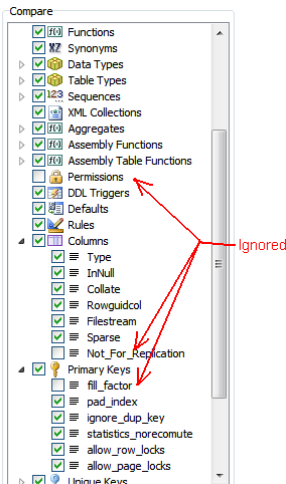
Tip: The content of the **Compare** tree depends on the **Schema Compare Rules** selected. Certain item types and attributes appear deselected by default as defined by the default options. To select/deselect any item, tick or un-tick the checkbox in front of the item. To select / unselect individual types of attributes, for example, to ignore index fill factor when comparing indexes in SQL Server databases, expand the Indexes item and un-tick the fill_factor item.

Tip: The default options drive what is initially selected for comparison in the **Compare** tree. If you often need to change the comparison options, you can modify their defaults to your liking in the SQL Assistant Options dialog. See Customizing Default Comparison Options topic for more details.

Ignore options

In the **Ignore** box you can choose generalized options that you want ignored for all applicable schema objects during their comparison. The following options are supported:

System objects – ignore all system schema objects, system schemas, and system databases.

Names (Constraints, Indexes) – ignore names of constraints and indexes. Compare their columns and attributes only. This option is helpful when working with system generated constraint and index names that vary in every schema.

Character case – ignore character casing, in other words, if checked, apply case insensitive comparison.

White spaces (DDL) – ignore leading and trailing white spaces when comparing procedural objects such as stored procedures, triggers, and so on.

Comments (DDL) – ignore differences in comments when comparing procedural objects such as stored procedures, triggers, and so on.

New line format (DDL) – ignore differences in line ending symbols when comparing procedural objects such as stored procedures, triggers, and so on.

> **Tip:** Line endings can differ on different systems, for example on Windows system text line typically end with a pair of CR+LF characters, while on Linux systems they typically end with a single LF character.

Synchronize options

Auto-fix target db and schema names – when comparing differently named databases or schemas, allow SQL Assistant to automatically replace source database and schema names with target names in the generated synchronization scripts.

Do not rename indexes and constraints – this option is used in conjunction with **Ignore Names (Constraints, Indexes)** option. When comparing indexes and constraints definitions and ignoring their names (typically system generated names), if definitions match, do not rename target constraints and indexes to match the source names.

Selecting Servers, Databases, Schemas, and Objects for Comparison

Connection and Name Filters

Multiple server connections are organized into logical connection groups. You can use the right-click menu to manage connections in place. To manage connection groups use either the SQL Assistant main Options dialog or the Multi-server Code Execution utility. For more information on managing connection groups and connection settings see Managing Connection Groups and Connection Settings topic in CHAPTER 15.

The Filter combo-box offers super-fast content filtering. Type the substring you want to use as a filter for database objects into the Filter box available above the object tree. Previously used filters are available in the drop-down portion of the Filter combo-box.

Tip: You can add new connections directly in the same dialog. Use the right-click menu in the connection tree. The same right-click menu can be used to modify saved connections.

Navigation

All server connections, databases, schemas, and schema objects are displayed in a single Object Tree. You can select only 2 items of the same type. The type must match the selected comparison **Scope** option.

Object Tree Legend

The following types of checkboxes could be displayed in the object tree.

☐   Item is not selected for schema comparison. This item is available for selection.

☐   Item is not available for selection because the item scope does not match the **Scope** option or you already have two items selected for comparison. In the latest case, If you want to change the selection, first unselect one of the selected items and then select another item.

☑   Item is selected for schema comparison.

▣   For an expandable schema, database or server item, this type of checkbox indicates that a "child" item has been selected in the object tree branch for this item.

Opening and Saving Projects

To open an existing schema compare project for modification or rerun

4. In the Schema Compare dialog, click the **Menu** drop-down in the right-top corner of the dialog
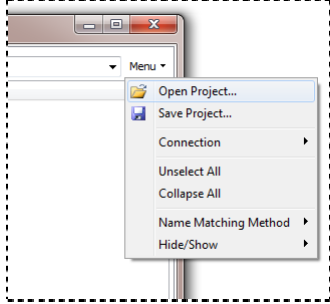
5. Click the 📁 **Open Project…** command. The **Open Schema Compare Project** dialog will appear.

6. Select the project file you want to open, then click the **Open** button. The Schema Compare dialog will be populated with previously selected database connections and project settings.

To save project settings in a project file:

4. In the Schema Compare dialog, click the **Menu** drop-down in the right-top corner of the dialog

5. Click the 💾 **Save Project…** command. The **Save Schema Compare Project** dialog will appear.

6. Select the project file you want to save project settings to, then click the **Save** button. It is a good idea to use "project" or similar prefix or suffix when naming project files in order to distinguish them from other XML files.

Schema Comparison Results

The second step in the Schema Compare dialog is where all comparison results are shown.

The following example screenshot demonstrates how comparison results are shown.

The dialog features 4 parts display of the results. There are the Source server schema objects tree and the Target server schema objects tree at the top and the DDL code views of the same at the bottom. The displayed items can be filtered using the predefined or custom filters at the top of the dialog to reduce the clutter and show only relevant differences. The DDL code differences navigation map is called change map, and it is displayed adjacent to the right side of the target object DDL code. For procedural objects there is also a separate tab showing logical differences in the programs using

Usage:

- The filter options available in the **Filter** box at the top of the Schema Compare dialog can be used to filter the comparison results. For specific details on the usage of the filter options see the following Filtering Results topic.

- The resizer control separating the the Source tree and Target tree parts can be used to adjust the size of each part.

- The resizer control separating the tree and the bottom DDL code comparison results can be used to adjust the size of top and bottom parts.

- Use standard tree view control navigation methods to collapse, expand, and navigate items in the Source schema object and Target schema object trees.

Color Coding

SQL Assistant uses color coding in the Source and Target schema object trees for visualizing the differences. Color coding provides a convenient way to see the comparison results at a glance. For your convenience the color legend is displayed at the top of the Schema Compare dialog in the **Filter** options box in front of the filter selection checkboxes.

It also use color coding for highlighting code differences in the Source and Target DDL code boxes. The color coding method and color schema are the same as in the integrated Code Compare utility. See Color Highlighting topic in CHAPTER 25, Code Compare Utility for more details.

Action Legend

The following types of checkboxes could be displayed in the **Action** column.

☐   Item is not selected for schema synchronization. This item is available for selection. If not selected, it will not be included in the change synchronization script.

▦   Item is not available for selection. This item is identical in the Source and Target databases and no synchronization is required.

☑   Item is selected for schema synchronization. The type of action CREATE, DROP, or ALTER is indicated in the Action column.

◼   For an expandable schema, database, or server item, this type of checkbox indicates that a "child" item has been selected in its tree branch for schema synchronization.

Resizing Content

To resize the Schema Compare dialog window, drag the resizer handle in the bottom-right corner of the window. See the screenshot at the beginning of the Working with SQL Assistant Popups topic for information on where to locate the resizer handle.

To adjust size of top and bottom parts within the dialog, drag the vertical split bar separating table and row lists.

Visual Comparison Results for Procedural Objects

In addition to textual comparison of DDL code differences, when procedurals objects are selected in the object tree, two additional tabs appear at the bottom of the Schema Compare dialog window:

- Logical Differences'

- Flow Differences

The Visual Code Compare utility is used to visualize the differences. See Visualizing Code Differences topic in CHAPTER 25, Code Compare Utility for more information.

Filtering Comparison Results

The **Filter** box provides multiple options for filtering the comparison results.

Category Filters

Use the category checkboxes to choose which groups of objects and their properties to show

Filter

☑ Objects in Source only   ☑ Objects in Target only   ☐ Identical objects   ☑ Different objects

For example, to quickly limit the list to new schema objects only, un-tick all checkboxes and tick only the **Objects in Source only** checkbox. To show the complete list, tick all checkboxes.

Name and Attribute Filters

The **Text** and **Filter Type** controls enable filtering comparison results by schema object names, by their attributes names, or by their attribute values.

For example, to filter the list to objects whose names contain substring "EMP", type EMP in the **Text** edit box and select "Schema object names" option in the **Filter Type** drop-down

To filter the list to show only tables or views having column names containing substring "EMP", type EMP in the **Text** edit box and select "Property names" option in the **Filter Type** drop-down.

The following **Filter Type** options are supported:

- **Schema object names** – filtering is applied to the schema object names level in the source objects and target objects trees. The text entered in the **Text** box is used as a name matching filter condition.

- **Property names** – filtering is applied to all kinds of attributes appearing below the object name level in the trees, including table and view column names, procedure and function parameters, type fields, and so on… The text entered in the **Text** box is used as a name matching filter condition.

- **Schema object and property names** – this works just the two options above combined.

- **Custom object filter** – complex expression based filtering rules are applied to object and attribute names. The logical **expression** entered in the **Text** box is used as a filter condition. The **expression** can refer not only to object names but also to object types, attributes names and their values. It can contain functions, regular expressions, and multiple logical conditions joined by logical AND, OR, and NOT operators as described below.

Custom Filter Expressions

If you choose **Custom object filter** option, you can specify a Boolean expression in the **Text** box that may contain complex name references, functions, comparison operators, and so on. For example, to hide all objects whose name contains _BAK and _OLD suffixes, specify

(Pos('_BAK', Name) = 0) and (Pos('_OLD', Name) = 0)

In the above example, the Pos function returns 0 if the value in the first argument cannot be found in the value of the second argument. The filter is satisfied if both _BAK and _OLD cannot be found in object names.

In the expressions you can refer to the following special elements

Name: string – the Name refers to object and attribute name. For example, to hide table DEPARTMENT_BAK_20161204 in the results, enter filter expression like
Name <> 'DEPARTMENT_BAK_20161204'

Value: string – the Value refers to attribute values. For example, to filter all elements with attribute value is equal USERSPACE1, you can use filter like
Value = 'USERSPACE1'

ObjectType: string – the ObjectType refers to the type of the object, for example, to show objects that are not

tables and having names containing EMP substring, enter filter expression like
(ObjectType <> 'Table') and (Pos('EMP', Name) > 0)

Level: integer – the Level refers to item level as it is displayed in the source and Target object trees in the results. For example, to show only second and third level objects, enter filter expression like
(Level = 2) or (Level = 3)

Attribute(aname: string) – the Attribute refers to the value of a particular object attribute. The actual attribute name must be specified as an argument in the Attribute reference. For example, to show only tables stored in USERSPACE1 table-space, enter filter expression like
Attribute('Table space') = 'USERSPACE1'

Tip: The applicable attribute names are data base type specific**.** To find out attribute names that can be used with the Attribute element, see column and expression names returned by the associated Schema Compare queries that can be found in the SQL Assistant's **Options** dialog. For more information about Schema Compare queries see The Anatomy of Queries topic later in this chapter.

Within the filter expression you can use functions listed in the following topic.

Tip: You can use the **RegExprMatch** function to apply regular expression based filters**.** This function returns True or False based on the value matching status. For example, to filter out all objects containing names containing numeric suffixes at the end of their names with numbers between 2000 and 2099, you can enter an expression like
*not RegExprMatch(Name, '*20[0-9][0-9]$')*

For more information about regular expressions syntax see Using Regular Expressions topic in CHAPTER 34, Integrated SQL Editors.

Functions Supported in Custom Filter Expressions

The following Pascal like functions can be used with filter expressions

function HasAttribute(aname: string): boolean

function RegExprMatch(str: string; expr: string): boolean

function ChildrenCount(chldObjType: string): integer

function IntToStr(i: Integer): String

function FloatToStr(e: Extended): String

function DateToStr(e: Extended): String

function TimeToStr(e: Extended): String

function DateTimeToStr(e: Extended): String

function StrToInt(s: String): Integer

function StrToFloat(s: String): Extended

function StrToDate(s: String): Extended

function StrToTime(s: String): Extended

function StrToDateTime(s: String): Extended

function Length(s: String): Integer

function Copy(s: String; from, count: Integer): String

function Pos(substr, s: String): Integer

function Uppercase(s: String): String

function Lowercase(s: String): String

function Trim(s: String): String

function CompareText(s, s1: String): Integer

function Chr(i: Integer): Char

function Ord(ch: Char): Integer

function ValidInt(cInt: String): Boolean

function ValidFloat(cFlt: String): Boolean

function ValidDate(cDate: String): Boolean

Saving, Reusing, and Managing Filters

Use [icon] toolbar button adjacent to the **Filter Type** drop-down to save the current filter definition. You will be prompted to choose a name for the saved filter. The filter can be referenced later by that name.

Use [icon] toolbar button adjacent to the **Filter Type** drop-down to reopen and reapply previously save filter. This will open the **Schema Compare Filters** dialog. Choose the required filter and click the OK button to apply it.

To make additional changes before applying the filter, or to add or delete filters, tick the Edit filter checkbox available at the bottom of the dialog. The Filters grid becomes editable, and the filters management toolbar will appear on the right hand side of the Filters grid.
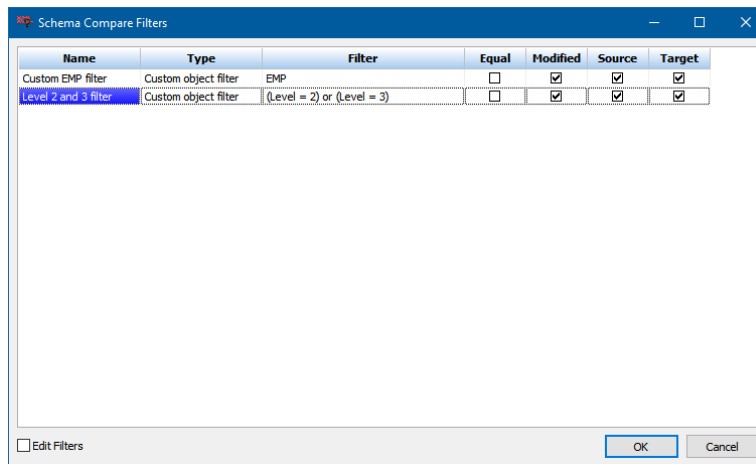
Printing Comparison Report, and Exporting it to Excel and PDF

The Comparison Report provides the following:

- Printing table scope comparison result summaries or saving as PDF versions.

- Documenting database changes

- Sharing results with coworkers who do not have the SQL Assistant software installed and cannot use the Schema Compare tools.

- Using Excel interface applying advanced filtering, sorting, and commenting which are important tools in the decision making process for which changes to synchronize or ignore.

Printing and saving comparison reports is a two-step process. First you click the [Report] button available at the bottom the Comparison Results dialog to display comparison results in a tabular format. After that use the toolbar buttons at the top of the table to generate a printable version of the reports or to save the table to an Excel file.

Important Note: The Report button is available on the last step of the comparison process when the comparison results for all objects are known.

The **Print** button generates printable version of the report that includes report header with the database connection details, selected comparison options, etc.. and sends it to a printer of your choice.

The **Preview** button displays printable version of the report on the screen.

The **Save as PDF** button saves printable version of the report to a PDF file.

The **Save as XLS** button saves tabular version of the report to editable Excel file. Unlike other report actions, this output format is not optimized for printing. The report data is exported to Excel so that you can manipulate it as needed and then save or print only what you want.

To return back to the graphical **Comparison Results** view, click the **Back** button at the bottom of the report table.

Schema Synchronization

By default all differences are included in the scope of the synchronization action. In the Schema Comparison Results you can exclude differences that you do not want to synchronize. To do so, un-tick checkboxes in the Action column on the right side of each row in the Source tree. When it is time to update the target schema, this row will not be considered for any pending changes.

Ticking or un-ticking a checkbox in a group row is equivalent to ticking or un-ticking all differences in that group. You can use this speedy selection method at any group row, for example, you can use it at the database row to quickly unselect everything and then at a specific table row to quickly select that table only.

To quickly select or deselect everything displayed using current filters in the comparison result trees, right-click anywhere in the Source tree. The context menu will appear. Choose **Select All** or **Deselect all** commands as required.

After you have selected everything you want to synchronize, click the **Next** button. SQL Assistant will generate the synchronization script for the selected schema objects and attributes and will display the resulting script. The following example screenshot demonstrates how the synchronization script may look like.
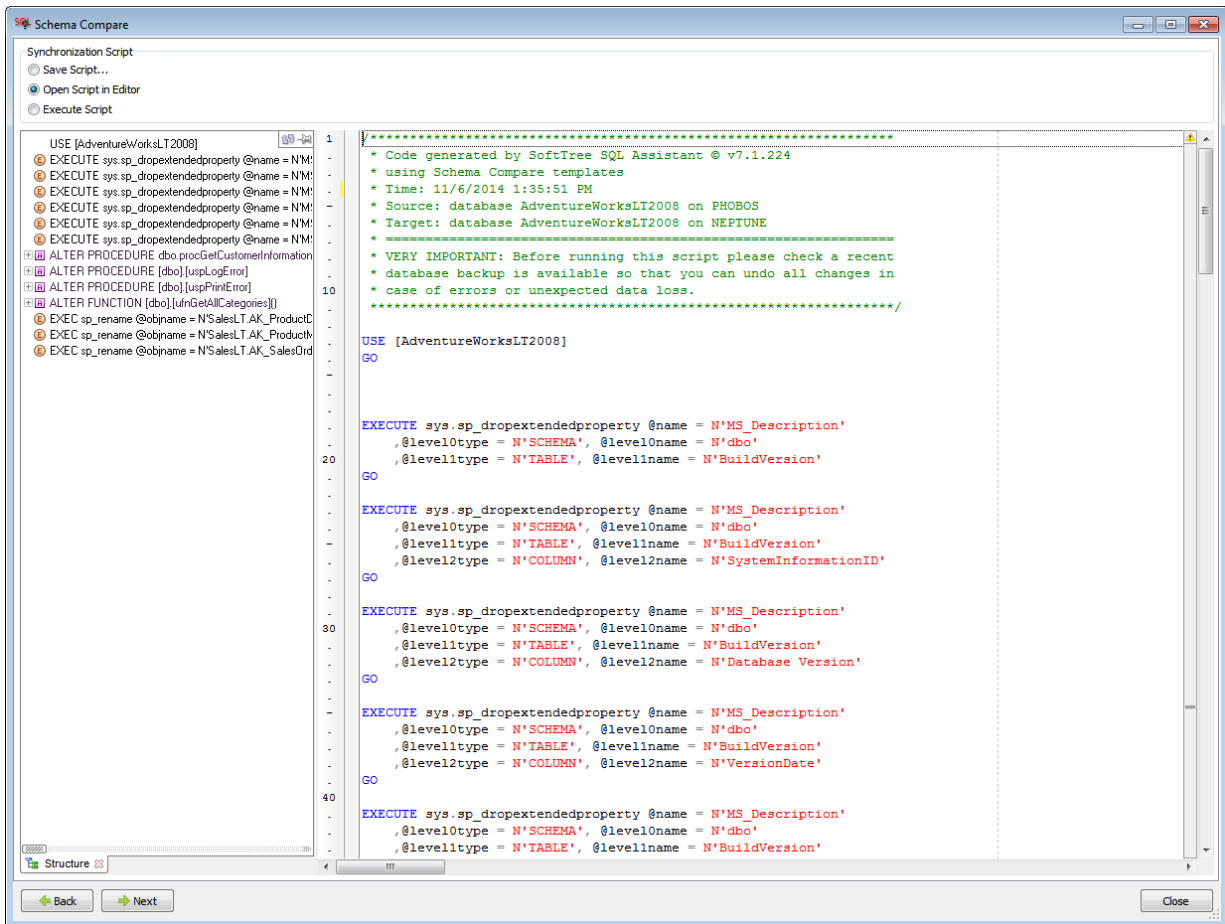
The generated synchronization script is displayed in a new code editor connected to the destination database server. Review the synchronization script and correct it if required. Note that the script syntax is validated automatically for syntax errors, but it is not checked for any logical errors or other issues that only a developer can recognize. The syntax errors are displayed on the syntax bar adjacent to the right side of the editor. For details on how to use the Syntax bar see CHAPTER 20, SQL Syntax Checker.

For your convenience the script Structure View is displayed to the left of the generated script with quick navigation link to SQL statements in the script. For instructions on how to use the Structure View see CHAPTER 4, Code Structure View and Bird's Eye View.

Three actions are available. You can:

- Save the script to a file on your computer.

- Open the script in external editor. In case if you started the Schema Compare tool from a development environment having SQL Assistant running as an add-on, the script will be opened in a new tab in the host development environment. Otherwise the script will be opened in SQL Assistant's integrated SQL Editor. See Professional SQL Editor IDE topic in CHAPTER 34, Integrated SQL Editors.

- Execute the script immediately.

Important Note: It is strongly recommended that you create a backup of the Target database before executing a schema synchronization script so that you can roll back all unwanted changes later.

Extending and Customizing Schema Compare Functions

Comparison Rules

The **Comparison Rules** are named collections of rules and various settings used for schema comparison and synchronization. The pre-defined Comparison Rules have names aligned with targeted database types so that they can be easily recognized. While it is possible to mix together settings and queries designed for different versions of the targeted database server software in a single comparison rules set, it is recommended that for manageability reasons when developing custom rules you create different collections of rules for different versions taking care of different database server features.

You can create new and customize existing Comparison Rules in the SQL Assistant's Options dialog on the **Schema Compare** tab.
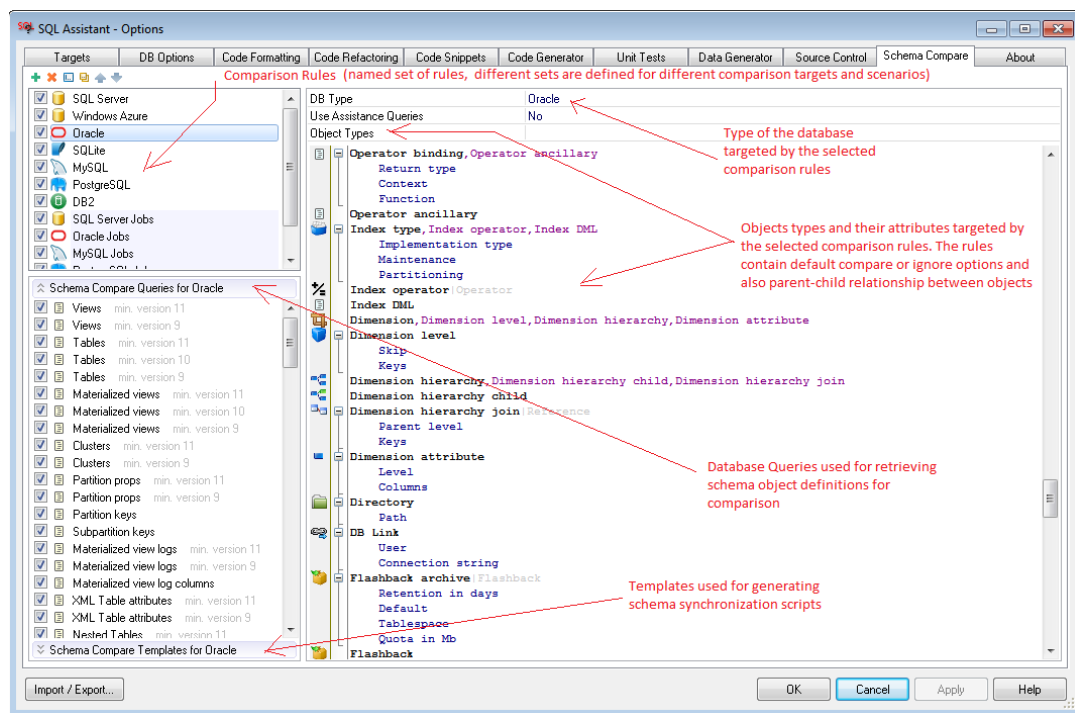
Adding, Copying, Deleting and Disabling Comparison Rules

Use the Options dialog, the **Schema Compare** tab to manage Comparison Rules and their settings

To add a new comparison rule set:

1. Right-click the **Comparison Rules** list in the top left window, and select the **Add** command. A new rule set will be added to the list with the input focus in the rule set name.

2. Enter new rule set name. In the right window, fill in property values for the new rule set.

3. In the left bottom window in the **Queries** section enter the required queries. In the right window, fill in appropriate properties for the new queries.

4. In the left bottom window in the **Templates** section enter the required templates. In the right window,

fill in appropriate properties for the new templates.

5. After you done with the **Queries** and **Templates** return to the rule set and enter the composite value of Object Types property. Specify objects types that you want to show in the list

To create a new comparison rule set from an existing one:

1. Select an existing rule set from the list box on the top left (the **Comparison Rules** list).

2. Right-click the **Comparison Rules** list in the top left window, and select the **Copy** command.

To delete a rule set:

1. Select the rule set from the list box on the top left and press the **Delete** button.

To disable a rule set:

1. In the top left window deselect the checkbox next to the rule set name.

To enable a rule set:

1. In the top left window deselect the checkbox next to the rule set name.

Note:

If a rule set is disabled, its definition remains in the SQL Assistant options, but the rules are not active and cannot be used. For more information on how to use and change SQL Assistant's options, see CHAPTER 48.

Additional Tips:

- Use the rules management icons available in the left-top corner of the Options dialog to create a new rule set or to rename, duplicate or delete code rule sets, their queries and templates.

  Note that the icon functions are sensitive to the location of the focus in the Schema Compare tab. For example, if you click the X button when a rule set is selected in the Comparison Rules window on the upper left, the selected rule set deleted entirely, including all queries and code templates included within the rule set. However, if the focus is set to the left-bottom list box containing template names and a template name is highlighted, clicking the X button deletes only the selected template.

- The content of the right side of the Schema Compare tab is also context sensitive. If a rule set is selected, the rule set properties are displayed in the right window. If a query is selected in the Queries section on the lower left, the query code and properties are displayed in the right window. If a synchronization template is selected in the Templates section on the lower left, the template code and properties are displayed in the right window.

- You can drag-and-drop rule set names in the left-top list to rearrange their order. You can use this method to push most commonly used rules to the top of the list and minimize the amount of scrolling and clicking required for customizing rules. Similarly you can drag-and-drop queries and templates in the left bottom lists to rearrange their order. However, it is important to remember that their order is very important as it controls the results of the schema comparison and code change synchronization.

Customizing Default Comparison Options

1. Open the Options dialog and select the **Schema Compare** tab.

2. In the top left window select Schema Comparison rule set that you want to customize.

3. On the right side of the dialog select required **DB Type** for the selected rule set.

4. On the right side of the dialog select **Use Assistance Queries** option. For all database servers types but SQLite and MS Access this option should be set to "No" value. For SQLite and MS Access this option should be set to "Yes" value. The "No" value means that the schema comparison engine can use SQL queries defined for the rule set in the **Queries** section. The "Yes" value means that the schema comparison engine cannot use queries, and it has to rely on the catalog data from SQL Assistant's data cache., for both database types SQL Assistance populates its data cache using non-query based programmatic interfaces to retrieve schema object definitions and attributes.

5. In the right window modify **Object Types** settings. The settings are saved in plain text format in a form of simple two-level tree having object types and their dependencies in the first level, and then their attributes as indented elements in the second level. To set a specific object type or attribute ignored by default in comparison options, prefix that object type or attribute name with a minus sign.
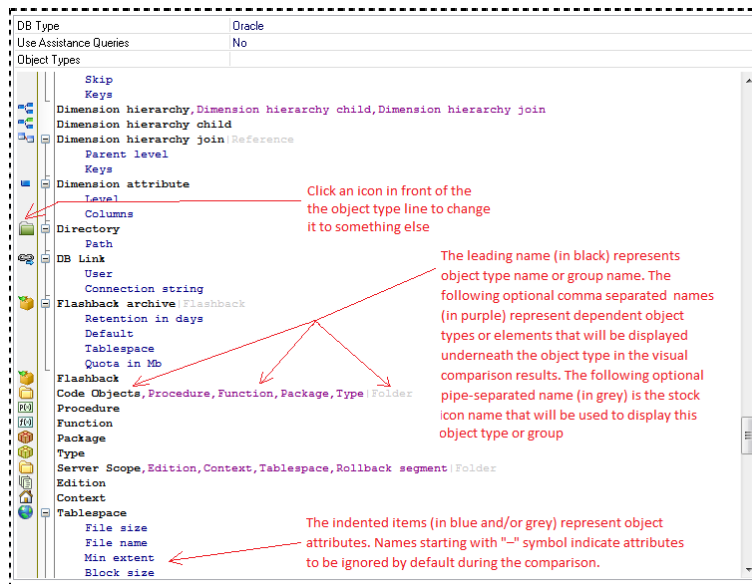
Below is an example of SQL Server specific rule set. In this example, "Permissions", column "Not for Replication", index "Fill Factor", assembly full "Filename" with path, and some other attributes are set to the "ignore" by default state. Every time you open the Schema Compare dialog and choose SQL Server rule set, they will not be selected by default in the Schema Object Tree.

```
User
    Default schema
Role
Application Role
Assembly
    Content
    -Filename
Partition Function
Partition Scheme
Table
    Data space
View
Procedure
Function
Synonym
```

```
Data Type
    Type
    Nullable
Table Type
    -Nullable
Sequence
    Type
    Start value
    Increment
    Min value
    Max value
    Cycling
    Cached
XML Collection
Aggregate
    Assembly
    Assembly class
    Return
Assembly Function
    Assembly
    Assembly class
    Assembly method
    Return
Assembly Table Function
    Assembly
    Assembly class
    Assembly method
-Permission
DDL Trigger
Default
Rule
Column
    Type
    Nullable
    Collate
    Rowguidcol
    Filestream
    Sparse
    -Not for replication
Primary Key
    -Fill factor
    Pad index
    Ignore dup key
    Statistics norecompute
    Allow row locks
    Allow page locks
Unique Key
    -Fill factor
    Pad index
    Ignore dup key
    Statistics norecompute
    Allow row locks
    Allow page locks
Foreign Key
    On delete
    On update
    -Not for replication
Check Constraint
    -Not for replication
Default Constraint
DML Trigger
Index
    -Fill factor
    Pad index
    Ignore dup key
    Statistics norecompute
    Allow row locks
    Allow page locks
Property
```

Notes:

- Object type names in the **Object Types** tree must appear at the start of the lines. Attribute names must be indented and must appear below object types they belong to. You can use spaces or tab characters for the indenting. At least one space or one tab character must be in front of the attribute name.

- Object type names might be followed by a comma separated list of dependent object types. This is to aid the comparison engine with calculating the proper display of items in the visual comparison results.

- To customize the appearance of an object type in the visual comparison results, click the icon in front of the type name. A list of stock icons will be displayed from which you can choose the desired icon. Optionally, you can enter the icon name as a pipe-separated text at the end of the line.
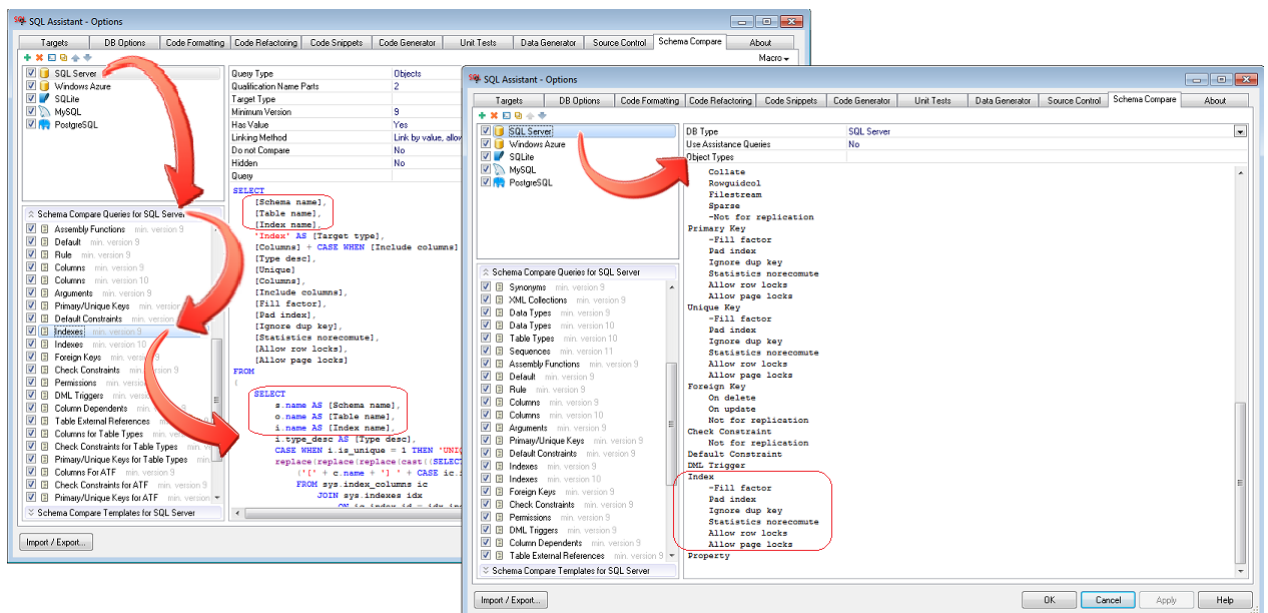
  In case no icon name is specified, the comparison engine will use the default icon for that object type or folder icon in case it does not recognize the object type.

- The names of object types and their attributes in the comparison options should match the data returned by the database queries specified for the rule set. If you have customized the prepackaged queries or added your own, you should also add matching object types and attributes to the **Object Types** settings.

  For example, if in the SQL Server rule set you customized the "Indexes" query and extended it by adding new [Table Type] column to the query definition and you want the value in that column to be comparable to values returned for other indexes, you need to add *Table Type* to the **Object Types** tree and inserted between the *Index* line and the next object type specified in the tree, in the above example, you would need to insert it between the *Index* and the *Property* lines.

  The following graphical diagrams demonstrate the changes required for adding [Table Type] as a new comparable attribute for Indexes.

Before the change:

After the change:
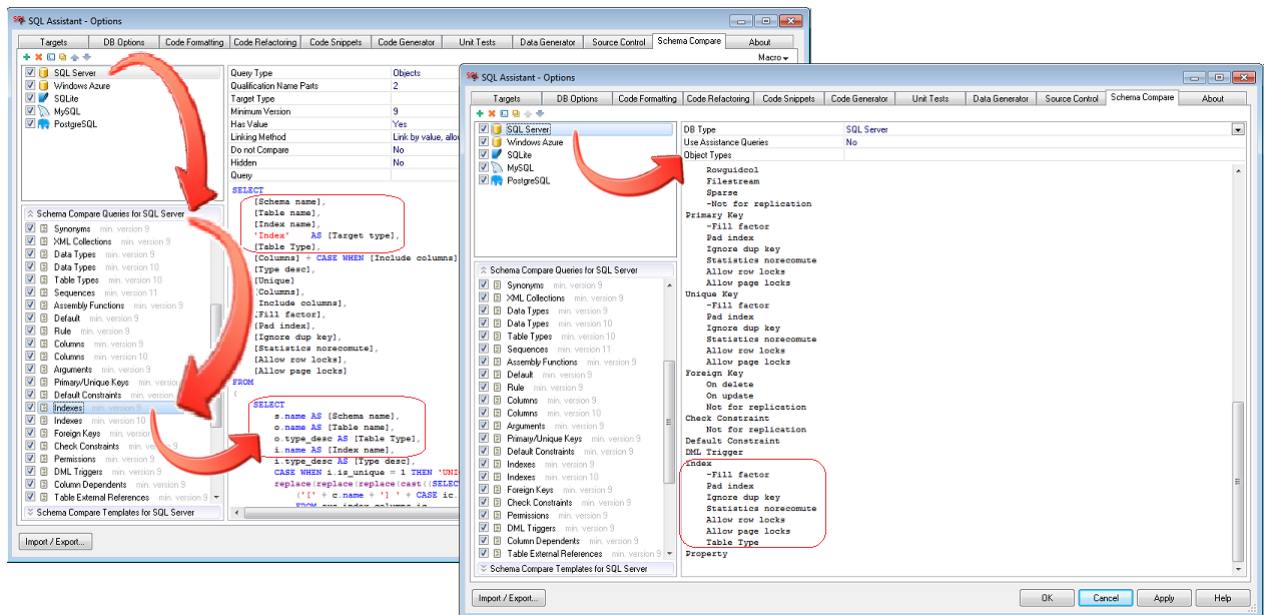
The Anatomy of Queries

Two types of queries are supported in the schema comparison rules:

- **Objects** – this query type is used to retrieve information about database and schema objects and optionally their attributes if they can be retrieved in one go.

- **Attributes** – this is an auxiliary query type used to retrieve object attributes. Queries of this type accompany predecessor **Objects** queries.

Queries retrieving data for schema comparison can have variable number of column. The order of columns in an **Objects** query is very important and must follow a certain pattern. The names of columns are not important and could be anything you like.
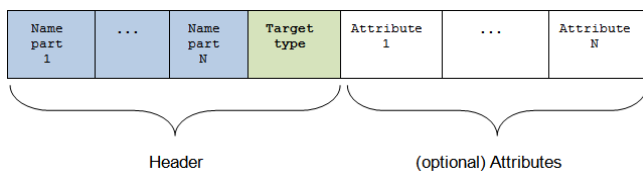
The header columns must return qualifying object name parts followed by target type, which is typically the same as schema object type. The number of header columns varies for different target types depending on their scope and name qualification. The query may reference additional non-header columns returning values of object attributes.

The following example "Schemas" query for SQL Server has 2 header columns and no attributes columns. The first column [Schema name] is "Name part 1." The second column with fixed value 'Schema' is "Target Type."

```
SELECT
    [name]        AS [Schema name],
    'Schema'      AS [Target type]
FROM sys.schemas
WHERE schema_id < 16384
ORDER BY [name]
```

The following example "Default constraints" query for SQL Server has 4 header columns and 3 attributes columns. The first column [Schema name] is "Name part 1", the second column [Table name] is "Name part 2", and the third column [Constraint name] is "Name part 3." The fourth column [Target type] with fixed value 'Default Constraint' is "Target Type." The rest are attribute columns "Attribute 1," "Attribute 2" and so son

```
SELECT
    s.name                AS [Schema name],
    o.name                AS [Table name],
    d.name                AS [Constraint name],
    'Default Constraint'  AS [Target type],
    c.name                AS [Column name],
    d.is_system_named     AS [IsSystem__DONT_COMPARE],
    d.definition          AS [Definition]
FROM sys.default_constraints AS d
    JOIN sys.objects AS o
        ON o.[object_id] = d.parent_object_id
    JOIN sys.schemas AS s
        ON s.[schema_id] = o.[schema_id]
    JOIN sys.[columns] AS c
        ON c.[object_id] = o.[object_id]
            AND d.parent_column_id = c.column_id
WHERE ('$SCHEMA_NAME$' = '' OR s.name = '$SCHEMA_NAME$') AND
    ('$OBJECT_NAME$' = '' OR o.name = '$OBJECT_NAME$')
```

Important Note: The value returned by the [Target Type] column is very important. The schema compare engine dynamically locates templates having their **Target Type** property set to the same value when it generates schema change synchronization scripts.

Query Properties

Query Type – Objects or Attributes. Objects – this query type is used to retrieve information about database and schema objects and optionally their attributes if they can be retrieved in one go. Attributes – this is an auxiliary query type used to retrieve object attributes. Queries of this type accompany predecessor Objects queries.

Qualification Name Parts – A zero-based index of name parts in the record header. In other words, this is the number of columns in the query preceding the Target Type column less one. It is expected that each preceding column refers to a qualifying part of the object. For example,

```
SELECT
    a.name                AS [Database name],
    s.name                AS [Schema name],
    o.name                AS [Table name],
    d.name                AS [Constraint name],
    'Default Constraint'  AS [Target type],

(the rest of the query goes here)
```

In the above query [Database name], [Schema name], [Table name], [Constraint name] fully qualify database constraint. In that case the Qualification Name Parts property should be set to 3. It tells the engine how many parts are in the path before the Target Type value. Note that column names in the query are unimportant. You can chose any appropriate names. But column position is very important, all qualifying parts must be specified before the column returning Target Type values.

Note: This property is used with **Objects** type queries only.

Target Type – the comparison scope (target type) for which the schema comparison engine will execute the query. If "Server" is specified, the query will be executed only in case server scope comparison is performed. If "Database" is specified, the query will be executed in case of server scope or database scope comparison. If anything else is specified or the parameter is empty, the query will be executed in case of server scope or database scope or schema scope comparison.

This is an optional property which is not required for schema object specific queries. It is required for database and server scope queries.

Important Note: For your convenience the customizable drop-down list for the **Target Type** property is populated with some common values. Sometimes you may need a value which is not in that list. If you cannot find the required value in the drop-down list, simply type it into the property value edit box.

Important Note: The comparison engine is allowed to use database scope queries when comparing schema level objects. This is done for performance reasons to avoid recursive queries and allow the engine to load in bulk definitions of schema objects and their attributes.

Loaded Objects – The type of objects returned by the query. The value of this property should be the same as the values returned in by the Target Type column or expression in the query. In case the query returns multiple object types, specify them as a comma-separated list in the **Loaded Objects** property. For example, the following query for SDQL Server returns both database roles and application roles. For that query the value of the **Loaded Objects** property should be set to *Role,Application Role*.

```
SELECT
    p1.[name] AS [Role name],
    CASE
        WHEN p1.[type] = 'A' THEN 'Application Role'
        ELSE 'Role'
    END AS [Target type],
    p2.[name] AS [Owner]
FROM sys.database_principals p1
    LEFT JOIN sys.database_principals AS p2
        ON p1.owning_principal_id = p2.principal_id
WHERE  p1.is_fixed_role = 0
    AND p1.principal_id <> 0
    AND p1.[type] IN ('A', 'R')
ORDER BY p1.name
```

Minimum Version – the minimum database server version (major version number) the query is compatible with. Within the same comparison rule set you can define multiple queries for the same Target Type with different Minimum Version values. The comparison engine will automatically select the query whose Minimum Version matches or exceeds the server version.

Note: For easy of management it is recommended to create different rule sets for different database server versions instead of creating single rule set with multiple version specific variations of same queries and templates.

Has Value – specifies whether the column following the [Target Type] column in the query returns object's value that can be used for object comparison. For example, table CHECK and DEFAULT constraints can be compared by their definition without looking at their system generated constraints names. For example, the CREATE TABLE command such as `CREATE TABLE test(col INT DEFAULT 1)` will produce 2 schema objects: table object named "test" and default constraint object with a system generated name. If executed again in a different database, the system generated name for the constraint will be different from the first one. The schema comparison engine can still compare both test tables and identify them as identical if constraint names are ignored and it can use the value returned in the column after [Target Type] as the constraint definition.

Note: This property is used with **Objects** type queries only.

Match Method – this property specifies how to compare objects in the source and target schemas. The following options are available:

- **Match by name** – the source and target schema objects and their properties should be matched by their names. For example, this is typically used for matching views and procedural object names, as

well as matching tables and table columns.

- **Match by value, no rename** – the source and target schema objects and their properties should be matched by their values rather than their names. If selected, the **Has Value** property must be set to "Yes", and the query itself should return the value for the source and target comparison. The value can be used in comparison results, but cannot be changed or renamed using simple ALTER TABLE commands. In case of differences found and selected for schema synchronization, the schema synchronization script will attempt to drop and recreate the entire schema object.

- **Match by value, allow rename** – the source and target schema objects and their properties should be matched by their values rather than their names. If selected, the **Has Value** property must be set to "Yes", and the query itself should return the value for the source and target comparison. The returned value can be used in comparison results, and it can be also changed or renamed using simple ALTER TABLE commands.

Note: This property is used with **Objects** type queries only.

Do not Compare – if set to "Yes", the query is an auxiliary query returning data for use by schema synchronization scripts only. The query is executed just like other queries, but the retrieved data is not displayed in the comparison results. For example, the "Table External References" query used in the predefined "SQL Server" rule set is used to retrieve table dependencies which the schema synchronization script needs to drop before it can change the selected table, and then recreate for synchronizing the differences, so it can restore the existing dependencies. The "Table External References" query has **Do not Compare** property value set to "Yes."

Hidden – if set to "Yes", the query is an auxiliary query returning data not displayed directly in the comparison results, but still used for object comparison. This property is reserved for future use and is not currently used.

Query – the actual text of the query. The text of the query may contain standard macro variables referenced in the Menu drop-down on the Schema Compare tab in the Options dialog. See Macro-variables and Dynamic Code Generation topic in CHAPTER 7 for more details.

Attribute Value Modifiers

Names of attribute type columns in schema comparison queries can be optionally tagged with special suffixes to indicate their special treatment. The following tags are supported:

__DONT_USE – instructs the engine to do create an attribute display for this column value. This tag effectively suppresses the column without modifying the result set of the query.

__DONT_COMPARE – instructs the engine to not compare values in this column. The column values can be still referenced in scheme change synchronization templates and used in conditional logic in templates, but their value differences between the source and target ignored.. You can also use the short notation __DC instead of the long notation __DONT_COMPARE.

__NAME_REFERENCE – instructs the engine that the value in this column is a reference to an attribute name. Basically, this is like a pointer, the value indicates which other attribute should be used for the comparison or further operations. You can also use the short notation __NR instead of the long notation __NAME_REFERENCE.

__HIDDEN – instructs the engine to not show values in this column in the visual comparison results. A good example of hidden columns is a column returning some internal system properties, or system generated identifiers, which are important for correlating different schema objects or elements together but the value itself is not that important. A case in point, internal numeric table object id is needed to locate its columns in the data dictionary, and yet that numeric value can be different for two structurally identical tables. The values in columns tagged with __HIDDEN can be still referenced in scheme change synchronization templates.

__AGG – instructs the engine to aggregate all values in the column. If the query with this column modifier returns multiple records, all values in that column are concatenated to a single comma separated list.

__SQL – instructs the engine that the value in this column contains SQL code and it should be compared using DDL comparison rules controlling comparison of spaces, comments, object name quoting, and so on...

Note: Multiple tags can be appended to the same column, for example, __DONT_COMPARE__HIDDEN.

The following sample query demonstrates the use of tags

```
SELECT
    s.name                AS [Schema name],
    o.name                AS [Table name],
    d.name                AS [Default name],
    'Default Constraint'  AS [Target type],
    c.name                AS [Column name],
    d.is_system_named     AS [IsSystem__DONT_COMPARE],
    d.definition          AS [Definition]
FROM sys.default_constraints AS d
    JOIN sys.objects AS o
        ON o.[object_id] = d.parent_object_id
    JOIN sys.schemas AS s
        ON s.[schema_id] = o.[schema_id]
    JOIN sys.[columns] AS c
        ON c.[object_id] = o.[object_id]
            AND d.parent_column_id = c.column_id
WHERE ('$SCHEMA_NAME$' = '' OR s.name = '$SCHEMA_NAME$') AND
      ('$OBJECT_NAME$' = '' OR o.name = '$OBJECT_NAME$')
```

In addition to attribute value modifiers, the engine recognizes the following several column names as columns returning special attributes. For example, in the above sample query, the "IsSystem" column is recognized as a special attribute. The special names are:

- **IsSystem** – The values in this column are expected to be either 0 or 1. The values indicate whether the object referenced in the result set features a system generated name that can be omitted when generating schema synchronization scripts and the target system can generate its own name for the target object.

- **IsSystemObject** – The values in this column are expected to be either 0 or 1. The values indicate whether the object referenced in the result set is a system object and it should not be included in schema synchronization script.

- **SyncConfig** – The value in this column contain additional instructions for the comparison engine indicating how the object needs be processed when generating schema synchronization scripts. The value is bitwise OR combination of following flags:

  1 – Generate CREATE statements for dependent logical "children" objects after CREATE statement for this object.
  2 – Generate DROP statements for dependent logical "children" objects before DROP statement for this object.
  4 – Do not generate ALTER statement for dependent logical "children" objects after altering this object.
  8 – Do not generate CREATE statement for this object.
  16 – This object is independent of its logical "parent" and it can be created or dropped independently.
  32 – This object is code based, for example, it's a stored procedure or view.
  64 – Do not generate synchronization statements for this object, this is typically a case for system objects.

  Here are default **SyncConfig** attribute values for standard object types:

Object Type	Flags (as Bitwise Expressions)	Value
Server, Database	1 OR 8	9

Schema	1 OR 2 OR 8	11
Comment, Property, Permission	16	16
Reference, Dependent	64	64
Column	8	8

Using Macros in Queries

The following standard macros used in snippets engine are also supported in templates: $DATE$, $TIME$, $SERVER$, $LOGIN$, DB, $USER$, $OSUSER$, $MACHINE$, SA_TARGET, $SA_VERSION$. In addition, the following schema comparison queries specific macros can be used in template codes:

DB_NAME - item's database name (level 1, database)

$SCHEMA_NAME$ - item's schema name (level 2, schema, user, role, partition function, etc.)

$OBJECT_NAME$ - item's object name (level 3, table, view, etc.)

$SUBOBJ_NAME$ - item's sub-object name (level 4, constraint, column, etc.)

$NAME_LEVEL_[number]$ - generic reference to an object at a certain hierarchy level. For example:

$NAME_LEVEL_1$ is the same as DB_NAME but a more generic name.

$NAME_LEVEL_2$ is the same as $SCHEMA_NAME$ but a more generic name.

$NAME_LEVEL_3$ is the same as $OBJECT_NAME$ but a more generic name.

$NAME_LEVEL_4$ is the same as $SUBOBJ_NAME$ but a more generic name.

$NAME_LEVEL_5$, $NAME_LEVEL_6$, and so on... Levels 5 and above have no schema scope analogues.

Tip: Use of generic references is more convenient in object hierarchies not based on database or schema names, such as jobs, schedules, resource governors, tablespace files, and so on, which are not schema bound in most database systems.

The Anatomy of Templates

The schema engine uses scriptable templates to generate database and schema change synchronization scripts. Templates work exactly like code snippets described in CHAPTER 7, Code Entry Automation using Code Snippets but in addition to that they can also call recursively other templates, for example, a "Table ("CREATE)" template can call "Column (CREATE)" template for each table column and that template in turn can call "Column definition" template to output column definition syntax required for the CREATE TABLE syntax.

Different templates are used for different object types. The "Target Type" property in the template used by the schema comparison engine to select the required template for each processed database and schema object type. Different templates are required to handle different types of changes such as ALTER, RENAME, CREATE, and DROP operations. The type of the template is defined by the "Template Type" template property.

Template Properties

Template Type – the type of the DDL operation the template is used for. The following types are supported:

- ALTER
- RENAME

- CREATE

- DROP

For ALTER DDL operation type, if ALTER template is not defined, the schema comparison engine will attempt to use DROP and CREATE templates.

Note: The **Template Type** value need not be specified for templates invoked by other templates, in other words for nested templates, which are reusable and not specific to an operation type. For example, a "Column definition" template should not have Template Type specified.

Target Type – the type of database or schema object the template is used for. Example types are: "Table", "Table function", "Check constraint", "Column", "Partition schema" and so on...

Note: For your convenience the drop-down list for the **Target Type** property is prepopulated with some common values. If you cannot find the required value in the list, simply type it into the property value edit box.

This is an optional property. It is not required for generic nested templates.

Minimum Version – the minimum database server version (major version number) the template is compatible with. Within the same comparison rule set you can define multiple templates for the same Template Type and Target Type with different Minimum Version values. The comparison engine will automatically select the template whose Minimum Version matches or exceeds the server version.

Note: For easy of management it is recommended to create different rule sets for different database server versions instead of creating single rule set with multiple version-specific variations of same queries.

Condition – An optional condition for using the template. Conditional expression must be specified n the following format:

```
<[operator][<|state_option>][{attribute_name}{comparison}{value}]>
```

A logical **operator** joining multiple conditions within the same conditional expression.

Operator	Meaning
&	AND operator
|	OR operator

An object **state**. The state element is optional

State	Meaning
+	If object exists in source only
-|	If object exists in target only
*	If objects are different
=	If objects are equal

Attribute name. The attribute name is optional.

Attribute	Meaning
Name of attribute column in the associated query	Check the specified attribute value.
.	If "." check the object name itself

Value **comparison** operator. The comparison operator is optional.

Comparison	Meaning
=	Equal
<>	Not equal

Value – an optional attribute value to check in this conditional expression. The value can be specified only if the attribute name is specified too.

Examples:

*Type – a template with this condition will be used only when the attribute named "Type" is different in compared objects.

IsSystem=0 - a template with this condition will be used only when the value of attribute named "IsSystem" is equal to zero.

Auto=0 & Nullable=NOT NULL - a template with this condition will be used only when the value of attribute named "Auto" is zero and the value of attribute "Nullable" is "NOT NULL"

Template – the code of the template

Using Macros in Templates

The templates support four types of macros:

- A number of standard macro variables supported by SQL Assistant in code snippets and in the Smart Database Refactoring templates.

- Simple macro variables returning comparison operation scope values such as source or target schema name, object name, and so on. They are described in detail in the following paragraphs.

- Simple macro variables returning values from columns of the associated database query, in other words, the attributes of the referenced schema objects. See The Anatomy of Queries topic for more details.

- Nested templates.

Use of SQL Assistant's Standard Macros

The following standard simple value macros supported in the code snippets are also supported in the schema comparison templates:

$DATE$
$TIME$
$SERVER$
$LOGIN$
DB
$USER$
$OSUSER$
$MACHINE$
SA_TARGET
$SA_VERSION$

For the description of the above macros refer to CHAPTER 7, Code Entry Automation using Code Snippets.

Use of Comparison Engine-specific Macros

The following comparison engine specific simple value macros can be used in the schema comparison templates:

$SOURCE$ - name of the relative top-level item in the source database. The returned value depends on the selected comparison scope and comparison options.

$TARGET$ - name of the relative top-level item in the target database. The returned value depends on the selected comparison scope and comparison options.

DB_NAME - item's database name (level 1, database)

$SCHEMA_NAME$ - item's schema name (level 2, schema, user, role, partition function, etc.)

$OBJECT_NAME$ - item's object name (level 3, table, view, etc.)

Example:
```
CREATE INDEXTYPE "$SCHEMA_NAME$"."$OBJECT_NAME$" FOR
$,\n|INDEX OPERATOR=INDEX OPERATOR$
USING $IMPLEMENTATION TYPE$$ WITH ARRAY DML {, |INDEX DML=INDEX DML}$
$WITH {PARTITIONING} PARTITION $$WITH {MAINTENANCE} STORAGE TABLES$
```

$SUBOBJ_NAME$ - item's sub-object name (level 4, constraint, column, etc.)

$NAME[;default]$ - object's name. The parameter's default value is used when the target object name is empty and the default value is known or specified in the macro. Here, object name is a name of schema object as specified in the **Object Types** tree, see Customizing Default Comparison Options topic for more information.

Example:
```
CREATE APPLICATION ROLE $NAME$
WITH PASSWORD = '/* {password} */' , DEFAULT_SCHEMA = $OWNER;dbo$
```

$VALUE[;default]$ - object's value. Default value (if specified) is used when the specified value is empty.

DDL - source object's DDL with optionally substituted target database and schema names. The substitution is applied in case the **Auto-fix target db and schema names** option is selected and target database or schema names differ from the source names.

$ALTER_DDL$ - This is the same as the DDL macro with additional change of the object's CREATE statement to ALTER or CREATE OR REPLACE depending on the target database type.

You can also append source and target reference modifiers to macro names. Use "**.SRC**" to refer to the source

value, and "**.TRG**" to refer to the target value. For example:

```
/*************************************************************
 * Code generated by SoftTree SQL Assistant © v$SA_VERSION$
 * using Schema Compare templates
 * Time: $DATE$ $TIME$
 * Source: $NAME.SRC$
 * Target: $NAME.TRG$
 *************************************************************/
```

Use of Macros Returning Schema Object Attributes

The attribute macros are simple value macros copying object attribute values from the database query associated with the template. The association between comparison templates and queries are based on the templates **Target Type** and **Condition** properties and the **Loaded Objects** types specified in the query properties.

$attribute_name[;default]$ - object's attribute value. The engine searches for attribute name specified in the **attribute_name** part using case insensitive search. If the specified attribute name cannot be found or it returns an empty string, then the default value is used.

Example

The "Application Role (CREATE)" template for SQL Server is associated with the "Database Roles" query. This "Application Role (CREATE)" template has its **Target Type** property value set to "Application Role" and the "Database Roles" query has its **Loaded Objects** property value set to "Roles,Application Roles." Below is the template code referencing the "Owner" attribute from the query and specifying "dbo" as the default value in case the value of the "Owner" attribute is empty (an empty string or NULL value).

The "Application Role (CREATE)" template:

```
CREATE APPLICATION ROLE $NAME$
WITH PASSWORD = '/* {password} */' , DEFAULT_SCHEMA = $OWNER;dbo$
```

The "Database Roles" query code:

```
SELECT
    p1.[name] AS [Role name],
    CASE
        WHEN p1.[type] = 'A' THEN 'Application Role'
        ELSE 'Role'
    END AS [Target type],
    p2.[name] AS [Owner]
FROM sys.database_principals p1
    LEFT JOIN sys.database_principals AS p2
        ON p1.owning_principal_id = p2.principal_id
WHERE  p1.is_fixed_role = 0
    AND p1.principal_id <> 0
    AND p1.[type] IN ('A', 'R')
ORDER BY p1.name
```

Use of Text Prefixes and Suffixes with Macros, and Default Values

The comparison engine supports optional prefixes and suffixes appended to the value returned by a macro variable, as well as the default values used in case of macro variable returning an empty value. Prefixes, suffixes, and default values can be specified using the following notation:

$Prefix{Macro}Suffix;Default$

Prefix and **Suffix** are text strings which are appended to the returned macro value. If the actual macro value is empty then the **Default** string will be returned. Each part before or after the {Macro} is optional. For example, you can specify prefix without suffix, or specify a default value without prefixes or suffixes specified. Note that the **Macro** could be a name of a simple macro variable or a name of a nested template.

Example:
```
CREATE INDEXTYPE "$SCHEMA_NAME$"."$OBJECT_NAME$" FOR
$,\n|INDEX OPERATOR=INDEX OPERATOR$
USING $IMPLEMENTATION TYPE$$ WITH ARRAY DML {, |INDEX DML=INDEX DML}$
$WITH {PARTITIONING} PARTITION $$WITH {MAINTENANCE} STORAGE TABLES$
```

In this example, the PARTITIONING in the highlighted part is a name of a simple macro variable referencing values returned by the PARTITIONING column of the preconfigured **Index type** query used for Oracle schema comparison. Every partitioning value is prefixed with "WITH " text and suffixed with " PARTITION " text, with the final result inserted into the schema synchronization script.

Additional Modifiers for Macro-variables

The following suffixes can be added to macro variable names used in the templates to indicate which specific value to insert in the place of the macro variable and how to format it.

.SRC – instructs the engine to use the original source value for the referenced name or attribute. This modifier can be used with object and column names.

.TRG – instructs the engine to use the target value for the referenced name or attribute in case it is different from the original name. This modifier can be used with object and column names. For example, when you compare 2 differently named schemas, the DDL for target objects in the synchronization script differs from the original, because the schema name is not the same. In the case $SCHEMA_NAME.SRC$ will return the schema name in the source database, while $SCHEMA_NAME.TRG$ - schema name in the target database.

.PREV – instructs the engine to use the value of the previous sibling object.

.STRING – instructs the engine to use convert the value to a valid string format and enclose it in single quotes. All nested single quotes are doubled.

Note: The modifiers can be concatenated together if required

Example: The below code is used in the "Table Object (RENAME)" template in the comparison rule set for SQL Server databases for renaming system generated constraint and index names in the target schema to match their system generated names in the source schema.

```
EXEC sp_rename @objname = N'$SCHEMA_NAME$.$NAME.TRG$', @newname = N'$NAME.SRC$'
```

Calling Nested Template

Call syntax for calling nested templates:

```
$separator|ObjectType1.SetModifier1=TemplateName1,ObjectType2.SetModifier1=TemplateName2,...,ObjectTypeN.SetModifier1=TemplateNameN$
```

Every part in the syntax is optional except the special "|" and "=" symbols separating the parts.

separator – an optional separator string used when concatenating text returned by nested templates. The following special symbols are supported in separator strings: \n - line break; \t - TAB character; \\ - "\" character.

Important Note: The **separator** string must be followed by the pipe "|" symbol. In case the separator string is not specified, the pipe symbol must be still specified.

```
CREATE PARTITION FUNCTION [$NAME$] ( $PARAMETER$ )
AS $FUNCTION TYPE$
FOR VALUES ( $, |PARTITION RANGE VALUE=NAME UNDELIMITED$ )
```

In this example, ", " (comma and space) is the separator string, and NAME UNDELIMITED is a name of the nested template.

ObjectType – an optional reference to the type of child object to which the nested template will be applied. If this parameter is omitted, nested template is applied to the object itself. For example,

Important Note: The object type with an optional modifier string must be followed by the equal sign "=" symbol. In case the object type is not specified, the equal sign symbol must be still specified.

```
CREATE PARTITION FUNCTION [$NAME$] ( $PARAMETER$ )
AS $FUNCTION TYPE$
FOR VALUES ( $, |PARTITION RANGE VALUE=NAME UNDELIMITED$ )
```

In this example, "PARTITION RANGE VALUE" is the object type, and NAME UNDELIMITED is a name of the nested template.

SetModifier – This property specifies custom objects subset to apply the nested template to. One of the following can be used:

Not specified - new selected source and target objects, all modified objects, all equal objects.

TARGET - equal objects, modified objects (source if selected, and target is not selected), selected new source objects, not selected new target objects

TARGET_OLD - same as TARGET but without new source objects

TARGET_ONLY - only target objects (equal, modified, new)

ONLY_NEW - only new selected source objects

TemplateName – the name of the nested template. If this parameter is specified, the engine searches for the specified template name. If this parameter is not specified, the engine searches for a template associated with the object type matching of **ObjectType** value and having the same template type as the calling template.

Here is an example of template code for MySQL specific rule set for column change synchronization template named "Column (ALTER)":

```
ALTER TABLE `$SCHEMA_NAME$`.`$OBJECT_NAME$` MODIFY COLUMN $|=COLUMN DEFINITION$;
```

When processing this template code the comparison engine performs the following steps:

1. The simple value macro $SCHEMA_NAME$ is replaced with the target schema name.
2. The simple value macro $OBJECT_NAME$ macro is replaced with the target table name.
3. The last part $|=COLUMN DEFINITION$ is a reference to the nested template named "Column Definition". This part is processed separately and then the output of that nested template is inserted into the output of the "Column (ALTER)".

Extending Schema Comparison, a Practical Example

The following example demonstrates how to compare SQL Server's server scope properties.

1. Double-click SQL Assistant's icon in the system tray to open the Options dialog and select the **Schema Compare** tab.

2. Right-click the **Comparison Rules** list in the top left window, and select the **Add** command from the context menu. A new rule set will be added to the list with the input focus in the rule set name. Enter "SQL Server Properties" name (without quotes), and then click the right window to set focus to the Properties table.

3. Fill in property values for the new rule set as specified below:

DB Type – SQL Server
Use SQL Assistance Queries – No
Object Types – Server
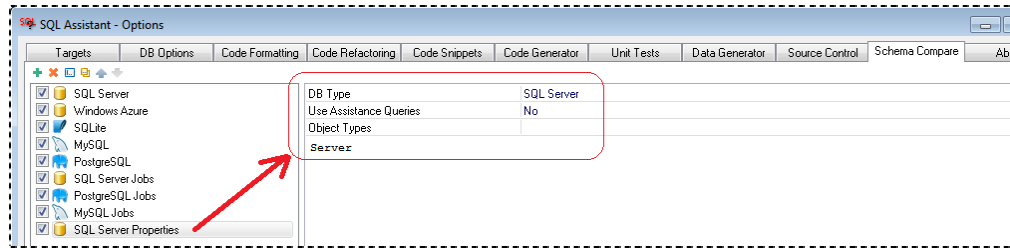
4. Right-click the left bottom window in the **Queries** section, and select the **Add** command from the context menu. A new query will be added to the list with the input focus in the query name. Enter "Server Properties" as the query name (without quotes), and then click the right window to set focus to the Properties table.

5. Fill in property values for the new query as specified below:
   Query Type – Objects
   Qualification Name Parts – 0
   Target Type – Server
   Minimum Version – 9
   Has Value – No
   Match Method – Match by name
   Do not Compare – No
   Hidden – No
   Query – *enter the following SQL query text*

```sql
SELECT propertyname,
       'Property' AS [Target Type],
       is_fixed AS is_fixed__HIDDEN,
       0 AS is_advanced__HIDDEN,
       0 AS is_dynamic__HIDDEN,
       CAST(SERVERPROPERTY(propertyname) AS NVARCHAR(255)) AS [value]
FROM
(
    SELECT 'BuildClrVersion' AS propertyname, 1 AS is_fixed
    UNION SELECT 'Collation', 1
    UNION SELECT 'Edition', 1
    UNION SELECT 'EngineEdition', 1
    UNION SELECT 'IsClustered', 1
    UNION SELECT 'IsFullTextInstalled', 1
    UNION SELECT 'IsIntegratedSecurityOnly', 0
    UNION SELECT 'IsSingleUser', 0
    UNION SELECT 'LCID', 1
    UNION SELECT 'ProductVersion', 1
    UNION SELECT 'ProductLevel', 1
    UNION SELECT 'ResourceVersion', 1
    UNION SELECT 'SqlCharSetName', 1
    UNION SELECT 'SqlSortOrderName', 1
    UNION SELECT 'FilestreamShareName', 1
    UNION SELECT 'FilestreamConfiguredLevel', 1
) AS t

UNION ALL
SELECT name  AS propertyname,
       'Property' AS [Target Type],
       0 AS is_fixed,
       is_advanced,
       is_dynamic,
       CAST([value] AS NVARCHAR(255))
FROM master.sys.configurations
ORDER BY 1 ASC
```
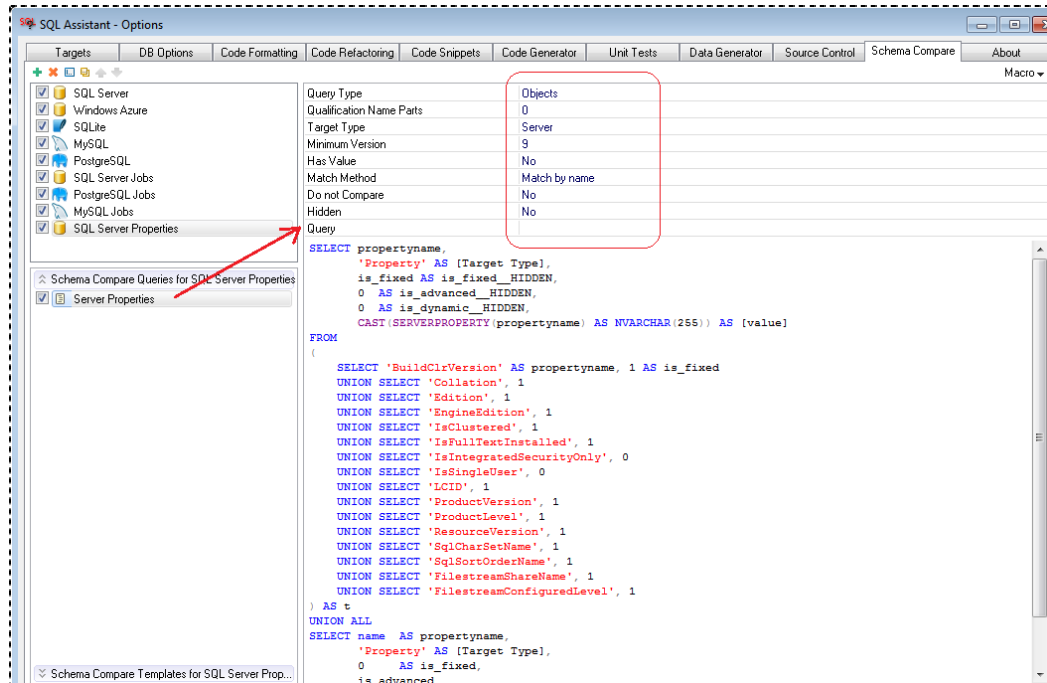
Note: The following steps are optional and only required if you want to automate generation of server properties synchronization scripts. Note that properties returning 1 in the **is_fixed** column cannot be changed through scripting, they require running SQL Server installation program or in case of SQL Server version or clustering configuration differences, a complete reinstall.

6. Click the **Schema Compare Templates for SQL Server Properties** bar to expand the **Templates** section.

7. Right-click the **Templates** section, and select the **Add** command from the context menu. A new template will be added to the list with the input focus in the template name. Enter "Server Properties (ALTER)" as the template name (without quotes), and then click the right window to set focus to the Properties table.

8. Fill in property values for the new query as specified below:
    Template Type – ALTER
    Target Type – Server
    Minimum Version – 9
    Condition – *leave this property value blank*
    Template– *enter the SQL script that can be used to update different server properties. The following example demonstrates how to do that for a few different properties, we will leave the rest for you to complete as an exercise.*

```
USE [master]
GO

IF '$NAME$' = 'awe enabled'
BEGIN
    EXEC sp_configure N'show advanced options', N'1'  RECONFIGURE WITH OVERRIDE
    EXEC sp_configure N'awe enabled', N'1' RECONFIGURE WITH OVERRIDE
    EXEC sp_configure N'show advanced options', N'0'  RECONFIGURE WITH OVERRIDE
    PRINT 'IMPORTANT: You must restart the server before the new
            "Use AWE Memory" settings will take effect.'
END
GO
```

```
IF '$NAME$' = 'cost threshold for parallelism'
BEGIN
    EXEC sp_configure N'show advanced options', N'1'  RECONFIGURE WITH OVERRIDE
    EXEC sp_configure N'cost threshold for parallelism', N'$VALUE$' RECONFIGURE
        WITH OVERRIDE
    EXEC sp_configure N'show advanced options', N'0'  RECONFIGURE WITH OVERRIDE
END
GO

IF '$NAME$' = 'IsIntegratedSecurityOnly'
BEGIN
    DECLARE @RegValue INT = (CASE WHEN '$VALUE$' = '1' THEN 1 ELSE 2 END)
    EXEC xp_instance_regwrite N'HKEY_LOCAL_MACHINE',
        N'Software\Microsoft\MSSQLServer\MSSQLServer', N'LoginMode',
        REG_DWORD, @RegValue
    PRINT 'IMPORTANT: You must restart the server before the new
        "Server Authentication" settings will take effect.'
END
GO

IF '$NAME$' = 'nested trigger'
BEGIN
    EXEC sp_configure N'show advanced options', N'1'  RECONFIGURE WITH OVERRIDE
    EXEC sp_configure N'nested trigger', N'$VALUE$' RECONFIGURE WITH OVERRIDE
    EXEC sp_configure N'show advanced options', N'0'  RECONFIGURE WITH OVERRIDE
END
GO
....
```
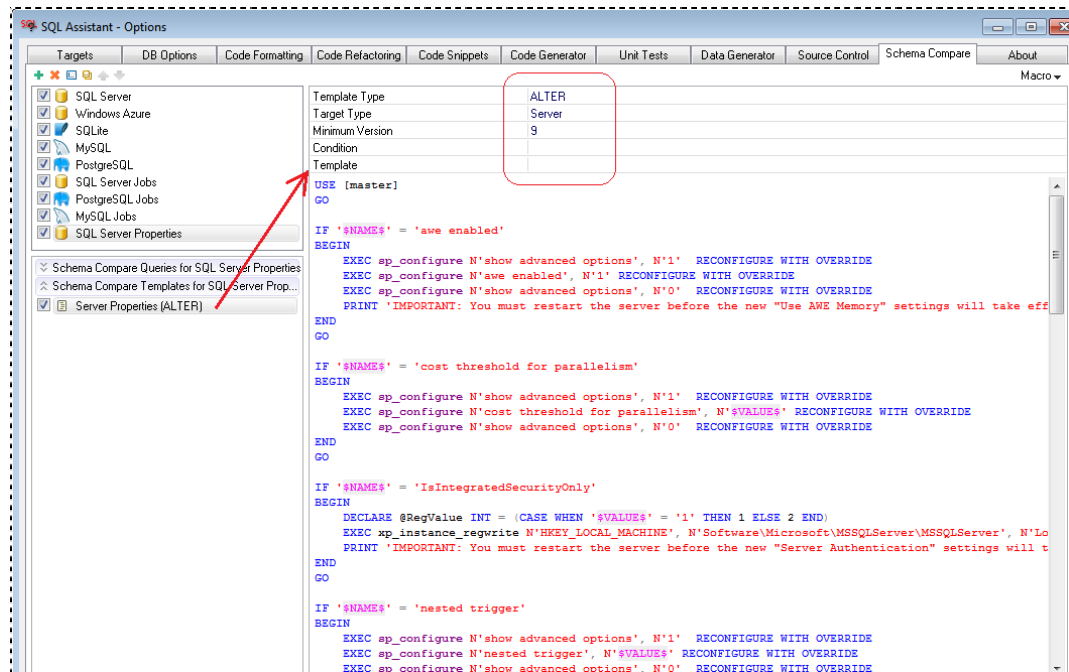

9. Click the **Ok** button to save the entered settings. The new Server Properties comparison rules can be used immediately.

Scheduling Automated Data Comparisons

The **Schedule** button on the Data Compare dialog enables you to automate periodic data comparison operations or to schedule data comparison runs at night or other quiet times when the database server is not very busy. This opens the Schedule dialog providing graphical interface to the data comparison command line interface described in the next topic. For information on how to manage scheduled tasks in SQL Assistant, see CHAPTER 50, Managing Scheduled Tasks

The scheduled task uses data comparison command line interface described in the next topic to run the comparison at the specific date and time.

Command Line Interface

To run schema comparison operations from a DOS command line window, use the following command:

```
sacmd sc srcconn:"src-connection-name" dstconn:"dest-connection-name" srcname:"src-object-
path" dstname:"dest-object-path" outfile:"sql-file-name" logfile:"log-file-name"
srcflags:"flag-chars" sas:"path-to-sa-settings-file"
```

The above command must be entered as a single line

Substitute values in the command as follows:

src-connection-name	The database connection name for the source database connection
dest-connection-name	The database connection name for the destination database connection
src-object-path	The fully qualified dot separated path to the source table, schema or database. This also defines the scope of the comparison. If you specify just **database** name, then all tables in the database will be compared. If you specify **database.schema** name, then all tables in the specified schema will be compared. If you specify **database.schema.table** name , then only the specified table will be compared
dest-object-path	The fully qualified dot separated path to the target table, schema or database. This also defines the scope of the comparison. If you specify just **database** name, then all tables in the database will be compared. If you specify **database.schema** name, then all tables in the specified schema will be compared. If you specify **database.schema.table** name , then only the specified table will be compared.

Important Notes:
- The source and destination path must have the same scope.
- For database servers without database element in the path, such as Oracle, DB2, MySQL, SQLite, and others use **noDB** name in place of the database name

srcrules:"rules-name"	Source schema comparison rule set name. This is an optional parameter. If not specified, the comparison engine will use default rules for the source database server type and version.

dstrules:"rules-name"	Destination schema comparison rule set name. This is an optional parameter. If not specified, the comparison engine will use default rules for the destination database server type and version.
sql-file-name	The output SQL file for the schema synchronization script.
log-file-name	The log file for the schema comparison log messages. This is an optional parameter. If not specified, log file is not generated.
flag-chars	Combination of flags for the schema comparison. The following symbols can be used in this parameter:

        s - Ignore system objects
        f - Ignore names (constraints, indexes)
        c - Ignore character case
        w - Ignore white spaces (DDL)
        x - Ignore comments (DDL)
        l - Ignore newline format (DDL)
        d - Ignore delimiters (DDL)
        n - Auto-fix target database and schema names
        r - Do not rename indexes and constraints

path-to-sa-settings-file	The full file name of the SQL Assistant settings file containing the required database connection parameters. This is an optional parameter. If not specified, the default path for the current user account is used.

Example:

```
cd "C:\Program Files (x86)\SQL Assistant 12"

sacmd sc srcconn:"DEV001 (sa)" dstconn:"PROD (sa)" srcname:"AdventureWorks.Sales"
dstname:"AdventureWorks.Sales" outfile:"C:\TEMP\SchemaSyncScript.xml" srcflags:"sbt"
sas:"%APPDATA%\SQL Assistant\13.0\sqlassist.sas"
```

Important Notes:

- The SQL Assistant settings file location is version and user profile specific. See the Notes in the Overview topic in CHAPTER 51 for details on how to find out the location of that file.

- You can find out the connection name in the DB Connections group of settings on the DB Options tab page in SQL Assistant Options. If a connection requires a user id and password, make sure that both are saved in the settings. The command line interface does not display interactive prompts and is unable to prompt for credentials during command processing. For more information about storing and managing database connections, see the Managing Database Connections topic in CHAPTER 48.

CHAPTER 28, Job Compare Utility

Overview

The Job Compare Utility is based on the Schema Compare Utility and provides similar functions and the user interface, but it is focused on the comparison of the scheduled automated database jobs and tasks.

You can compare job differences in two database servers of the same types having the same or different versions. The utility ignores version specific differences and compares only objects and attributes supported by both versions.

The Job Compare utility supports flexible comparison filters enabling you to choose to include or exclude particular object types, particular object attributes and properties, to choose to ignore minor differences, such as differences in user privileges, extra spaces in the code, and so on...

Note: The Job Compare utility supports scriptable interface comprised of comparison rules, database queries, and templates similar to queries and templates used by the Schema Compare Utility for schema synchronization. It enables users to extend the existing functionality, adding new features for supporting new object types and properties and altering behavior of pre-configured comparison features. See the Extending and Customizing Schema Compare Functions topic for more details on customizing the Job and Schema Compare.

The Job Compare utility can be invoked from SQL Assistant's menus. The **Compare Code and Data → Compare Jobs** command is available from either the right-click context menu in the target editor or the top-level menu (if the target editor top-level menu integration is enabled). This command opens the **Job Compare** dialog. In this dialog you can select the target servers, jobs, schedules, programs, pre-defined conditions, and other related objects to compare. The following topics describe the usage of the Job Compare

How Job Comparison Engine Works

The schema comparison and job comparison share the same engine. For job comparison the engine utilizes job specific comparison rules, queries and templates predefined in SQL Assistant's options. For details on how the engine works see How Schema Comparison Engine Works topic.

The Job Compare Dialog

Navigation

The Job Compare dialog is a wizard-like dialog that guides you through a 3-step job comparison and synchronization process:

- Step 1: Choose database connections and comparison options.

- Step 2: Run the comparison engine. Review the comparison results.

- Step 3 (Optional): Review, edit, save and execute the job synchronization script.

Use the **Next** button at the bottom of the dialog to advance to the next step. Use the **Back** button to go back to the previous step. If you click the **Back** button while there is an active comparison operation in progress, the operation is automatically aborted.

To close the dialog, at any time click the **Close** button in the right bottom corner.

Comparison Options

Job Compare Rules – The set of comparison queries and templates designed for specific database server type and version that will be used for the selected Source and Target items. Typically the rule names are matching database server types but in fact they are just names and can be anything you like. It is important to choose rules that match your database type and version in order to produce correct comparison results and generate valid job synchronization scripts.
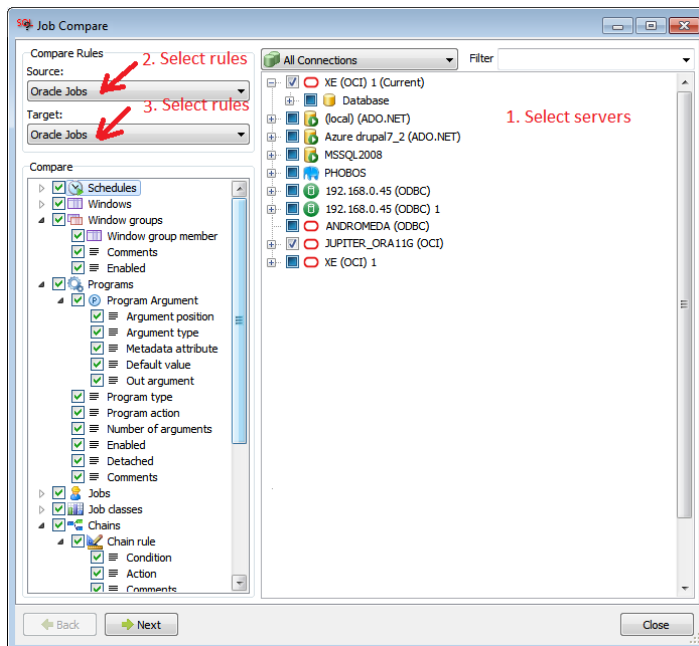
Compare options tree –The collection of job related object types and their attributes available for comparison. In the options tree you can choose which types of objects and their attributes you want to compare. If you want to ignore certain object types and/or some of their attributes, for example, Comments attribute of Program object, unselect that attribute in the **Compare** list.

Selecting Servers for Comparison

Connection and Name Filters

Multiple server connections are organized into logical connection groups. You can use the right-click menu to manage connections in place. To manage connection groups use either the SQL Assistant main Options dialog or the Multi-server Code Execution utility. For more information on managing connection groups and connection settings see Managing Connection Groups and Connection Settings topic in CHAPTER 15.

The Filter combo-box offers super-fast content filtering. Type the substring you want to use as a filter for database objects into the Filter box available above the object tree. Previously used filters are available in the drop-down portion of the Filter combo-box.

Tip: You can add new connections directly in the same dialog. Use the right-click menu in the connection tree. The same right-click menu can be used to modify saved connections.

Navigation

All server connections, databases, schemas, and schema objects are displayed in a single Object Tree. You can select only 2 items of the server type.

Object Tree Legend

The following types of checkboxes could be displayed in the object tree.

☐	Item is not selected for job comparison. This item is available for selection.
☐	Item is not available for selection because the item is not a server or you already have two servers selected for comparison. In the latest case, If you want to change the selection, first unselect one of the selected items and then select another item.
☑	Item is selected for job comparison.

Job Comparison Results

The second step in the Job Compare dialog is where all comparison results are shown.

The results dialog features 4 parts display of the results. To the top portion is split into 2 parts – the Source server objects tree and the Target server objects tree. The content of both parts can be filtered using the predefined filters at the top of the dialog to reduce the clutter and show only relevant differences. The bottom portion is split into 2 parts too. The left box shows DDL code of the object selected in the Source tree. The right box shows DDL code of the object selected in the Target tree. The code differences navigation map is displayed adjacent to the right side of the Target DDL code box.

Usage:
- The **Filter** options box provided at the top of the Job Compare dialog can be used to filter the comparison results.

  To quickly limit the list to new objects only, un-tick all checkboxes and tick only the **Objects in Source only** checkbox. To show the complete list, tick all checkboxes.

  The **Text** and **Filter Type** controls enable filtering comparison results by object names, by their attributes names, or by their attribute values.

- The resizer control separating the the Source tree and Target tree parts can be used to adjust the size of each part.

- The resizer control separating the tree and the bottom DDL code comparison results can be used to adjust the size of top and bottom parts.

- Use standard tree view control navigation methods to collapse, expand, and navigate items in the

Source object and Target object trees.

Color Coding

SQL Assistant uses color coding in the Source and Target object trees for visualizing the differences. Color coding provides a convenient way to see the comparison results at a glance. For your convenience the color legend is displayed at the top of the Job Compare dialog in the **Filter** options box in front of the filter selection checkboxes.

It also use color coding for highlighting code differences in the Source and Target DDL code boxes. The color coding method and color schema are the same as in the integrated Code Compare utility. See Color Highlighting topic in CHAPTER 25, Code Compare Utility for more details.

Action Legend

The following types of checkboxes could be displayed in the **Action** column.

☐　　Item is not selected for job synchronization. This item is available for selection. If not selected, it will not be included in the change synchronization script.

☐　　Item is not available for selection. This item is identical in the Source and Target databases and no synchronization is required.

☑　　Item is selected for job synchronization. The type of action CREATE, DROP, or ALTER is indicated in the Action column.

☑　　For an expandable item, this type of checkbox indicates that a "child" item has been selected in its tree branch for job synchronization.

Resizing Content
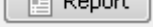
To resize the Job Compare dialog window, drag the resizer handle in the bottom-right corner of the window. See the screenshot at the beginning of the Working with SQL Assistant Popups topic for information on where to locate the resizer handle.

To adjust size of top and bottom parts within the dialog, drag the vertical split bar separating table and row lists.

Printing Comparison Report, and Exporting it to Excel and PDF

The Comparison Report provides the following:

- Printing comparison results or saving them as PDF versions.

- Documenting database changes

- Sharing results with coworkers who do not have the SQL Assistant software installed and cannot use the Job and Schema Compare tools.

- Using Excel interface applying advanced filtering, sorting, and commenting which are important tools in the decision making process for which changes to promote or ignore.

Printing and saving comparison reports is a two-step process. First you have to use the button shown in the Comparison Results view to display comparison results in a tabular format. Note that the button is visible in the Comparison Results view only. After that use the toolbar buttons at the top of the table to generate a printable version of the reports or to save the table to Excel file.

The **Print** button generates printable version of the report that includes report header with the database connection details, selected comparison options, etc.. and sends it to a printer of your choice.

The **Preview** button displays printable version of the report on the screen.

The **Save as PDF** button saves printable version of the report to a PDF file.

The **Save as XLS** button saves tabular version of the report to editable Excel file. Unlike other report actions, this output format is not optimized for printing. The report data is exported to Excel so that you can manipulate it as needed and then save or print only what you want.

To return back to the graphical **Comparison Results** view, click the **Back** button at the bottom of the report table.

To advance to the **Schema Synchronization** step, click the **Next** button

Job Synchronization

By default all differences are included in the scope of the synchronization action. In the Job Comparison Results you can exclude differences that you do not want to synchronize. To do so, un-tick checkboxes in the Action column on the right side of each row in the Source tree. When it is time to update the target database, this row will not be considered for any pending changes.

Ticking or un-ticking a checkbox in a group row is equivalent to ticking or un-ticking all differences in that group. You can use this speedy selection method at any group row, for example, you can use it at the database row to quickly unselect everything and then at a specific job row to quickly select that job only.

To quickly select or deselect everything displayed using current filters in the comparison result trees, right-click anywhere in the Source tree. The context menu will appear. Choose **Select All** or **Deselect all** commands as required.

After you have selected everything you want to synchronize, click the **Next** button. SQL Assistant will generate the synchronization script for the selected objects and attributes and will display the resulting script.

The generated synchronization script is displayed in a code editor connected to the Target database server. Review the synchronization script and correct it if required. Note that the script syntax is validated automatically for syntax errors, but it is not checked for any logical errors or other issues that only a developer can recognize. The syntax errors are displayed on the syntax bar adjacent to the right side of the editor. For details on how to use the Syntax bar see CHAPTER 20, SQL Syntax Checker.

For your convenience the script Structure View is displayed to the left of the generated script with quick navigation link to SQL statements in the script. For instructions on how to use the Structure View see CHAPTER 4, Code Structure View and Bird's Eye View.

Three actions are available. You can:

- Save the script to a file on your computer.

- Open the script in external editor. In case if you started the Job Compare tool from a development environment having SQL Assistant running as an add-on, the script will be opened in a new tab in the host development environment. Otherwise the script will be opened in SQL Assistant's integrated SQL Editor. See Professional SQL Editor IDE topic in CHAPTER 34, Integrated SQL Editors.

- Execute the script immediately.

Important Note: It is strongly recommended that you create a backup of the Target database before executing a job synchronization script so that you can roll back all unwanted changes later.

CHAPTER 29, Code Search & Replace in Files

Overview

SQL Editor Professional Edition is equipped with a file search and replace utility that provides advanced capabilities for file search and replace operations. You can use it to very quickly find and replace strings and text patterns across multiple files and folders. Regular expression engine and Template engines enable you to create sophisticated searches, preview replace, perform batch operations on multiple files, and more. The utility provides a number of additional functions to create automatic backup copies of files before they are modified, to log all operations, to run repeatable searches, and so on…

To open the file search and replace utility, in the SQL Editor click the Search menu and select **File and Replace in Files…**menu. The Find in Files dialog will open.
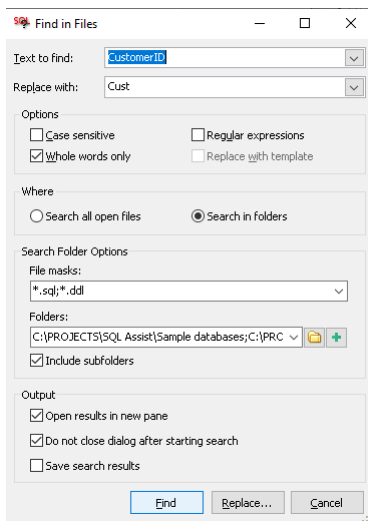
Supported File Search Options

Text to find - To run search for a new text string or expression, enter it in the box. To repeat search for strings that you searched for most recently, you can choose them from the drop-down list.

Replace with - To replace instances of the string or expression in the Text to find box with another string, enter the replacement string in the Replace with box. To delete instances of the string in the Text to find box, leave the Replace with field blank. To replace the text with one of the recently entered Replace with values, you can select them from the drop-down list.

Case-sensitive – By default the search operations are not case sensitive. This option converts the search to case sensitive search.

Whole words only - When selected, the search results will include only instances of the Text to find string that

are matched in complete words. For example, a search for "TableA" will return "TableA" but not "TableAB"

Regular expression - if selected, this option indicates that the value in the Text to find is a regular expression used for the string matching. The search and replace utility supports powerful PCRE regex engine, which provides the same syntax and functions as Perl regular expressions engine.: Please see http://perldoc.perl.org/perlre.html for the syntax description. It enables you to generate random values with predefined formatting, prefixes, and suffixes, as well as value ranges. For example regular expressions and syntax see Using Regular Expressions topic in CHAPTER 18, Generating Test Data.

Replace with template – This option can be used only if the Regular expressions option is selected for the search. It enables you to specify capture groups and replacement patterns.

A capture group delineates a sub-expression of a regular expression and captures a substring of an input string. You can use captured groups within the regular expression itself (for example, to look for a repeated word), or in a replacement pattern. For detailed information see http://perldoc.perl.org/perlre.html.

The following simple example demonstrates how to search for text containing straight bracket symbols "[" and "]" surrounded by one or more spaces or tabs and remove all white spaces around the brackets.

Text to find: `[\x20\t]*([\[\]]+)[\x20\t]*`

Replace with: `\1`

In the above example, a substring like "   [TableA   ]   " after the search and replace operations get replaced with "[TableA]"

Where: Search all open files – searches only in files currently opened in the SQL Editor.

Where: Search in folders – searches in files in one of more folders and their subfolders as specified in the additional search options below

File mask – the file mask for the files to search. To search for all files, specify *.*. Using a file type specific mask helps to avoid searching in large binary files and that helps to improves performance of search operations when scanning folders with many files.

> **Tip:** You can enter semicolon-separated list of file masks to search in multiple file types simultaneously, for example, *.sql;*.ddl.

Folders – use the drop-down menu to select folders you selected in previous searches. Use the yellow **Folder** button to the right of the Folders field to open a graphical Browse for Folder dialog and use to select a folder. Use the **Plus** button to the right of the Folders field to add additional folders to the folder search list. You can also enter semicolon-separated list of folders directly into the **Folders** field.

Include subfolders – check this option to enable recursive file search within selected folders and subfolders.

Open results in new pane – this option is selected by default. The results of each search operation are output to a new Search Results pane attached to the bottom of the SQL Editor main frame window. This enables you to run multiple searches with different search criteria and compare their results. If unchecked, a single Search Results pane is reused for all results, the results of the current search will replace the content of the previous search results.

Do not close dialog after starting search – This option keeps the Find in Files dialog window open after the search started. Use this option if you are going to run multiple searches.

Save search results – if selected, this will automatically save search results to XML or text file of your choice. You will be prompted for the file name after the search operation completes.

> **Tip:** If you don't select this option, you can still use right-click menu in the Search Results pane to save the results

Working With Search Results

The search results by default are shown a tree like structure indicating in which file and folder the results are found. The **Show Folders** toolbar button [icon] and the matching right-click menu can be used to display the results in a flattened format.
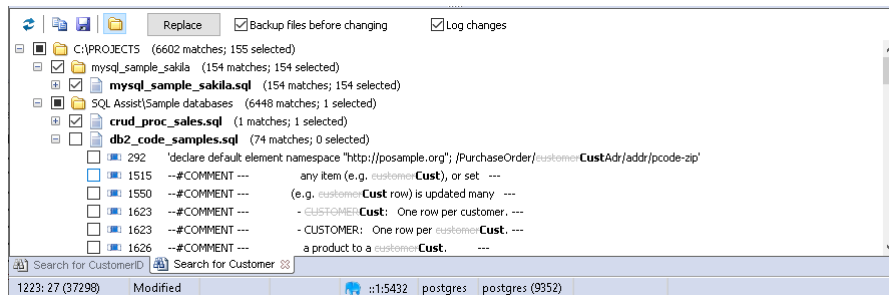
To save results to an XML or text file, click the **Save** toolbar button [icon]. You will be prompted to select the output file. If nothing is specifically selected in the results, all lines from the Search Results pane will be saved to the output file. If one or more lines are selected in the results, only their text will be saved to the output file.

To copy results to the Clipboard use the copy button [icon]. If nothing is specifically selected in the results, all lines the Search Results pane will be copied to Clipboard. If one or more lines are selected in the results, only their text will be copied to the Clipboard.

[icon] **Tip:** Hold down the Ctrl key to select multiple items in the Search Results pane. To select a range of items, use hold down the Shift key.

The display of the results is very similar for both Search and Search and Replace operations. In the Replace mode, a checkbox is shown in front of every folder, file and text match. You can use them to select and unselect which occurrences of the matching text you want to modify. In the Replace mode, the matches show both the text before the change using strikethrough font and also how it will appear after the change, as demonstrated on the above screenshot.

The match and selection counters at the right of each folder and file name show how many matches have been found and how many of them you have selected for the Replace operation.

To preview the found match in context, double-click its occurrence shown in the result. The file containing that match will open in the SQL Editor, and the matching text occurrence will be automatically found and highlighted for you in the text of the file.

[icon] **Important Note:** If you modify text directly in the editor, it's strongly recommended that you rerun the search to avoid conflicting file changes that may result in incorrect summary results.

Using the

Backing up files before performing batch updates

Use the **Backup files before changing** option available at the top of the Search Results pane to have SQL Assistant automatically save copies of all files you selected for text replacements. The copied files will have date time suffix added to their file name and their files extensions modified too. For example, a copy of file named proc_rename.sql, will be saved as proc_rename_backup_20200130_105015.~sql.

Note: The changes to the extensions help with preventing the saved copied from showing in recurring Search and Replace operations.

CHAPTER 30, Code Search & Replace in Databases

Overview

In the course of any database development, searching for data and code referring to certain objects is a daily routine. Searching is typically required across multiple database schemas, across multiple databases, and sometimes across multiple database servers. SQL Assistant provides the advanced search facility that enables you to search for both source code and table data across multiple domains. You can use full power of regular expressions to find exactly what you want as well as to use regular expressions to search for multiple variations of search terms, running so-called fuzzy searches. For example, you can use that to search for a column name referenced in dynamic SQL string within stored procedures, which otherwise cannot be found via regular database object dependencies and find all the stored procedures where that column is used. There are many other applications for the search functions.

SQL Assistant utilizes concept of Connection Groups for running multi-server code and data searches. You can define as many groups as you want and associate same or different connections with different groups. Database server connections and groups can be defined directly in the **SQL Assistant – Multi-server Code Search and Replace dialog**. See Managing Connection Groups and Connection Settings topic in CHAPTER 15, Executing SQL Scripts on Multiple Servers for more information.

Important Notes:

- In databases with case insensitive settings, code and data searches are case insensitive too. Searching for "TEST" and "test" will produce identical results. On contrary, in databases with case sensitive settings, searching for "TEST" and "test" will produce different results.

- Code Search and Replace function is designed for quick text replacing within the database code objects such as changing expressions and formulas coded within views and stored procedures, changing column references, and so on. Do not use it for renaming objects and columns in the database. Instead use SQL Assistant's Code Refactoring methods which allow you to quickly and safely reorganize and restructure your database code. Read CHAPTER 8, Smart Database Refactoring for more information on using code refactoring methods.

Running Fast Single-Server Code Search

For a quick on-the-sport source code search in the current database server you can use the light version of the code search function that is accessible via SQL Assistant's menu, look for **SQL Assistant → Search & Replace → Quick Code Search…** menu command. This command will open up **Search Code** dialog.

Note: In addition to source code of various procedural objects SQL Assistant queries attributes of tables and other objects searching for matching substrings in their names and also for matching column names.

1. In the Search Code dialog enter the substring you want to find within text of stored procedures, functions, views, triggers, and other types of source code objects.

2. This step is required if you are working with a multi-database server, such as SQL Server or Sybase ASE. Select one or more databases to search.

3. Choose how to search the source code. Two options are available in the **Code search method** drop-down:

- **Query system views –** This is a server side search based on querying system catalog views using regular SELECT statements containing the search criteria in their WHERE clause. This method is very fast, but it may fail in certain cases if the object source code is stored in chunks. For example, if a text of stored procedure is stored in the system catalog in 2 chunks and the search substring is split, in other words it begins in one chunk and ends in another, a simple SQL query will not be able to locate that substring occurrence.

- **Perform SQL code scanning –** This is a client side in memory search. SQL Assistant retrieves source code object definitions from the database server and performs in memory search for the specified substring.

4. Click the **Next** button to start the search.

5. After the search operation is complete, review search results. If another search is needed, click the **Back** button, change search criteria and run it again, otherwise click the **Close** button to close the Search Code dialog.

   Note: The display of code search results and the navigation is virtually the same as code display and navigation in Smart Database Refactoring dialogs. See the Layout section in Refactoring Wizard Dialog topic in CHAPTER 8, Smart Database Refactoring.

Searching Code Across Multiple Servers

The multi-server code search method can be used to search code across multiple servers, databases, and database schemas.

1. In SQL Assistant's menu, select **Search & Replace → Code Search & Replace (Multi-server)…** menu command. This command will open up **Multi-server Code Search and Replace** dialog.
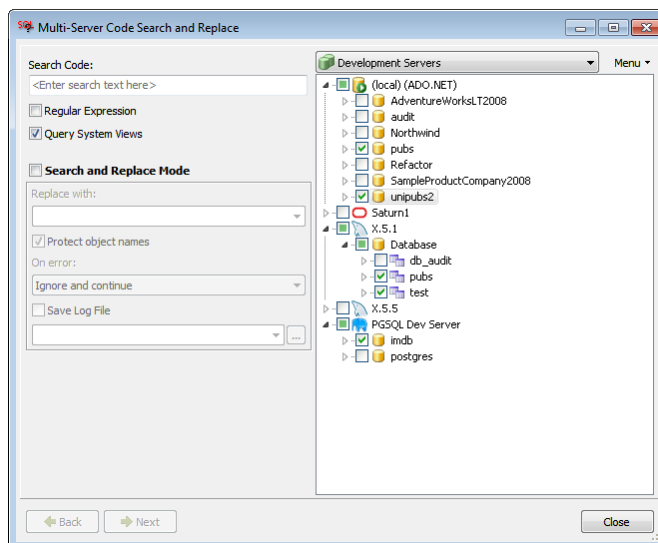
2. Enter substring or regular expression that you want to search for. If you want to use regular expression, tick **Regular Expression** checkbox below the search field. For example, to find all

objects and attribute names containing "TextToFind" as a separate word, you can enter "\bTextToFind\b" into the search box without double quotes.

3. Choose how to search the source code. If you check the **Query system views** checkbox, SQL Assistant will run server side code search querying system catalog views using regular SELECT statements containing the search criteria in their WHERE clause. If you leave the **Query system views** checkbox unchecked, SQL Assistant will perform client side in memory search retrieving each object code and reverse engineering its DDL.

    Note: The **Query system views** search method is much faster than reverse engineering DDL of each object, but it does not allow using regular expressions.

4. On the right side of the dialog select servers, optionally select specific databases and specific schemas if you want to narrow the search scope.

    Note: Use the right-click menu to add more connections to the list of severs or to modify existing connections. See Managing Connection Groups and Connection Settings topic in CHAPTER 15, Executing SQL Scripts on Multiple Servers for more information on managing database server connections.

5. Click the **Next** button

6. After the search operation is complete, review search results. If another search is needed, click the **Back** button, change search criteria and run it again, otherwise click the **Close** button to close the Multi-server Code Search dialog.

    Note: The display of code search results and the navigation is virtually the same as code display and navigation in Smart Database Refactoring dialogs. See the Layout section in Refactoring Wizard Dialog topic in CHAPTER 8 for more information.

Replacing Code Across Multiple Servers

The multi-server database code search & replace method is an extension of the multi-server database code search method. It does require that you run the search operation first and allows you to review the search results before running mass replacing. It also allows you to pick and choose objects and specific code occurrences for replacing and skip what does not need to change.

The **Multi-server Code Search and Replace dialog** guides you through a 3-step code replacement process:

*   Step 1: Enter required search and replace parameters and options.

*   Step 2: Preview and edit the proposed changes.

*   Step 3: Execute the replace operations and review the output results and logs.

Tip: If the replacement operation fails for some reason, you can click the Back button to go back to step 2 where you can review and edit the code for any failed items. After making necessary corrections, click the Next button again to rerun the process.

Warnings: The multi-server database code search & replace method is not source control aware. If you are using SQL Assistant's source code control interface, it is recommended that after running the database code replace operations you refresh source code control workspace files from the database. For more information see Getting Source Code from Database Server topic in CHAPTER 23, Database Source Code Control Interface.

To run the database code search and replace operation:

1. Follow steps 1 to 4 described in the previous Running Code Search Across Multiple Servers topic for instructions on running the search and the supported search options.

2. Select **Search and Replace Mode** checkbox.

3. Enter replacement string into **Replace With** field.
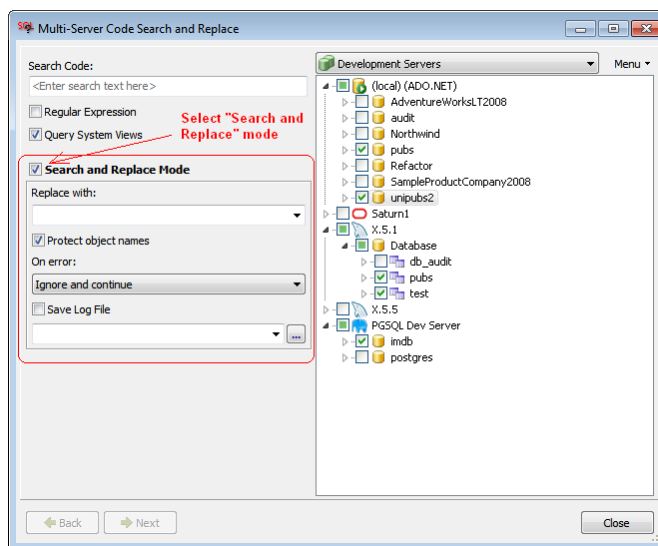
4. Select **Protect Object Names** field to improve the safety of code search and replace operations. This option instructs SQL Assistant to ignore changes in the first line of CREATE OR REPLACE and ALTER commands to prevent the original object name. For example, if this option is selected and you are replacing *TestString* with *TestString2*, and a procedure named *ProcedureTestStringHere* is found, containing *TestString* in the code, SQL Assistant will not replace TestString in the procedure name in the ALTER PROCEDURE *ProcedureTestStringHere* AS … command.

5. Enter optional error handling and process logging options.

    On error – this option instructs SQL Assistant what to do in case a database error occurs during replace operations, whether to abort the processing or continue with the next table in the table list. The default is not to abort the work and continue after errors.

    Save Log File -– If this option is checked, SQL Assistant writes processing status messages to the log file specified in the **Log** option.

    Log – The name of the output log file. This name must be specified if **Save Log Option** is checked.

6. Click the **Next** button to start the search.

7. Review search results. The code search results will appear similar to the example results on the following screenshot of the Multi-server Code Search and Replace dialog. Choose what to replace and what not to replace. The display of search results and the navigation is described in detail in the following topic "Multi-server Code Search and Replace dialog."

8. Click the **Next** button to run the replace operation for the selected objects and code occurrences.

9.   Review replace operation logs. If satisfied with the results, click the **Close** button to close the dialog, or click the Back button to run another search and replace.

Multi-server Code Search and Replace dialog

Layout

Step 2 of the Multi-server Code Search and Replace dialog is where you will spend most of your time reviewing proposed changes and, if necessary, providing additional input. In Step 2, the dialog is split into three parts: the code replacement operation description in the top section of the dialog, the object tree navigator on the left side, and the SQL editor on the right side. Items in the object tree are represented by icons indicating their object type and text labels as illustrated in the screenshot below. See the Object Tree Legend table following the screenshot for a key to icon meanings. Checkboxes in front of item icons can be used to select and unselect objects you want to change.

Object Tree Legend

The following types of checkboxes with optional overlay icons could be displayed in the object tree.

- Item is not selected for code replacement, no action will be taken for this item.

- Item is selected and ready for code replacement.

- Item may need code replacement, but SQL Assistant is unsure how to replace it correctly. Your input in the SQL editor is required before the object code can be altered in the database. You should select this item and manually correct the code in the SQL editor box before advancing to Step 3.

- For an expandable schema or database item, this type of checkbox indicates that only a subset of "child" items has been selected in the object tree branch for this item. For a simple item, this type of checkbox indicates that changes in the database completed successfully. This type of checkbox displays only if you click the Back button to return to Step 2 to review and edit an initial code replacement operation.

- Item changes in the database failed. This type of checkbox displays only if you click the Back button to return to Step 2 to review and edit an initial code replacement operation.

Navigation Status Bar Legend

The navigation bar is available on the right side of the Multi-server Code Search and Replace dialog in step 2. The status bar provides color coded interactive location indicators enabling you to quickly jump to a line of code where a change is pending. Click on a location indicator to jump to the associated line of code.

Embedded SQL Editor

The Multi-server Code Search and Replace dialog features an embedded SQL Editor you can use to modify code before it is altered in the database. For more information on the supported features, see CHAPTER 3, Code Assistants and SQL Intellisense.

Note that the editor supports a dual-mode interface:

- **Read-only mode with change highlighting** of proposed changes for the current refactoring operation and selected item. In this mode, the right side status bar is used for quick change navigation. Color-coded marks are used to indicate ending changes and other sensitive elements in the code.

- **Edit mode**, a full featured SQL editor with SQL Intellisense and other editor features. This mode is activated automatically as soon as you start typing anything. In this mode the right-hand side status bar is used to show syntax check results. To switch back to **Read-only mode with change highlighting,** select a different object in the object tree on the left.

Important Notes:

- If you start modifying code of an object that is not selected in the object tree, SQL Assistant will automatically select it.

- You should review all selected objects and their pending changes before you click the **Next** button on the Multi-server Code Search and Replace dialog.

Tip: The Multi-server Code Search and Replace dialog window is resizable. A border of the window can be dragged to change the size of the window. You can also use the dialog window **Maximize** button to open it full screen and allow more room for the dialog controls, including the SQL Editor

Process Log Legend

The process log is displayed in step 3 of the Multi-server Code Search and Replace dialog. The following icons indicate operation status conditions in the refactoring log.

✅ The operation or step completed successfully

⬛ The operation or step failed

❌ Indicates code lines that generated specific error messages returned by the database server during object alteration.

Using Context SQL Search

Please see CHAPTER 37, Improving Code Reusability for the description and instructions on using advanced context based code search methods.

CHAPTER 31, Data Search & Replace

Overview

In the course of any database development, searching for data and code referring certain objects is a daily routine. Searching is typically required across database schemas, across databases, and in many instances across database servers. SQL Assistant provides the advanced search facility that enables you to search for both source code and table data across multiple domains. You can use full power of regular expressions to find exactly what you want as well as to use regular expressions to search for multiple variations of search terms, running so-called fuzzy searches.

SQL Assistant utilizes concept of Connection Groups for running multi-server code and data searches. You can define as many groups as you want and associate same or different connections with different groups. Database server connections and groups can be defined directly in the **SQL Assistant – Multi-Server Data Search & Replace** dialog. See Managing Connection Groups and Connection Settings topic in CHAPTER 15, Executing SQL Scripts on Multiple Servers for more information.

Important Notes:

- In databases with case insensitive settings, code and data searches are case insensitive too. Searching for "TEST" and "test" will produce identical results. On contrary, in databases with case sensitive settings, searching for "TEST" and "test" will produce different results.

- The search and replace operations can be applied to database tables and views. This includes full range of tables and views, including "proxy" tables referencing external files and data sources, materialized views, and so on... However only updatable tables and views allow data replacements. An attempt to replace data in a read-only table or view will lead to a database side error. It is your responsibility to select only updateable objects for data replace operations.

Searching Data Across Multiple Servers

The multi-server data search method can be used to search for occurrences of specific values across multiple servers, databases, database schemas, and tables. Before you use this search method, be sure to check all required database server connections have been already entered and saved in SQL Assistant settings. See Managing Connection Groups and Connection Settings topic in CHAPTER 15, Executing SQL Scripts on Multiple Servers for more information.

1. In SQL Assistant's menu, select **Search → Search Data (Multi-server)…** menu command. This command will open up **Multi-server Data Search** dialog.
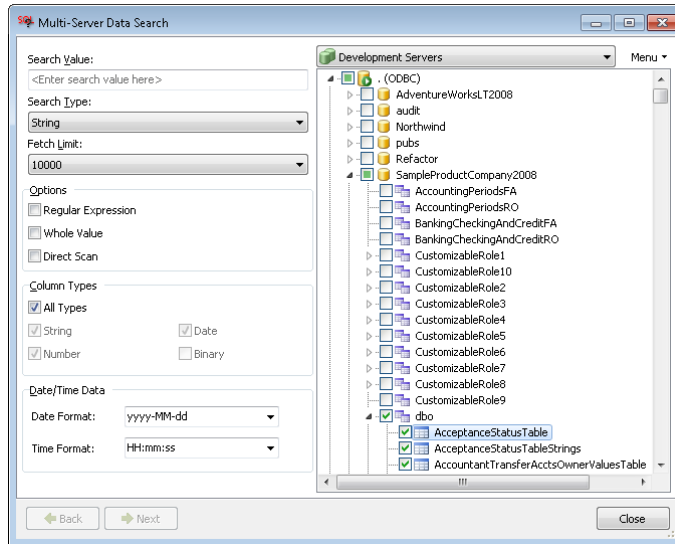
2.   Enter substring or regular expression that you want to search for. If you want to use regular expression, tick **Regular Expression** checkbox in Options box below. For example, to find all objects and attribute names containing "TextToFind" as a separate word, you can enter regular expression "\bTextToFind\b" into the search box without double quotes.

3.   In the **Search Type** drop-down choose data-type of the value to search for. The default is String, but you can also search for dates, times, and numbers.

   Notes:

- Do not confuse Search Type and column data types. If Search Type is set to String, SQL Assistant will automatically convert values in all searchable columns to their string representations. This for example can be used for partial date, time, and numeric value searches. This will also allow you to use regular expressions against all column types

- Partial matches and regular expressions are supported in **String** type searches. For all other data types the search is always performed for the whole value.

4.   Specify **Fetch Limit** or leave the default value. The fetch limit is the maximum number of matching records per table SQL Assistant will retrieve from that table in search results. For example, if the limit is set to 100 and 1 million records have been found satisfying the search parameters ina particular table, only the first 100 records from that table will be returned and displayed in search results. This is to prevent run-away data retrieval.

5.   Complete additional search options and date-time formats.

   Regular expression – check this checkbox if the search string is a regular expression.

   Whole Value – check this checkbox to disable partial and substring matches.

   Direct scan – this option is reserved for later user and cannot be changed.

   Date-format – the format to use for converting date values and date portion in date-time values to their string representations

   Time format – the format to use for converting time values and time portion in date-time values to their string representations

   Note: See Handling Date and Time Values topic in CHAPTER 13, Scripting, Exporting, and Importing Data, for details on supported date and time value format masks.

6. On the right side of the dialog select servers, optionally select specific databases and specific schemas if you want to narrow the search scope.

    Note: Use the right-click menu to add more connections to the list of severs or to modify existing connections. See Managing Connection Groups and Connection Settings topic in CHAPTER 15, Executing SQL Scripts on Multiple Servers for more information on managing database server connections.

7. Click the **Next** button to start the search process.

8. After the search operation is complete, review search results. The tables with found matches are displayed on the left side of the dialog in the Object Tree.
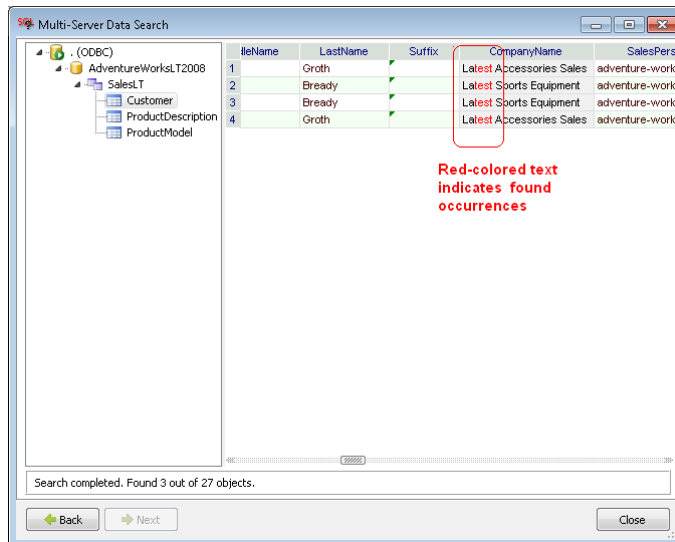
    Click tables in the tree to display matching records from these tables.

    Notes: Note that the red-colored text within table cells with gray background indicates found text occurrences. In certain cases you may need to scroll the view to locate colored cells. In case the cell value contains multi-line value and the match is not in the first visible line, no red text will be visible. Double-click the cell to see complete multi-line value ina separate popup window..

9. If another search is needed, click the **Back** button, change search criteria and run it again, otherwise click the **Close** button to close the Multi-server Data Search dialog.

Replacing Data Across Multiple Servers

The multi-server data search & replace method is an extension of the multi-server data search method. It does require that you run the search operation first and enables you to bulk replace all found values or pick and choose what to replace and what not. Refer to the previous Running Data Search Across Multiple Servers topic for instructions on running the search and search options.

In order to activate the Search and Replace mode, the **Replace** option must be enabled. The follow example screenshot shows the location of that option on the **Multi-server Data Search & Replace** dialog.
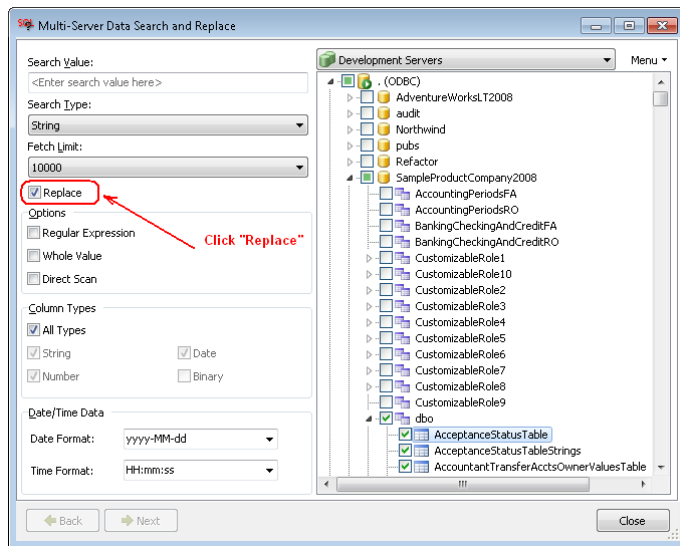
Make sure you tick the **Replace** checkbox before you start the search. This option instructs SQL Assistant to prepare the found data for updates.

The data search results will appear similar to the example results on the following screenshot of the **Multi-server Data Search & Replace** dialog.
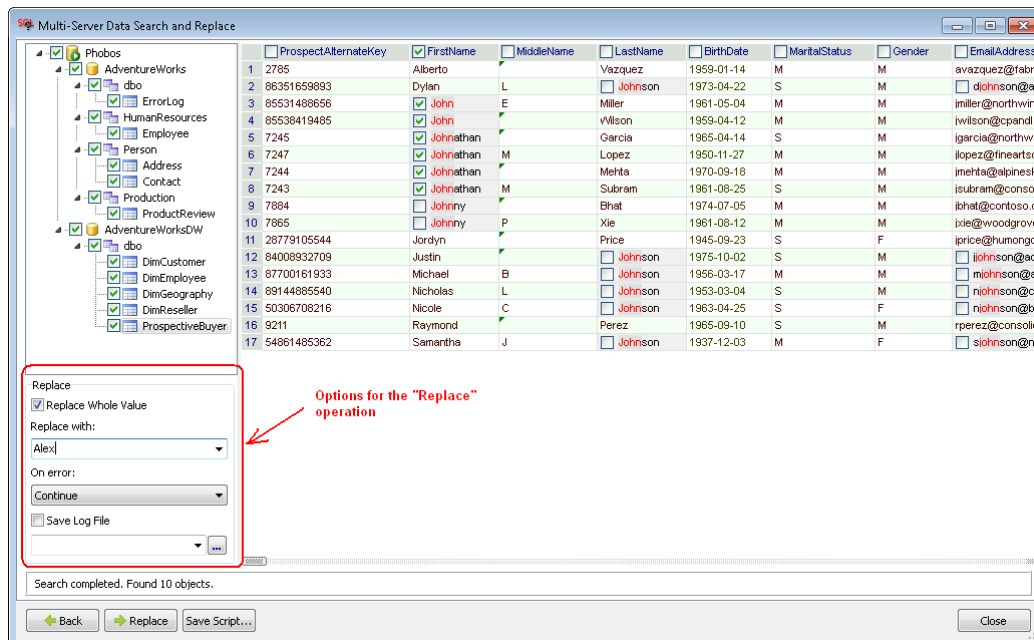
The **Multi-server Data Search & Replace** dialog hosts 3 important controls:

The **Table tree** contains references to the tables with data matching your search criteria. Click each table in the

table tree to review the found data matches. Clear checkboxes appearing to left of tables whose values you do not want to touch.

The **Replace** box provides options for mass replace operations across multiple servers and multiple tables.

Important Note: The Replace operation is applied to the selected tables, columns and cells only.

The **Replace** box provides the following options:

- **Replace Whole Value** – tick this checkbox if you want to replace the entire data value in the cells with the matching data. Do not confuse this with the Search Whole values option. For example, you can search for text columns containing substring "John" and find 3 values: "King John", "John", "John Who" and specify "Alex" as the new value to replace with. If **Replace Whole Value** checkbox is ticked, all 3 found matching values will be completely replaced with "Alex" and result will be "Alex", "Alex", "Alex". If the checkbox is not ticked, substring replacement will be performed and the result will be "King Alex", "Alex", "Alex Who"

- **Replace with –** this is the replacement value.

- **On error –** this option instructs SQL Assistant what to do in case a database error occurs during replace operations, whether to abort the processing or continue with the next table in the table list. The default is not to abort the work and continue after errors.

- **Save Log File -–** If this option is checked, SQL Assistant writes processing status messages to the log file specified in the **Log** option.

- **Log –** The name of the output log file. This name must be specified if **Save Log Option** is checked.

Data Preview and Column and Cell Controls – The right side of the **Multi-server Data Search & Replace** dialog is occupied by a special control is used for both previewing the search results and controlling the scope of replace operations. It is very important to understand well how to use this control for correctly performing the replace operations. The following controls can be used:

- Checkboxes in individual cells can be used to select which cells to replace to touch and which not. Note that if a cell has no matching value, ticking or clearing the checkbox in that cell has no impact on the replace operations.

- Checkboxes in the column headers can be used as shortcuts to quickly tick or clear all checkboxes in all cells belonging to that column. Note that if a column has no cells with matching values, ticking or clearing the checkbox in that column has no impact on the replace operations.

Working with Data Search Results Interface

NULL Values

SQL Assistant renders cells containing NULL values as empty cells. A little green mark is displayed in the top-left corner of a NULL cell to differentiate it from cells containing empty string and spaces only.

If you mouse-over a cell with a green mark, you will see a popup hint with word "NULL."

Long and Multi-line Text Values

Long values exceeding column width are rendered partially, as much as fits the width. 3 dots (also called ellipses) are rendered in each cell with non-fitting data to indicate the data overflow effect. You can resize the column, to see the entire value.

For multi-line text values containing end-of-line and carriage return characters, only the first line is displayed. A little red mark is displayed in the top-left corner of a cell containing multi-line text to differentiate it from cells containing single-line values. You can see the entire text value if you double-click the cell. The text will be displayed in a separate Cell Value window described in the next topic.

If you mouse-over a cell with a red mark, you will see a popup hint with instructions for displaying the entire value.

Tip: To quickly resize a column so it fits the entire content in all cells, double-click the right-edge of the column header. SQL Assistant will calculate the required width and resize this column as needed.

Expanded Cell View

The fetched query results do not always display well in the Data Preview grid. Some fetched values can be long or have multiple lines. For a better view of the results of such queries, double-click the cell whose value you want to see. This will open the **Cell Value** popup window. To see a value in some other cell, double-click that cell. There is no need to close the **Cell Value** popup window, it will refresh automatically.

Tips:

- The title bar of the **Cell Value** popup window indicates row and column of the cell containing the value.

- The popup window can be moved and resized as needed. It will remember its size and position and stay on top of other windows until it is closed.

- If the value displayed in the **Cell Value** popup window contains long lines, tick the **Line Wrap** check box to make text wrap at the right edge of the display area.

Scrolling Content

Use standard scroll bars available in the **Data Preview** control to scroll the content. In addition, you can use regular keyboard navigation keys and the mouse wheel control if such is available.

Resizing Content

To resize the **Data Search & Replace** window, drag the right-bottom edge of the window.

To resize individual columns in the data grid, drag the right-edge of the column header left or right. Note that when you place mouse pointer over the right edge of a column header the cursor shape changes to resize shape as on the following screenshot.

Make sure the cursor takes the right shape before dragging the column edge.

Tip: When column width is too narrow to fit the content, 3 dots (also called ellipses) are rendered in each cell with non-fitting data to indicate the data overflow effect. To quickly resize a column so it fits the entire content in all cells, double-click the right-edge of the column header. SQL Assistant will calculate the required width and resize this column as needed.

CHAPTER 32, Visual Bookmarks

Overview

The visual bookmarks feature offers innovative methods of quickly navigating code. Visual bookmarks capture small screenshots of the area of the target editor where bookmarks are made which provide instant access to that area with a single click on the bookmark icon. They also provide a quick way of previewing code using simple mouse-over events. Visual bookmarks make it unnecessary to memorize nameless bookmark numbers or to scroll many pages of code to find a small bookmark marker displayed at the right edge of the editor window.

Visual Bookmarks complement other quick code visualization and navigation options provided by SQL Assistant. For more information see the Overview of Code Structure View and Overview of Bird's Eye View topics in CHAPTER 4.

Visual bookmarks are represented by small numeric icons typically displayed on the right hand side of the editor. When the bookmarked code line appears within the visible area of the editor, it is highlighted using the distinct color associated with the bookmark. If the bookmarked line is above the visible area of the editor, its docked icon appears in the top right corner of the screen. Likewise, when it is below the visible area, its docked icon appears in right bottom corner of the screen. As you scroll the code, bookmark icons automatically move with their lines and can change their position depending on the direction of the scrolling.

Each bookmark has a distinct number and color. The same color is used for highlighting the bookmarked line and the background color of the bookmark icon.

When the mouse pointer is placed above the docked bookmark icon, a small Help window containing a screenshot of the bookmarked area appears near the pointer. When the bookmark icon is clicked, SQL Assistant scrolls the content of the editor window and places cursor at the beginning of the bookmarked line.

The following example screenshots demonstrate the appearance of visual bookmarks. If you are reading this manual in electronic format, enlarge the view to see more details.

Bookmark icons and lines
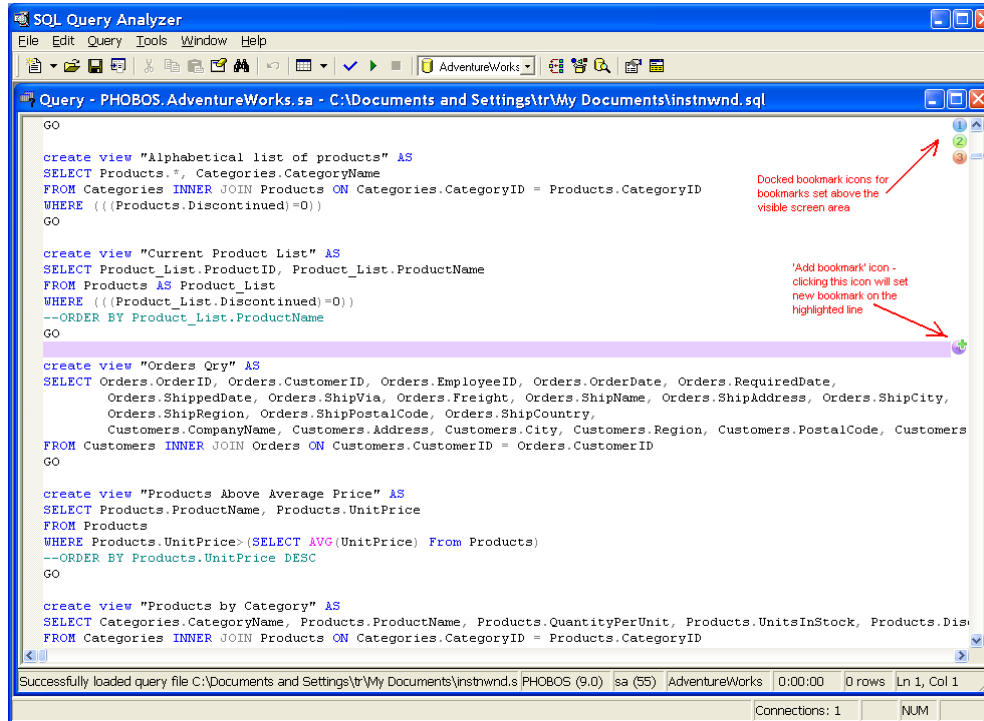
Bookmark mouse-over Help windows
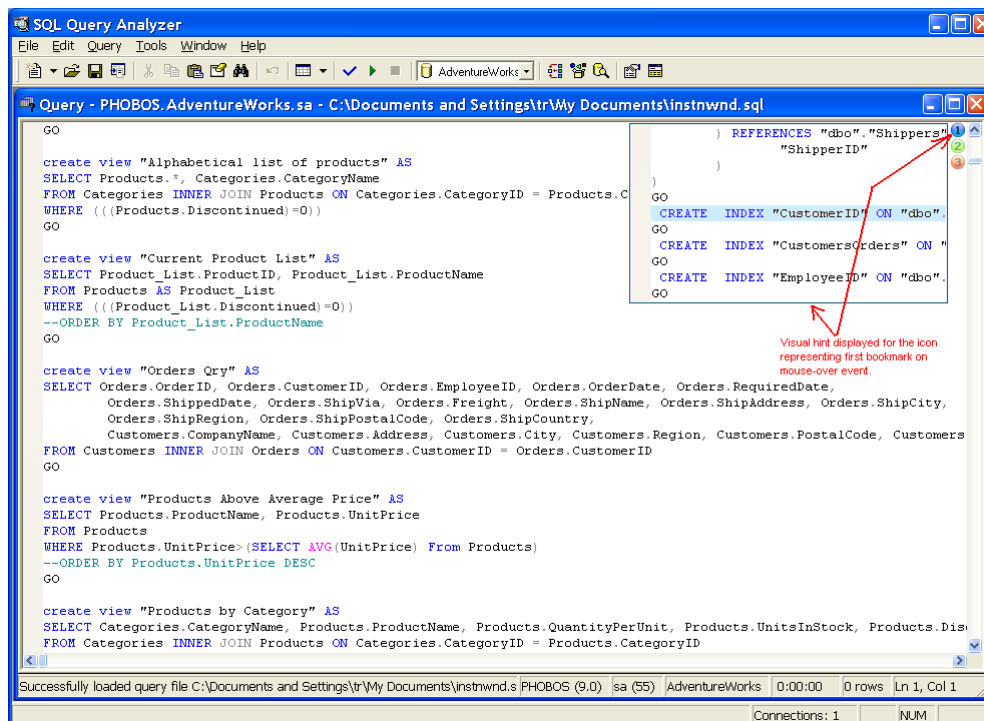
Note that the Help window border is the same color as the bookmark icon.

Bookmarks Enumeration

The first nine bookmarks in a single script are enumerated using single digits ranging from 1 to 9 and are marked with different colors. If there are more than nine bookmarks in a script, the additional bookmarks are displayed using unnumbered gray colored circles.

To-Do Tasks, and Other Special Tags

SQL Assistant automatically scans scripts loaded into SQL editor and recognizes comments containing the following substrings.

- TO DO
- TODO
- BUG
- HACK

The substring match is case-insensitive and can be located anywhere within comment lines, as in the following example:

```
/* to do: implement check for NULL values later */

SELECT SalesPersonID, SalesOrderID, YEAR(OrderDate) AS SalesYear
FROM Sales.SalesOrderHeader
WHERE SalesPersonID IS NOT NULL

-- Here we search for zero amount in the Balance column
-- Ask Mark what to do in case of NULL values
```

Blue marks on the Syntax Bar indicate the location of tags within the script, as shown in the following screenshot.
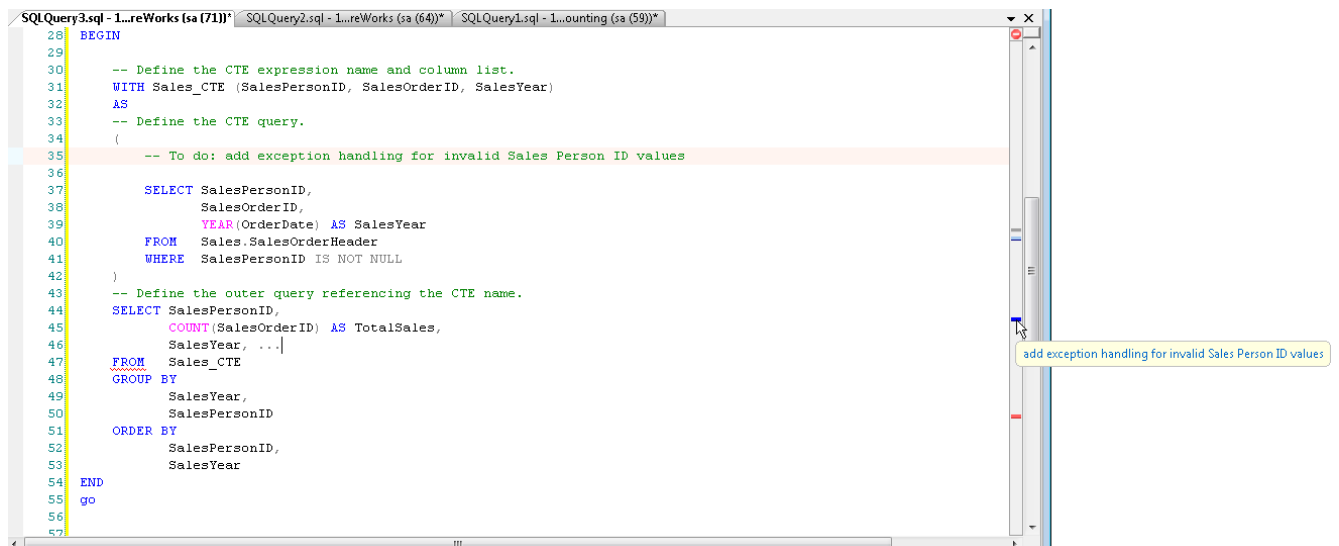
If you rest the mouse over a blue mark on the syntax bar, a balloon window appears near the blue mark displaying a portion of the comment text.

If you click a blue mark, SQL Assistant scrolls the content of the editor window To-Do line in the code and positions cursor at that line.

Working with Visual Bookmarks

Creating Bookmarks

The following instructions describe the default Visual Bookmarks behavior. For details on customizing bookmark behavior and shortcuts, see the [Customizing Bookmark Handling Options](#) topic in CHAPTER 48, Customizing SQL Assistant's Behavior.

To add a new bookmark with a single mouse click:

1. Make sure the code line where you want to add new bookmark is visible.

2. In that line move mouse pointer horizontally to the right edge of the editor window. An *Add Bookmark* icon will appear along the edge and the line under the mouse pointer will be highlighted. Note that the *add bookmark* icon looks similar to other bookmark icons except that it has a plus sign symbol painted over the icon.
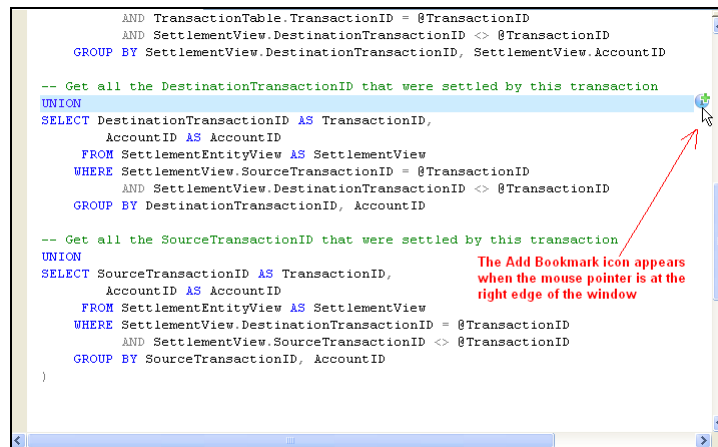
3. Click the *Add Bookmark* icon.

    Note: The bookmark numbers are assigned automatically as sequential numbers.

To add a new bookmark using hot keys:

1. Make sure the line where you want to add new bookmark is visible.

2. Place the edit caret on the line to be bookmarked.

3. Make sure the Num Lock is keyboard state is set. Press the Alt key along with one of the numeric keys on the extended keyboard., for example Alt+Num1. Note that the selected numeric key defines the numeric bookmark id. You can use key combinations ranging from Alt+Num0 to Alt+Num9.

To add a new bookmark using menus:

1. Make sure the line where you want to add new bookmark is visible.

2. Place the edit caret on the line to be bookmarked.

3. Right-click on the selected line and select the SQL Assistant → Bookmarks → Add Bookmark menu command. If top-level menus integration is enabled, the same result can be achieved using the top-level menus in the editor. See the Using Context and Top-level Menus topic in CHAPTER 3 for more details about SQL Assistant menu integration options.

Removing Bookmarks

The following instructions are valid with default settings for Visual Bookmark behavior. For details on customizing bookmark behavior and shortcuts, see the Customizing Bookmark Handling Options topic in CHAPTER 48, Customizing SQL Assistant's Behavior.

To remove a bookmark with a single mouse click:

1. Make sure the bookmarked line is visible and highlighted. The bookmark icon should be visible at the right edge of the editor screen

2. Rest the mouse over the bookmark icon. Note that the bookmark icon changes to the *Remove Bookmark* icon, which looks similar to the bookmark icon except that it has a minus sign symbol painted over the icon.

3. Click the *Remove Bookmark* icon to remove the bookmark.

To remove all bookmarks:

Right-click anywhere in the target editor workspace, then choose the SQL Assistant → Bookmarks → Remove Bookmarks menu command. If top-level menus integration is enabled, the same result can be achieved using the top-level menus in the editor. See the Using Context and Top-level Menus topic in CHAPTER 3 for more details about SQL Assistant menu integration options.

Jumping to Bookmarked Line

To jump to a bookmarked line with a single mouse click:

Click the associated bookmark icon. SQL Assistant will automatically scroll the editor and place the edit caret in the beginning of the bookmarked line.

To jump to a bookmarked line using hot keys:

Simultaneously click the Alt key and the numeric that matches the bookmark number. For example, use Alt+1 to jump to the first bookmark, use Alt+2 to jump to the second bookmark, and so on.

Loading Saved Bookmarks from Comments

SQL Assistant supports specially formatted code comments that it recognizes as "portable" bookmark markers. You can move or rename a file, or updated procedural code in the database, and its portable bookmarks will remain intact, while the regular bookmarks get lost as soon as you close the script in the target editor.

Bookmark comments are recognized by the presence of the word "bookmark" enclosed in square brackets (see the example code below). When a file is opened in the editor or a unit of procedural code is loaded into editor from a database, SQL Assistant scans comments in the loaded code for the presence of the [bookmark] pattern. It automatically creates visual bookmarks in code areas containing the these markers. Following are examples of valid bookmark markers.

```
/* a  multi-line comment here [bookmark] more text
here */
-- a single line comment here [bookmark] more text here
-- any comment here [bookmark:3] any comment here
-- [bookmark:1]
```

The markers can be in either of two formats:

[bookmark] – this marker causes SQL Assistant to assign the next available number to the bookmark identifier.

[bookmark:n] – this marker causes SQL Assistant to create a bookmark identifier with a numeric value of "n," where n is a digit from 0 to 9. The user specified number is assigned to the bookmark unless that number is already assigned to another bookmark, in which case, the user-specified value is ignored and the next available number is used. Note that only numeric values from 0 to 9 can be used. All additional bookmarks are displayed without numeric identifiers.

CHAPTER 33, To-Do Tasks and Reminders

Overview

The Tasks and Reminders graphical interface provides a central place for accessing all saved bookmarks and To-Do tasks. For details on how to save bookmarks in specially formatted comments and To-Do tasks, read the Overview and Loading Saved Bookmarks from Comments topics in CHAPTER 32, Visual Bookmarks. The interface allows you to see items across multiple projects and database systems and to quickly access them with a simple double-click action.

SQL Assistant uses several techniques for locating bookmarks and to-do tasks within the code comments saved in files and in databases. It uses non-interrupting background processing to scan files in the user selected file system folders and databases. This process monitors selected locations for file and database changes, and updates the Tasks and Reminders view automatically if it finds that changes have been made in these places.

The **Tasks and Reminders** pane can be opened from SQL Assistant's menu in the target or by using the right-click menu in the SQL Assistant system tray application. In either case, select the **SQL Assistant → Tasks and Reminders → Show** command from menu. To use the target editor's menus, the menu integration option must be enabled. For details, see the Manually Invoking SQL Assistant Popups topic in CHAPTER 3.
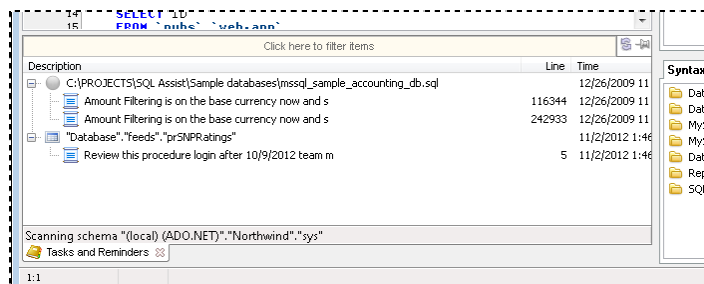
Before you can use the Tasks and Reminders feature, you must let SQL Assistant know where to search for bookmarks and To-Do items. Use the **SQL Assistant → Tasks and Reminders → Options** menu command, or right click menu in the **Tasks and Reminders** pane, and choose the **Options** command. The Tasks and Reminders Options dialog opens (see the "Options" topic below).

By default, the monitoring process checks for file changes in near real time and wakes up every 30 seconds to rescan modified database objects for new changes. To refresh the **Tasks and Reminders** pane manually, select the **SQL Assistant → Tasks and Reminders → Update** menu command. Alternatively, you can right-click in the **Tasks and Reminders** pane and choose the **Update** menu command from the popup menu.

Options

The Tasks and Reminders Options dialog allows you to specify in which locations SQL Assistant should search for saved bookmarks and To-Do items.
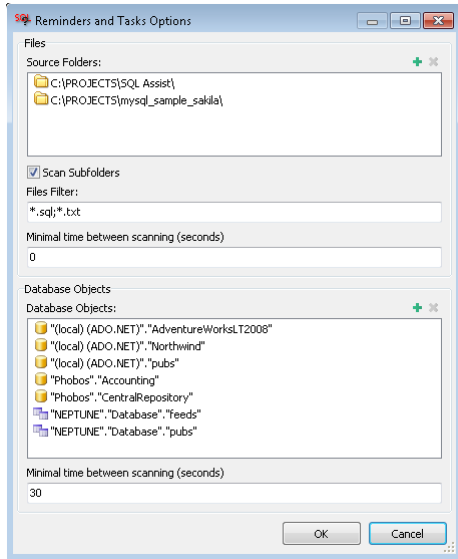
The following options can be configured using the Options dialog:

Files section

> **Source Folders** – Specifies one or more folders that SQL Assistant will search for source code files. This list is used in conjunction with the File Filter and Scan Subfolder options.
> Use the 🞤 🗙 icons above the top right corner of the Source Folders list box to add new folders to the list or delete previously added folders.
>
> **Scan Subfolders** – Specifies that SQL Assistant will recursively scan files in subfolders of the selected Source Folders.
>
> **File Filter** – Specifies a comma-separated list of file masks for the files to be scanned in Source folders and subfolders. You can use standard * and ? wildcard characters. By default, SQL Assistant only checks files with the .SQL and .TXT extensions.
>
> **Minimal time between scanning (seconds)** – The time interval in seconds for the monitoring process to rescan files in the Source folders. By default this value is set to 0, meaning that SQL Assistant will check modified files only using near real-time file monitoring methods. If this value is set to non-zero value, it will rescan all relevant files at the specified time interval. The actual rescan interval may vary if scanning of folders and files takes a long time.

Database Objects sections

> **Database Objects** – Specifies the list of databases, schemas, and individual objects that SQL Assistant will scan. You can add entire databases or individual schemas or objects to the list.
> Use the 🞤 🗙 icons above the top right corner of the Database Objects list box to add new objects to the list or delete previously added objects.
> 📌 **Note:** Only database objects accessible using database connections registered in SQL Assistant

settings can be added to the list. If a required database connection is not listed, cancel the dialog and add that connection to SQL Assistant options. See the Managing Database Connections topic for specific instructions on registering new database connections

Minimal time between scanning (seconds) – The time interval in seconds for the monitoring thread to rescan databases. By default this value is set to 30, meaning that SQL Assistant will check modified database objects every 30 seconds. If this value is set to zero value, it will not scan databases automatically; only a manual refresh will trigger new scan.

Working with Tasks and Reminders Interface

Navigation

The Tasks and Reminders view contains combination tree-view controls that allow you to see all details at a glance and yet quickly collapse folders, files, databases, and objects you do not want to see. Use standard tree view navigation methods to collapse and expand individual branches

Opening Source Files and Navigating to Bookmarked Lines

Simply double-click the specific item you want to jump to. SQL Assistant will open the source file or database object in the current target editor and scroll the content to the selected line with a bookmark or To-Do comment.

Filtering the Tasks and Reminders List

Click the yellow bar at the top of the Tasks and Reminders window and start typing the substring you want to find in the comments of bookmarked lines. The Tasks and Reminders list will be filtered and only items containing the matching text will be displayed.

To clear an existing filter, erase the text from the filter box.

Scrolling Content

Use the standard scroll bars in the **Tasks and Reminders** window to scroll the content. You can also use regular keyboard navigation keys and the mouse wheel.

Resizing Content

To resize the **Tasks and Reminders** window, drag the top edge of the window up or down. Note that when you place mouse pointer over the top edge of the **Tasks and Reminders** window the cursor shape changes to resize shape as on the following screenshot.

Make sure the cursor takes the right shape before dragging the window edge.

To resize individual columns in the treeview, drag the right-edge of the column header left or right. Note that when you place mouse pointer over the right edge of a column header the cursor shape changes to resize shape as on the following screenshot.

Make sure the cursor takes the right shape before dragging the column edge.

Tip: When column width is too narrow to fit the content, 3 dots (also called ellipses) are rendered in each cell with non-fitting data to indicate the data overflow effect. To quickly resize a column so it fits the entire content in all cells, double-click the right-edge of the column header. SQL Assistant will calculate the required width and resize this column as needed.

Refreshing and Clearing the Tasks and Reminders List

The **Tasks and Reminders** pane can be refreshed from SQL Assistant's menu in the target or by using the right-click menu in the SQL Assistant system tray application. In either case, select the **SQL Assistant →
Tasks and Reminders → Update** command from menu. To use the target editor's menus, the menu integration option must be enabled. For details, see the Manually Invoking SQL Assistant Popups topic in CHAPTER 3.

The refresh operation runs in the background as a separate process. Depending on how many databases and folders are configured for the Tasks and Reminders feature, the update may take anywhere from several milliseconds to several minutes to run. If for whatever reasons you need to stop the already running update, select the **SQL Assistant → Tasks and Reminders → Stop Update** command from menu.

To clear the Tasks and Reminders List, select the **SQL Assistant → Tasks and Reminders → Clear** command from menu. The command will purge the disk cache and clear the list. However, the next Update operation will populate it again with the freshly located items.

CHAPTER 34, Integrated Development Environments

Overview

SQL Assistant provides two integrated database development environments:

- **Standard SQL Editor** - SQL Assistant Standard Edition provides simple version of SQL editor (pictured below) supporting Notepad-like single document interface with all standard code edit and code execution functions typically expected from a SQL editor, including SQL syntax highlighting, SQL Intellisense, code execution, data retrieval and output and a fully integrated SQL Assistant add-on with all its functions accessible directly from the editor. The database connections are handled by SQL Assistant. The editor can connect to any database type supported by SQL Assistant.
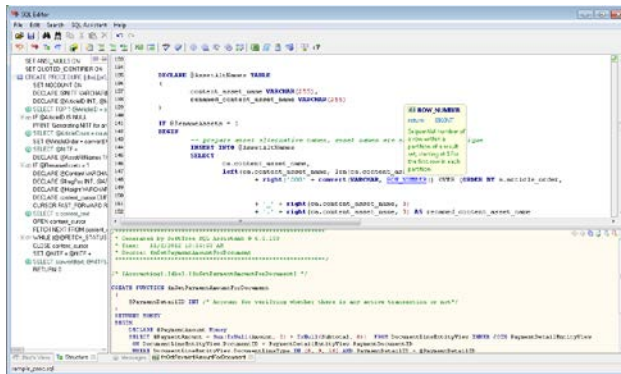
- **Professional SQL Editor IDE -** SQL Assistant Professional Edition provides an advanced version of SQL editor (pictured below) supporting tabbed and multiple-document interface with concurrent connections to multiple database servers, advanced code refactoring, code folding, macros for code entry automation, and many other unique features not available in the Standard Edition.
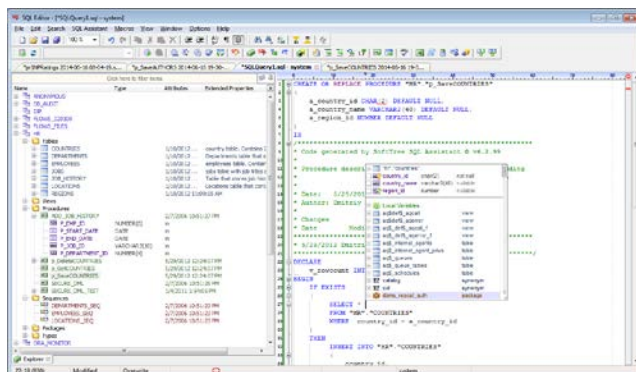
Both code editors support all standard edit methods, SQL syntax highlighting, as well as the complete set of tools and functions provided by embedded SQL Assistant add-on.

Both code editors can be started as standalone applications invoked via menus and toolbars of SQL Assistant add-ons and plug-ins loaded within the target development environments. The version invoked depends on the edition of SQL Assistant licensed by the user.

Tip: If your development environment doesn't support Windows-enabled editor components for database development, you can add a custom shortcut using target's supported customization features to launch the SQL Assistant's code editor. The editor can be accessed via the sqleditor.exe program (or sqleditor_standard.exe for Standard license type users) located within the **bin** subfolder of the SQL Assistant installation folder.

Following is an example of how to do that for the Sublime Text 2 general code editor:

1. Launch Sublime Text and select the **Preferences → Key Bindings - User** menu command. This will open "user keymap" file in Sublime editor.

2. Insert the following text in the file replacing the [,] placeholder

```
[{
    "keys": ["f10"], "command": "exec", "args":
    { "cmd": ["C:\\Program Files\\SQL Assistant 6\\data\\sqleditor.exe"] }
}]
```

    Click the **File / Save** menu to save changes in the keymap file.

After you save the keymap file, pressing F10 within Sublime Text will launch the SQL Editor with integrated SQL Assistant. You will then be able to connect to a database server and work effortlessly with SQL files, objects and data stored in the database.

Standard SQL Editor

It is assumed that users of SQL Assistant software are computer sophisticated and do not require specific instructions for using basic text editing functions for text changes, content scrolling and navigation, and for clipboard based operations. The editor supports most common Windows hotkeys such as these used in Windows Notepad and many other programs so that the users should find the editor's interface familiar.

Connecting to Databases

The editor connects to a database server automatically as soon as you perform any database related operation or enter a SQL keyword. The database connectivity is described in detail in CHAPTER 2, Connecting to Your Database.

Working with Databases

All database operations including code execution, data preview, SQL Intellisense, etc… are handled by the integrated SQL .Assistant add-on. Please refer to other chapters in this User's Guide for details on specific operations and SQL Assistant functions..

Professional SQL Editor IDE

The professional version of SQL Editor provides fully featured advanced development environment for multi-database development. The integrated development environment (IDE) provides all the features that database develop or administrator may need in their day to day work.

PDB and Edition Views of Your Oracle Database Server

With Oracle database server connections SQL Editor supports session specific views of your database servers. The selection of the current pluggable database (PDB) and the current editor drive the content of the DB Explorer, the content of SQL Intellisense suggestions, and many other session context specific functions. The Edition and Database drop-downs available on the right of the main toolbar can be used to change the current session view and connection settings, as demonstrated on the following example:
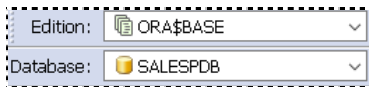
Note that if you connect to an Oracle server instance not configured for multi-tenancy, the **Database** drop-down is not shown on the toolbar.

Tabbed and MDI Layouts

The SQL Editor supports the modern tabbed document layout as in Visual Studio and in other advanced editors allowing working with multiple documents in a limited screen space, as well as, the classical multiple-documents interface (MDI) when screen space is not an issue and so multiple documents can be displayed side by side or overlapping.

The settings that change between the tabbed and MDI layouts are controlled from the **Window** menu. In case of using MDI layout, the Window menu also provides additional commands for managing and rearranging opened documents within the editor's main window.

Tip: You can use keyboard shortcuts to navigate between open tabs and MDI windows. Ctrl+Tab switches focus to the next tab / window. Ctrl+Shift+Tab switches focus to the previous tab / window.

Tab Management Functions

Reordering tabs - drag tabs horizontally to reorder them in the tab strip

Tab positioning – use **Position** commands in tab-strip right-click menu to display tabs at the top or at the bottom of SQL Editor MDI window.

Multi-row tabs – this mode is enabled by default. Tabs for the opened editors that do not fit in a single row in the tab-strip appear in the second row, those that do not fit in the second row, appear in the third row, and so on.

In a single row mode all tabs are show in the same row, as many as fit in a single row tab-strip. The remaining tabs can be scrolled one by one or accessed directly using the Tab Manager.

You can use **Position -> Multiple Rows** menu to switch between multi-row and single-row modes.

Closing tabs – use **File –> Close** command in the top menu, or Ctrl+F4 keyboard hotkey to close the current tab. To close all tabs, use **File –> Close All** in the top menu or **Close All Pages** command in the tab-strip right-click menu. To close all but the current tab, use **Close All Other Pages** command in the tab-strip right-click menu. For your convenience the right-click menu in the tab-strip provides several types of additional commands for closing multiple tabs, for example, for closing all unmodified editors only, or closing only tabs to the left of right.

Sticky tabs – sticky tabs persist between SQL Editor IDE sessions and also ignore mass tab closing operations. To make a tab sticky, right click it in the tab-strip and select **Sticky Tab** menu. To undo tab stickiness, use the same menu again. Note that the background color of sticky tabs is slightly different from regular tabs.

Running tabs – the background of tabs actively executing SQL code is rendered with a yellow shade to help identify running operations.

Cloning tabs – to instantly clone editors including their content and database connections, right click the editor's tab in the tab-strip and select **Clone Tab** menu.

Text Change Map

Marks along left border of the editor indicate states of editor lines:

- New line (green)
- Modified line (yellow)
- Saved line (dark blue)

Search and Replace Functions

The search and replace functions can be invoked from the top level **Search** menu and by using Ctrl+F and Ctrl+H hot keys.

Search and Replace Options

Case sensitive - Makes search case-sensitive, i.e. words "Some" and "some" will be different.

Whole words - Finds only whole words, i.e. word "some" cannot be found inside "somewhat".

Regular expressions - Activates regular expressions syntax for input text fields ("Search for" and "Replace with"). See the following topic for more details on using regular expressions.

Prompt on replace - Enables confirmation message on doing "Replace all" operation, for each found occurrence.

Global - Search within entire text in the editor.

Selected text - Searches only within selected block of text. Note that search in selection makes new smaller

selection, so you can't find all matches in selection by pressing "Find next" button.

Search from cursor - Starts search forward from the current edit caret position. Note: that option is ignored for "Replace all" operation.

Search entire scope – Warps the search operation, e.g continue searching from the opposite text edge (i.e. continue from text beginning when ending is reached, or continue from text ending when beginning is reached with backward search).

Forward - Starts search forward from the current edit caret position. If "Search entire scope" is selected, wraps the search and continues searching from the start of text, otherwise stops at the end of text

Backward - Starts search forward from the current edit caret position. If "Search entire scope" is selected, wraps the search and continues searching from the end of text, otherwise stops at the start of text

Using Regular Expressions

The Find/Replace dialog supports PCRE like regex engine, which provides similar syntax and functions as Perl regular expressions engine.: Please see http://perldoc.perl.org/perlre.html for the complete syntax description.

There are few minor differences between the syntax supported by the SQL Assistant SQL Editor and the Perl regular expressions syntax:

- Regex engine allows changing the case on replacements. See the following help paragraphs for details.

- Regex engine doesn't support non-capturing groups.

- "r" modifier is not supported.

- "g" modifier is the same as "U".

- \g and \G are not used.

- \h and \H have another meaning

- Regex engine recognizes only \x0D\x0A or \x0A or \x0D (depending on the file format) as newline characters when using "^", "$" and ".". Unicode line endings are not supported with ^ $ in the current implementation, but could be matched using \x meta-character.

- Regex engine uses \z to match any newline character.

Simple matches

Any single character matches itself, unless it is a metacharacter with a special meaning described below.

A series of characters matches that series of characters in the target string, so the pattern "bluh" would match "bluh" in the target string. Quite simple, right?

You can cause characters that normally function as metacharacters or escape sequences to be interpreted literally by 'escaping' them by preceding them with a backslash "\", for instance: metacharacter "^" match beginning of string, but "\^" match character "^", "\\" match "\" and so on.

Examples:

foobar   matches string 'foobar'

\^FooBarPtr   matchs '^FooBarPtr'

Escape sequences

Characters may be specified using an escape sequences syntax much like that used in C and Perl: "\n" matches a newline, "\t" a tab, etc. More generally, \xnn, where nn is a string of hexadecimal digits, matches the character whose ASCII value is nn. If You need wide (Unicode) character code, You can use '\x{nnnn}', where 'nnnn' - one or more hexadecimal digits.

\xnn	char with hex code nn
\x{nnnn}	char with hex code nnnn (one byte for plain text and two bytes for Unicode)
\t	tab (HT/TAB), same as \x09
\n	newline (NL), same as \x0a
\r	car.return (CR), same as \x0d
\f	form feed (FF), same as \x0c
\a	alarm (bell) (BEL), same as \x07
\e	escape (ESC), same as \x1b

Examples:

foo\x20bar   matchs 'foo bar' (note space in the middle)

\tfoobar   matchs 'foobar' predefined by tab

Character classes

You can specify a character class, by enclosing a list of characters in [], which will match any one character from the list.

If the first character after the "[" is "^", the class matches any character not in the list.

Examples:

foob[aeiou]r   finds strings 'foobar', 'foober' etc. but not 'foobbr', 'foobcr' etc.

foob[^aeiou]r   find strings 'foobbr', 'foobcr' etc. but not 'foobar', 'foober' etc.

Within a list, the "-" character is used to specify a range, so that a-z represents all characters between "a" and "z", inclusive.

If you want "-" itself to be a member of a class, put it at the start or end of the list, or escape it with a backslash. If you want ']' you may place it at the start of list or escape it with a backslash.

Examples

[-az]   matches 'a', 'z' and '-'

[az-]   matches 'a', 'z' and '-'

[a\-z]	matches 'a', 'z' and '-'
[a-z]	matches all twenty six small characters from 'a' to 'z'
[\n-\x0D]	matches any of #10,#11,#12,#13
[\d-t]	matches any digit, '-' or 't'
[]-a]	matches any char from ']'..'a'

Metacharacters

Metacharacters are special characters which are the essence of Regular Expressions. There are different types of metacharacters, described below.

Metacharacters - line separators

^	Start of line
$	End of line
\A	Start of text
\Z	End of text
.	Any character in line
\z	Single line break. Instead of **$** metacharacter it matches line break characters, not only line break position. Possible sequences: $0D0A ; $0D ; $0A

Examples:

^foobar	matches string 'foobar' only if it's at the beginning of line
foobar$	matches string 'foobar' only if it's at the end of line
^foobar$	matches string 'foobar' only if it's the only string in line
foob.r	matches strings like 'foobar', 'foobbr', 'foob1r' and so on

The "**^**" metacharacter by default is only guaranteed to match at the beginning of the input string/text, the "**$**" metacharacter only at the end. Embedded line separators will not be matched by "**^**'' or "**$**''.

You may, however, wish to treat a string as a multi-line buffer, such that the "^'' will match after any line separator within the string, and "**$**" will match before any line separator. You can do this by switching on the modifier **/m**.

The **\A** and **\Z** are just like "**^**'' and "**$**'', except that they won't match multiple times when the modifier **/m** is used, while "**^**'' and "**$**'' will match at every internal line separator.

The "**.**'' metacharacter by default matches any character, but if you switch off the modifier **/s**, then '**.**' won't match embedded line separators.

Regex engine works with line separators as recommended at www.unicode.org (
http://www.unicode.org/unicode/reports/tr18/):

"**^**" is at the beginning of an input string, and, if modifier **/m** is On, also immediately following any occurrence of \x0D\x0A or \x0A or \x0D (if You are using Unicode version of regex engine, then also \x2028 or \x2029 or \x0B or \x0C or \x85). Note that there is no empty line within the sequence \x0D\x0A.

"**$**" is at the end of a input string, and, if modifier **/m** is On, also immediately preceding any occurrence of \x0D\x0A or \x0A or \x0D (if You are using Unicode version of regex engine, then also \x2028 or \x2029 or \x0B or \x0C or \x85). Note that there is no empty line within the sequence \x0D\x0A.

"**.**" matches any character, but if you switch off the modifier **/s** then "**.**" doesn't match \x0D\x0A and \x0A and \x0D (in case of using Unicode version of regex engine, then also \x2028 and \x2029 and \x0B and \x0C and \x85).

Note that "**^.*$**" (an empty line pattern) does not match the empty string within the sequence \x0D\x0A, but matches the empty string within the sequence \x0A\x0D.

Metacharacters - predefined classes

\w   an alphanumeric character (including "_")

\W   a non-alphanumeric

\d   a numeric character

\D   a non-numeric

\s   any space (same as [\t\n\r\f])

\S   a non space

\h   hexadecimal digit

\H   non-hexadecimal digit

\l   letter character (any language)

\L   non-letter character (any language)

\c   identifier character (a-z, A-Z, 0-9, _)

\C   non-identifier character

\g   Latin letter (a-z, A-Z, _)

\G   non-Latin letter

\k   European digit

\K   non-European digit

Predefined classes \w, \W, \d, \D - for any language.

You may use \w, \d, \s, \h, \l, \c, \g, \k within custom character classes.

Examples:

foob\dr   matches strings like 'foob1r', ''foob6r' and so on but not 'foobar', 'foobbr' and so on

foob[\w\s]r   matches strings like 'foobar', 'foob r', 'foobbr' and so on but not 'foob1r', 'foob=r' and so on

Metacharacters - word boundaries

\b   Match a word boundary

\B   Match a non-(word boundary)

A word boundary (\b) is a spot between two characters that has a \w on one side of it and a \W on the other side of it (in either order), counting the imaginary characters off the beginning and end of the string as matching a \W.

Metacharacters - iterators

Any item in a regular expression may be followed by metacharacters - iterators. Using this metacharacters you can specify number of occurrences of previous character, metacharacter or sub-expression.

*   zero or more ("greedy"), similar to {0,}

+   one or more ("greedy"), similar to {1,}

?   zero or one ("greedy"), similar to {0,1}

{n}   exactly n times ("greedy")

{n,}   at least n times ("greedy")

{n,m}   at least n but not more than m times ("greedy")

*?   zero or more ("non-greedy"), similar to {0,}?

+?   one or more ("non-greedy"), similar to {1,}?

??   zero or one ("non-greedy"), similar to {0,1}?

{n}?   exactly n times ("non-greedy")

{n,}?   at least n times ("non-greedy")

{n,m}?   at least n but not more than m times ("non-greedy")

So, digits in curly brackets of the form {n,m}, specify the minimum number of times to match the item n and the maximum m. The form {n} is equivalent to {n,n} and matches exactly n times. The form {n,} matches n or more times. There is no limit to the size of n or m, but large numbers will chew up more memory and slow down r.e. execution.

If a curly bracket occurs in any other context, it is treated as a regular character.

Examples:

foob.*r   matches strings like 'foobar', 'foobalkjdflkj9r' and 'foobr'

foob.+r   matches strings like 'foobar', 'foobalkjdflkj9r' but not 'foobr'

foob.?r   matches strings like 'foobar', 'foobbr' and 'foobr' but not 'foobalkj9r'

fooba{2}r   matches the string 'foobaar'

fooba{2,}r   matches strings like 'foobaar', 'foobaaar', 'foobaaaar' etc.

fooba{2,3}r   matches strings like 'foobaar', or 'foobaaar' but not 'foobaaaar'

A little explanation about "greediness".

"Greedy" takes as many as possible; "non-greedy" takes as few as possible. For example, 'b+' and 'b*' applied to string 'abbbbc' return 'bbbb', 'b+?' returns 'b', 'b*?' returns empty string, 'b{2,3}?' returns 'bb', 'b{2,3}' returns 'bbb'.

You can switch all iterators into "non-greedy" mode (see the modifier /g).

Metacharacters - alternatives

You can specify a series of alternatives for a pattern using "**|**" to separate them, so that fee|fie|foe will match any of "fee", "fie", or "foe" in the target string (as would f(e|i|o)e). The first alternative includes everything from the last pattern delimiter ("**(**", "**[**", or the beginning of the pattern) up to the first "**|**", and the last alternative contains everything from the last "**|**" to the next pattern delimiter. For this reason, it's common practice to include alternatives in parentheses, to minimize confusion about where they start and end.

Alternatives are tried from left to right, so the first alternative found for which the entire expression matches, is the one that is chosen. This means that alternatives are not necessarily greedy. For example: when matching foo|foot against "barefoot", only the "foo" part will match, as that is the first alternative tried, and it successfully matches the target string. (This might not seem important, but it is important when you are capturing matched text using parentheses.)

Also remember that "**|**" is interpreted as a literal within square brackets, so an expression [fee|fie|foe] would matching [feio|] only.

Examples:

foo(bar|foo)   matches strings 'foobar' or 'foofoo'

Metacharacters - sub-expressions

The bracketing construct (...) may also be used for define r.e. sub-expressions (after parsing You can find sub-expression positions, lengths and actual values in MatchPos, MatchLen and GetMatch properties and substitute it in template strings by Substitute).

Sub-expressions are numbered based on the left to right order of their opening parenthesis.

First sub-expression has number '1' (whole r.e. match has number '0' - You can substitute it in regex engine.Substitute as '\0').

Examples:

(foobar){8,10}   matches strings which contain 8, 9 or 10 instances of the 'foobar'

foob([0-9]|a+)r   matches 'foob0r', 'foob1r' , 'foobar', 'foobaar', 'foobaar' etc.

Metacharacters - back-references

Metacharacters \1 through \9 are interpreted as back-references. \<n> matches previously matched sub-expression #<n>.

Examples:

(.)\1+   matches 'aaaa' and 'cc'

(.+)\1+   also match 'abab' and '123123'

(['"]?)(\d+)\1   matches '"13" (in double quotes), or '4' (in single quotes) or 77 (without quotes) etc

Modifiers

Modifiers are used for behavior changing.

There are many ways to set up modifiers.

Any of these modifiers may be embedded within the regular expression itself using the (?...) construct.

i	Do case-insensitive pattern matching (using installed in you system locale settings).
m	Treat string as multiple lines. That is, change "^" and "$" from matching at only the very start or end of the string to the start or end of any line anywhere within the string.
s	Treat string as single line. That is, change "." to match any character whatsoever, even a line separators, which it normally would not match.
g	Non standard modifier. Switching it Off You'll switch all following operators into non-greedy mode (by default this modifier is On). So, if modifier /g is Off then '+' works as '+?', '*' as '*?' and so on
x	Extend your pattern's legibility by permitting whitespace and comments (see explanation below).
r	Non-standard modifier. When modifier R is switched off languages support is turned off, i.e: \w = \c, \W = \C, \l = \g, \L = \G, \d = \k, \D = \K

The modifier **/x** itself needs a little more explanation. It tells the regex engine to ignore whitespace that is neither backslashed nor within a character class. You can use this to break up your regular expression into (slightly) more readable parts. The **#** character is also treated as a metacharacter introducing a comment, for example:

(

(abc) # *comment 1*

| # *You can use spaces to format r.e. - regex engine ignores it*

(efg) # *comment 2*

)

This also means that if you want real whitespace or **#** characters in the pattern (outside a character class, where they are unaffected by **/x**), that you'll either have to escape them or encode them using octal or hex escapes. Taken together, these features go a long way towards making regular expressions text more readable.

Perl extensions

(?imsxr-imsxr)

You may use these extensions for changing behavior modifiers on the fly. When they are inlined into sub-expression, they effect the sub-expression only

Examples:

(?i)Santa-Claus	matches 'Santa-claus' and 'Santa-Claus'
(?i)Santa-(?-i)Claus	matches 'Santa-Claus' but not 'Santa-claus'
(?i)(Santa-)?Claus	matches 'Santa-claus' and 'santa-claus'
((?i)Santa-)?Claus	matches 'santa-Claus', but not 'santa-claus'

Inline Comments

(?#text) is used as a comment. Comments are ignored by regex engine. Note that ")" symbol completes comment and cannot be esacaped, so there is no way to put a literal ")" in the comment text.

Matching Words Navigation

When you click an item within the code such as variable name, object name, column name, and so on, all instances of the same item are highlighted in the code. You can move between highlighted items using any of the following methods:

- Pressing Alt + Right Arrow or Alt + Left Arrow keys.

- Using **Search → Find Current Word, Next** menu or **Search → Find Current Word, Prior** menu.

- Using toolbar icon ▼ or ▲.

Advanced Text Processor

The **Advanced Text Processor** utility enables you to harness the power of regular expressions (regex) to transfer the data or scripts, perform data cleanup operations, and many other things. The regular expressions are executed in a single-step or multi-step process with conditional logic applied in between to evaluate the

results or locate next text fragment for the following operations.

The transformation rules and settings can be saved and reused later. If you often need to apply the same set of transformations to one of more files or simple search & replace operations, this is the utility to use.

The Advanced Text Processor allows you to assign transformation rules to categories for easy filtering and grouping. For example, you can organize all your data cleanup rules in the a single category called Data Cleanup. You can create as many categories as needed.

Please refer to the [Using Regular Expressions](#) topic for more information on using regular expressions and the supported regular expressions syntax and dialect.

Running the Advanced Text Processor

1. Open the data file or script whose content you want to transform

2. Click **Search → Advanced Text Processor** menu. The **Advanced Text Processor – Select Rules** dialog appears.
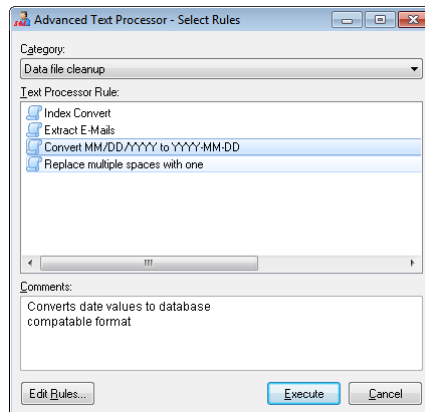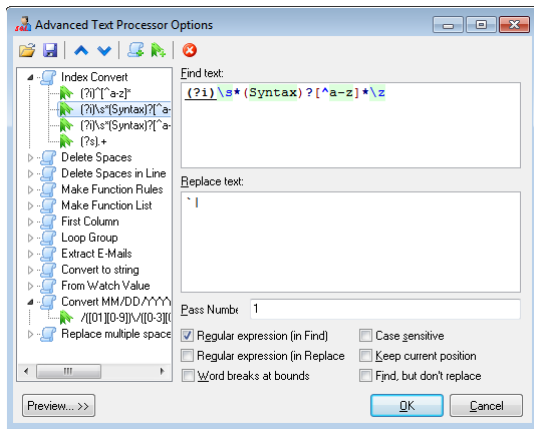

3. Choose the **Category** containing transformation rules you want to use.

4. Select **Text Processor Rule** to execute.

5. Click the **Execute** button

Tip: The text processing rules are also accessible in the editor's right-click menu

Configuring Text Processing Rules

Click **Options → Advanced Text Processing** menu to open the **Advanced Text Processing Options** dialog.

Use the toolbar icons at the top of the **Advanced Text Processing Options** dialog to open and save configuration files and to make changes.

This icon opens Advanced Text Processing configuration files. The default configuration file is %APPDATA%\SQL Assistant\13.0\SqlEditor.mrt. Note that %APPDATA% is a system environment variable. If you do not know the value of this variable, you can display it by opening the DOS command window, and executing the command echo %APPDATA%. This file is loaded automatically on the SQL Editor startup.

This icon saves changes in the Advanced Text Processing configuration files.

This icon arranges position of the selected item in the Rules tree. It moves the selected item up.

This icon arranges position of the selected item in the Rules tree. It moves the selected item down.
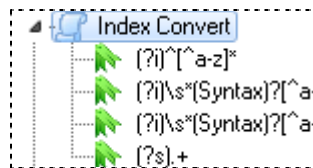
This icon creates new Regex Rule Group adding it to the end of the current file.

This icon creates new Regex Rule in the selected Regex Rule Group.

Tip: If a Rule Group item is selected in the Rules Tree, the new Rule is added to the end of the group.

If a Rule is selected in the Rules Tree, it created a new nested Rule.

This icon deletes selected item from the Rules Tree

Text Processing Rule Group Properties

Rule Groups are represented by 📜 icons in the Rules Tree. The following Rule Group properties are supported:

Display Name – This is the group name. This name is displayed in dialogs and menus related to the Advanced Text Processor.

Category – This used to organize transformation rule groups in categories. When adding a new group or editing an existing group you can select either an already existing category from the **Category** drop-down list, or type in new category name.

Group Index – Do not use this property. This property is reserved for internal use only.

Shortcut Key – You can use this property to assign a hotkey to the rule group. The hotkey enables you to invoke the rule group for the text selected in the editor using the chosen hotkey. To enter the hotkey, press it while the focus is in the **Shortcut Key** field, for example to assign Ctrl+1, hold down Ctrl key while pressing 1 key. To remove previously select hot key, set the focus the **Shortcut Key** field and then press the Backspace key.

Target – This property specifies target type, whether operation is be applied to the whole text in the editor or to the selection only. This property applies to top level rules in the group only. Sub-rules are always applied to the text processed by their parent rules.

Text Evaluation Condition (Regex) – An optional regular expression applied to the text before executing the rules. If the evaluation condition is specified and does not pass during rule group execution, the operation is aborted without any changes. The value for Text Evaluation Condition (Regex) is optional.

Comments – This property is used to enter comments describing the selected Rule Group and how to use it. The comments are optional.

Text Processing Rule Properties

Rules are represented by 🏹 icons in the Rules Tree. The following Rule properties are supported:

Find text – A regular expression or plain text to find in the rule's target The top level rule target is defined by their Rule Group's **Target** property. Sub-rules search only the text elements returned by their parent rules.

Replace text - A regular expression or simple text used as a replacement for the text occurrences returned by the Find Text expression.

Pass number – The order in which the Rule is applied. The order value is zero based, in other word, the first executed rule must have 0 in the **Pass Number** field.

For example, you can setup a Rule Group with three rules. The first two rules have no correlation between them and can be applied in any order. Specify 0 for the **Pass number** value. The third rule depends on the results of the first two rules and it needs to be executed after the first two are executed. Specify 1 for the third Rule.

Regular expression in Find – If this checkbox is checked, the value of the **Find text** field is treated as a regular expression. If not checked, the value is treated as a plain text.

🦖 **Tip**: When using plain text searches, you can refer to special symbols using backslash as the escape symbol. For example, to refer to a tab character, specify \t. When using regular expression, use regular expressions compatible syntax to refer to special symbols.

Regular expression in Replace – If this checkbox is checked, the value of the **Replaced text** field is treated as a regular expression. If not checked, the value is treated as a plain text.

Tips: When using plain text replacements, you can refer to special symbols using backslash as the escape symbol. For example, to refer to a tab character, specify \t. When using regular expression, use regular expressions compatible syntax to refer to special symbols.

The replacement regular expression may contain references to the found text, for example:

```
Find text: [\x20\t]*([\|\{\}\[\]\(\)<>,=']+)[\x20\t]*

Replace text: \1
```

In this example the search is performed for text surrounded by spaces and tabs and replaced with the found text excluding spaces and tabs

Case sensitive – If this checkbox is checked, the search is case sensitive. This property is only used with plain text searches (e.g. when the **Regular expression in Find** checkbox is not checked)

Keep current position - If this checkbox is checked, after applying the search and replacement rule, the next search operation starts from the beginning of the replaced text, otherwise the next search operation starts from the character following the replaced text.

Word breaks at bounds –If this checkbox is checked, the found text must be bound by word breaks. For example,

```
Target text: abc123 something else abc 123.

Find text: abc

Replace text: xyz
```

In this example the search will find two occurrences of *abc*, but will replace only the second occurrence because that *abc* is a separate word The first *abc* occurrence is part of *abc123* word and it will be ignored.

Find, but do not replace - If this checkbox is checked, the search is performed as usual and the scanning position is moved to the found text, but the replacement is not performed. You may use it to perform nested search operations within the same Rule Group, e.g. find some text using one condition and then perform an additional search within the previously found text only.

Fast Synchronous Renaming of Multiple Identifiers

The synchronous word editing feature ("sync edit" here and later) allows simultaneous renaming of multiple repeating identifiers appearing in the selected text, such as variable names, object names, and so on… This method is more efficient as compared to running multiple search and replace operations and enables you to preview all changes immediately in WYSIWYG mode.

To use this feature

1. Select a block of text containing several repeating identifiers that you want to change.

2. The sync edit icon appears on gutter near the ending of the selected block of text. Click this icon or the same icon in the top level toolbar to active the sync edit mode for the selected region. In active sync edit mode the icon border changes to inset style.

   Note: In order to activate the sync edit mode, your selection must contain at least one repeating identifier.

3. Click any repeating identifier and edit its new name. All instances of this identifier in the selected

block will be renamed as well. Note that as you make the changes in the identifier with the focus, all other matching identifiers change synchronously.

4.   If required, repeat the previous step for other identifiers in the selected block.

5.   Click the sync edit ![icon] icon again to deactivate the sync edit mode.

Tip: The sync edit feature supports multiple independent regions.   After you activate sync edit for one region, you can then select another block of text and then activate sync edit for that block. Up to 16 independent regions can be used.

Tip: The following keyboard shortcuts can be used with the sync edit feature.

- Ctrl+Shift+S - Activates sync edit mode for the selected region.

- Ctrl+Shift+W – Expands sync edit mode for the entire document.

- Ctrl+Shift+C – Cancels sync edit mode for the current region.

The following series of screenshots demonstrate how the sync edit mode is working in case of a very simple script.

Code Views, Code Folding, and Code Navigation

The SQL Editor provides several convenience features easing the coding process.

Zoom

You can quickly zoom in and out by changing the zoom ratio in the Zoom drop-down available in the toolbar area. You can choose one of the predefined values or type in a custom value.

Word Wrap, Ruler, Column Markers

When text extends beyond the visible code pane, it can automatically wrap to the next line. You can turn on and code wrapping using **View → Word Wrap** menu.

The Ruler and Column Markers help you to control line length and your code appearance. You can customize their display and colors in the SQL Editor options using **Options → SQL Editor…** menu. See the Customizing SQL Editor Options and Behavior topic in this chapter for more details.

Incremental Search

An incremental search is performed letter-by-letter as you type in a search string.

To activate an incremental search, press Ctrl + E hot key or use **Search → Incremental Search** menu. Start typing the text you want to find. The SQL Editor highlights the first set of characters found in the current document that match the search string entered.

If you made a mistake and need to remove a character from the entered search string, press the Backspace key.

To stop the incremental search, press the Esc key.

Matching Identifier Navigation

See the Matching Words Navigation topic for more information.

Line Jumps

To quickly jump to a specific line in the code, press the Alt+G hotkey. The **Enter Line Number** dialog appears. Enter new position or just a line number and press the OK button.

Bookmarks

The SQL Editor supports advanced Visual Bookmarks feature which is documented in the CHAPTER 32, Visual Bookmarks
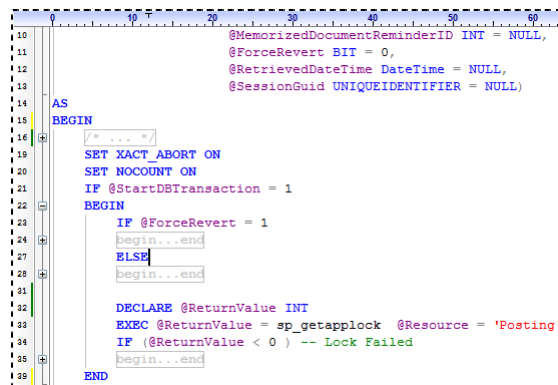
Code Regions, Folding, and Outlining

By default, all text is displayed in the SQL Editor, but you can choose to hide some code from view. In long code files, it is convenient to be able to collapse or hide one or more regions so that you can focus on the part of the file that you are currently working on. The SQL Editor lets you select a region of code and make it collapsible, so that it appears under a plus sign [+] displayed in the gutter area followed by their region type identifier. It also enables you to perform a limited set of edit operations on collapsed regions, such as delete, copy and move.

Tip: In addition to region folding signs shown in the gutter area, for custom user-defined regions **-- #region** their region specific navigation stripes are shown on the Syntax Bar. The position of a stripe on the Syntax Bar marks the relative position of the corresponding region in the code. Click on a stripe to quickly jump to its corresponding region in the code. It works similarly to working of TO-DO stripes, code syntax errors and other similar marks shown on the Syntax Bar for speedy code navigation.

Four kinds of regions are supported:
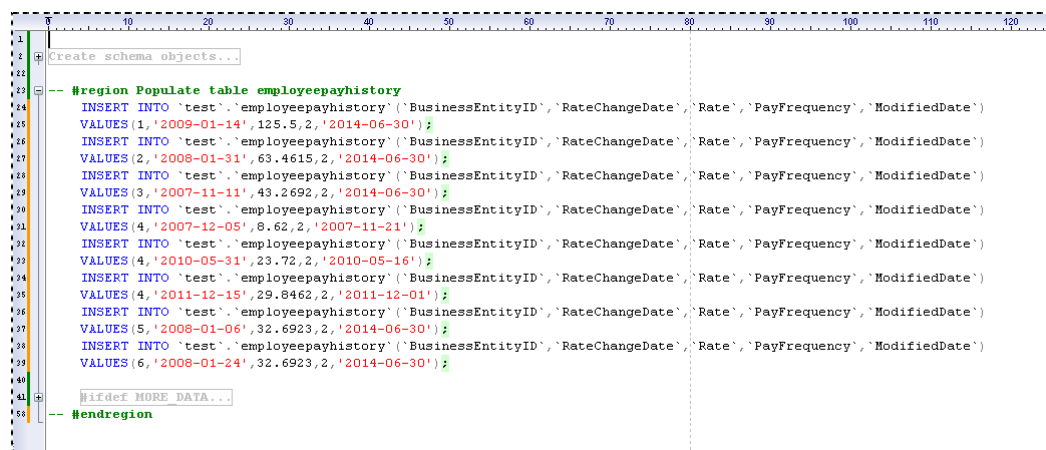
- **Standard SQL syntax based regions** - Standard regions are automatically identified by SQL syntax and are similar to what is used by the Code Structure View to display a structural collapsible map of the document. See CHAPTER 4, Code Structure View and Bird's Eye View for more details.

- **Custom regions –** You can define your own regions to enclose long and related blocks of code using specially formatted comments **-- #region [region name here]** and **-- # endregion**

- **SQL Preprocessor regions** – The regions of this kind are defined by SQL Preprocessor directives for conditional code execution, for example **-- #if defined [variable name here]** and **-- #endif**. Refer to the previous screenshot demonstrating collapsed region for **– #if defined MORE_DATA** code example. For more information about SQL Preprocessor and its directives see Using SQL

[Preprocessor for Advanced Code Execution](#) topic in CHAPTER 14.

- **Selected block of text** - You can also collapse the selected text and make the editor treat it as a single region independent of the code syntax and structure. Select a block of text and then use **View → Collapse Selection** menu command.

Tip: The content of collapsed regions can be previewed without expanding them. When you hover over the collapsed region with the mouse a popup window appears with the region preview.

You can turn on and off code folding using **View → Code Folding** menu.

To collapse a region and then expand it by clicking the [+] and [-] symbols displayed next to the first line of the region. To quickly collapse all available regions or expand them use **View → Full Collapse** and **View → full Expand** menu commands. Similarly to collapse regions in the selected block of code only, use **View → Collapse in Selection** and **View → Expand in Selection** menu commands.

Code Outlining

The code outlining feature outlines code regions with matching BEGIN…END keywords, for custom regions, and for SQL Preprocessor regions.

Note that the BEGIN…END outlining is supported only for code regions with BEGIN and END keywords appearing in separate lines and not mixed with other keywords.

Line Numbering

You can turn line numbering on and off in the SQL Editor options using **Options → SQL Editor…** menu. See

the Customizing SQL Editor Options and Behavior topic in this chapter for more details.

Hyperlinks

You can embed URLs in the document you are editing, and use them as hyperlinks.

```
SELECT 'Let''s jump to http://www.softtree.com web site'
FROM |
```

To turn on and off this feature, click **View → Hyperlinks** menu.

Code Structure and Code Page Views for Fast Code Navigation

Code structure and page view enable fast code navigation between different parts of the code when working with large scripts. They are covered in detail in CHAPTER 4, Code Structure View and Bird's Eye View.

Split Screen for Synchronous Off-line Code Editing

The SQL Editor supports horizontal split view function enabling you to work with the same file in two different windows synchronously. The top part provides an off-line editor not connected to the database not running background syntax checks and other on-line edit mode operation, The bottom part provides regular code editor connected to the database with all SQL Assistant functions enabled. The split view enables you

- To see and edit two different fragments of the same file simultaneously

- To effectively work with large files not dragged down by background database operations.

To split the view, use horizontal splitter control or control handler displayed in the right top part of the ruler bar.
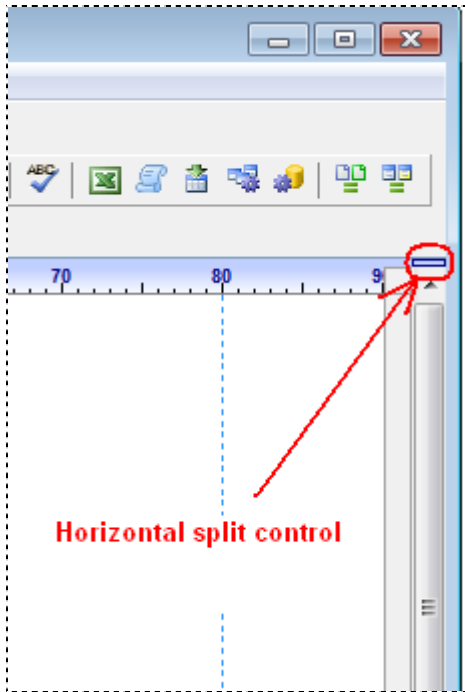
Mouse-over the splitter control and drag it down to split the view.

Note, when the cursor is properly positioned over the splittter control, the cursor shape changes to a double arrow shape as on the following screenshot.

Make sure the cursor takes the right shape before dragging the splitter.

File Operations, Formats, and Encoding

Text files can be saved with a variety of encoding and different code pages. If after opening a file you see its content with broken text not displayed correctly, Unfortunately the encoding and format aren't saved in the file so it could be anyone's guess as what encoding to use to pen the file with original settings. The SQL Editor provides you with a number of options available in the File menu for choose different file formats, code pages, character sets, and encoding. Use commands in the following menus to choose the required encoding for file open and save operations.

File → File Encoding - use this to choose ASCII, UTF and other encoding types

File → File Format – use this to choose file format between, Windows, Unix, Mac. This affects which symbols are used for line breaks.

File → Character Set – use this to choose your national character set

File → Code Page – use this to choose a code page.

Printing and Documenting Your Code

Printing

The SQL Editor provides all standard print features available from the operating system, along with a number of features specific to the SQL Editor. For example, you can include line numbers or omit collapsed regions in print outs, include highlighting, staples and so on, as well as customize headers and footers, colors and other options.

Use **File → Page Setup** and **File → Print Setup** menu commands to customize the printing options. The options are self-descriptive and easy to use.

To print your code in colors, open the **Page Setup Options** dialog and then select RGB item in the **Colors** drop-down list.

To preview how a print outs would look like with the currently selected options before it is printed, select **File → Page Setup** menu command.

Saving Code for Documentation Purposes

The SQL Editor provides two methods for saving code for documentation purposes

- **Save to RTF files**– this method can be invoked using **File → Export to RTF…** menu command. This enables you to save the current script in the editor to a RTF file, including all formatting, fonts, colors, text highlighting, and so on.

- **Save to HTML files**– this method can be invoked using **File → Export to HTML…** menu command. This enables you to save the current script in the editor to a HTML file, including all formatting, fonts, colors, text highlighting, and so on.

Connecting to Databases

The editor connects to a database server automatically as soon as you perform any database related operation or enter a SQL keyword. The database connectivity is described in detail in CHAPTER 2, Connecting to Your Database.

Working with Databases

All database operations including code execution, data preview, SQL Intellisense, etc… are handled by the integrated SQL .Assistant add-on. Please refer to other chapters in this User's Guide for details on specific operations and SQL Assistant functions.

Running SQL Queries

The SQL Editor enables you to run SQL code from editor tabs and MDI windows. The default hot key is F5. If any text is selected in the current editor, only the selected text is run. Otherwise, the entire editor content is run.

Batch delimiters and statement terminators control code execution. For specific details see Handling of Batch Delimiters in CHAPTER 14, Executing SQL Scripts.

You can run SQL code in multiple editor tabs and windows concurrently. Tabs with running code have yellowish color. After code execution is completed, tab colors change back to their regular color. In addition, in running tabs the status bar color changes to bright yellow. If the code running takes more then a few seconds then the timer statistics are show in the yellow part of the status bar too.
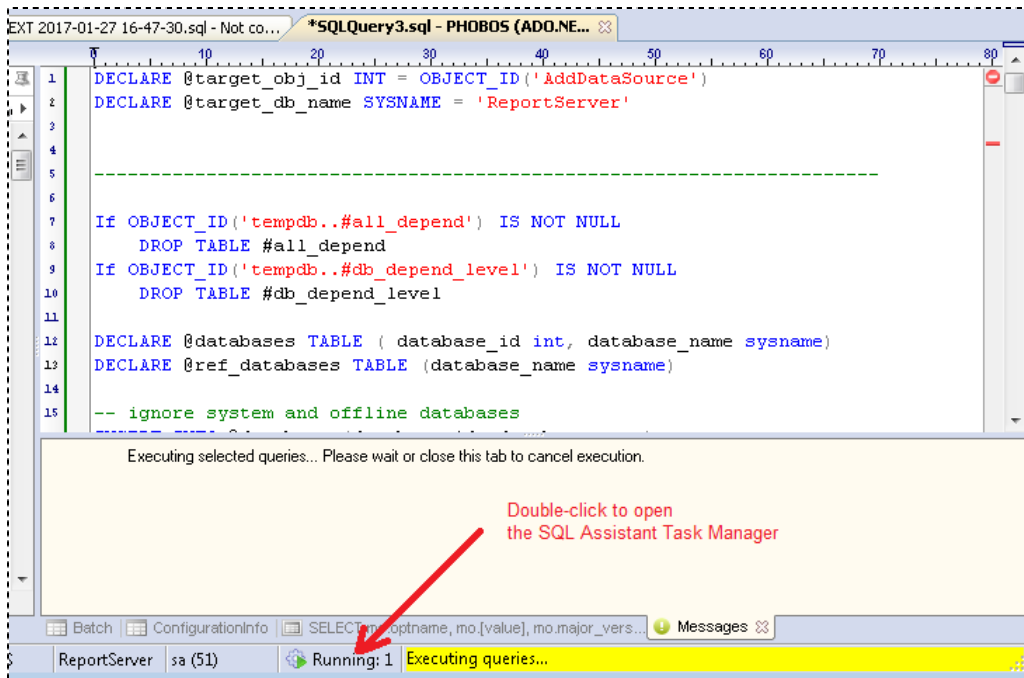
Tip: The Task Manager area in the status bar shows number of running tasks. This number includes the currently running tab, other tabs with running SQL queries, and all other background tasks that register with the **SQL Assistant Task Manager**, such as data export/import, code generation, and so on… To see more details about the running tasks, their current progress, and to manage them, double click the Task Manager area. The Task Manager will appear on the screen. For more information about the Task Manager and managing tasks, see CHAPTER 42, Task Manager and Database Session Monitor.

Using Source Code Control

Refer to CHAPTER 23, Database Source Code Control Interface for configuration instructions and usage details.

Recording Editor Macros for Repetitive Text Operations

The SQL Editor fully supports macro recording for keyboard and mouse actions, cursor positioning and other text editor operations. A macro consists of series of commands that you can group together as a single command to accomplish a task automatically. Macros allow you to automate repetitive actions. Over 200 macro commands are available to you to automate repetitive actions.

You can record macros as well as create and edit them manually. Typically you would use the recording facility to record the commands you perform manually and them manually tweak the recorded macro code to make it universal so that it can be reused with other files.

To record a new macro

1. Select **Macro → Record** menu to start the recording session.

2. Perform the required keyboard and mouse operations.

3. Select **Macro → Stop** menu to end the recording session.

To rename or modify the recorded macro

1. Select **Macro → New / Edit** menu. The Macro Editor dialog appears.
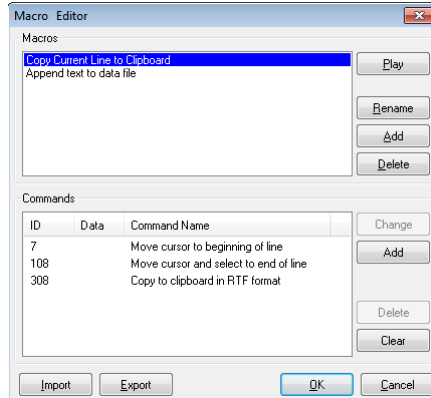
2. Select the macro you want to rename or modify in the **Macros** list.

3. Click the Rename button. enter new name and then press the Enter key.

4. To edit the macro, delete unneeded or add additional commands as required. Use the buttons to the right of the **Commands** list for the macro manipulations.

- The **Change** button opens the **Select macro command** dialog that you can use to replace the selected macro command with another command and/or change parameter values for the selected command.

- The **Add** button inserts a new command below the selected command or at the end of the command list if nothing is selected.

- The **Delete** button deletes the selected command from the commands list.

- The Clear button clears the command list effectively deleting all commands in the selected macro.

Use the **Add** button to the right of the **Macros** list to add a new macro, which you can then edit manually.

Use the **Delete** button to the right of the **Macros** list to delete an existing macro, which you no longer need.

To export macros and their commands to an external configuration file, use The **Export** button. The exported file can be used for backup purposes or shared with colleagues who can import your file into their system using the **Import** button.

Macro Commands

The Macro Editor supports over 200 various commands for automation of repetitive code entry operations. Each command internally is identified by its unique numeric ID number. Command descriptions are used to when editing macros. The **Select macro command** dialog is used to enter new and change already entered commands.
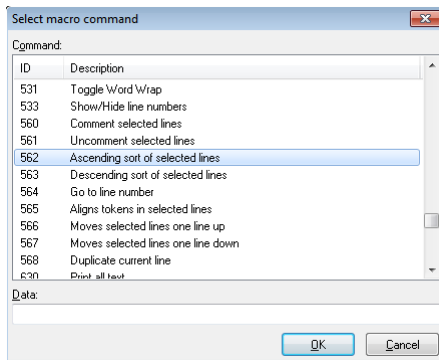
The following commands are supported. For your convenience they are grouped by categories and sorted in alphabetic order.

Category	Command
Block operations	Block Copy & Paste above selected block
Block operations	Block Copy & Paste below top of file
Block operations	Block Copy & Paste end of file
Block operations	Block Copy & Paste start of file
Block operations	Block Cut & Paste end of file
Block operations	Block Cut & Paste start of file
Bookmarks	Goto Bookmark 0
Bookmarks	Goto Bookmark 1
Bookmarks	Goto Bookmark 2
Bookmarks	Goto Bookmark 3
Bookmarks	Goto Bookmark 4
Bookmarks	Goto Bookmark 5
Bookmarks	Goto Bookmark 6
Bookmarks	Goto Bookmark 7
Bookmarks	Goto Bookmark 8
Bookmarks	Goto Bookmark 9
Bookmarks	Toggle Bookmark 0
Bookmarks	Toggle Bookmark 1

Bookmarks	Toggle Bookmark 2
Bookmarks	Toggle Bookmark 3
Bookmarks	Toggle Bookmark 4
Bookmarks	Toggle Bookmark 5
Bookmarks	Toggle Bookmark 6
Bookmarks	Toggle Bookmark 7
Bookmarks	Toggle Bookmark 8
Bookmarks	Toggle Bookmark 9
Change case	Lower case to current or previous word
Change case	Lower case to current selection or current char
Change case	Title case to current or previous word
Change case	Title case to current selection
Change case	Toggle case to current or previous word
Change case	Toggle case to current selection or current char
Change case	Upper case to current or previous word
Change case	Upper case to current selection or current char
Deleting text	Delete char at cursor (i.e. delete key)
Deleting text	Delete current line
Deleting text	Delete everything
Deleting text	Delete from cursor to beginning of line
Deleting text	Delete from cursor to end of line
Deleting text	Delete from cursor to next word
Deleting text	Delete from cursor to start of word
Deleting text	Delete last char (i.e. backspace key)
Indents and Tabs	Indent selection
Indents and Tabs	Insert Tab char
Indents and Tabs	Tab key
Indents and Tabs	Unindent selection
Inserting text	Break line at current position, leave caret
Inserting text	Break line at current position, move caret to new line
Inserting text	Insert a character at current position
Inserting text	Insert a whole string
Inserting text	Soft break line at current position, move caret to new line
Macros	Cancel macro recording

Macros	Play macro
Macros	Start macro recording
Macros	Stop macro recording
Markers	Collect marker (jump back)
Markers	Drops marker to the current position
Markers	Jump to matching bracket (change range side)
Markers	Swap marker (keep position, jump back)
Miscellaneous	Aligns tokens in selected lines
Miscellaneous	Ascending sort of selected lines
Miscellaneous	Comment selected lines
Miscellaneous	Descending sort of selected lines
Miscellaneous	Duplicate current line
Miscellaneous	Go to line number
Miscellaneous	Moves selected lines one line down
Miscellaneous	Moves selected lines one line up
Miscellaneous	Page Setup dialog
Miscellaneous	Print all text
Miscellaneous	Print preview
Miscellaneous	Print selected text
Miscellaneous	Show/Hide line numbers
Miscellaneous	Show/Hide non printed text/characters
Miscellaneous	Toggle Word Wrap
Miscellaneous	Uncomment selected lines
Navigation with columnar selection	Move cursor and column select left one char
Navigation with columnar selection	Move cursor and column select right one char
Navigation with columnar selection	Move cursor and column select up one line
Navigation with columnar selection	Move cursor and column select down one line
Navigation with columnar selection	Move cursor and column select left one word
Navigation with columnar selection	Move cursor and column select right one word
Navigation with columnar selection	Move cursor and column select to beginning of line
Navigation with columnar selection	Move cursor and column select to end of line
Navigation with columnar selection	Move cursor and column select up one page
Navigation with columnar selection	Move cursor and column select down one page
Navigation with columnar selection	Move cursor and column select right one page

Navigation with columnar selection	Move cursor and column select left one page
Navigation with columnar selection	Move cursor and column select to top of page
Navigation with columnar selection	Move cursor and column select to bottom of page
Navigation with columnar selection	Move cursor and column select to absolute beginning
Navigation with columnar selection	Move cursor and column select to absolute end
Navigation with columnar selection	Move cursor and column select to first char of line
Navigation with columnar selection	Move cursor and column select to last char of line
Navigation with columnar selection	Move cursor and column select left and up at line start
Navigation with columnar selection	Move cursor to specified position and column select
Navigation	Move cursor down one line
Navigation	Move cursor down one page
Navigation	Move cursor left and up at line start
Navigation	Move cursor left one page
Navigation	Move cursor left one word
Navigation	Move cursor right one char
Navigation	Move cursor right one page
Navigation	Move cursor right one word
Navigation	Move cursor to absolute beginning
Navigation	Move cursor to absolute end
Navigation	Move cursor to beginning of line
Navigation	Move cursor to bottom of page
Navigation	Move cursor to end of line
Navigation	Move cursor to first char of line
Navigation	Move cursor to last char of line
Navigation	Move cursor to specified position
Navigation	Move cursor to top of page
Navigation	Move cursor up one line
Navigation	Move cursor up one page
Navigation with regular selection	Move cursor and select down one line
Navigation with regular selection	Move cursor and select down one page
Navigation with regular selection	Move cursor and select left one char
Navigation with regular selection	Move cursor and select left one word
Navigation with regular selection	Move cursor and select left one page
Navigation with regular selection	Move cursor and select left and up at line start

Navigation with regular selection	Move cursor and select right one char
Navigation with regular selection	Move cursor and select right one word
Navigation with regular selection	Move cursor and select right one page
Navigation with regular selection	Move cursor and select to absolute beginning
Navigation with regular selection	Move cursor and select to absolute end
Navigation with regular selection	Move cursor and select to beginning of line
Navigation with regular selection	Move cursor and select to bottom of page
Navigation with regular selection	Move cursor and select to end of line
Navigation with regular selection	Move cursor and select to first char of line
Navigation with regular selection	Move cursor and select to last char of line
Navigation with regular selection	Move cursor and select to top of page
Navigation with regular selection	Move cursor and select up one line
Navigation with regular selection	Move cursor and select up one page
Navigation with regular selection	Move cursor to specified position and select
Scrolling	Scroll down one line leaving cursor position unchanged
Scrolling	Scroll down one page leaving cursor position unchanged
Scrolling	Scroll left one char leaving cursor position unchanged
Scrolling	Scroll left one screen leaving cursor position unchanged
Scrolling	Scroll right one char leaving cursor position unchanged
Scrolling	Scroll right one screen leaving cursor position unchanged
Scrolling	Scroll to absolute beginning leaving cursor position unchanged
Scrolling	Scroll to absolute end leaving cursor position unchanged
Scrolling	Scroll to absolute left leaving cursor position unchanged
Scrolling	Scroll to absolute right leaving cursor position unchanged
Scrolling	Scroll up one line leaving cursor position unchanged
Scrolling	Scroll up one page leaving cursor position unchanged
Search & Replace	Find All
Search & Replace	Find Current Word Next
Search & Replace	Find Current Word Prior
Search & Replace	Find Dialog
Search & Replace	Find First
Search & Replace	Find Last
Search & Replace	Find Next
Search & Replace	Find Previous

Search & Replace	Go to next search mark
Search & Replace	Go to previous search mark
Search & Replace	Incremental Search
Search & Replace	Replace Again
Search & Replace	Replace All
Search & Replace	Replace Dialog
Search & Replace	Replace First
Search & Replace	Replace Last
Search & Replace	Replace Next
Search & Replace	Replace Previous
Search & Replace	Reset search marks
Search & Replace	Search Again
Selection modes	Column selection mode
Selection modes	Line selection mode
Selection modes	Marks the beginning of a block
Selection modes	Marks the end of a block
Selection modes	Normal selection mode
Selection modes	Reset selection
Selection modes	Set insert mode
Selection modes	Set overwrite mode
Selection modes	Toggle insert/overwrite mode
Standard actions	Copy selection to clipboard
Standard actions	Copy to clipboard in RTF format
Standard actions	Cut selection to clipboard
Standard actions	Delete current selection
Standard actions	Paste clipboard to current position
Standard actions	Perform redo if available
Standard actions	Perform undo if available
Standard actions	Select entire content, move cursor to the end
Text folding	Collapse all blocks in the text
Text folding	Collapse block at current line
Text folding	Collapse ranges in selection
Text folding	Collapse selected block
Text folding	Collapse/expand block at current line

Text folding	Collapse/expand nearest block
Text folding	Expand all collapsed blocks in the text
Text folding	Expand block at current line
Text folding	Expand ranges in selection
Text folding	Toggle Folding
Tools	Auto completion popup
Tools	Auto correct all words
Tools	Auto correct current word
Tools	Code parameters tool tip
Tools	Code templates popup
Tools	Insert character popup

Customizing SQL Editor Options and Behavior

Use **Options → Editor…** top level menu in SQL Editor to customize editor's options and behavior. The following options can be customized:

Insert mode - Inserts text at the cursor without overwriting existing text. If Insert Mode is disabled, text at the cursor is overwritten. Note that you can use the Ins key to toggle Insert Mode in the editor without changing this default settings.

Auto indent mode - Positions cursor under the first nonblank character of the preceding nonblank line when you press Enter. Note, that SQL Assistant's automatic syntax-based code auto-indenting/formatting is applied after the editor and overrides editor's settings. SQL Assistant's automatic syntax-based code auto-indenting/formatting can be customized in the Options dialog.

Backspace unindents - Aligns the insertion point to the previous indentation level (outdents it) when you press Backspace, if the cursor is on the first nonblank character of a line.

Group undo - Undoes your last editing command as well as any subsequent editing commands of the same type, if you press Alt+Backspace or choose Edit → Undo menu.

Group redo - If it is set Redo will involve group of changes.

Keep caret in text - Allows moving edit caret only within text. All trailing blanks and empty lines are ignored

Double click line Highlights the line when you double-click any character in the line. If disabled, only the selected word is highlighted.

Fixed line height - Prevents line height calculation. Line height will be calculated by means of Default Style. If not enabled, text line height may vary depending on the fonts you selected for syntax highlighting (you can choose different fonts for different element types).

Persistent blocks - Keeps marked blocks selected when the cursor is moved using the arrow keys, until a new block is selected.

Overwrite blocks - Replaces a marked block of text with whatever is typed next. If Persistent Blocks is also selected, text you enter is appended following the currently selected block.

Show caret in read only mode - Shows edit caret in read only mode. (a file is opened in read-only mode)

Copy to clipboard as RTF - Copies selected text in RTF format, including fonts and syntax colors

Enable column selection - Enables column selection mode.

Hide selection - Hides selection when editor loses focus.

Hide dynamic - Hides dynamic highlighting when editor loses focus.

Enable text dragging - Enables drag & drop operations for text movement.

Collapse empty lines - Collapses empty lines after a text range when this range is folded.

Keep trailing blanks - Keeps any blanks you might have at the end of lines. If not checked, trailing blanks are automatically discarded.

Float markers - If it is set, markers are linked to the text, so they will move with text during editing. Otherwise they are linked to the edit caret position, and stay unchanged during editing. Also markers save scroll position.

Undo after save Keeps undo buffer unchanged after save, otherwise resets the undo buffer

Disable selection - Disables any text selection.

Draw current line focus - Draws rectangle around current text line 9but only when the editor has input focus).

Hide cursor on type - Hides mouse punter when you type text in the editor

Scroll to last line - When this option on, you may scroll to the last line of text, otherwise you can scroll to the last page. When this option is off and total text height less then the editor's client are height, its vertical scroll bar is hidden.

Greedy selection - If this option is set, the selection grabs extra column/line during column/line selection modes.

Keep selection mode – The selection enabled for caret movement commands (like in BRIEF).

Smart caret - Optimizes the edit caret movement (up, down, line start, line end). the caret is moved to the nearest position on the screen.

Word wrap - Wraps long text lines at the right side of editors client area.

Word break on right margin - Wraps text at the right margin instead of right side of the editor's client area. You can setup margin positions anywhere within the client area.

Optimal fill - Begins every auto indented line with the minimum number of characters possible, using tabs and spaces as necessary.

Fixed column move - Keeps position of the edit caret before editing text, this position is used when moving the current up or down.

Variable horizontal scroll bar - Sets maximum range of the horizontal scroll bar to the maximal width of visible lines only, not the entire text; hides horizontal scroll bar if all visible lines fit the client area width.

Unindent keep align - Restricts unindent operations when at least one of the selected lines can not be

unindented.

Gutter options

Visible - Shows or hides the left side gutter.

Line numeration - Shows line numbers.

Width - Width of the gutter, default is 40 pixels.

Color - Color of the gutter, default "Button Face".

Right margin options

Visible – Shows line at the right margin of the editor.

Right margin - Sets the right margin of the editor. The default is 80 characters.

Color - Color of the right margin line.

Other options

Editor font - Default editor's font, which is used for displaying plain text without syntax highlighting.

Number font - This font is used to display line numbers.

Background color - Background color of the editor window.

Undo limit - Specify the number of keystrokes that can be undone. The default value is 32,767 (32K).

Tab stops - Sets tabs that the cursor will move to when you press Tab key. Enter one or more integer numbers separated by spaces. When multiple tab stops are entered, the numbers indicate specific columns in which the tab stops are placed. If each successive tab stop is not larger than its predecessor, you will receive an error. When only one tab stop is specified, it indicates the number of spaces to jump each time you tab.

Collapse level - Specifies level of text ranges affected by the "Collapse all" command. If this value is equal -1, all text ranges are collapsed.

Tab mode -Specifies processing of the Tab key. The following values are supported::

- **Use tab character**: Inserts tab character. If disabled,

- **Insert spaces:** Inserts space characters.

Customizing Syntax Highlighting

SQL Editor Professional Edition offers reach capabilities for customizing SQL code syntax highlighting. It enables you to use different fonts, colors,, display styles and even borders for different syntax elements.

Use **Options → Syntax Highlighting…** top level menu in SQL Editor to open the syntax highlighting options. This menu opens the **Style Collection** dialog box. To change display style for a particular type of syntax item, select item type in the item types list box, then customize its rendering options using controls available on the on the right side of the dialog.

The following options can be used:

Style type – This drop-down list provides preselected set of styles for quick customization. You can choose one of the following

- **Font style and colors** – allows setting almost all available display parameters for the selected item type including font, colors, vertical text alignment, borders. The background color cannot be changed.

- **Back and fore-ground** – allows setting all available display parameters including font, colors, vertical text alignment, borders, background color, and so on

- **Only background** – allows setting only background colors and borders. This style is typically used for customizing display of selected text selection

- **Custom font** – allows setting all available display parameters including font, colors, vertical text alignment, borders, background color, and so on

Names and all other options appearing in the Syntax Highlighting Options dialog are self descriptive and do not require additional usage instructions.

CHAPTER 35, Document Manager and Code History Add-on

Overview

SQL Assistant includes **Document Manager and Code History** add-on for a number of supported target environments. This is a dual-purpose add-on, which provides you with the following functions:

1. Automatically restores editor tabs when you reopen target editor environment.

2. Provides access to named file and unnamed script change history and creates periodic snapshots of code changes cataloging them as file revisions. It also enables comparing different revisions with each other, as well as with the head revision, and the script in the current editor tab.

3. Visualizes file context when closing target editor environment. the standard Save Files dialog is replaced with Document Manager's Save documents dialog enabling you to see content of the opened files and their specific changes and choose what to save, instead of the default "dumb" prompt to save or not to save all changes at once.

Important Note: The Code History function is not a substitute for a Source Code Control System, The Code History function stores script revisions for a limited time only in a local file system. It does not support team development, release labeling, and other common functions typically supported by Source Code Control Systems. If you need a full featured Source Code Control System, refer to CHAPTER 23, Database Source Code Control Interface.

Enabling and Customizing Document Management Interface

You can enable or disable the document management interface and to customize its behavior in SQL Assistant's **Options** dialog. For all target environments supporting the **Document Manager and Code History** add-on, the **Document Manager** section appears in the target options on the **Targets** tab as shown on the following screenshot.
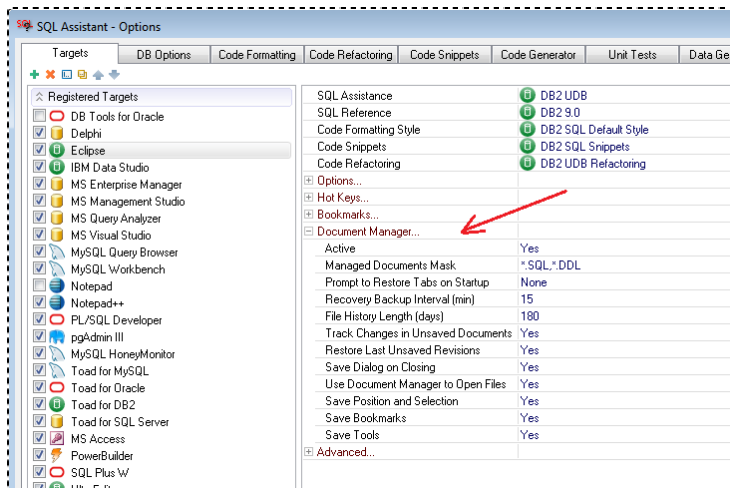
Note that for target environments not supporting the **Document Manager and Code History** add-on, the **Document Manager** section is hidden in the options and cannot be customized.

Restoring Tabs, Connections, Bookmarks, Edit Positions

The Restore Tabs function enables you to reopen editor tabs that were opened in previous target editor sessions along with the associated SQL Assistant's bookmarks, scroll and edit positions, and other tools opened in that document's previous session, so that you can conveniently continue working with your scripts as you left them at the time of target editor closing. Note, that the database connection context is restored too.
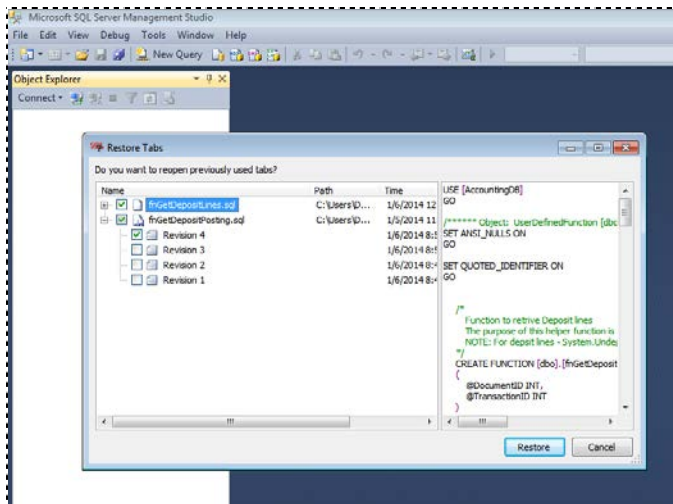
By default the **Restore Tabs** dialog is displayed on startup. To open the **Restore Tabs** dialog at a later time, use **Recent Documents** command in SQL Assistant's menu. In the **Restore Tabs** dialog you can choose specific scripts and their versions that you want to reopen. For more information about script and versions, see Saving Code Change History topic in this chapter.

The following icons are used in the dialog to indicate types of documents that can be restored

Icon	Description
	A named file with unsaved changes. A previous instance of the target editor was closed or crashed before changes in this file changes were saved
	A named file with saved changes.
	A unnamed script with unsaved changes. This script was opened in previous target editor instance with unsaved changes.
	A file deleted from the disk with a revisions still available in the code history. .
	Icons with a little lock overlay in the left-top corner indicate locked files and scripts. Locked documents are not updated in the code history log and no new revisions are saved until they are unlocked.

Tip: Mouse-over a file in the list and rest the mouse pointer for a second. File type description, and file's original connection information will be displayed in the mouse-over balloon.

In the dialog tick checkboxes in front of the document names and then click the **Restore** button to reopen them in the target editor. Note that by default the last saved versions are restored. To restore a particular version, expand the document branch and tick the checkbox in front of the required version as demonstrated on the previous screenshot. To quickly select or deselect all documents, use the right-click context menu and select **Select All** or **Deselect All** command.

For your convenience a preview of the selected document is displayed in the Preview box on the right side of the **Restore Tabs** dialog. If you select a particular revision, the preview highlights code changes.

By default The highlighting shows changes as compared to the head revision. The highlighting method and the change-map are the same as in the **Code Compare** dialog. For more information on the code compare styles and controls, see Using Code Compare Dialog in CHAPTER 25. To see changes relative to the prior revision instead of the head revision, right-click the revision name and then in the content menu choose **Compare –> Prior Revision**. To hide the changes, right-click the revision name and then in the content menu choose **Compare –> None**.

In SQL Assistant's options you can configure to display a simply Restore Tabs prompt with Yes/No buttons instead of instead of the Restore Tabs dialog or to automatically restore all tabs without prompting the feature by selecting the **None** option. See Customizing Add-on Behavior topic later in this chapter for more details.

Tips:

- Only non-empty editor tabs are restored. Tabs having unsupported content types are ignored by the add-on.

- Tabs are restored in a single target editor instance only. If you open two or more target editor instances, no prompt to reopen tabs will appear in the second instance and consequently no tabs will be restored there. If your editor crashes, before you open a new one, use the Windows Task Manager to verify the crashed process is closed completely. For example for SQL Server Management Studio verify ssms.exe or sqlwb.exe are not listed in the processes list. Terminate them If they are still in the task list, otherwise you will not be able to restore tabs and recover unsaved changes.

- The original database context is restored in the restored tabs, but that only works if the connection parameters were saved for the connection. See Ad-hoc and Remembered Connections topic in CHAPTER 2, Connecting to Your Database for more details. Connections cannot be restored for tabs with ad-hoc connections. In the latest case, the current editor's connection is reused for the restored tabs. To quickly see which connection will be used, mouse-over a file in the list and rest the mouse pointer for a second. The original connection information will be displayed in the mouse-over balloon.

Code Change History

While you work with your script, the add-on monitors text changes in the editor. If it finds recent changes, it automatically caches all changes and saves them as script revisions in the special **History** subfolder located in %APPDATA%\SQL Assistant folder. Please note that the default location of this subfolder varies in different Windows versions. If you do not know the value of %APPDATA%\ environment variable, open DOS command prompt, and execute echo %APPDATA% command.

Saved file revisions are renamed according to NNNNNNNNV_FILENAME_YYYYMMDD_HHMMSS.sql naming convention: The following parts are used in the file name:

- NNNNNNNN - this is the hash value of the full file path, which is used to distinguish similarly named files opened from different locations.

- V – file source type: value 1 – represents auto-generated file created from an unnamed script tab; value 0 represents existing file opened from the file system.
- FILENAME - source file name without extension.

    Note: File name may repeat for unrelated files stored in different source folders and for unsaved scripts whose names are auto generated by the target editor.

- YYYYMMDD_HHMMSS – file version timestamp.

Tip: By default cached versions are stored for up to 180 days. You can customize the cache expiration time for your target editor in SQL Assistant's Options. on the Targets tab, select your editor, and then expand the Document Manager options group, and change **File History Length (days)** value.

Reopening Recently Opened Files and Unnamed Scripts

The Recent Documents function enables you to reopen named files and unnamed scripts that you had opened and edited in the target editor in the last 180 days or as specified in SQL Assistant options, as well as reviewing revisions of the currently opened script in a vertical pane docked to the left side of the editor window.

Use **SQL Assistant → Recent Documents…** menu command to open the **Recent Documents** dialog.
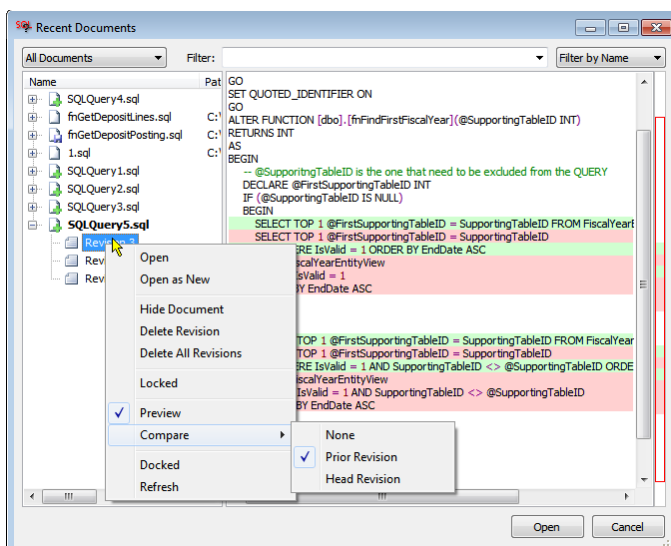

The **Recent Documents** dialog is similar to and provides the same interface and functions as the **Restore Tabs** dialog. In addition it provides flexible search and filtering by file names, and by content, as well as, it allows you to manage documents and their revisions using simple right-click menus. The following functions are supported via the right-click menu.

Open – Opens the selected document or revision as a new tab. If the selected document is already open, actives its tab.

Open as New – Opens a copy of the selected document or revision as a new tab and as a new script.

Hide Document – Hides selected document or revision in the document list, but does not delete the associated files or revision history.

Delete Revision – Deletes the selected revision from the list and also deletes the associated file from the file system

Delete All Revision – Deletes the selected document with all its revisions from the list and also deletes the associated revision files from the file system.

Lock– Locks the selected document. Locked documents are not updated in the code change history log and no new revisions are saved until they are unlocked.

Unlock – Unlocks the selected document.

Refresh – Forces refresh of the **Recent Documents** table of contents.

Docked – Opens the **Document Revisions** pane. An example of the **Document Revisions** pane is displayed on the next screenshot.
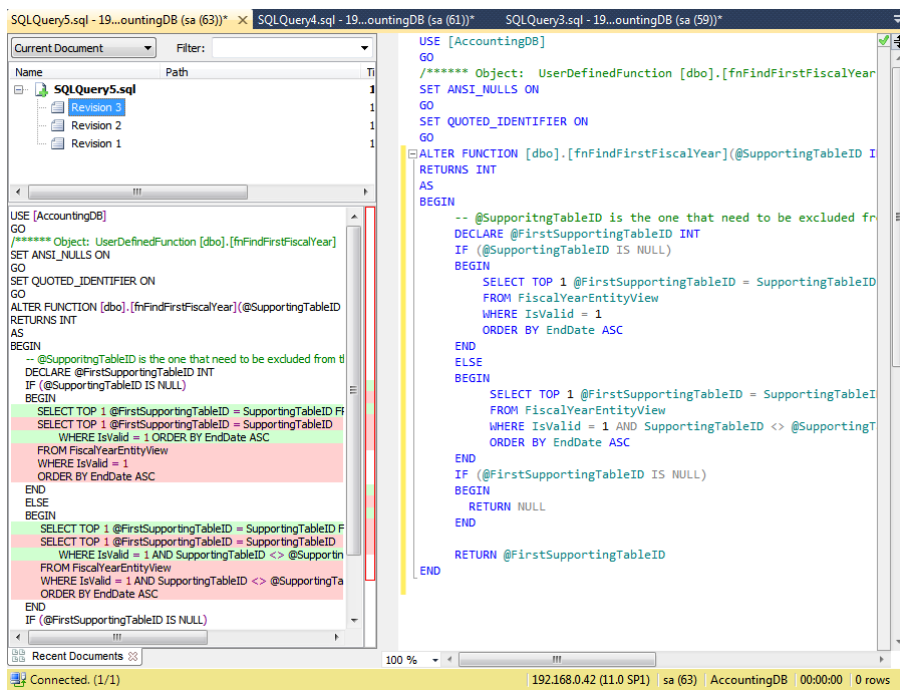
The **Document Revisions** pane enables 1-click access to the revisions of the script in the current editor tab. To see revisions in other documents, change the filter in drop-down list box displayed in the left-top corned from the **Current Document** to the **All Documents** item.

The Filter drop-down list and the Filter box provide the filtering functions you can use to quickly locate the required documents.

To filter by file name:

1. Select **Filter by Name** item in the drop-down list available in the top-right corner of the Recent Documents dialog

2. In the **Filter** box, enter partial or full file name to use as a filter. Note that the entered text can appear anywhere within file name. The entered filter is applied automatically as you type the filter text.

To filter by file content:

1. Select **Filter by Content** item in the drop-down list available in the top-right corner of the Recent Documents dialog

2. In the **Filter** box, enter text to use as a filter. Note that the entered text can appear anywhere within file content.

3. Press the Enter key to apply the entered filter.

Comparing Script Versions

Method 1

1. Use **SQL Assistant → Recent Documents…** menu command to open the Recent Documents dialog.

2. Locate the document whose historical versions you want to compare and expand that document's branch.

3. Locate and select the required version. A preview of that version script will appear on the right side of the dialog in the Preview box. All changes in that version will be automatically compared to the head version and highlighted in the Preview. To compare to the previous version instead, right-click the selected version, and from the context menu choose **Compare –> Previous Revision**

Method 2

1. Use **SQL Assistant → Compare Code and Data → Compare Script Versions…** menu command to compare current and historical code versions. This command will open **SQL Assistant – Script Versions Compare** dialog preloaded with the current script.

2. Use the available controls to pick one of the previous versions to compare against the current script. Note that this Compare dialog is a variation of the Code Compare utility. It provides the same type of user interface and code comparison functions. For more information on the usage of the Code Compare function, read CHAPTER 25, Code Compare Utility.

Saving Tabs, Bookmarks, Edit Positions

The **Save Documents** dialog replaces default Save Changes message box displayed by target environment when you close with unsaved changes in one or more tabs. The **Save Documents** dialog shows changes in all tabs enabling you to choose what to save and what not, review revisions, choose destination file names all in one place.

Tabs with unsaved changes are listed on the left side of the dialog. To see content of a particular tab and its revision highlights, click that tab in the tab list. The content of the selected tab appears on the right side of the dialog. The content is color coded to indicate new, modified, and deleted lines. Pink and light green highlights along with ✚ and ✖ marks indicate changes in the text. To document change map is displayed to the right of the text and can be used for quick text navigation. For more information on how to use the map, read CHAPTER 25, Code Compare Utility

The **Save Documents** dialog recognizes changes in existing files and in unnamed scripts. It automatically pre-selects existing files so that their changes are flushed to the disk when you click the **Save** button.

The dialog also saves copies of your changes to the file change cache in the History folder as described in the Overview topic in this chapter. In addition to file changes, it saves current cursor positions and bookmarks so that your documents can be restored on the editor startup as they were at the time of the editor closing.

Note: The Document Manager automatically saves code changes to cache in the **History** folder. Even if you choose to not to save changes in files, you can still restore and access unsaved scripts and changes from the History cache, until the cache expired. By default old cached versions of unsaved files are purged from the cache after 180 days.
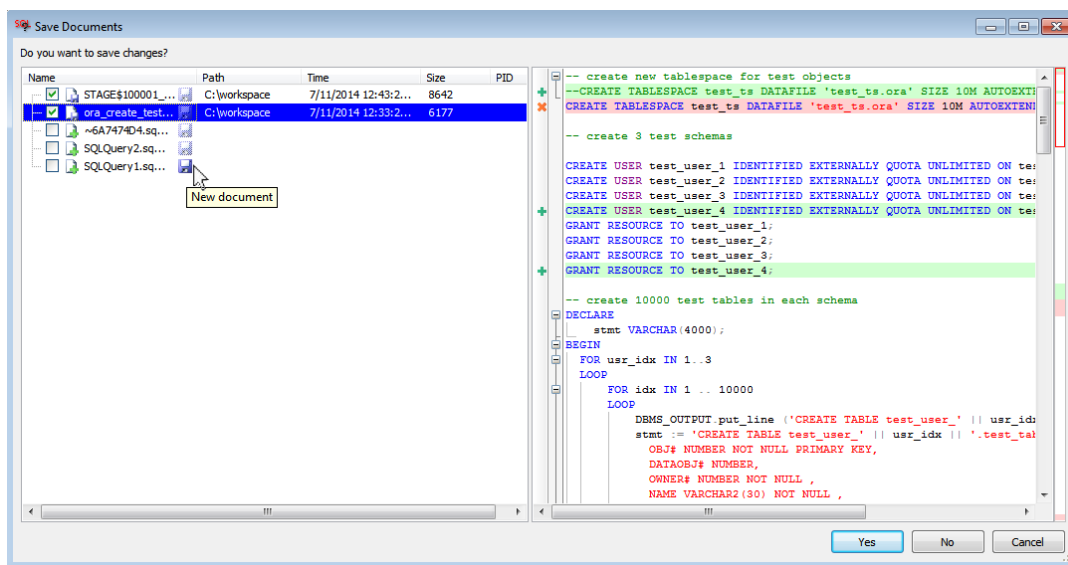
The following icons can appear in the list of unsaved tabs and files.

This icon represents an existing file with unsaved changes.

This icon represents an existing file with changes already save or with no changes. This file icon can appear for non-modified files opened in the target editor and also after you use the Save File icon to save changes in a particular file.

This icon represents an unnamed script never saved before.

Use checkboxes to the left of document icons to select which files you want to save to the disk, and then click the **Yes** button to save selected changes and to close the **Save Documents** dialog.

To quit the target editor environment without saving any changes, click the **No** button.

To cancel closing the target editor environment and to return back to the editor, click the **Cancel** button.

CHAPTER 36, Testing Database Performance

Overview

You can use the Database Benchmark utility to conduct database workload and scalability testing, to test your database code under stress and to eliminate slow SQL database performance. You can also use it to identity areas in your database that might be sensitive to user concurrency issues and deadlocking. It enables you to deploy changes to your database schemas and code with a confidence, as well as dramatically improve your database performance and application scalability.

The testing can be performed interactively in a real time or scheduled to run during off pick hours or simultaneously using multiple computers in a computer farm or an array of virtual machines..

The Database Benchmark supports 2 load generation methods:

- Using database calls to test your database the same way your applications would use them.

- Using external applications reproducing production like application workload, real user behavior and real data I/O patterns.
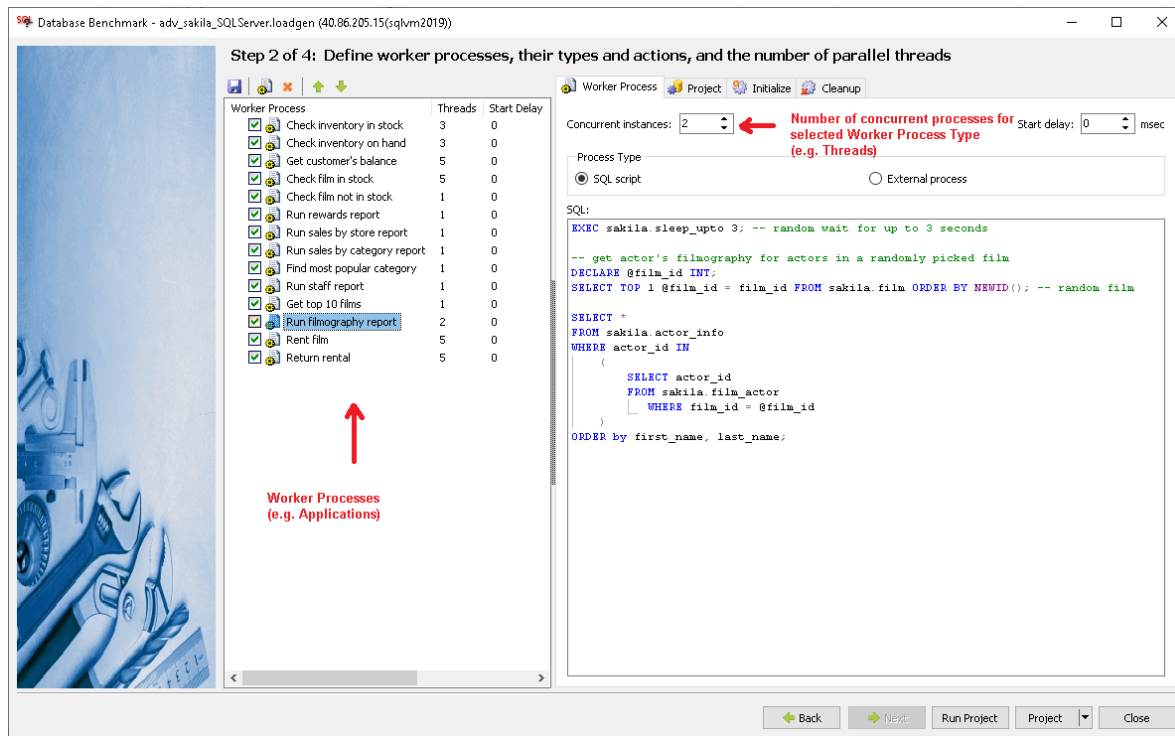
To launch the Database Benchmark, use right-click menu in the SQL Editor. Alternatively, you can right-click the SQL Assistant icon in the Window system tray and, in the right-click menu, navigate to the SQL Assistant submenu. Choose **Code and Data Generators □ Database Benchmark** menu command.

Common Concepts

The Database Benchmark can be used to generate database workloads. To simplify workload generation management, the Database Benchmark supports load generation projects. Load generation projects are useful for:

- Testing different database usage scenarios.

- Testing your database performance under stress test.

- Establishing performance benchmarks; testing application and database performance after code changes, software and hardware upgrades and comparing them to the previously set benchmarks.

- Testing scalability of your database dependent applications with more users and/or more data.

- Persistent storage of project configuration parameters and load generation rules within a set of project configuration files.

Worker Processes and Threads

The Database Benchmark uses **Worker Processes** to simulate database applications. Each Worker Process represents one application. To test multiple applications running concurrently, add a separate Worker Process for each application to the Database Benchmark project.

To simulate multiple users running the same application concurrently configure number of **Concurrent Instances** (e.g. **Threads**) to run the Worker Process. One concurrent instance represents one user running an instance of an application. If you add 10 Worker Processes to the project and configure each Process to be run by 10 concurrent instance, in total the Database Benchmark will create and run 100 threads simulating 100 concurrent database users.

Each **Thread** runs the associated unit of work continuously in loop for as long as specified in the **Total Load Duration** project scope parameter. See Project Scope Options topic for more details

Worker Processes can be of two types:

- **SQL Script** – used for executing database SQL scripts simulating typical application use conditions.
- **External Process –** used for running user applications, which are expected to connect to the database being tested and to run the real-life database operations.

Benchmark and Workload Templates

Ready to run templates are available for the following database types:

- Amazon Redshift

- SQL Server

- Azure SQL Database

- DB2 UDB

- MySQL

- MariaDB

- Oracle

- PostgreSQL

- SQLite (limited support)

Three kinds of templates are preinstalled for all listed above database types excluding SQLite:

Sakila template – the objective of this template is creating database workload mimicking behavior of a real world application and its impact on the database operations. The template is modeled after well known Sakila database originally developed by Mike Hillyer of the MySQL AB documentation team

Scale template - the objective of this template is creating intense load and scalability benchmarking environment. This is a modification of the Sakila template configured for high concurrency and gradual load generation.

Pgbench template - the objective of this template is creating intense load and performance benchmarking environment. This template is modeled using commonly used pgbench performance benchmarking application originally developped for PostgreSQL, which was modeled after TPC-C benchmark

Using Database Benchmark in Conjunction with Test Data Generator

It is a good idea to use the Database Benchmark against a well populated database in order to obtain good quality test results. The performance impact could change dramatically with more data stored in database tables and with larger IO workloads. You can use the Test Data Generator tool to pre-populate your database with large amounts of data before running database load tests.

Tip: It is also possible to invoke Test Data Generator projects during load testing to simulate batch processing and bulk loading of large amounts of data. For that purpose, configure one or more Worker Processes in the Database Benchmark Project to run as external process. In the external process properties use Test Data Generator's command line interface to run the required Test Data Generator project. For more information see CHAPTER 18, Test Data Generator Command Line Interface topic.

Scalability

The performance of the load test and it is impact on the database server being tested might be constrained by the computing power of the system (computer or virtual machine) on which you are running the Database Benchmark database. Using bigger hardware (vertical scalability) for the Load Test Generator system is not always faster—but it can usually handle more load and run more Worker Processes generating the load, but non necessarily generating bigger throughout. All network cards have a limitation on the maximum throughput they can handle, which is typically 100MB/s. That is the reason that more hardware does not automatically improve the quality of load testing.

Scaling horizontally, adding more systems running the Database Benchmark concurrently, eliminates the computing power constraints and the network card throughput on the Database Benchmark side. To support this horizontal scalability, SQL Assistant allows you to schedule running of the Database Benchmark projects, so that you can have multiple systems running the same load tests at the same time against the same database server. It also allows you to save test results from multiple systems to a common database table in a shared repository database providing you with a complete result set for further analysis of the database server

and application performance during heavy workloads.

Working with Database Benchmark

Opening and Saving Projects

To open an existing load generation project to modify project details or to execute a load generation run:

1. Use right-click menu in the SQL Editor to launch the Database Benchmark,. Alternatively, you can right-click the SQL Assistant icon in the Window system tray and, in the right-click menu, navigate to the SQL Assistant submenu. Choose **Code and Data Generators □ Database Benchmark** menu command.

2. Click the 📁 button displayed in the top left corner of the Database Benchmark dialog. The Open Database Benchmark Project dialog opens.

3. Select the project file you want to open, then click the **Open** button. The Database Benchmark dialog will be populated with Worker Processes and their individual settings.

To save all settings in a project file.

1. Click the 💾 button in the top left corner of the Database Benchmark dialog. The Save Database Benchmark Project dialog opens.

2. Select the project file to which you want to save project settings and click the **Save** button. To distinguish project files from other XML files, it is a good idea to use "project" or a similar prefix or suffix when naming project files .

Adding Worker Processes to a Project

To add new processes to a Database Benchmark project:

1. Click the 🔧 button in the top left corner of the Database Benchmark dialog. A new Worker Process is added to the list with the default name **New Worker Process** and with an input focus.

2. Type new name for the added process. It is recommended to use self descriptive names so that later you could easily figure out what the process is used for.

3. On the right side of the Database Benchmark dialog fill in process properties. See Worker Process Scope Options topic later in this chapter for more information on the supported Worker Process properties.

Removing Worker Processes from a Project

To remove processes from a Database Benchmark project:

1. On the Database Benchmark window select the Worker Process you want to delete.

2. Click the ✖ button in the top left corner of the Database Benchmark dialog.

3. Save your project changes as described in the "Opening and Saving Projects" topic.

Tip: To disable a Worker Process without deleting its definition, deselect the checkbox in front of the Worker Process.

Disabling and Enabling Worker Processes

To disable or enable a specific process:

1. In the project tree, select the case you want to modify.

2. Deselect the check-box in front of the case name to disable a case. Select the check-box to enable a selected case.

Modifying Database Workload Generation Options

Each Worker Process in the project has its own separate set of properties for the workload generation. To modify the properties.

1. In the worker processes list, click name of the Worker Process whose properties your want to modify. Make sure the Worker Process name is selected and the checkbox in front of the process name renames selected too. The Worker Process properties pane will be displayed at the right side of the Database Benchmark window.

2. Choose the appropriate generation method and options. For specific details on available data methods and options, see the topic Worker Process Scope Options in this chapter.

3. Repeat steps 2 and 3 for other Worker Process in the active project

Saving and Analyzing Load Test Results

At the end of the project run, the Database Benchmarks opens a new **Results** tab with the results of the processing

Note: The **Results** tab displays aggregate results for all worker processes. If you setup multiple Worker Processes in your project, to see their detailed results, click on each enabled worker process. To see all results at once, click white space in the Worker Process List so that no workers appear selected.

If you haven't selected in the project scope options to automatically save results to a file or database table, it is still not too late to save them. click the Save Results hyperlink displayed in the right-top corner to save results to a file. You can choose to save results to an XML file, CSV or tab-separated file. See the Project Scope Options topic in this chapter for details on the output file format and its sample content.

If you saved results to a file, you can use common analytical tools such as Microsoft Excel or specialized tools to slice and dice the load test data and to chart various statistics.

If you configured the project to save results to a database table, you can use SQL queries with various aggregate and analytical functions available in your database to analyze the results. In the simplest case the following sample query can be used to retrieve results from the log table for a particular test run and schenario

```
SELECT start_time,
       duration
FROM   dbo.SA_LOADGEN_LOG
WHERE  process_name = 'Data feed processing'
       AND id = '20140524 011527'
ORDER BY start_time
```

Tip: If you setup several different projects for different load scenarios or database configurations, you can compare their results side by side to find out scenario with best performance and to find out scenarios leading to performance anomalies.

Scrolling Content

Use the scroll bars in the **Database Benchmark** window to scroll the panes. Alternatively, you can use the keyboard navigation keys or mouse wheel.

Note that the table data generation properties pane scrolling is a bit different from other panes. This pane has two parts: fixed columns, which are not scrollable, and data generation properties columns, which are scrollable.

Resizing Content

To resize the **Database Benchmark** window, drag the top edge of the window up or down, left or right.

To adjust sizes of left and right panes of the Database Benchmark window, place mouse pointer over the right edge of the Worker Processes list box. The cursor shape changes to resize shape as on the following screenshot.

Drag the edge to adjust pane size. Make sure the cursor takes the right shape before dragging the pane edge.

To resize individual columns in the Worker Processes list, drag the right-edge of the column header left or right. Note that when you place mouse pointer over the right edge of a column header the cursor shape changes to resize shape as on the following screenshot.

Make sure the cursor takes the right shape before dragging the column edge.

Tip: When column width is too narrow to fit the content, 3 dots (also called ellipses) are drawn in each cell with non-fitting data to indicate the data overflow effect. To quickly resize a column so it fits the entire content in all cells, double-click click on the right-edge of the column header. SQL Assistant will calculate the required width and resize this column as needed.

Running Database Load Test

To generate database workload:

1.   Start the Database Benchmark. See the Overview topic for more details.

2. Create new or open an existing project. See the Opening and Saving Projects topic for more details.

3. If required, add or remove Worker Processes to the project. Choose required Project Scope Options, and Worker Process Scope Options. as described in the following topics.

4. Click the **Generate** button to start the load test or click **Schedule…** button to schedule it to run at a later time.

You can also use the command line interface to run Database Benchmark projects either from command window or from other applications. See the Command Line Interface topic for more details.

Scheduling Database Benchmark Project Runs

To schedule a Database Benchmark project run in unattended mode:

1. Open the project you want to schedule for unattended execution. See the Opening and Saving Projects topic for details.

2. Click the **Schedule...** button in the right bottom corner of the Database Benchmark window. The **SQL Assistant – Task Schedule** dialog will appear.

3. Enter the task schedule properties.

Task Schedule

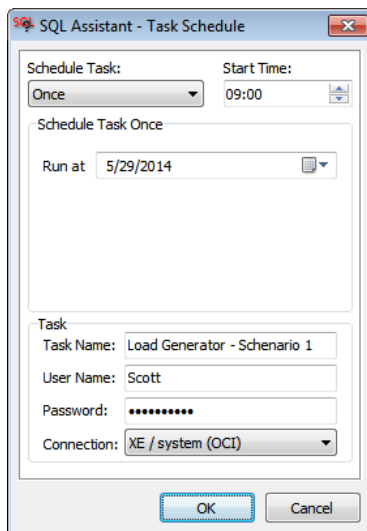To schedule a one-time run, in the Schedule Task drop-down select **Once**. Enter a date and time to start the task.

If you select the **Daily** option, you can enter the recurrence interval for the task and the date and time to start the task. An interval of 1 produces a daily schedule, and an interval of 2 produces an every other day schedule. The task will start at the specified time each day.

If you select the **Weekly** option, you can enter the recurrence interval for the task, the date and time to start the task, and the days of the week in which to start the task. An interval of 1 produces a weekly schedule, and an interval of 2 produces an every other week schedule. The task will start at the specified time on each of the specified days.

If you select the **Monthly** option, you can enter the months in which you want to start the task and the weeks and days of the month in which you want to start the task. You can also specify that you want to start a task on the last day of each selected month.

Task Properties

Task Name – the name of the scheduled task. This is a required property. By default, the Database Benchmark project name is used for the task name. The name cannot contain special characters not allowed for use in file names.

User Name – the name of the Windows user account that will run the task. To specify a local user name, enter the name in `.\user` format or `machine\user` format. To specify

domain user name, enter the user name in `domain\user` format.

Important Notes: Do not confuse the Windows user account that will run the task with the database connection user account. The Windows user account is used to schedule and start the process. The database connection user account is used to connect to the database server. The database connection user is specified in the connection properties of the connection associated with the task. It may be different from the Windows user account. However, if the connection properties are set to use **"Windows Authentication"** method, the database connection will be attempted within the security context of the Windows user account specified for the task.

See CHAPTER 2, Connecting to Your Database for more information on supported connection methods and their properties.

Password – the password of the Windows user account that will run the task.

Connection – the database connection the task will use for the project run.

See the CHAPTER 15, topic Managing Connection Groups and Connection Settings for more information on how to add, modify, and delete connections.

4. Click the OK button to create the scheduled task and to close the **SQL Assistant – Task Schedule** dialog

Database Benchmark Options

Project Scope Options

To configure project scope options, active the **Project** tab page.
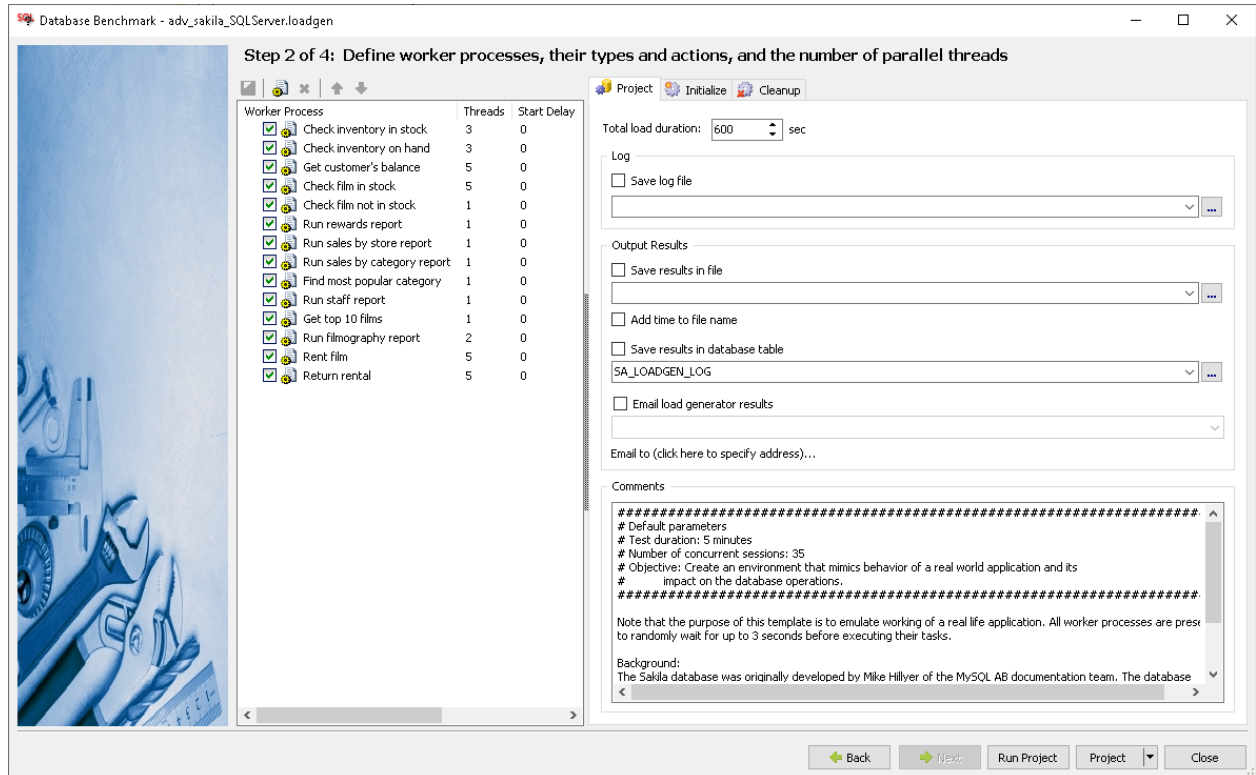
The following options are supported:

Total Load Duration

> The total duration for project run. At the end of the total run SQL Assistant forcibly terminates all all threads of all Worker Processes participating in the project run.

Log

> **Save Log File** – this option specifies whether a log file is written for the project run. The log file is used for troubleshooting problems that might occur during the project run. The log file is written in plain text format and contains progress of work and other diagnostic messages.

> **Log File** – specifies the log file name. A log file is only created only if the **Save Log File** option is enabled.

Output Results

> **Save Results in File** – this option specifies whether an output file is written for the project run. The output file records the results of the load test in the selected output format. The output data can be used for analyzing load test results.

The output file is written at the end of the project run. The output file format depends on the selected output file extension. The following formats are supported:

- **XML** – output results are recorded in XML format. Each node in the output file contains 7 named properties:

name	Name of the worker process
status	Execution status, SUCCESS or FAILURE
threaded	Unique tread id within the named Worker Process which was used for running the associated SQL script or external process.
runid	Run Id for the specified Tread ID. Each run performed by a Thread generates unique Run ID
start	Start time for the specified Run ID
duration	Run duration in milliseconds as decimal number with up to 2 digits after decimal point.
error	Error messages if any returned during worker process run

Here is a small sample demonstrating of XML file content.

```
<lgres>
  <item name="Data feed processing" status="SUCCESS" runid="1" threadid="7576"
start="5/24/2014 1:14:26 PM" duration="13.79" error=""/>
  <item name="Data feed processing" status="SUCCESS" runid="2" threadid="7576"
start="5/24/2014 1:14:26 PM" duration="52.62" error=""/>
  <item name="Data feed processing" status="SUCCESS" runid="3" threadid="7576"
start="5/24/2014 1:14:26 PM" duration="10.57" error=""/>
  <item name="Data feed processing" status="SUCCESS" runid="4" threadid="7576"
start="5/24/2014 1:14:26 PM" duration="11.43" error=""/>
  <item name="Data feed processing" status="SUCCESS" runid="5" threadid="7576"
start="5/24/2014 1:14:26 PM" duration="13.66" error=""/>
  <item name="Data feed processing" status="FAILURE" runid="61" threadid="6416"
start="5/24/2014 1:15:22 PM" duration="5267.32" error="Execution error: Primary
key violation"/>
</lgres>
```

- **CSV –** output results are saved in a comma-separated value file. The values are the same as in the XML output format. See XML format description above for more details.

- **TXT** – output results are saved in a tab-separated value file. The values are the same as in the XML output format. See XML format description above for more details.

Add Time to File Name – this option, if enabled, causes SQL Assistant to add a date and time suffix to the output file name. This ensures that every project run generates unique output file name and does not override previous files.

Save Results in Database Table

This is similar to **Save Results in File** except that results are loaded into a database table at the end of the project run. By default the results are saved into SA_LOADGEN_LOG table which is created automatically in the current database if it does not exists. If you want to save in a specific database, specify fully qualified table name in **database.schema.table** format or **schema.table** format if your database server does not support 3 part object names. The results table has the following structure

```
CREATE TABLE dbo.SA_LOADGEN_LOG
(
    ID                  VARCHAR(15) NULL,
    PROCESS_NAME        VARCHAR(255) NULL,
    THREAD_ID           INT NULL,
    RUN_ID              INT NULL,
    STATUS              INT NULL,
    START_TIME          DATETIME NULL,
    DURATION            DECIMAL(8, 2) NULL,
    ERROR_MESSAGE       VARCHAR(1000) NULL

)
```

The columns are the same as in the XML output format. See XML format description above for more details, with an addition of ID column which stores session id of the Database Benchmark .run based on the load test run startup time.

Important Notes:. For long running load tests generating a lot of results, saving results to a table may take a while.

Send Results

This group of options is used in conjunction with the Output Results group of project scope options. It allows you to automatically email project execution output results. The results can be optionally emailed after each project run. It is intended for use in automated unit tests invoking SQL Assistant's command line interface for running database workloads. See the Command Line Interface topic in this chapter for more details.

The following options are supported:

To – semicolon-separated email addresses of the mail recipients

From – the email address of the message sender

SMTP Host – server name or IP address of your SMTP email server

SMTP Port – SMTP server port used by your server. The default SMTP port is 25.

Requires Authentication – specifies whether your email server requires user authentication

User – specifies a valid user name for your email server logon. This option must be specified if your server requires user authentication and the **Requires Authentication** option is selected.

Password – specifies a valid password for the user name specified on the **User** option. This option must be specified if your server requires user authentication and the **Requires Authentication** option is selected.

Email Database Benchmark Results– specifies whether SQL Assistant should attach the project output file to the email message. This option is ignored if the **Save Unit Test Results** option is not enabled or if an output file name is not specified on the **Output Results** tab page.

Send Results – specifies when SQL Assistant sends project output email. The following values are supported:

- **Always** – project results are automatically emailed after each project run

- **On Failure** – project results are emailed if at least one Worker Process run fails.

Worker Process Scope Options

The following Worker Process scope options are supported:

Concurrent Instances

The number of threads allocated to run the selected Worker Process. Each thread executes tasks specified for the worker process concurrently with other thread of the same Worker Process.

Start Delay

The time in milliseconds to wait before launching next processing thread of the selected Worker Process.

Use this option to increase the load gradually. For example, if you configure a Worker Process with 10 threads and enter 10000 for the delay, the Database Benchmark will create a thread which will begin running the task, then 10 seconds it will later create a second thread, 10 seconds later it will create a third thread, and so on. After 90 seconds all 10 threads of the Worker Process will be up and running

Process Type

Specifies Worker Process type. Two types are supported:

- **SQL Script** – executes SQL batch script containing 1 or more SQL commands
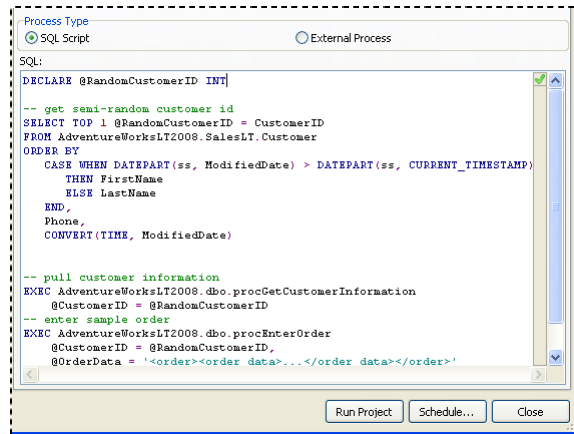
In the **SQL** box enter SQL commands you want the Worker Process to execute continuously during project run. After the script completes, a new script run is performed.

- **External Process** – executes external process such as a batch file or executable program.
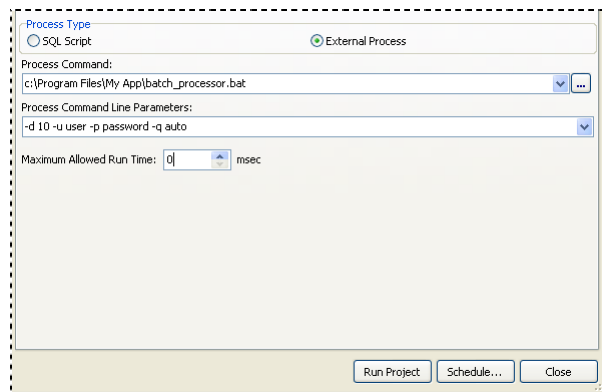
In the **Process Command** field specify full path to the executable file you want the worker to run continuously during project run.

📌 **Note:** Do not add command line parameters to this field.

In the **Process Command Line Parameters** field specify optional parameters for the executable file. If no parameters are required, leave this field blank.

Maximum Allowed Run Time in milliseconds – if the specified executable quits automatically after completing its work, leave this value as zero. Zero value means that the Database Benchmark does not need to do anything to terminate the process. Otherwise enter some value to have the Database Benchmark automatically terminate the launched process after the specified time if it does not quit by that time. After the process is terminated a new instance of the process is launched.

📌 **Tip:** You can use **External Process** worker type to run other SQL Assistant utilities supporting command line interfaces, such as for example, the Bulk Test Data Generator, which can be used to generate large number of database IO (input/output) operations. Or to run the unit tests specified in a Unit Testing Framework project and measure how they perform under stress.

Command Line Interface

To run a database load generation project from a DOS command line window, use the following command:

```
sacmd lg:"path-to-project-file " sas:"path-to-sa-settings-file" conn:"myserver (userid)"
```

Substitute values into the command as follows:

path-to-project-file	The full file name of the Database Benchmark project file
path-to-sa-settings-file	The full file name of the SQL Assistant settings file containing the required database connection parameters. This is an optional parameter. If not specified, the default path for the current user account is used.
myserver (userid)	The database connection name

Example:

```
cd "C:\Program Files (x86)\SQL Assistant 12"

sacmd lg:"C:\Projects\Performance\dev_pos.loadgen" sas:"%APPDATA%\SQL
Assistant\13.0\sqlassist.sas" conn:"DEV001 (sa)"
```

📌 **Important Notes:**

- Database Benchmark project files are XML files you save using the Database Benchmark graphical interface.

- The SQL Assistant settings file location is version and user profile specific. See the Notes in the Overview topic in CHAPTER 51 for details on how to find out the location of that file.

- You can find out the connection name in the DB Connections group of settings on the DB Options tab page in SQL Assistant Options. If a connection requires a user id and password, make sure that both

are saved in the settings. The command line interface does not display interactive prompts and is unable to prompt for credentials during command processing. For more information about storing and managing database connections, see the <u>Managing Database Connections</u> topic in CHAPTER 48.

CHAPTER 37, Improving Code Reusability

Overview

As opposite to traditional object oriented programming (OOP) languages code reuse is difficult to achieve in traditional database-side applications. There are multiple obstacles, but among them the lack of support for grouping of similar or related procedures and functions in classes, packages, encapsulating object/class specific methods, luck of inheritance and so on. Unless you are working with an Oracle database or some recent DB2 UDB version that support somewhat limited OOP implementation and code packaging, you are out of luck. On the other hand, a typical modern relational database system contains many user created stored procedures, functions, views, types, triggers, and other procedural objects spread across multiple databases and schemas, which are often hidden from a plain view. On top of that, it is very difficult to identify in that code repeating code fragments, with repeating or similar business logic contained in them Compare that to traditional OOP languages storing most of their project code in flat text files typically located in a single folder with subfolders, and which are easy to search. Modern development tools for OOP languages index the code and compile code class hierarchies for fast code navigation and analysis enabling you to quickly locate objects and their methods including inherited and overloaded methods and properties.

The SQL Assistant 's **Code Context** feature attempts to improve the situation with database code reusability and enable you to deal with the virtues of DRY (don't repeat yourself).

The Code Context implements the following:

- Stores copies of database code from multiple servers, databases, and schemas in a single location in a local "full text search" (FTS) code repository file.
- Compiles full text indexes for super fast code search by context with results ranking
- As you enter new code or make code changes, it provides real time suggestions for similar and related code already available in your databases so that you can have a look and decide for yourself whether you can reuse that code or you need to code anew.
- Enhances development team collaboration, automatically finding related code written by other developers.

FTS Code Repository

SQL Assistant maintains a local "full text search" (FTS) code repository that it uses internally for code indexing, super fast context search, and for search results ranking. The repository is located in saCodeRepo.db file in the %APPDATA%\SQL Assistant\CodeRepo folder. Note that %APPDATA% is a system environment variable. If you do not know the value of this variable, you can display it by opening the DOS command window, and executing the command echo %APPDATA%.

The FTS code repository must be populated with the database code before you can use the Context Search features. To load code repository with the code from a specific database server:

1. Open your SQL editor and connect it to the required database server.

2. Right click in the editor and from SQL Assistant's menu select **Search and Replace → Update FTS Code Repository** command.

3. Allow SQL Assistant to process the repository update. Depending on the amount of code it may take anywhere from several seconds to several hours. A progress bar will appear in the Code Context page at the bottom of the target editor window. You can continue coding while the

update is running..

4. If you want to load repository with codes from another database server, reconnect your editor to the other database server and repeat steps 2 and 3. For more servers repeat the same steps again. Alternatively, you can schedule automated FTS repository updates that SQL Assistant can run in the background or overnight. For specific instructions on how to do that, see the " To schedule code repository updates" section at the end of this chapter.

Managing FTS Code Repository

To update FTS code repository for the current database server:

1. Open your SQL editor and connect it to the required database server.

2. Right click in the editor and from SQL Assistant's menu select **Search and Replace → Update FTS Code Repository** command.

3. Allow SQL Assistant to process the repository update. Depending on the amount of code it may take anywhere from several seconds to several hours. A progress bar will appear and the Code context page at the bottom of the target editor window. You can continue coding while the update is running..

4. If you want to update repository for another database server, reconnect your editor to the other database server and repeat steps 2 and 3. For more servers repeat the same steps again.

To completely rebuild the code repository database

1. Open your SQL editor and connect it to a database server.

2. Right click in the editor and from SQL Assistant's menu select **Search and Replace → Context SQL Search** command.

3. In the **Context SQL Search** window, right click the list box and from the popup menu select the **Rebuild FTS Code Repository** command.
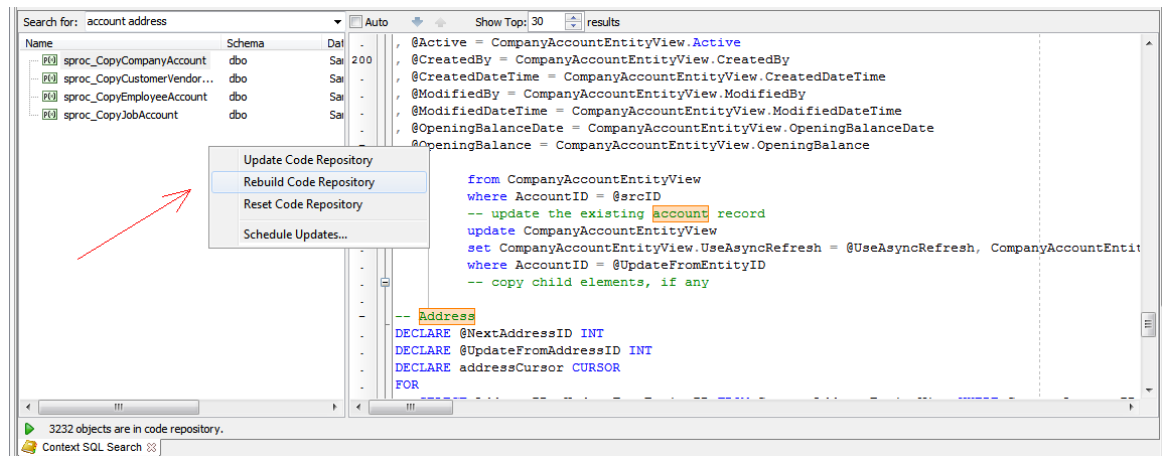
Allow SQL Assistant to process the repository update. Depending on the amount of code it may take anywhere from several seconds to several hours. A progress bar will appear and the Code context page at the bottom of the target editor window. You can continue coding while the update is running..

Important Note: The **Rebuild** command truncates code repository database and then populates it with the code of the current database server. If the repository contained code from other database servers, change your database connection and then use the **Update** command to populate repository with the other database server's code. Repeat it for as many servers as required.

To truncate the code repository database:

1. Open your SQL editor

2. Right click in the editor and from SQL Assistant's menu select **Search and Replace → Context SQL Search** command.

3. In the **Context SQL Search** window, right click the list box and from the popup menu select the **Reset FTS Code Repository** command.

To schedule code repository updates:

1. Open your SQL editor and connect it to a database server.

2. Right click in the editor and from SQL Assistant's menu select **Search and Replace → Context SQL Search** command.

3. In the **Context SQL Search** window, right click the list box and from the popup menu select the **Schedule Updates** command. the Schedule Updates dialog will appear.

4. Fill in the required schedule time and frequency.

5. Repeat steps 1 to 4 for other database servers if required.

Note: It is recommended that you update the SQL Assistant's code repository often in order to keep it up to date with your latest database changes.

The Light Bulb

As you edit the code in the editor, SQL Assistant automatically checks in the background if it can find similar or related codes in the code repository. It displays the light bulb icon 💡 next to the code fragment for which it found one or more matches. Click the light bulb icon to see what it found. The Code Context popup will appear on the screen as in the example pictured below.
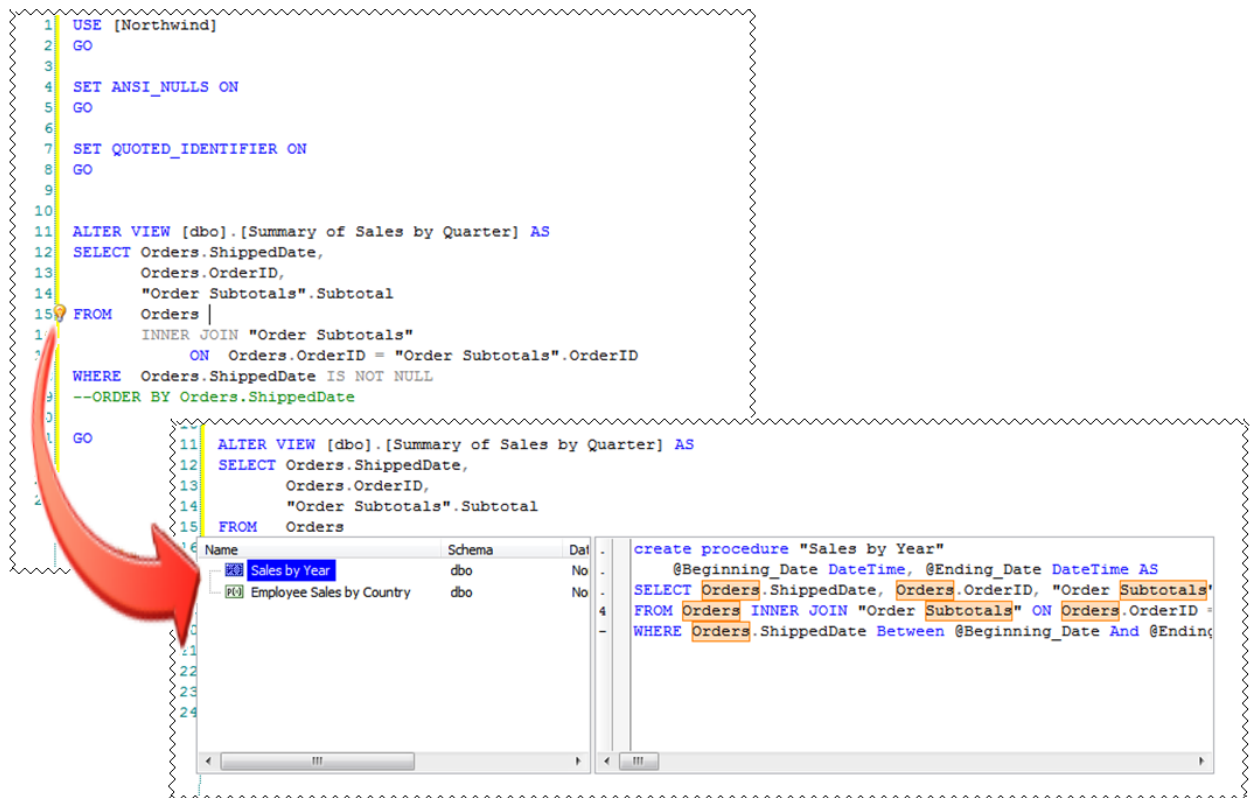
In this example, the user worked with "Summary of Sales by Quarter" view. SQL Assistant found similar code in 2 stored procedures "Sales by Year" and "Employee Sales by Country" and shown the light bulb icon line 15. When the user clicked that icon, SQL Assistant brought up the Code Context popup listing the items and matching fragments of their code. The matching tokens are highlighted with light orange background color.

Important Note: For the Light Bulb feature to be functional, the SQL Assistant's code repository must be populated with your database SQL code. See the Code Repository topic for details on how to populate and manage SQL Assistant's code repository.

Advanced Context-based SQL Search

The context search provides you with advanced methods for very fast searches of your database SQL code across multiple databases and database servers provided that you have populated SQL Assistant's code repository with their stored SQL code.

To open the context search results window, click **Search and Replace → Context SQL Search** command. The Context SQL Search window appears at the bottom of the target editor.

The context search can be used in two different ways. The state of the **Auto** checkbox controls if the searches are performed with search results displayed automatically or a manual input is required to initiate the search.

Automatic search

In this mode, as you type the text in the editor, SQL Assistant picks words before or under cursor and automatically searches for matching text in the code repository. The search terms are copied to the **Search for** input box automatically. If you select several word in the text, the context search is performed for the entire selection. If the selection contains multiple words, the context search uses so called NEAR search to find SQL code containing all specified words within a certain proximity of each other. The behavior of the NEAR search can be customized in SQL Assistants options. See the Tuning Code Context Behavior topic in this chapter for more details.
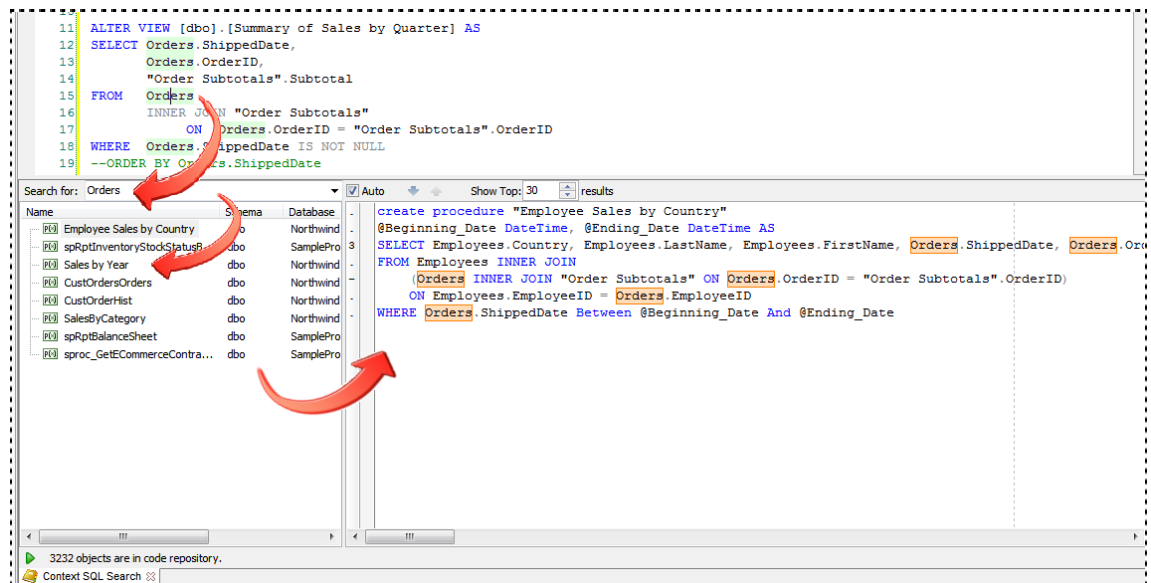
Note: Automatic searches are performed in the background so you can continue working with the editor. The search results are ranked by relevancy and displayed automatically.

Manual search

In this mode you need to enter search terms into the **Search for** input box and press the Enter key to initiate the search.
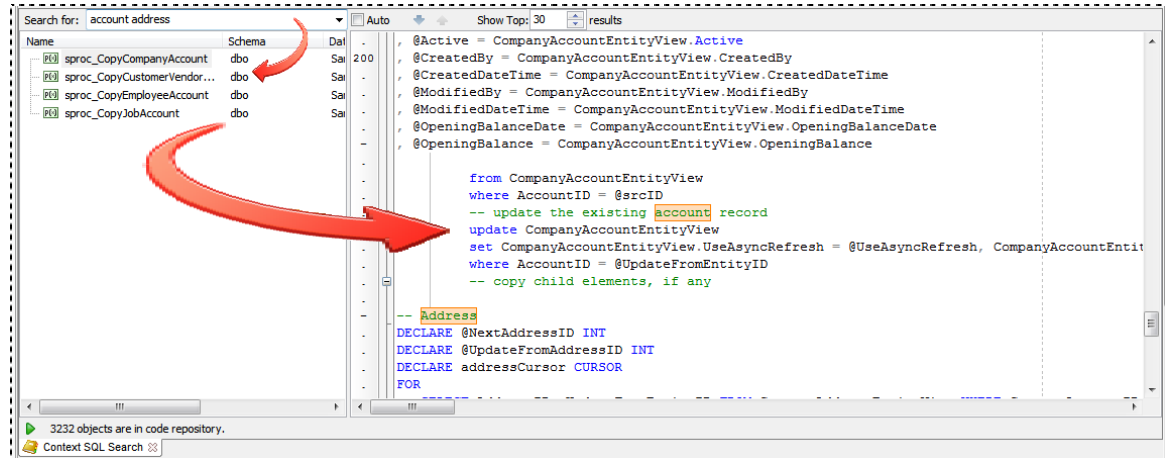
Searching for multiple loosely connected words

If you enter multiple words to search for, the context search uses so called NEAR search to find SQL code containing all specified words within a certain proximity of each other. The behavior of the NEAR search can be customized in SQL Assistants options. See the Tuning Code Context Behavior topic in this chapter for more details.

Wildcard searches

In the manual mode you can also do wildcard searching with the "*" character. If you do a wildcard search for *cat**, it will search for all text that starts with *cat*, so it will find *category*, *catalog*, etc.

Phrase searches

To search for SQL code containing a specific phrase with a set of entered words in a specified order with no other intervening words, enclosing the search phrase in double quotes ("). For example *"Sales Order"*

Using AND, OR, NOT operators

There are currently three supported operations that can be used with the search criteria:

The AND operator determines the intersection of two sets of search terms. Try using a combination of a phrase search with an additional loosely connected word joined by AND operation, for example, *"Sales Order"AND Cancel*. This search finds all procedural objects in your code repository containing "Sales Order" phrase and also containing word Cancel or its variations like Cancelation.

The OR operator calculates the union of two sets of search terms. For example, search for *Shipper OR Sender,* finds all objects in your code repository containing either the word Shipper or the word Sender or both.

The NOT operator (using the unary "-" operator) may be used to compute the relative complement of one set of search terms with respect to another. For example, search for *Shipping -Order* finds all objects in your code repository containing the word Shipping and not containing the word Order.

Important Note: For Context-based SQL Search feature to be functional, the SQL Assistant's code repository must be populated with your database code. See the FTS Code Repository topic for details on how

to populate and manage SQL Assistant's code repository.

Tuning Code Context Behavior

Code Context working and behavior can be customized in the **Common** section on the **Targets** tab in SQL Assistant's Options dialog.

To customize the settings:

1. Double-click the SQL Assistant icon in the Windows System Tray. The **SQL Assistant - Options** dialog will appear.

2. Click the **Common** section on the left side of the Options dialog screen.
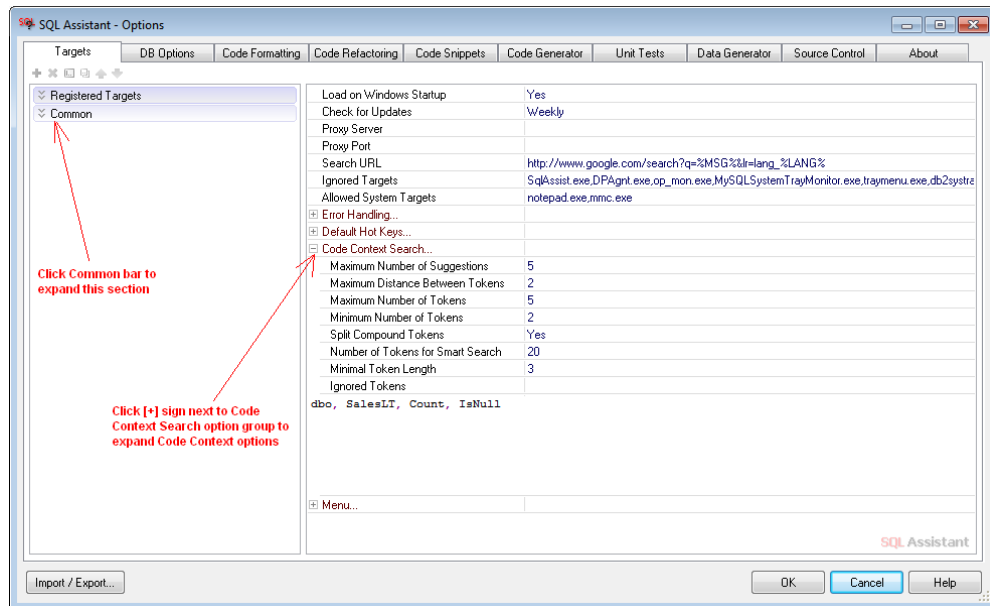
3. Expand the **Code Context Search…** group of options

4. Change the options as required. To tune the behavior try a step-by-step process changing 1 or 2 options at a time, checking results and then if needed changing same or other options again until you obtain the satisfactory results.

The following options are supported by the Code Context:

Maximum Number of Suggestions

The maximum number of suggest items that may appear in the Light Bulb popup. do not confuse this with the number of items that can be listed in the Code Context Search window, which has its own options and controls.

Maximum Distance Between Tokens

The maximum number of words between search tokens, in other words, the proximity of search tokens within the text. For example, if the value is 2, and in the Code Context Search window you enter "industry list" search terms, the context search will only search for objects in the code repository containing text referencing both "industry" and "list" and these two words are not separated by more than two other words between them.

Maximum Number of Tokens

The maximum number of words the Code Context can pick up from the code in the editor for searching similar code in the code repository.. For example, if the value is 4, up to four words are used as a criteria for searching for similar code. This option is used in conjunction with the **Minimum Number of Tokens** to improve the quality of context search results. The optimal values vary for different database systems.

Minimum Number of Tokens

The minimum number of words the Code Context can pick up from the code in the editor for searching similar code in the code repository.. For example, if the value is 2, at least two words are used as a criteria for searching for similar code. This option is used in conjunction with the **Maximum Number of Tokens** to improve the quality of context search results. The optimal values vary for different database systems.

Split Compound Tokens

If set to Yes, the Code Context automatically breaks words like IndustryList, Product_Code into separate words Industry List and Product Code when searching for similar code in the code repository.

Number of Tokens for Smart Search

The number of words before the edit caret that the Code Context can evaluate for their usability in context searches and from which it can pick words for searching similar code in the code repository. If any matches are found, the Light Bulb indicator is displayed.

Minimal Token Length

Te minimal number of characters in a word for that word to be used in context searches. Note that this option does not affect the context search behavior for non essential words like "a", "the", "or" and others which are always ignored by the code repository text indexes.

Ignored Tokens

The words you want always ignored in code context searches. Enter the list of words in the edit box below the **Ignored Tokens** line**.**

CHAPTER 38, Entity Relationships, Graphical Dependencies, and Data Flows

Overview

SQL Assistant provides integrated database documentation and modeling tools with a multi-faceted view of the data organization, data flow, and dependencies, which sets it apart from many traditional database diagramming and modeling tools primarily focused on entity relationships. It enables you to develop new models from scratch or reverse-engineer your existing database schemas and import them into the model. You can start with a reverse-engineered version and then continue developing it adding new objects and making changes in the imported objects as required. All changes are saved in a composite database model file saved to the file system on your workstation. At any time in the modeling process you can generate a database-change script and execute it in one or more environments to apply your changes to the database schemas.

Each database model contains three facets:

- **ER diagram** - An entity relationship (ER) diagram shows the relationships of entity sets stored in a database schema. ER diagrams illustrate the logical and physical structure of database schemas:
- **Code dependencies diagram** – A code dependencies diagram visualizes dependencies between various database schema objects.
- **Data flow diagram** – A data flow diagram documents data dependencies and visualizes data movement in your applications. It is an open ended diagramming tool enabling you diagrams documenting your application working.

You can open and work concurrently on multiple database models. Each model is opened in separate **Database Model** dialog. The following topics describe in detail how to use the **Database Model** dialog and built-in diagramming tools.

ER Diagrams
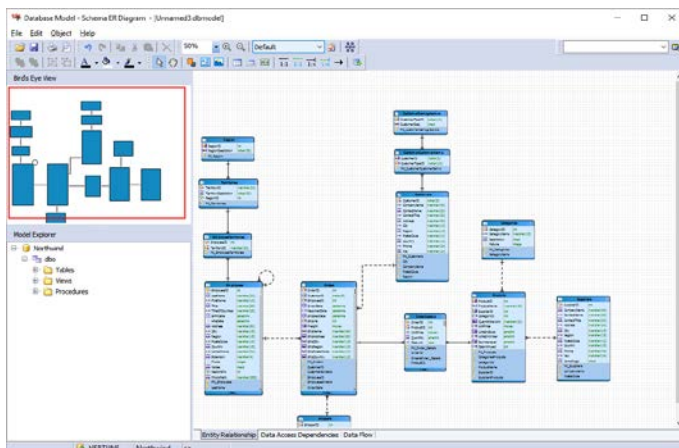
Tables are the basic elements of an ER diagram. They can be imported from an existing database schema or be created in the Database Model project and later created in the database during execution of the **Database Change Script**.

All tables in the model are shown in the **Tables** node of the **Model Explorer**, they may or may not appear in the

diagrams.

You can use simple drag-and-drop method to bring tables from the **Tables** node to the ER diagram.

To remove a table or other object from ER diagram, click that object and then press the Delete key on the keyboard.

Double-clicking on a table in the Tables node or in the ER diagram opens the Edit dialog, where you can view and update table properties including columns, constraints, and indexes.

An **identifying relationship** exists when the primary key of the parent entity is included in the primary key of the child entity. In other words, parent entity's key is part of the child entity's identification. A good example of such relation is "Orders" and "Order Details."

A **non-identifying relationship** exists when the primary key of the parent entity is included in the child entity but not as part of the child entity's primary key. In other words, parent entity's key is not used for the child entity's identification. For example, data in "Orders" and "Shipping Address" are related, but "Shipping Address" is not part of the "Orders" entity identification

Code Dependencies Diagrams

The Code Dependencies diagrams help with managing database schema and code dependencies. The Code Dependencies diagrams visualize physical relations between various objects in the database.
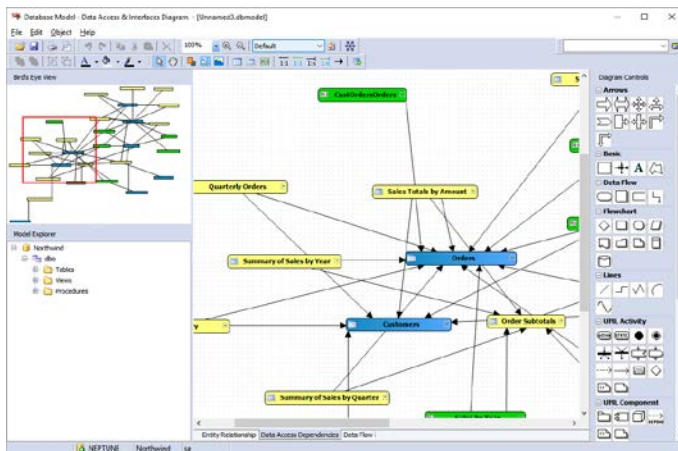
Data Flow Diagrams

The Data Flow Diagrams are created by users for the purpose of documenting internal and external data flows and their process dependencies. You can use UML graphical elements, connector lines, text labels, images, and other diagramming components to document how the data moves through your database.
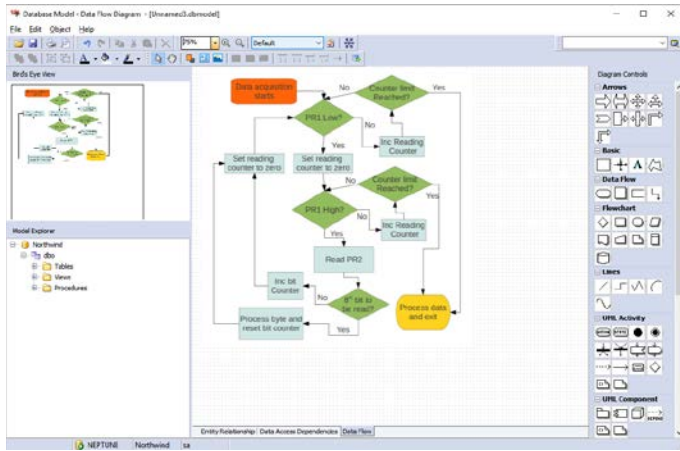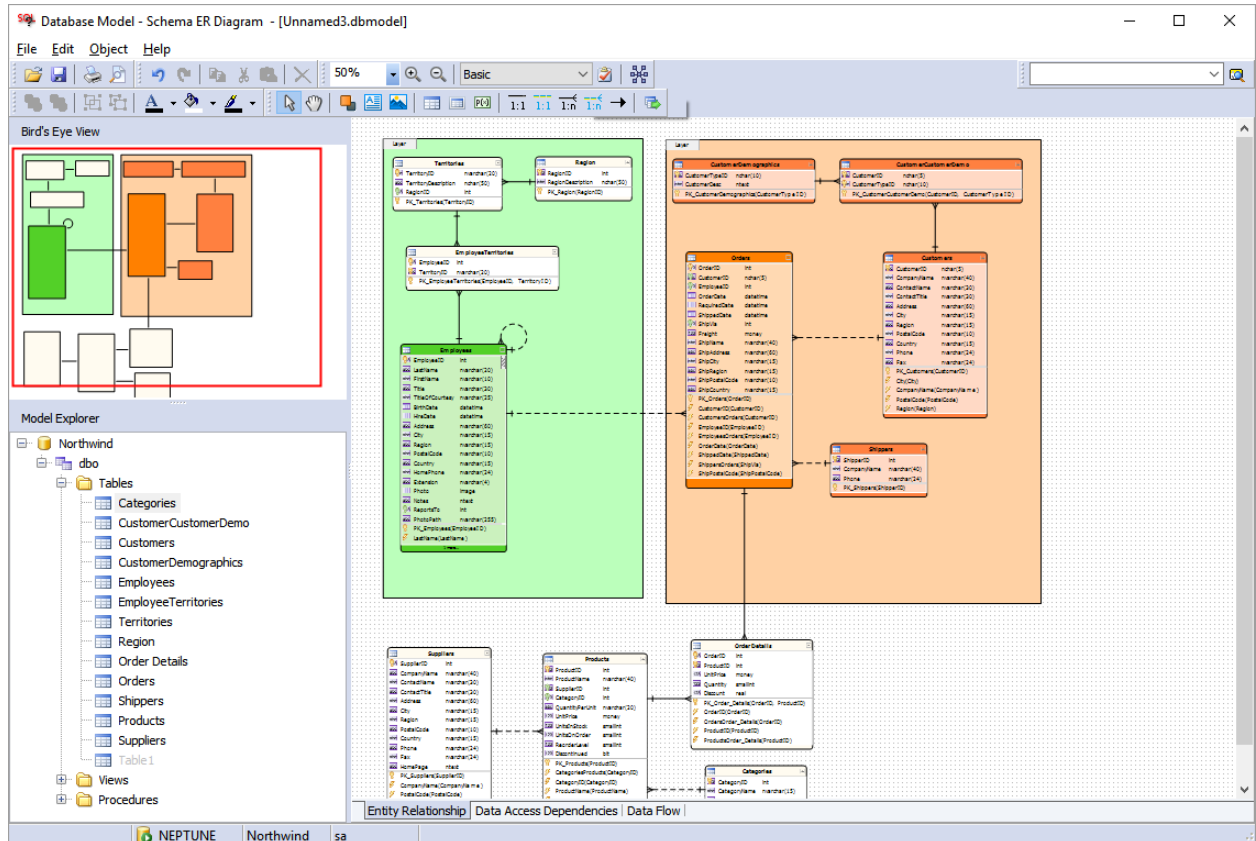
Consider using it together with the Data Lineage tools that can help yopu discover and visual your application data lineage. See CHAPTER 45, Analyzing Application Data Lineage for additional information.

The Database Model Workspace

Main Components and Controls

The **Database Model** workspace has five main parts described below.

The top level menu and toolbars provide quick access to all main functions supported by the database modeling.

The **Bird's Eye View** window displays a miniature view of the active diagram. The window appears adjacent to the left side of the Database Model workspace. The sole purpose of the **Bird's Eye View** is to ease the navigation while working with large diagrams. In **Bird's Eye View,** a red rectangle is used to identify the area currently displayed in the active diagram window. Note the red rectangle is not shown if the diagram fully fits in the workspace. . Dragging the red rectangle causes the diagram window to scroll to display the section covered by the rectangle.

The **Model Explorer** window appears adjacent to the left side of the Database Model workspace. . The content of that window references all schema objects available in the model. Double-clicking a schema object in the **Model Explorer** opens object-type specific properties editor dialog. You can also use drag-and-drop to add objects to the active diagram from the **Model Explorer.**

To delete an object from a model, delete it from the Model Explorer. Note that deleting an object from a diagram is not the same as deleting it from a model. An object remains in the model until it is deleted from the Model Explorer.

The **Workspace** is the area on the center hosting three tabs for the database model diagrams. See the

following topic for detailed instructions on working with the diagrams.

When **Data Access Dependencies** or **Data Flow** tabs get activated, an additional **Toolbox** window with "Diagram Controls" title is displayed to the right of the **Workspace**. The **Toolbox** window provides additional controls and stock icons that you can use in the diagrams.

Working with Diagrams

Single Object and Multi-Object Operations

For all operations concerning schema object database properties you work with a single object at a time. Typically you double-click it or right-click it and then select **Edit** from the context menu to open the Edit dialog where you can modify objects' properties.

For operations concerning object position and appearance, you can operation on both single and multiple objects.

Objects added to a layer can be also operated in groups. For example, you can move the entire group by moving its layer.

Selecting Objects

To select a single object, click the object you want to select. Most objects are agnostic to where you click them, but some like layers have a special area in the left-top corner used as a hot-spot for selection. This special area is indicated by a different background color.

To add more objects to the selection: Hold down Shift key and click additional objects to add them to the selection.

To select a group of objects in a specific area: Use mouse lasso effect - while holding down the left mouse button drag a rectangle around the objects you want to select.

To select all objects in a diagram: Press Ctrl+A key to select all objects.

To select a connector line, click the line you want to select.

Grouping and Ungrouping Objects

Grouping objects is typically a temporary operation which you can use to bundle different objects together and then manipulate them in groups while reorganizing the diagrams.

Another use of grouping is keeping related objects together as a single group. However Layers provide a richer set of options that can be used for object grouping.

To group objects, select multiple objects using any appropriate method, then click the **Group** icon on the

toolbar.

To ungroup previously grouped objects, select the group, and then click the ▦ **Ungroup** icon on the toolbar.
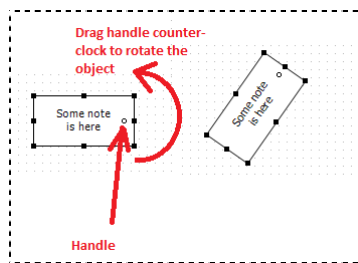
Moving, Rotating, and Resizing Objects

Moving and resizing objects is the same as moving and resizing windows. If multiple objects are selected, they are moved as a group. If objects are grouped using the Group function, all objects in the group are moved together. If objects are placed onto a layer, the entire layer can be moved with all the contained objects.

On contrary, the rotate and resize operations are applied always to individual objects. However if the selected objects have been previous grouped together using the group tool, then resize operations effect all objects in the group.

📌 **Note:** Selected objects have three dots displayed on each side of the object. Drag the dots to change object's dimensions.

Objects of certain type, for example text notes and images can be rotated, A special handle is shown in the object's area that you can drag sideways to rate the object. The following example demonstrates this technique

📌 **Note:** The rotation handles are visible only when objects are selected.

Adjusting Connector Lines

In certain cases you may want to adjust positions and shape of connector lines. Select a line that you need to move or change shape. If selected line has two legs, two resizing controls are shown close to line end-points. If the selected line also has a middle part, then a mid-point resizing control is also show as demonstrated on the following image. Drag the resizing controls to adjust positions of the line parts.
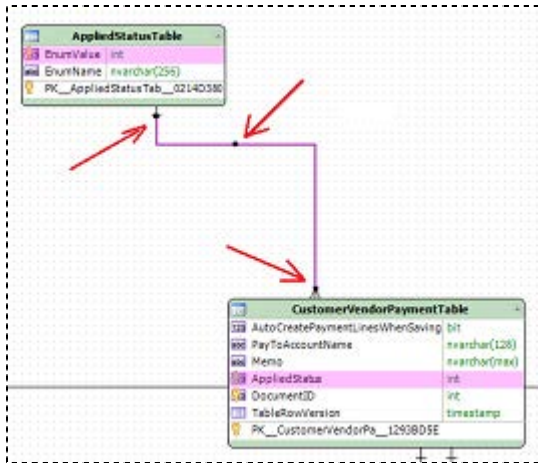
Note: The resizing controls are visible only when lines are selected.

Appearance

SQL Assistant supports themed interface for database model diagrams allowing you to not only set an overarching theme for the schema objects controlling which elements and object attributes to show in the diagrams , but also individual styles based on color, effects and font components. Different appearance settings can be used for different types of schema objects. For simplicity all types of schema objects are grouped in three categories: Tables, Views, and Procedures. Note that Procedures category includes all types of procedural objects support by SQL Assistant, including but not limited to stored procedures, functions, macros, and triggers.

Several predefined themes are available out of the box. You can also create your own custom themes as described in the following topic. In case all themes are deleted or themes configuration files cannot be located a predefined **Basic** theme is used. The Basic theme cannot be customized, this theme is static and is used only if everything else fails.

Themes can be applied to the model on the fly using the **Themes** drop-down list.

In addition to themes you can customize visual properties of individual objects using different fonts and colors. Object scope customizations override visual attributes of the selected theme. Use right-click context menu to change visual properties of specific objects.

To change global diagram options including display of gridlines, object position snapping, control of rulers, and other global properties, use the **Diagram Options** dialog accessible via top level **Object -> Diagram Options** menu and icon on the toolbar.

Customizing and Creating Themes

The **Theme Manager** dialog allows you to customize diagramming themes. To open the **Theme Manager** click top-level **Object -> Theme Manager** menu.
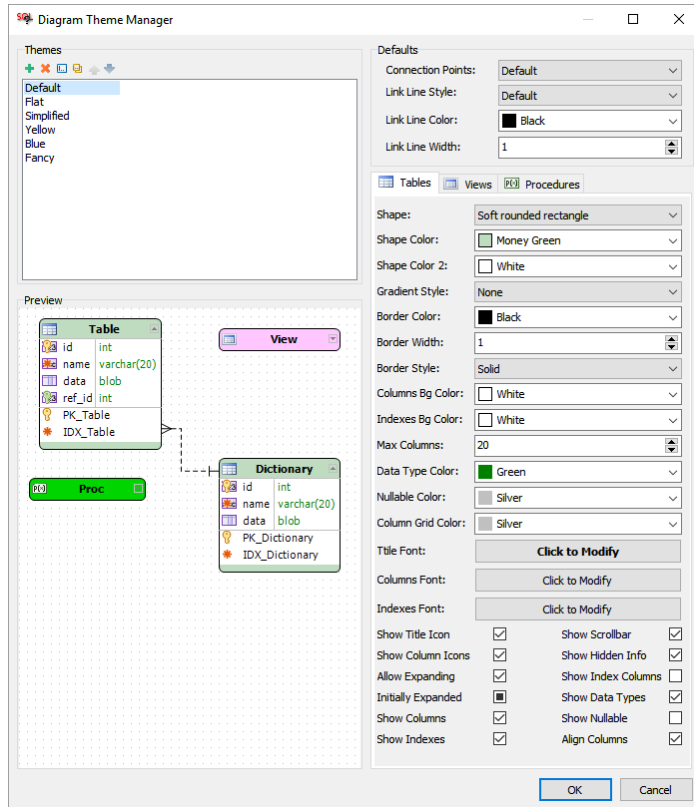
The **Themes** list shows all available themes. You can use toolbar buttons above the list to add, delete and copy themes.

Following is a description of each toolbar button:

	Moves the selected theme up one line
	Moves the selected theme down one line down.
	Renames the selected theme
	Inserts a new theme with basic settings.
	Deletes the selected theme.

The **Preview** box shows how objects will appear in the model if selected theme is applied to the model.

The best method to learn the many available options in the Theme Manager is to try changing them and seeing changes applied to the **Preview** box. When you are satisfied with the look of the modified them, click the OK button to close the **Theme Manager** dialog and save your changes.

Navigation

You can select shapes in the diagram with the mouse as described in the <u>Selecting Objects</u> topic , but sometimes it is more convenient to use the keyboard. Use the Arrow Right, Arrow Left, Arrow Top and Arrow Down keys to move current selection to the adjacent objects.

Pan Mode

Use the **Hand** [icon] icon on the toolbar to shift to Pan mode. Mouse down and dragging will pan the diagram

Zoom

You can quickly zoom in and out by changing the zoom ratio in the **Zoom** drop-down available in the toolbar area. You can choose one of the predefined values or type in a custom value. Or you can use Ctrl + mouse wheel to zoom in and out of the diagram.

Search

To quickly locate the required object by its name you can use the **Search** box available in the toolbar area. Click the **Search** box, type full or partial name of what you want to find and press the Enter key.

Bird's Eye View

In **Bird's Eye View,** a red rectangle is used to identify the area currently displayed in the active diagram window. Note the red rectangle is not shown if the diagram fully fits in the workspace. . Dragging the red rectangle causes the diagram window to scroll to display the section covered by the rectangle.

Auto-layout

SQL Assistant supports automatic layout function that can be used to rearrange entire diagrams. After the diagram is rearranged the same function can be applied again to make small adjustments to object positions. You would typically use the auto-layout after importing a set of new schema objects into the model.

SQL Assistant auto-layout function uses force-directed graph drawing algorithm for drawing graphs in an aesthetically pleasing way. It attempts to position the objects (a.k.a. nodes of a graph) in two-dimensional space so that there are as few crossing lines (a.k.a. edges) as possible, by assigning forces among the set of edges and the set of nodes , based on their relative positions, and then using these forces either to simulate the motion of the edges and nodes or to minimize their energy.

To apply the auto-layout to the active diagram, select **Edit -> Automatic Layout** menu or click the **Auto-Layout** [icon] icon on the toolbar.

Working with Layers

Layers are used to help organize objects in the diagrams. Typically, related objects are added to the same layer; for example, you may choose to add all your tables storing data for Account Receivables to one layer while all tables related to Accounts Payables to another layer. You can then choose different colors for different layers to make the diagram look aesthetically pleasing.

To change layer caption, double-click the grey box in the left-top box of the layer, or alternatively right-click the layer area and select **Edit** command from the context menu. You can type the layer caption.

Tips:

- The Del key cannot be used when editing layer caption. This key is reserved for other uses. If you need to erase part of the text entered, use the Backspace key.

- Note caption text can have multiple lines. Use the Enter key while entering text to break it by lines. But you should keep the caption short as not to make the text overflow layers' control box. You should not enter more than 2 lines of text.
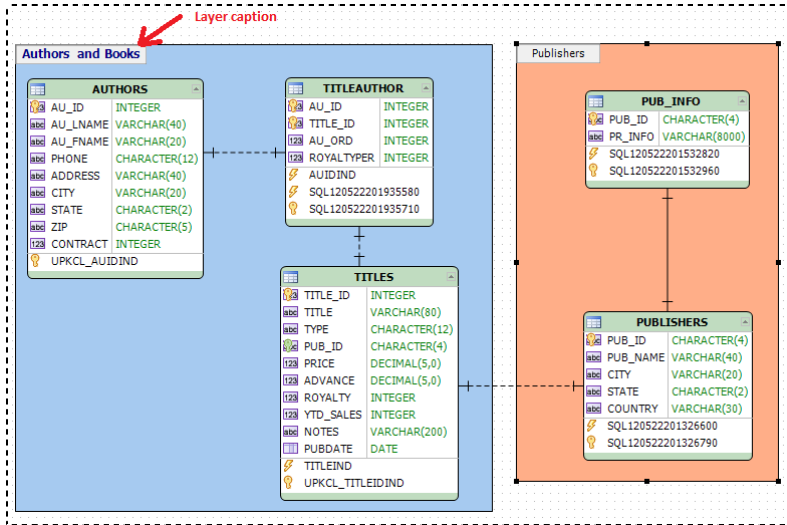
All objects placed onto the same layer are logically grouped. If you move layer to a new position in the diagram workspace, all objects in the layer will move along with it.

To add new layer to a diagram, click the **Layer** icon on the toolbar then click in the workspace where you want the new layer to appear.

To add an object to a layer, simply drag and drop it directly onto a layer. To drop multiple objects at once, select a group of objects as described in the Selecting Objects topic and then drag and drop the entire selection onto a layer.

Adding Schema Objects

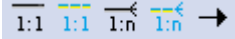To add logical schema objects to an active diagram

1. Click the appropriate object type on the toolbar

2. Click in the diagram workspace where you want to add the object. A box with the object type icon will be added to the diagram.

3. Double-click the box to open **Edit** dialog for the newly added object. Enter required object properties and then click the .Ok button to close the **Edit** dialog.

Adding Relations and Dependencies

You can add logical relations to tables on the ER diagram

1. Click the appropriate relation type on the toolbar

2. Click a child table in the diagram workspace.

3. Click a parent table in the diagram workspace. A relation line will appear between the two tables.

4. Double-click the relation line to open the **Edit** dialog. Choose specific **Source** and **Reference** columns for the relation and then click the .Ok button to close the **Edit** dialog.

Tip: If you click the same table twice, a self-referenced relation will be added to that table.

Using Notes

To add notes to a diagram, click **Add new text block** icon on the toolbar and then click in the diagram workspace where you want to add a new note.

To edit note text and change its appearance:

You may edit the note text by double-clicking it to switch to text Edit mode, or simply right click it and choose Edit command from the context menu. Using the context menu you can also change color and font, text block back-to-front relative position, and remove the note.

Tips:

• The Del key cannot be used when editing notes text. This key is reserved for other uses. If you need to erase part of the text entered, use the Backspace key.

• Note text can have multiple lines. Use the Enter key while entering note text to break text by lines.

Using Images

Images allow you to add rich content to your diagrams. They are most useful on the workflow diagrams enabling you to add custom shapes and objects to the diagram as well as create the entire diagram from a single images copied from your database application documentation.

Click **Add new image** icon on the toolbar. A Select Image File dialog will appear. Select the image that you need and click OK to close the file dialog. After that click in the diagram workspace where you want to add a new image.

Using Stock Icons

A rich collection of stock icons is provided to you for use with **Data Access Dependencies** and **Data Flow** diagrams. When a diagram of one of those types is activated, the **Toolbox** window appears adjacent to the right side of the **Database Model** workspace.

To add a stock icon from the toolbox, click the required icon, then click in the diagram workspace where you want the icon inserted.

Printing Diagrams

The printing functions and options used to create printouts of your diagrams can be found under the **File** menu. The printing options are pretty much the same as in most other Windows programs, and do not require special description.

Saving Diagrams to Images and PDF Files

You can save your diagrams to PDFs and various image file formats so that you can share them with colleagues who do not have the SQL Software installed on their computers.

Saving to image files also enables you to open and edit the diagrams in other tools, as well as use advanced printing functions provided by many graphics editors.

To save your diagram to PDF or image file, select **File -> Export...** menu. Choose the file name and format and then click the **Save** button.

Note: To view a PDF file, you must have a PDF reader installed on your computer such as the Acrobat Reader, available from Adobe Systems.

Working with Model

Creating Blank Model

To start with a blank model in the target editor select SQL Assistant menu and then select **Database Modeling...** menu command. This will open new instance of the **Database Model** workspace connected to the same database as the editor from where you opened it.

To continue, follow instructions in the Working with Diagrams topic for adding new objects to the diagrams,

If you already have the **Database Model** workspace open, you can choose **File -> New Model** menu. In this case, the new model will appear in the same dialog.

Reverse-engineering Existing Database Schemas

1. In the **Database Model** workspace click the **Import** icon on the toolbar. This will open the **Import Schema Objects Wizard** dialog
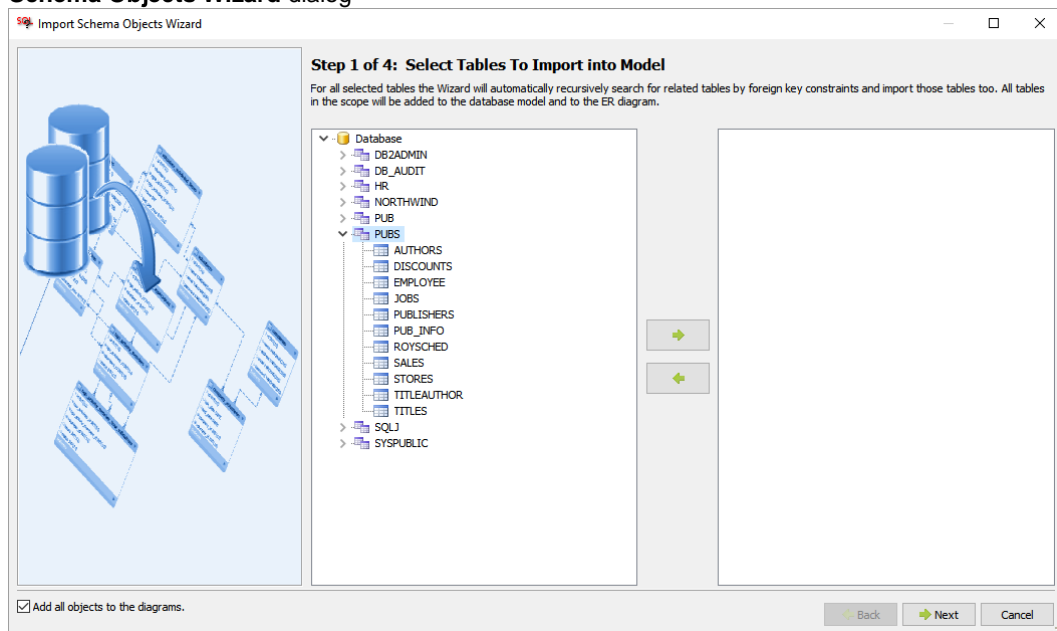
2. Follow instructions provided by the dialog to select schema objects to add to the model. This is a four steps simple and quick procedure.

   Tip: If you uncheck **Add all objects to the diagrams** option at the bottom of the dialog, the imported objects will be added to the model but not added to the model's diagrams.

Adding Existing Schema Objects to Model

This is essentially the same procedure as creating a new diagram by reverse-engineering existing databases and schemas. The same **Import** method is used. However if you select objects that are already in the model,

the import process will ignore them and they will not be repeated.

Generating and Executing Database Update Scripts

Two types of scripts can be generated:

- **Database Schema Change Script** – this type of script contains SQL commands required to synchronize database design with the current model state.

  Important Note: The **Schema Change Script** does not guarantee that the existing data in the database will be preserved during the execution of the script. If you need to ensure the data is safe, review the generated script before executing it, and if deemed necessary, update it as required.

- Database Schema Create Script – this type of script contains SQL commands required to recreate the schema objects referenced in the model.

Both types of scripts can be generated using appropriate commands in the top-level **File** menu.

Refreshing Model from Database

If your database has recent changes not reflected yet in the mode, you can refresh the model from the database. To start this operation, click top-level **File -> Refresh Model from Database** menu.

Saving and Reopening Models

Saving and opening model files is the same as saving and opening files of any other type. Use top-level **File -> Save**... and **File Open...** menus.

CHAPTER 39, SQL Code Visualizer and Database Documenter

Overview

Important Note: The Code Visualizer and Visual Database Documenter are experimental features. All experimental features are not fully supported. Use them at your own discretion. If you find a bug or have an enhancement suggestion, please report it to the support team.

SQL Code Visualizer

SQL Code Visualizer is a tool for visualizing SQL code and presenting it in a form of flow diagrams and sequence diagrams. To learn how to use this tool and what it can do, open an existing procedure in SQL Assistant's SQL Editor. In the editor select **SQL Assistant -> Code Visualizer** menu. The Code Visualizer pane will open in the bottom part of the editor.
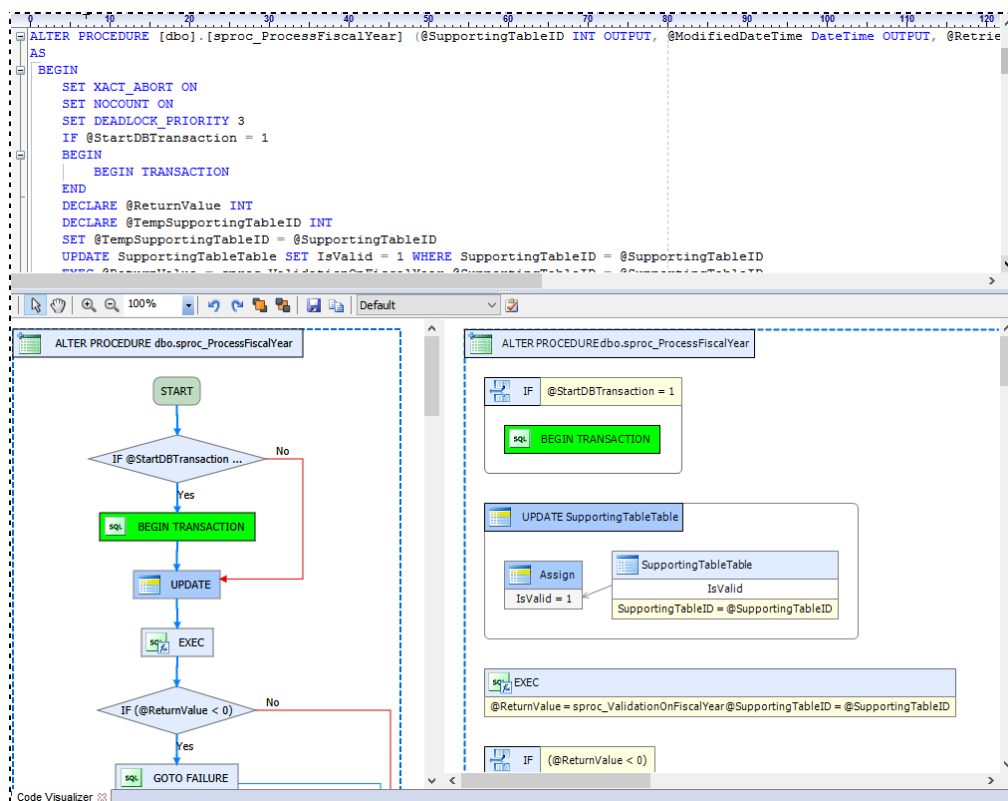
The flow diagram of the code is rendered on the left side of the pane. A special version of the sequence diagram is rendered on the right.

You can use the toolbar at the top of Code Visualizer pane to zoom in and out, to move elements of the

diagram around if you wish to adjust their positions for better visibility. You can also save the diagrams to a PDF or image file.

SQL Assistant provides flexible customization options for styling the diagrams and for choose the level of details displayed. Several predefined themes are available out of the box and can be selected from the **Themes** drop-down in the toolbar area. To customize the existing styles or add your own, click the Themes button on the Code Visualizer toolbar. This will open the **Visual Code Themes** dialog where you can customize many different options.

Expand the style names and then expand the Blocks and Lines groups of options to customize visual appearance of the diagrams as desired. To customize the content of the diagrams use the top level Transformation Options available in the right-top corner of the **Visual Code Themes** dialog.

Visual Database Documenter

The Visual Database Documenter is conceptually similar to the SQL Code Visualizer but it works at the schema and database levels. It generates visual documentation for objects of a given schema or database, inclu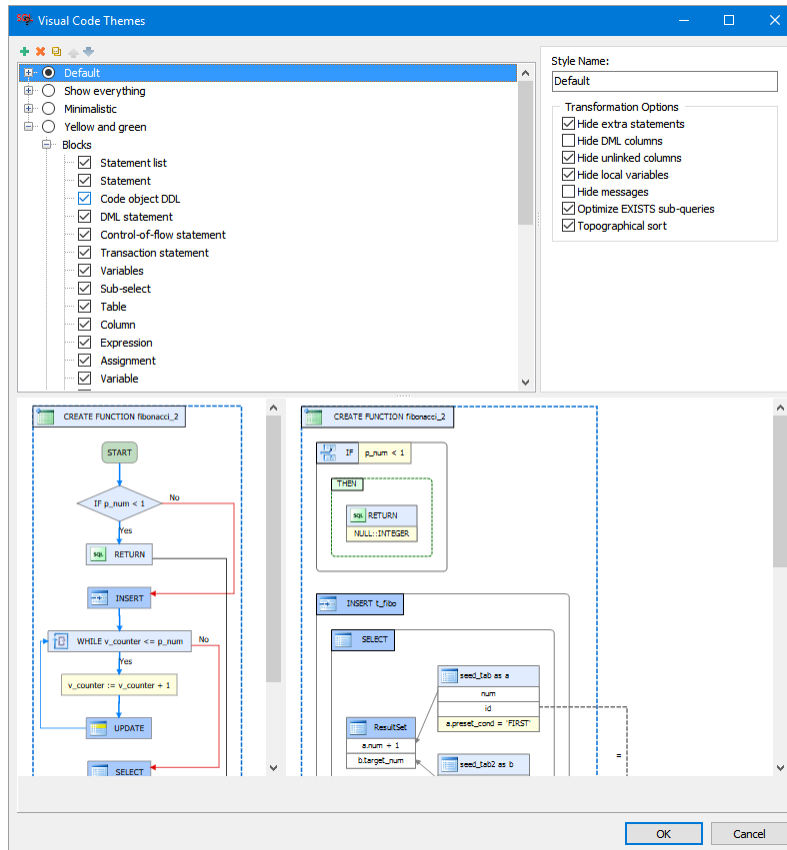ding automatic generation of database entity relationship diagrams (ERD, see CHAPTER 38, Entity Relationships, Graphical Dependencies, and Data Flows for details), code dependencies diagrams, flow and sequence diagrams for procedural schema objects, as well as definitions of tables and views.

The Visual Database Documenter can be accessed from DB Explorer's menu. Right-click database name or schema in the explorer and choose **Visual Database Documenter** from the context menu. The Visual Database Documenter dialog will appear
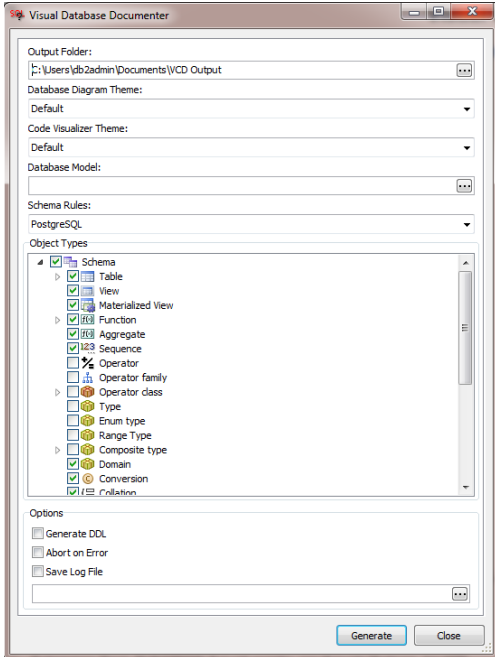
In the dialog you can choose appropriate output folder for the generated documentation files, choose themes for the database ERD diagram and for the Code Visualizer, which you want to use in the current session. Select object types that you want to be documented. In most cases you would want to choose only you objects that you need to document your database design. Too many objects of secondary type if selected may cause the output to be very big and the documentation difficult to navigate because of its big size.

Once you are done with the selection of options, click the **Generate** button and then wait for the Visual Database Documenter to complete. For a large database generating documentation files and especially code flow diagrams may take a while. Upon completion, the Visual Database Documenter will launch your default web browser and open the index page of the generated documentation files. The result should look similar to what you can see on the following screenshot.
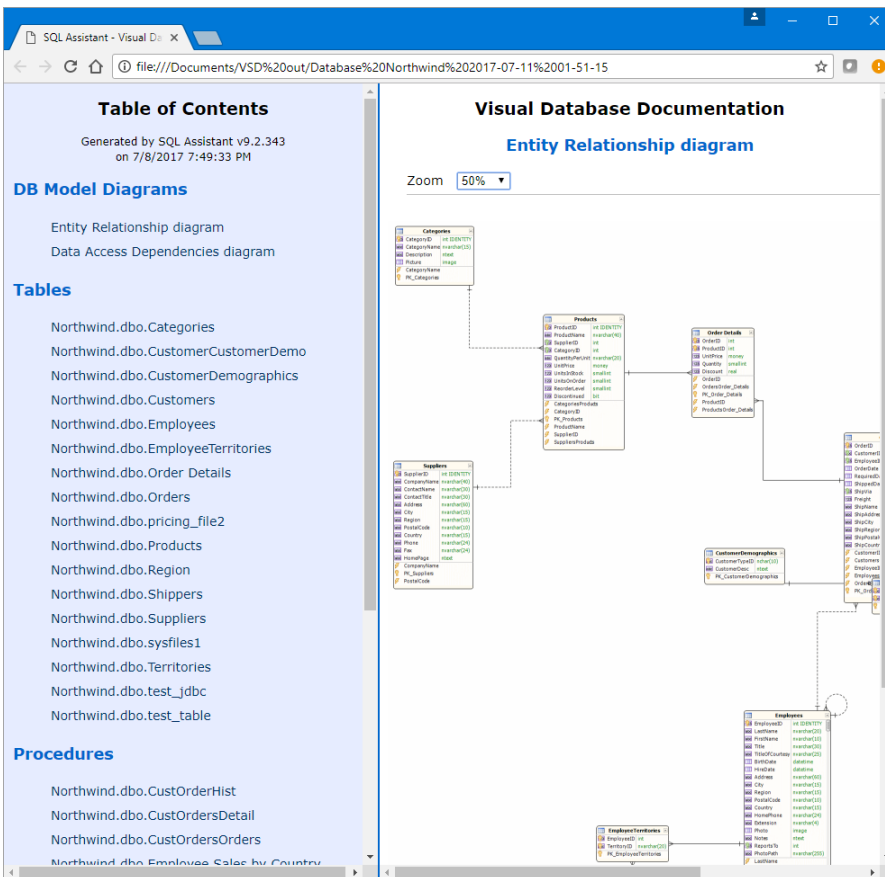
CHAPTER 40, Sanitizing and Obfuscating Database Data

Overview

When the test data generated using the Test Data Generator isn't good enough, or just takes too much time to develop and maintain the data cases required for a robust application testing, one can copy the real production data to the test database and then sanitize it to have all customer specific accounts, names and other personally identifying details removed or obfuscated so that it cannot be reognized anymore. The **Data Saniteser** utility provides you with the tools for that task.

Working with Data Sanitizer

1. To open the **Data Sanitiser** utility, in SQL Assistant's menu select Code and Data Generators menu branch, and then select the Data Sanitizer command. This will open the Data Sanitizer dialog.
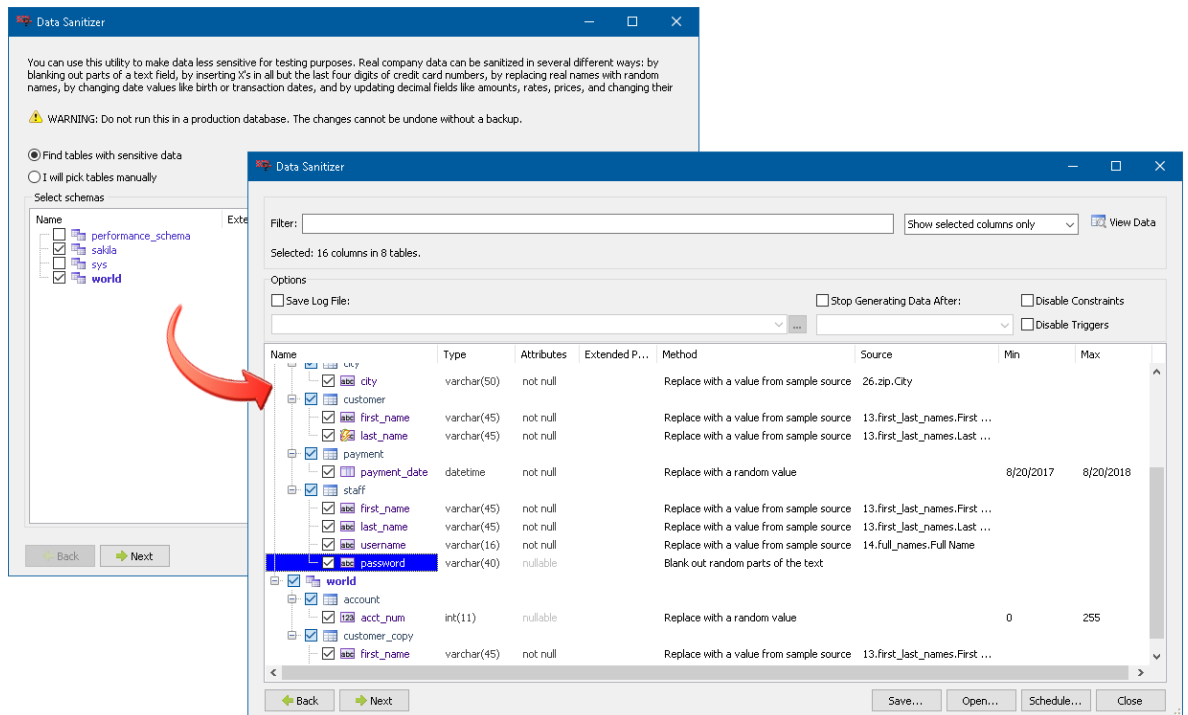
2. Choose how you want to identify tables that require data obfuscation or cleaun up. You can let the utilituy scan your database for schema definitions and to take data samples for analysis, then suggest the tables that appear to be good candidates for , or you can manually select which tables to sanitize.

3. Click the Next button to advance to the next step. Review the columns suggested for data sanitation. You can change the selection.

4. Choose the data onfuscation and/or cleaun up methods and parameters using rhe fields displayed to the right of table column names..

   You can choose from one of the following six methods for each selected column:

   - Blank out random parts of the text
   - Replace all, but last 4 characters with X's
   - Replace all, but first 3 characters with X's
   - Replace with a random name
   - Replace date part with random date
   - Replace numeric value with random number.

   For methods that support replacing original values with random values you can choose how to generate the random value, min and max range, or to choose a value from one of the predefined or custom data libraries. See Data Generation Options topic in CHAPTER 18, Generating Test Data for more information on data libraries and data ranges.

5. Click the Next button to start the data update process immediatelly, or click the Schedule button to schedule it for a later run.

Opening and Saving Projects

To open an existing data sanitizer project for modification or rerun

1. In the Data Sanitizer dialog, click the **Open** button right-bottom part of the dialog. The **Open Data Sanitizer Project** dialog will appear.

2. Select the project file you want to open, then click the **Open** button. The Data Sanitizer dialog will be populated with previously selected database connections, tables, columns, and project settings.

To save project settings in a project file:

1. In the Data Sanitizer dialog, click the **Save** button in the right-bottom corner of the dialog . The **Save Data Sanitizer Project** dialog will appear.

2. Select the project file you want to save project settings to, then click the **Save** button. It is a good idea to use "project" or similar prefix or suffix when naming project files in order to distinguish them from other XML files.

Scheduling Data Sanitization During Quiet Hours

The **Schedule** button on the Data Compare dialog enables you to schedule the process to run unattended at night or other quiet times when the database server is not very busy. This opens the Schedule dialog providing graphical interface to the data comparison command line interface described in the next topic. For information on how to manage scheduled tasks in SQL Assistant, see CHAPTER 50, Managing Scheduled Tasks

The scheduled task uses data sanitizer's command line interface to run the process at the specific date and time.

CHAPTER 41, Cloning Databases, and Schema Objects

Overview

It's often required to create a clone of an existing database or database schema for troubleshooting production issues, for testing and development of new application versions, and for other reasons. And sometimes it's the only option when no other operational environment is available for replicating or restoring a copy of an existing database or schema. SQL Assistant Professional Edition provides the **Clone It** utility supporting that function. As of SQL Assistant version 13.0, this utility supports the following database servers

- Microsoft SQL Server 2008, and later
- Azure SQL Database 12.x
- Oracle 11.1, and later
- MySQL 5.x, and later
- MariaDB 5.x, and later
- PostgreSQL 9.x, and later
- Amazon Redshift 1.x, and later
- Pivotal Greenplum 4.3.x, and later
- SQLite 2.x, 3.x

Future versions of **Clone It** will support other types of database servers as well.

 Important Note: The **Clone It** tool is not designed for creating exact binary copies of databases, and schema objects. It does not operate with database files. It does generate SQL scripts required for recreating databases, their schemas, and schema objects within a different namespace. It makes best effort to preserve the code integrity of the cloned procedural objects and views replacing within their code all source schema name references with the destination schema name references. When cloning referential integrity constrains it also updates their references to point to new tables in the cloned schemas.

If you need an exact binary copy of your database or schema cloned to a different database server.

1. Connect to the source database server.
2. Use the commands available in the DB Explorer's "Backup and Restore" menu to backup your database to a file.
3. If required to clone it to a different server, copy the resulting backup file using available operating system commands from the source server to the destination server.
4. Connect to the destination server.
5. Use the commands available in the DB Explorer's "Backup and Restore" menu to restore your database from the backup file.

If you only need to clone your database schema without data to a different database server:

1. Connect to the destination server.
2. In the DB Explorer's right-click the destination database. In the context menu choose **New…** command.

**Important Note:** The right-click context is very important. You do need to right-click the destination database, and not a schema or schema object below the destination database level.

The **New DB Object** dialog will open listing the options available for the selected context. If you right-clicked in a wrong place, then you *will not see* the Schema option listed there.

In the **New DB Object** dialog choose Schema and click the Ok button. This will open the **New Schema** dialog. Note that the dialog pictured above is just an example for a specific version of DB2 UDB. The content of the dialog varies for different database types and their versions.
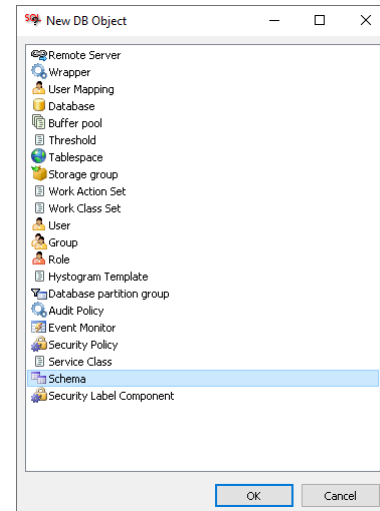
3. Follow instructions provided in the **New Schema** dialog to create a new empty schema.

4. Connect to the source database server.

5. In the DB Explorer locate the schema you want to clone. Right-click it and choose the **Compare Schema** command in the content menu. This will open the **Compare Schema dialog**, which you can use to compare your existing source schema and the new empty schema in the destination database. Run the Schema Compare and let it replicate the source schema objects to the destination schema.

   For more information about the Schema Compare tools and specific usage instructions please refer to CHAPTER 27, Schema Compare Utility

Working with Clon It

The **Clone It** interface breaks the complex cloning task into several interactive steps to make it easier for you to choose the required scope, options, and the database and schema objects to clone. You can pause or cancel the cloning process at any time. We recommend that you first read the Overview section to learn how the Clone It works conceptually, what it does and does not, and what other tools you can use if you cannot find what you need in the **Clone It** functions.

The Clone It tool is accessible from several different places, you can invoke it from the Command Selector, from the top level **Data Tools** menu in SQL Editor Professional Edition, and from the DB Explorer. The first two methods will invoke the Clone It dialog based on the current connection context for the current database and schema. While using the DB Explorer you can invoke it for any schema object whether it's in the current schema or not, as well as for other schemas and databases.

Pretty much like when copying files in the same folder, depending on what you selected to clone, you will need to chose a unique name for the new database, schema, or schema object selected.

To create a clone of a database

1. Connect to the source database server.

2. In the DB Explorer right-click the database you want to clone. If you connected to a single database server, right-click anywhere with in an empty space of the DB Explorer. In the Context menu choose the **Clone It…** command. This will open the Clone Database dialog.
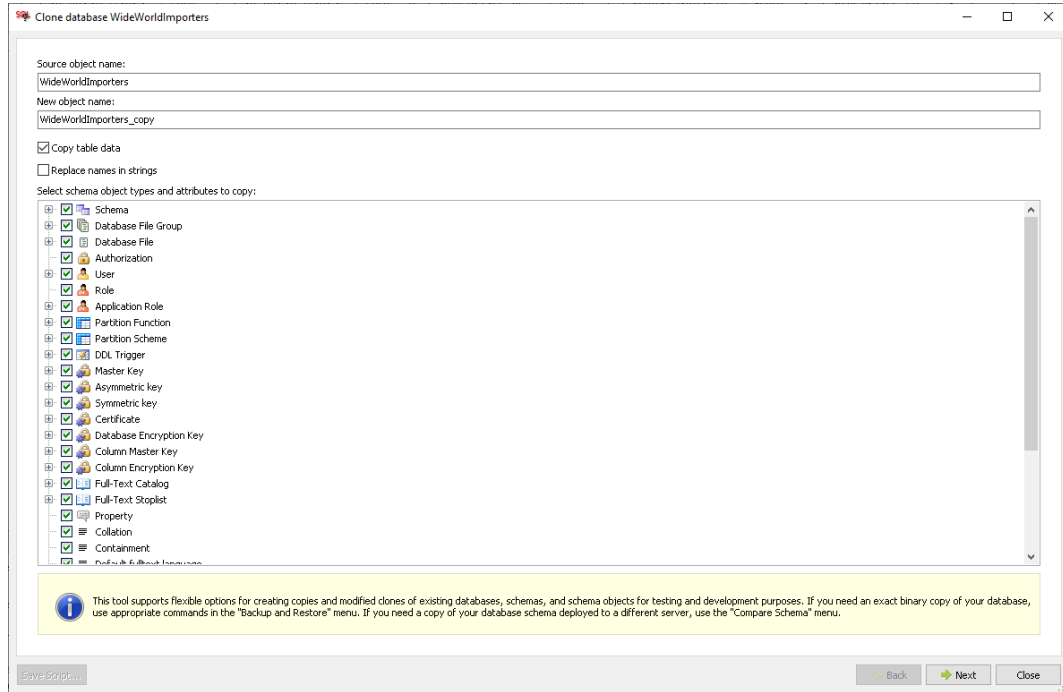
3. Choose the options and types of database objects you want to clone. It's not required to select everything. For example, you may want to ignore full text index options for tables and do not clone them, or when cloning a database, you may choose to do not clone the same certificates and encryption keys.

   Note: In this step you can also choose whether to copy table data along with the schema object definitions. If you uncheck the **Copy table data** option, only the settings and schema object definitions will be copied without the data. If you choose to copy the data, before you proceed further be sure to check you have enough free disk space on the database server to store a copy of the data.. An additional space is also required for transactional logs and rollback segments for the intermediate transaction data the database server will generate internally while executing the data copy operations. The amount of this additional space is typically 1.5 size of the largest table selected for cloning. If you choose to clone temporal tables, an additional space is required for their historical data archives.

   Important Note: The historical data for the temporal tables is not copied. Only the latest data is copied over to the cloned instance.

   Click the **Next** button to proceed to the next step

4. In step two, you can choose which specific database and schema objects to clone. You can also preview the SQL scripts generated for the selected objects.

   Note: Cloning of some types of objects requires additional inputs provided by you. For example, if you choose to clone credentials for a login based on a user name and password, you would need to enter password for to be created login. When cloning databases and their files, you may also need to enter new file path for the database files to avoid conflicts with existing database files, and so on.

   The following screenshot demonstrate the interface available for the object selection and for the additional input parameters:
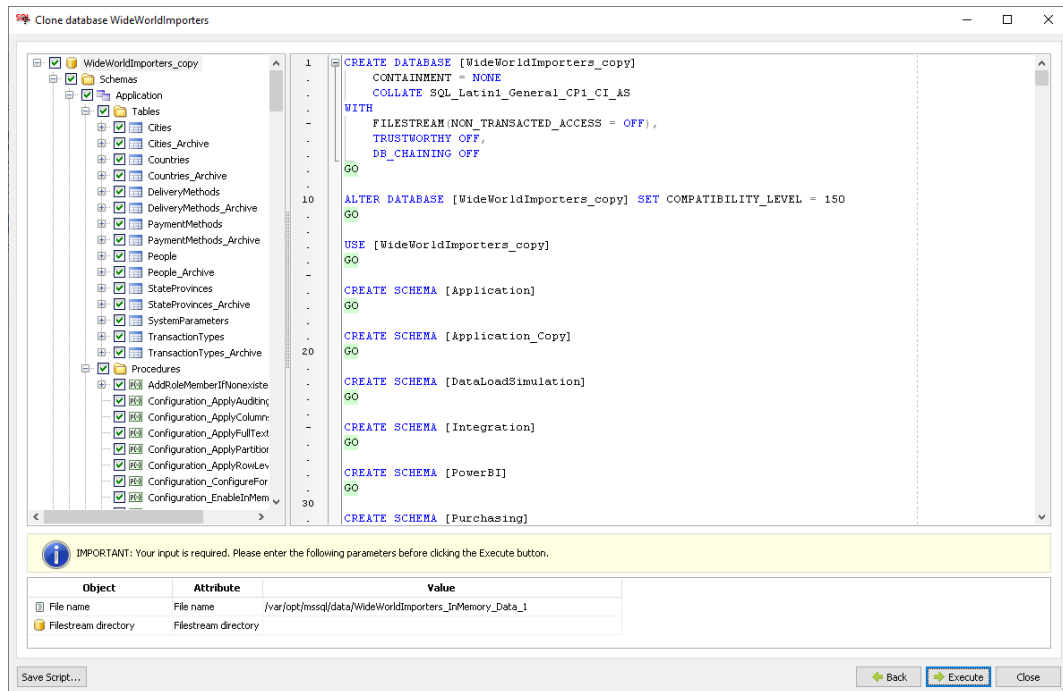
After you are done with refining the selection, click the **Execute** button to start the cloning process.

You can also click the **Save Script** button on the left side of the dialog to save the generated SQL script for a later execution.

Note: SQL Assistant uses different methods for different database types to clone their data.

To create a clone of database schema or schema object

Cloning a schema or schema object conceptually is very similar to cloning a database. Please refer to the previous section for more details. However there are few differences that you need to know before you use the Clone It utility with a schema or schema objects:

- When cloning a database, SQL Assistant requires new name for the new database instance. All schema objects within the database are containerized and so their names remain the same and do not changed.

- When cloning a schema, SQL Assistant requires new name for the new schema instance. That schema name change impacts all views, user defined functions, stored procedures, triggers, and other types of procedural objects having in their source codes references to fully qualified schema object names. The references to the source schema are automatically replaced with the new schema name. For example, if you choose to clone schema *dvd_store* to new schema named *dvd_store_copy*, and if within that schema there is a user defined function *get_customers_balance* containing the following code fragment

```
SELECT COALESCE(SUM(f.rental_rate),0) INTO v_rentfees
FROM dvd_store.film f, dvd_store.inventory i, dvd_store.rental r
WHERE f.film_id = i.film_id
   AND i.inventory_id = r.inventory_id
   AND r.rental_date <= p_effective_date
   AND r.customer_id = p_customer_id;
```

In the new schema the above code will be transformed to the following version

```
SELECT COALESCE(SUM(f.rental_rate),0) INTO v_rentfees
```

```
FROM dvd_store_copy.film f, dvd_store_copy.inventory i, dvd_store_copy.rental r
WHERE f.film_id = i.film_id
    AND i.inventory_id = r.inventory_id
    AND r.rental_date <= p_effective_date
    AND r.customer_id = p_customer_id;
```

- Similarly, if the tables in the source schema feature foreign key constraints referring to other tables in the same schema, the cloned tables will be automatically modified to have their foreign key constraints refer to the appropriate tables in the cloned schema.

- For any fully qualified schema object name in the code or referential constrain with references to schema objects outside of the source schema, their references will not be changed. For example, if there is a table named *dvd_store.customer* with *geo_region_id* column having its foreign key constraint referring to a primary key table named *geography.regions*, the cloned table *dvd_score_copy.customer* will also have its foreign key constraint referring to the same primary key table the *geography* schema.

CHAPTER 42, Tab Manager, Task Manager, and Database Session Monitor

Overview

SQL Assistant provides a set of tools for monitoring both client side and server side operations and their progress. The Tab Manager tool enables preview or all open editor tabs, their connections and active operations. The Task Manager tool enables monitoring of SQL Assistant background operations and their progress of work. It can be also used to terminate unwanted operations forcedly. The Session Monitor tool enables monitoring database activities and database users. It the database server supports progress of work reporting for long running operations, the tool can show progress of work for active queries.

In the integrated SQL Editors by default all three tools are opened in windows docked to the bottom of the editor's main window so that they can be seen in any tab. You can dock them to other sides as well as float them as standalone windows. The docking position and floating state can be changed via context menu as well as the top most position. For details working with docked and floating windows see SQL Assistant Windows and Appearance topic in CHAPTER 3, Code Assistants and SQL Intellisense.

In other target environments both tools have their windows by default opened in the floating state.

Tab Manager

You can use the **Tab Manager** to quickly switch between tabs, close them, review their context, file associations, and database connections without activating tabs.
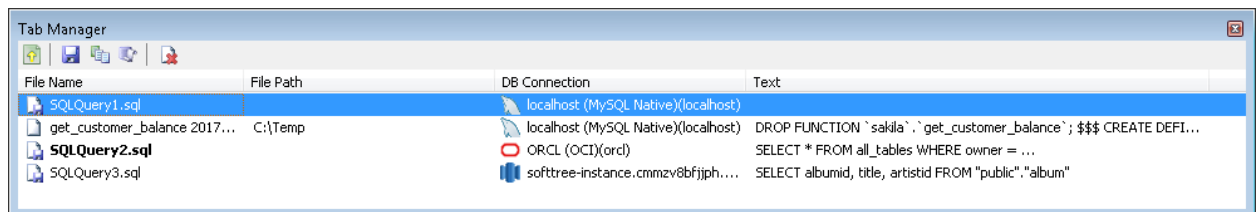
🎀 **Tip:** The **File Name** is always the same as the tab name. If the **File Path** value is empty, the tab contents has not been saved to a file, and the File Name represents what would be the file name when the tab is saved with default options. If the **File Path** is value is not empty, then the tab is associated with the actual file on the disk as specified by the **File Path** and **File Name** values.

The **Tab Manager** supports the following actions that become available when a specific file name is selected in the tab list.

    Activate tab - Activates the editor tab associated with the selected tab name.

    Save tab contents – Saves tab contents to disk. If the tab content has never been saved before, the system **Save File** dialog is displayed, and you can choose the file path and name, otherwise tab changes are saved to the associated file.

Show revisions – This command launches the **Document Manager** for the selected tab and shows previous revisions of the tab and file contents. For more information on working with document revisions, see CHAPTER 35, Document Manager and Code History Add-on.

Show SQL history – Shows history of SQL commands executed in the selected tab. For more information about this function see Using Code Execution History topic in CHAPTER 15.

Close tab – Closes the selected tab. If there are unsaved changes in the tab, you will be prompted to save them.

To open the **Tab Manager, i**n the target environment select **SQL Assistant -> Tab Manager** menu.

Tips:

- Double-clicking a tab name listed in the Tab Manager activates and sets focus to the editor tab associated with that tab, However, this method does not work for all supported target environment.

- The data-grid showing open tabs supports data sorting and other data related manipulations supported by SQL Assistant data-grids. For more details see Working with Table Data Preview Interface CHAPTER 11, Data Display and Editing.

- The right-click menu in the data-grid showing open tabs provides an additional command for quickly locating referenced files in the Windows Explorer. Right-click the required file name and then choose **Open Containing Folder in Explorer** command. Windows Explorer will open and automatically expand the path specified in the File Path value and then highlight the chosen file.

Task Manager

You can use the **Task Manager** to monitor SQL Assistant background operations such as running database queries, data import and export operations, data transfers between servers and more. For most operation types progress of work reporting is available within the **Task Manager**.

The **Task Manager** supports the following actions that become available when a specific operation is selected in the task list.

Show document - Activates the editor tab associated with the running operation. Note that not all operations have windows associated with them. For example this action is not available for data scripting, data import/export operations, and some other operation types.

Cancel database operation or **Cancel task**– In case of a single database operation such as query execution from an editor tab, it sends signal to the database to cancel the active database operation. In case of multi-server operations, for example, data transfer tasks, multi-server code execution, multi-server search and replace, and so on… it executes the Cancel function, which is typically the same as clicking the Cancel button in the dialog window associated with that operation type. In all cases the effect of the **Cancel** action is not always immediate.

Kill connection – Kills database connection associated with the active database operation. Note that in case of multi-server operations the effect of this action is not clear. It depends on the operation current state and selected options.

To open the **Task Manager, i**n the target environment select **SQL Assistant -> Execute/Schedule SQL -> Task Manager** menu.

Tips:

- Double-clicking an operation listed in the Task Manager activates and sets focus to the editor tab associated with that operation, However, this method does not work for all supported target environment.

- The data-grid showing running operations supports data sorting and other data related manipulations supported by SQL Assistant data-grids. For more details see Working with Table Data Preview Interface CHAPTER 11, Data Display and Editing.

Important Note: The percentage of the task progress shown in the **Progress** column of the **Task Manager** indicates SQL Assistant's task progress. Do not confuse this with the progress of database back-end operations. For example, if the task is "Executing queries" and it's associated with running SQL statements in an editor tab, the progress shown is measured as a percent of SQL statements already executed relative to the total number of statements pending execution.

Session Monitor

You can use the **Session Monitor** to monitor database sessions. This enables you to determine the users who are currently logged in to the database and the queries they are running.

Multiple Session Monitors can be opened for different servers simultaneously

By default the **Session Monitor** shows active database sessions and the SQL queries being executed. To display all sessions including idle sessions, select "**All sessions"** in the **Show** drop-down list in the toolbar area,

Tip: Blocked sessions are shown with pink background.

By default the Session Monitor window content refreshes every 5 minutes. To change the frequency use the **Auto-refresh** drop-down list in the toolbar area. To force a manual refresh, click the **Refresh** icon in the toolbar.

You can use the **Kill** icon to kill the selected session and to cause it to be disconnected from the database server and all its activities stopped as soon as possible.

Tip: The data-grid showing running operations supports data sorting and other data related manipulations supported by SQL Assistant data-grids. For more details see Working with Table Data Preview Interface CHAPTER 11, Data Display and Editing.

Important Note: The progress of work percentage, time remaining, and other progress related details are database type and version specific. Not all database servers support such statistics. For more details see the following topic describing **Session Monitor** capabilities and supported customizations

Session Monitor Capabilities and Customizations

The driving SQL queries for the **Session Monitor** tool are stored in SQL Assistant settings and they can be customized just like all other queries in SQL Assistant configuration. You can find them in the **Options** dialog, **DB Options** tab, **DB Queries** section.
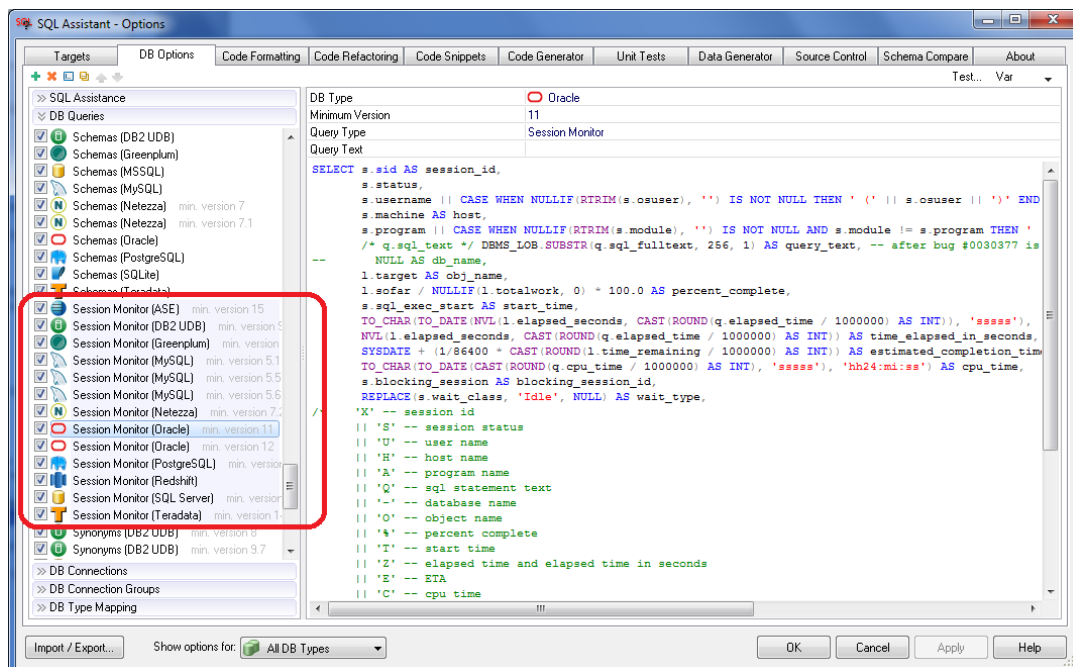
The queries are database type and version specific.

Important Note: The query text for the **Session Monitor** must begin with a commented line having **CAPABILITIES:** prefix, as in the following example

-- CAPABILITIES: XSUHAQDO%TZECBW

The control characters following the CAPABILITIES prefix describe the monitoring query capabilities, in other words, which columns are populated in the query with actionable values and should be displayed in the Session Monitor window. Here are the supported control characters:

X	session id
S	session status
U	user name
H	host name
A	program name
Q	SQL statement text
D	database name
O	object name targeted by the current operation
%	percent complete
T	start time
Z	elapsed time and elapsed time in seconds
E	ETA for the current operation
C	CPU time
B	blocking session
W	wait type

Custom columns

In addition to predefined columns, you can specify any number of custom columns that you want to appear in the **Session Monitor**. For the additional columns, you do not need to reference them in the CAPABILITIES section.

CHAPTER 43, Command Selector

Overview

The **Command Selector** (also known as Command Palette) provides quick access to most SQL Assistant commands and functions in a single searchable popup window. It relieves the need to remember where to locate many of SQL Assistant menus and shortcuts that are sometimes buried several levels deep in the regular menus The **Command Selector** window can be opened using Ctrl+Shift+Space hotkey. This default hot key can be customized in SQL Assistant's options in Targets -> Common -> Default Hot Keys section. See Customizing Functional Hot Keys topic in CHAPTER 48, Customizing SQL Assistant behavior for more details.'

Immediately after opening the Command Selector window, start typing the filer criteria to locate the function you look for. The text in the filter can be located anywhere within the command/function name. For example to locate all commands that can be used to create new objects in the database, type "new", to locate all commands that can be used to export data to Excel or from Excel, type "excel.
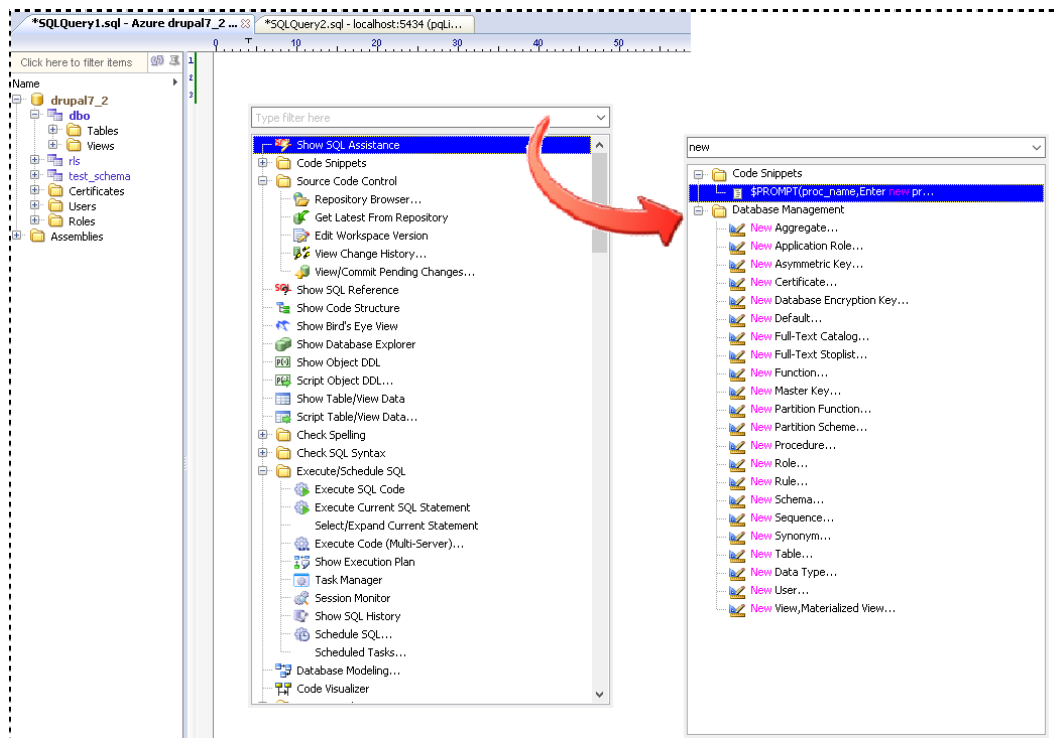
Important Note: Many commands are editor's context specific. For example, if you want to see all cascading object references to a particular database view, or all dependencies for a table column, please make sure the cursor in the editor is within the name of the view or column name when you open the **Command Selector** window. Another technique is double-clicking the subject name to highlight it in order to ensure that the right name reference is picked up by the command selector as shown on the following screenshot
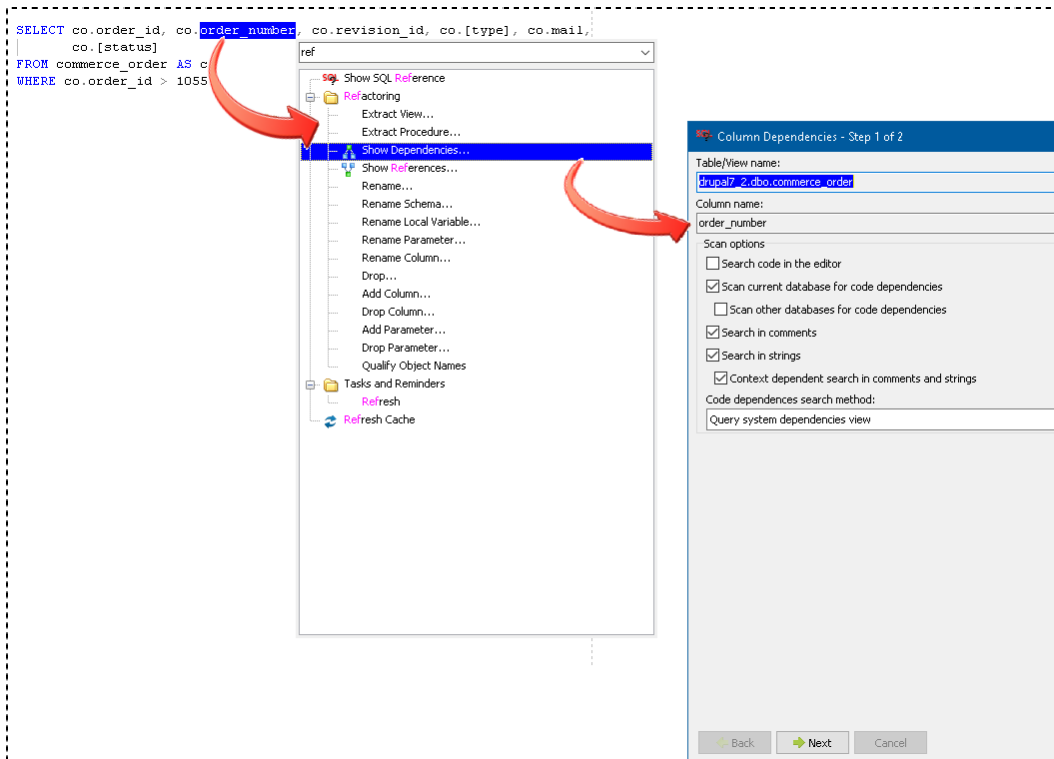
Tip: The Command Selector window is resizable. Drag the edges of the window to change its size to your liking. The new size will be remembered for the current target development environment. If you use several different development environments, repeat it as need in each environment.

Configuration

To reduce the clutter you can use the right-top menu in the Command Selector to choose groups of commands that you want to show or hide in the Command Selector as demonstrated on the screenshot below. Your selection will be applied immediately and also remembered for future use.
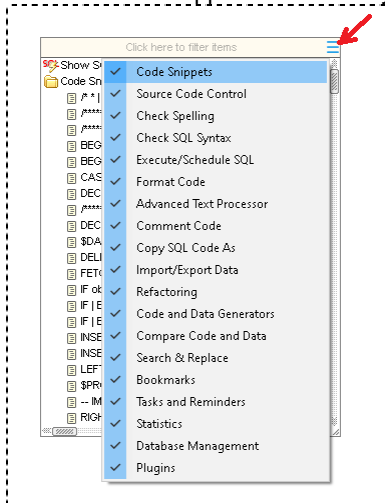
CHAPTER 44, Visual Database Management

Overview

SQL Assistant provides a comprehensive set of management dialogs for visual database schema design and administration tasks including database and schema objects, user and security management, backup and restore tasks, and much more. The management dialogs are available for a number of supported database systems and their versions. They can be accessed via context menus in the Database Explorer, using plugins, and the Command Selector.

The following screenshots for the Table Properties dialog demonstrate different methods of accessing the management dialogs. The Table Properties dialog is picked only as an example. Similar dialogs are available for most other object and management functions.

Using the context menus
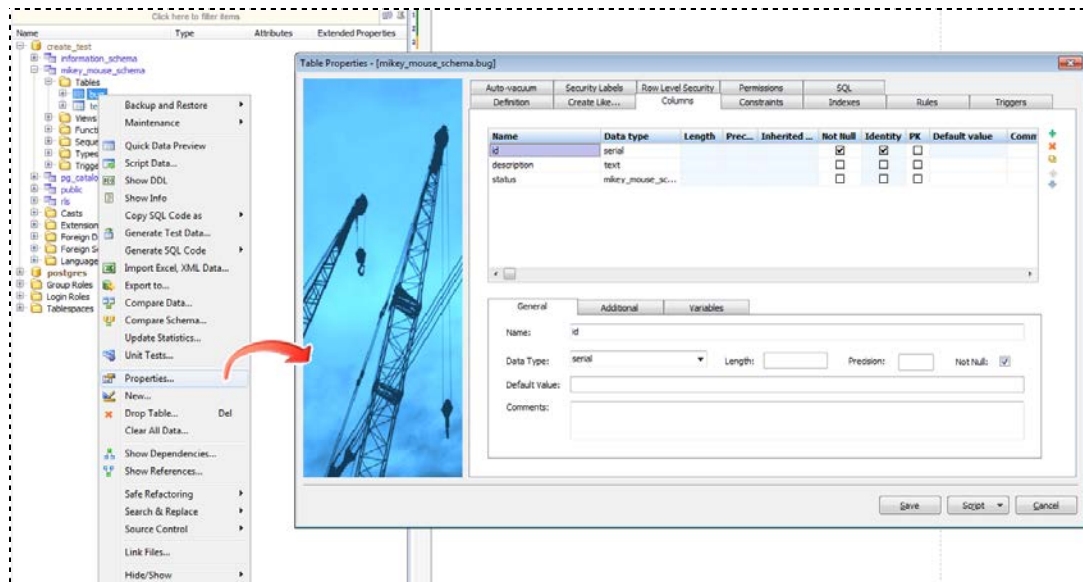
Using the Command Selector
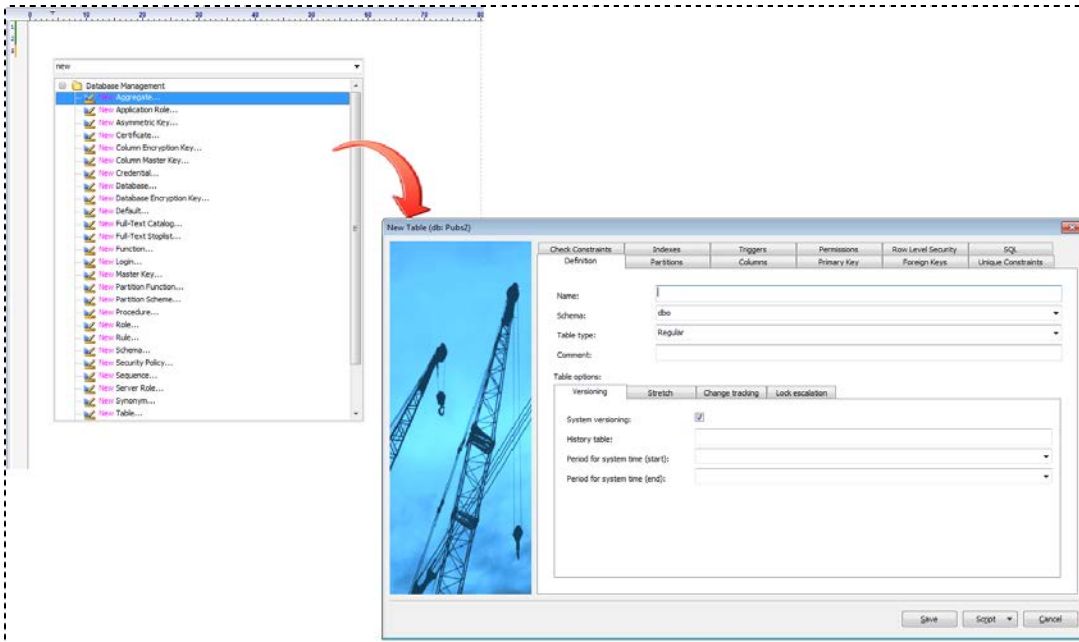
Using hot-track and mouse over hints
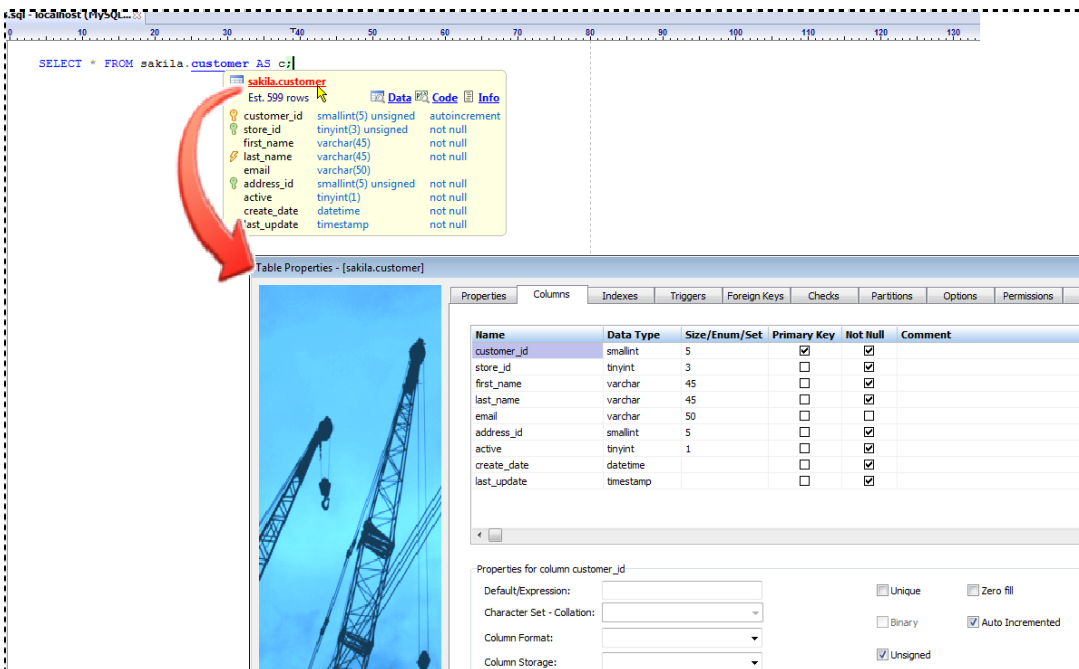
Usage

Version aware - The management dialogs are database version aware. Only the features supported by the

version of the database server you are connected to are accessible in the dialogs. The management dialogs have been designed specifically to generate database version specific SQL syntax.

Instant help - Mouse-over hints for all controls and input fields provide complete in-line help system. If you are not sure about usage of a specific field and what to know what you enter or if a value is required or can be omitted, rest the mouse pointer over the field for a couple of a seconds and a popup window will appear wit the relevant details. Additional tips and instructions are provided within the dialog's user interface as well.

Change now or later, you choose - The database changes can be applied immediately, saved to a script, or clipboard, or scheduled to be applied at a later time.

Comprehensive management interface – Each management dialog provides complete coverage of all the features of the selected objects so that you can design and manage them in a single place and apply as a single logical unit of work. For example, if you design a new table, you can visually define its attributes, columns, constraints, indexes, partitions, triggers, security and permissions, and so on… all in the same dialog. SQL Assistant will generate the required SQL script, which you can execute immediately or save for later.

Concurrent changes and multiple monitors – you can create new or modify multiple existing objects concurrently whether you are working with a single database or multiple databases of different types. If you have multiple monitors, for better efficiency and convenience you can move different dialogs to different monitor and SQL Assistant will remember their positions.

Creating New Objects

You can use the Command Selector in the editor, or the right-click menu in the Database Explorer and then the **New…** command to launch the management dialog for new objects. But there is an important difference between the two. In the Command Sector you can find many of the New [objects type] commands supported by the SQL Assistant, but not all of them. Some management dialogs are operational context dependent and require that the "parent" object /container is selected before the dialog can be used. For example, to create a new table index, you first need to choose a table for which a new index is to be created. With the Database Explorer, the situation is quite different, the **New…** menu features two different forms. In case the current operational context is non ambiguous and supports only objects of specific type, the New menu caption is object type specific as shown on the following screenshot for the Sequences folder. The command caption in this case is New Sequence. Selecting that command immediately opens the New Sequence dialog.
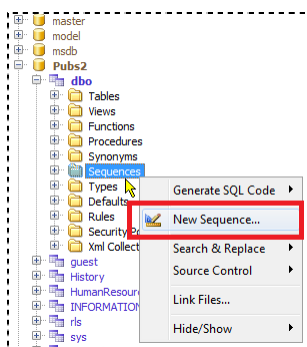
In case the selected context supports objects of different types, the menu caption is generic text **New…** as shown on the following screenshot. Clicking that menu opens the **New DB Object** dialog. Using that dialog you can choose the type of object whose management dialog you want to launch.
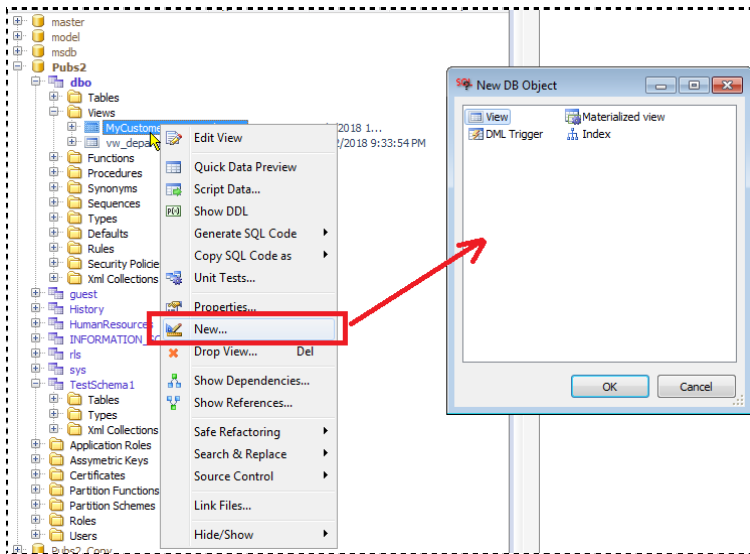
Updating Existing Objects

In most situations you would select the required object in the Database Explorer, right-click it, and then choose the **Properties…** command.

In certain cases you can also open the dialog directly from the editor. In case the type of object in the current context supports mouse over hints, rest the mouse over the object name, wait for the hint to appear, in the hint click the object name appearing in the header line to open up the graphical management dialog as demonstrated on the following screenshot.
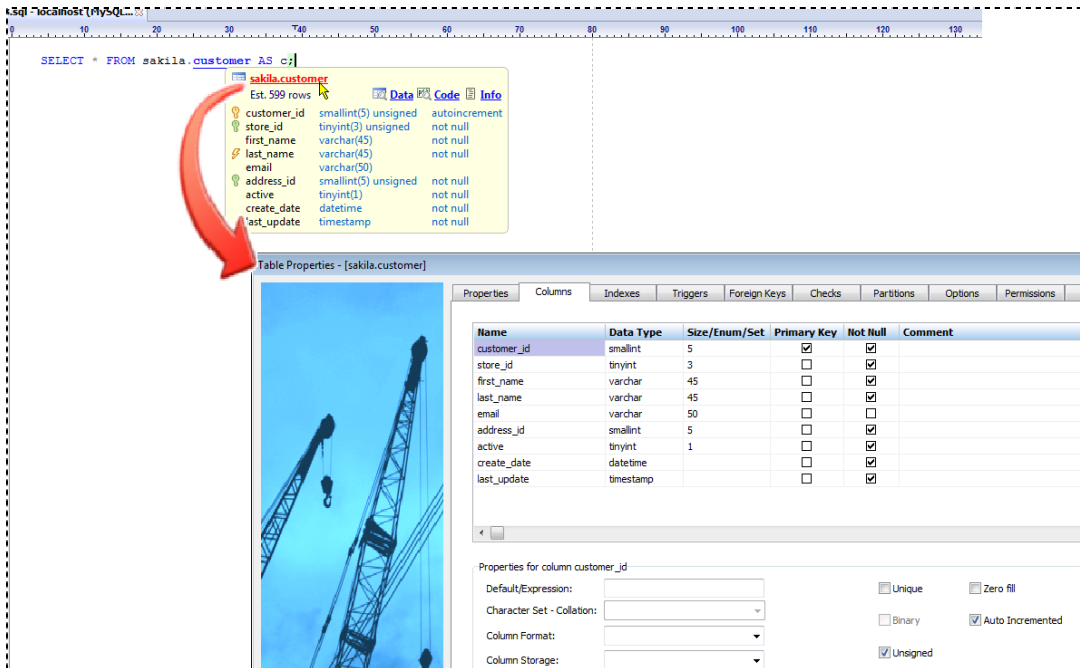
Database Management Operations

Most database and server scope management commands are available in both the Command Selector and the right-click menu in the Database Explorer. In the Command Selector you can find them within the Database Management folder, while in the Database Explorer they are available at the top of the right-click menu for Database and Server scope objects as demonstrated below.
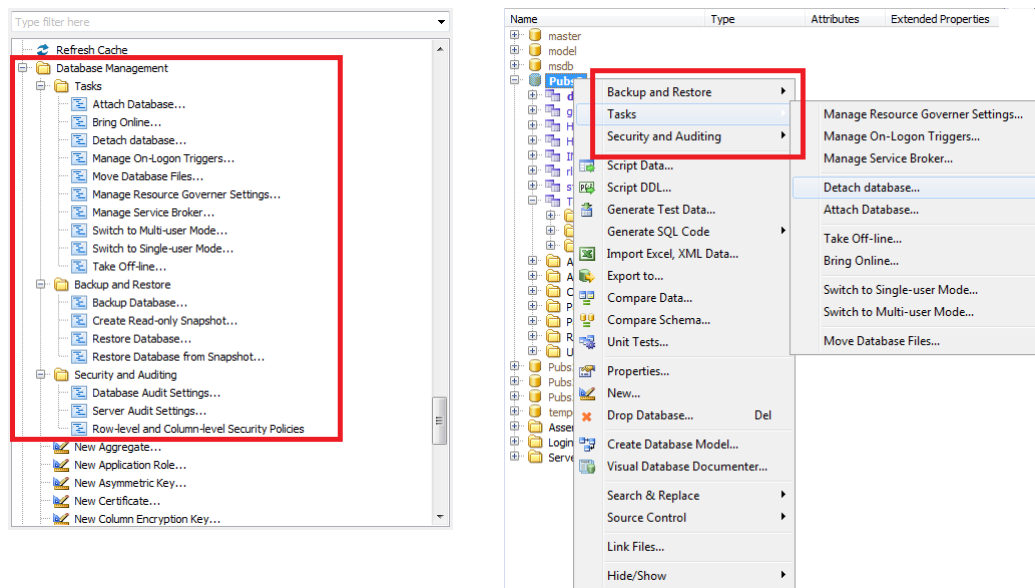
Database Change Preview and Manual Corrections

All management dialogs feature special **SQL** tab which provides a preview of the database change script that would be executed when you click the **Save** button. You can use that tab to verify that the generated script will change exactly what you want. If for whatever reason you want to modify the script, for example, to add additional comments or to adjust the script to execute it differently, you can edit the script directly in place before clicking the **Save** button. You can also click the **Script** button and in the context menu choose **Open in Editor** command. The script will open in the SQL editor and you can continue working with the script using the full featured editor interface.
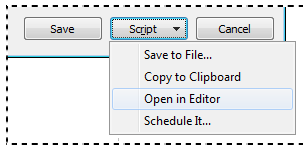
Error Handling

The database change scripts are generated with transactions in mind so that in case of an error during the change operations all important changes are automatically rolled back. However, that is not always possible. Not all database types and not in all scenarios allow transactional changes. In such scenarios partial changes can be applied. In the event of an unexpected error, and should reopen the dialog and correct or undo the previous changes causing the error. Here is an example for partial changes. A complex table change script may succeed in changing table columns and indexes, but then fail with insufficient permissions error when trying to grant table access permissions to a selected group of users. The column and index changes will remain after such error.

Smart Schema Refactoring

The management dialogs are designed to deal with complex change types. In many cases when a simple change using ALTER command is not supported, the management dialogs may generate complex scripts for rebuilding tables and other objects aiming to minimize or avoid any potential data loss. A good example of that behavior would be converting a non-partitioned table to a partitioned table. The change script is generated in such a way that it saves the data in the current table, rebuilds the table and its dependencies, and then copies the saved data into the new table.

Scheduling Database Changes

It's not always desired or possible to apply the changes immediately during regular business hours while the target objects are in use. In such situations you can use the **Schedule It** option to have the change script executed by SQL Assistant at a later time. Or you can use **Save Script** option to save it to a SQL file and then execute the script using other utility of your choice.
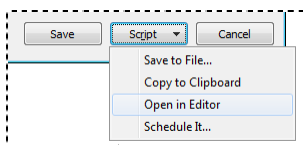
CHAPTER 45, Analyzing Application Data Lineage

Overview

The Application Data Lineage utility can perform static analysis of your files to discover and visualize data origin, and how the data moves to and from your databases, and what happens to it in flight. It can zoom in down to the column level lineage. It provides better intelligence about your data helping with the following tasks:

- Documenting application data dependencies, visualizing application services, modules, and routines that read and write to database tables, including indirect table access through views, procedures, and other forms of data access methods

- Planning application upgrades and database migrations

- Preparing for database upgrades, schema changes, and other database maintenance tasks

- Tracing data related errors back to the root cause

The Application Data Lineage utility can ingest multiple types of files, analyze their contents, and locate references to database operations. It can analyze the following types of files

- Application log files in different formats

- Low level driver and ODBC trace files

- Application source files from many types of programming systems, including Java, C#, Python, Perl, C/C++, SQL files, and a number of others

The Application Data Lineage utility parses all input files and locates text elements that might be relevant to execution of various database queries.

It uses a customize set of regular expression based patterns to find the relevant text.

It also looks to application context specific details such as service names, module names, class names, function names. It also uses a set of regular expressions to extract relevant regex groups containing the context information.

It filters and aggregates the results

It then identifies names of database objects referenced in the queries, and traces their dependencies within the database. For example, if an application executed SQL query refers to a database stored procedure, the utility parses the procedure code and locates tables and their columns that are referenced in the procedure code and how they are accessed.

From the table access patterns it extracts names of referenced columns and operation types breaking them down to write or read classes.

It finally brings all relevant data together and prepares it for quick interactive visualization.

The results of the file parsing and analysis are saved locally. That saved data is used for interactive data lineage diagrams rendered in the project workspace.

File processing is typically a one time operations. However new scanning might be required in case you choose to add more application components to your projects and rerun the analysis to trace their data dependencies.

Important Note:

- Many modern applications use advanced frameworks providing high level APIs for CRUD database operations and for managing data states. The APIs enable developers to avoid coding actual database queries and avoid referencing actual database tables and columns directly, and enable them to focus on developing application classes and their business logic. The actual database queries are generated by such frameworks in the application runtime making it difficult to locate all data dependencies using static code analysis.

- The second important issue with analyzing the application code is that it doesn't tell us which parts of the application code have been deprecated, and which are still being actually used by the application and which are not and have been deprecated.

- We highly recommend analyzing application log files for an accurate discovery of the data lineage.

Quick Start

Preparing for Data Lineage Analysis

Typical steps required for the data lineage analysis:

1. Identify all relevant application components you want to analyze. If you plan on finding all data flows and their dependencies for a given database, consider not only the front end applications, but also various back-end processes, such as APIs, data feeds, data pipelines, ETL / ELT jobs, and so on…

2. Copy application logs files for the components that need to be analyzed to a location accessible from your computer. **Make sure to copy logs from different applications and or services to different subfolders. Separation of log files is important for their correct classification**.

   Tip: Copy only the relevant logs files. Do not copy generic log files that cannot help with the data lineage analysis, and which do not contain references to database operations. For example, generic logs generated by IIS web servers and similar systems are of no use for the data lineage analysis, they log HTTP requests and responses only. You need the application logs generated API services which are engaged through HTTP requests sent to web servers and converted to API service calls. For more information see the following Application Log Files topic.

3. In case you don't have good application log files, enable low level logging at the database driver level or preferable ODBC tracing if your application is using ODBC and let them run for a while to capture and log enough records for the data lineage analysis.

   Tip: The application log files are preferable over low level driver and ODBC trace files. The application log files typically provide all important application context, while the low level trace files typically don't have application specific context logged, they are application abstract and do not provide details about application modules And function names executing the code.

4. Download the source code of the applications you want analyzed from their source code repositories to a location accessible from your computer.

Running Data Lineage Analysis

1. Launch the Application Data Lineage wizard. The Application Data Lineage wizard can be launched from several places including the right click menu for the SQL Assistant system tray icon, from a relevant Tools menu.

2. Choose the log files to analyze. If you want to skip log analysis, uncheck the **Analyze application log files** option and click the Next button. If you choose to analyze application log files, be sure to associate them with specific application context. If your database is shared by multiple applications, services, and scripts, and they generate different log files, enter their log files on multiple lines as demonstrated on the following screenshot.
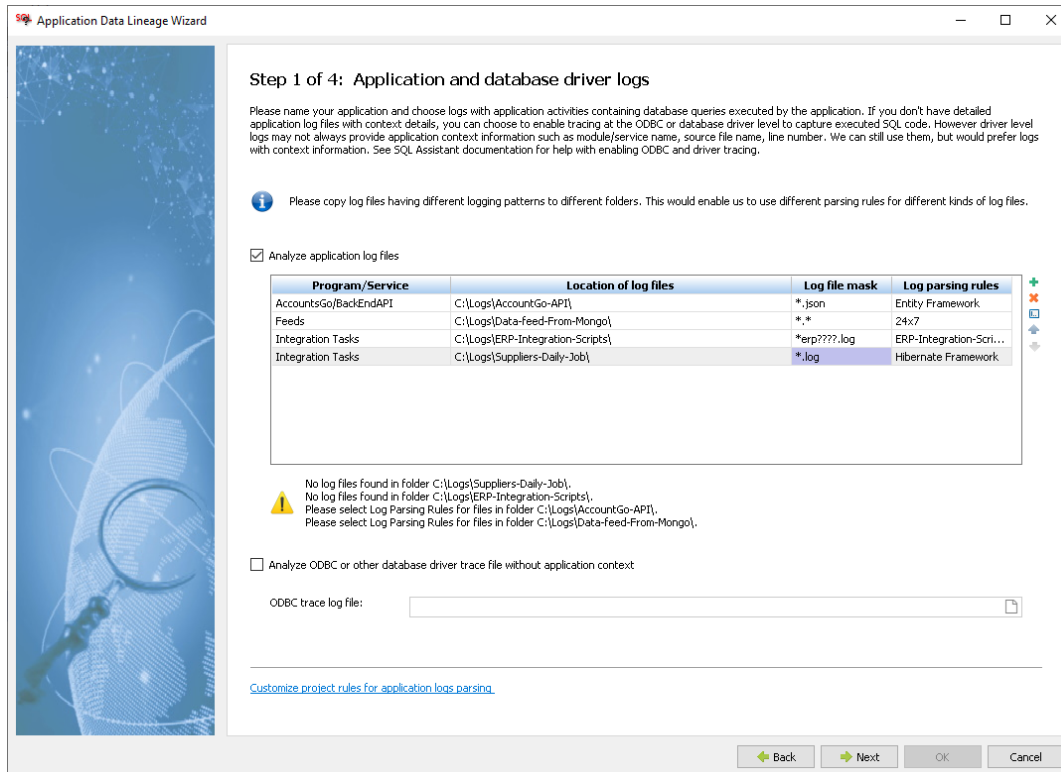
Use toolbar buttons adjacent to the right of the grid to add, delete, and reorder lines as required.

Use **Program/Service** field to associate each set of log files with their component such as a program, service, or a module. Click the field and then click **[…]** button that is going to be visible in the active field.

Tip: The association references are important for generating better data lineage results, and useful for aggregating and displaying data dependencies using user friendly service and module names.

Choose file parsing rules in the **Log parsing rules** field. It is critical for the analysis that the parsing rules match format of your files. In case of custom application log files, you would need to customize the rules, choose **Create New…** item in the Log parsing rules drop-down.

Important Note: For a successful data lineage tracing it's absolutely critical to choose correct file parsing rules. Be sure to test them before you continue to the next step. See the following Creating, Customizing, and Testing Parsing Rules topic for more information on creating, customizing, and testing parsing rules.

If you have ODBC or driver trace files that you want analyzed too, check the **Analyze ODBC or other database driver trace files** option and select the trace file you want to analyze. The parsing rules for such files are predefined, however you can still customize them in case your database drivers generate uncommon trace files.

Click the Next button to proceed to the next step.

3. Choose the application source code files to analyze. If you want to skip static source code analysis, uncheck the **Scan and analyze application code** option and click the Next button. If you choose to analyze the source code, for each application or component in scope enter root folder of their source code

files rather than common project root folder. Including folders containing lots of irrelevant files like video files, images, and similar will not alter the results, but it may greatly increase the processing time. .
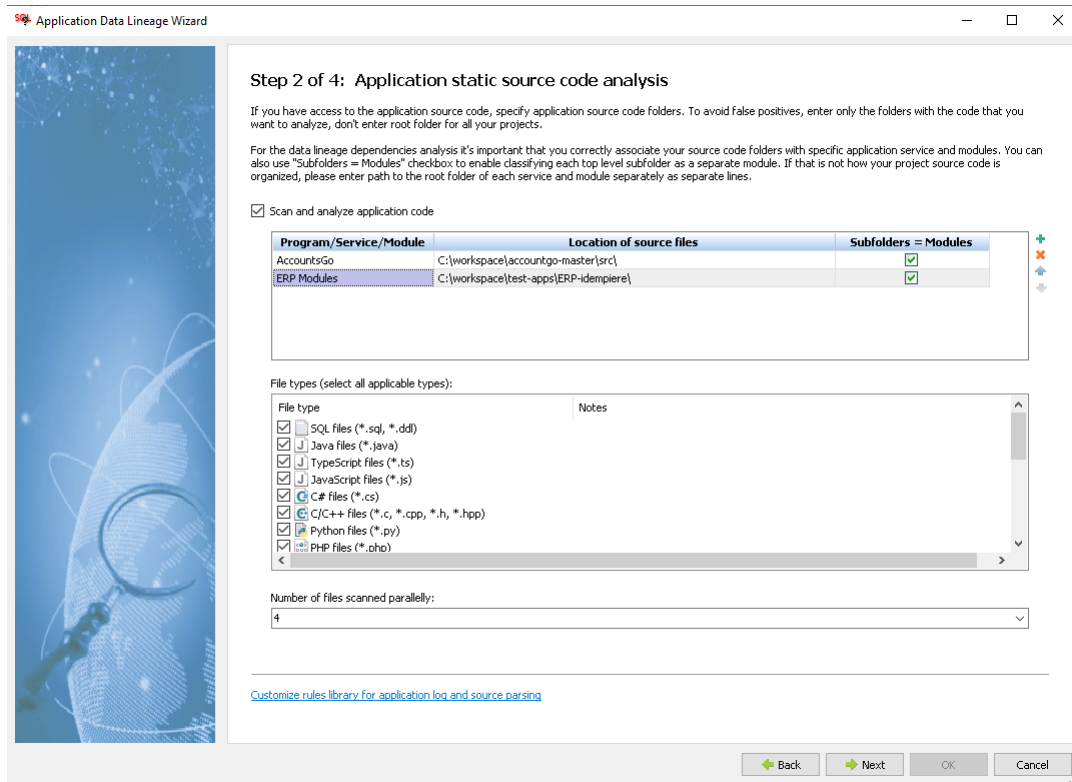
Use toolbar buttons adjacent to the right of the grid to add, delete, and reorder lines as required.

Use **Program/Service/Module** field to associate each set of source code files with their application component such as a service or a module. If your projects source code is structured such as each module or service has its source files in a separate top-level sub folder, you can use **Subfolders = Modules** option and avoid specific path to each service and module source files separately.

Tip: The association references are important for generating meaningful data lineage results, and useful for aggregating and displaying data dependencies using friendly service and module names.

Tip Parsing rule are predefined for all supported programming languages. If you need to customize them or you want to analyze other types of source code files, you can click the **Customize rules library for application** log and source parsing link. You can also use that link to test source code parsing before you continue to the next step. See the following Creating, Customizing, and Testing Parsing Rules topic for more information on creating, customizing, and testing parsing rules.

Click the Next button to proceed to the next step.

4. Choose the database connections for the analysis and associate each of them with specific application components. If the same database is used by different applications and services, enter the same connection multiple times so that you can associate it with multiple components.
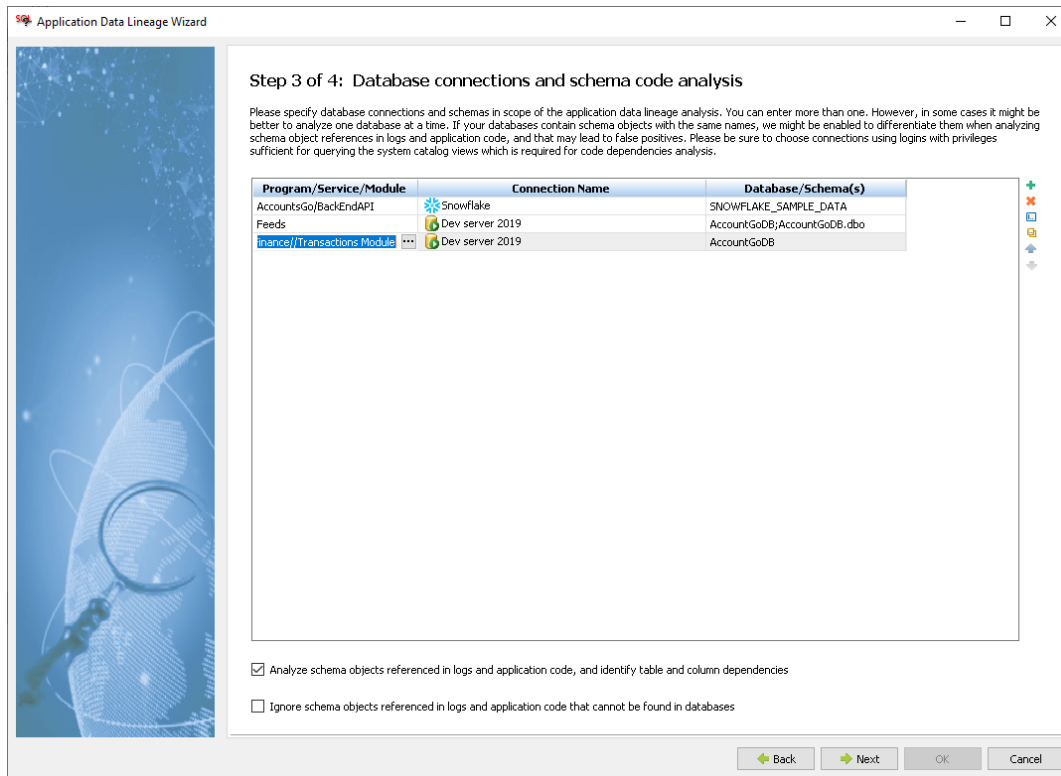
![Tip icon] **Tip:** The association references are important for generating meaningful data lineage results, and useful for aggregating and displaying data dependencies using friendly service and module names.

![Tip icon] **Tip:** By default the Application Data Lineage performs in depth analysis of database dependencies expanding view definitions, tracing stored procedure calls, parsing procedural code to identify specific table and column dependencies referenced in that code. The database code analysis could be very time consuming when working with very large databases. If you want to skip that processing, uncheck "**Analyze schema objects…**" checkbox. By default it also catalogs all schema objects referenced n the parsed logs and code even if it cannot locate them in the databases specified in the project. To ignore such objects, check the "**Ignore schema objects…**" checkbox.

Click the Next button to proceed to the next step

5.   At this point the data lineage wizard has all required inputs and can start analyzing the specified files. Click the Next button to start file parsing and analysis.

6.   After the analysis is complete click OK button to complete the wizard and proceeed to the Data Lineage workspace where you can interactively explore the data lineage diagrams, add your own anotations and save the project. See the following topc Working with Data Lineage Diagrams

Application Log Files

Having well formatted detailed application logs is extremely important for an intelligent data lineage analysis. Simply put good application logs containing records for database operations provide most value for data lineage analysis.

There are many popular logging frameworks in use today supporting well defined logging methods, flexible log

format patterns, and logging levels. Native logging facilities are available in .NET, Java, Python, and other programming systems. Different logging level control how much information is logged. For the data lineage analysis it's important to have all database operations logged along with their application context, such as process name, module name, class name, function name, ideally with a reference to specific line numbers in the source files. Logging database operations sometimes requires turning on DEBUG or similar logging level for a period of time to capture the database specific activities. The Application Data Lineage analyzer can consume the logged application context information and convert it to structured data which can be used for data lineage multi-level aggregation with drill-down capabilities. Here are examples of log records containing application and operation context details.

Log records in a well structured text based log file may look like the following

```
2021-02-18T16:58:30.1541866+00:00 – DEBUG: AccountUpdaterSvr – BalanceCalc –
ThreadDailyAggr – Running EXEC sp_UpdateDailyBalance
```

Similar records in JSON formatted log files may look like the following

```
{"date":"2021-02-
18T16:58:30.1541866+00:00","level":"DEBUG","appname":"AccountUpdaterSvr","logger":"Balan
ceCalc","thread":"ThreadDailyAggr","message":"Running EXEC sp_UpdateDailyBalance"}
```

From the above log records the Application Data Lineage analyzer can gather the name of the service "AccountUpdaterSvr", specific class name handling daily account balances "BalanceCalc", name of the execution thread "ThreadDailyAggr", and from parsing the message text learn name of the database stored procedure "sp_UpdateDailyBalance". It can then locate the mentioned procedure in the associated database, parse its code, and find out which tables are accessed and their columns, which of these columns get updated, and which are not.

Enabling ODBC Tracing

If your application uses ODBC based connections, you can utilize ODBC tracing to generate log files. ODBC tracing is inferior to normal application logging. It does not provide any application operations context, the logging occurs outside of the application, and the trace files contain database operation details only. Still it is a valuable source of data lineage intelligence when not other logs are available.

The ODBC tracing can be activated in Windows settings. To enable the trace option on Windows

1. Type ODBC in the Windows search box, you will see a link to open the ODBC Data Source Administrator utility.

2. The **Tracing** tab of the ODBC Data Source Administrator dialog box lets you configure the way ODBC function calls are traced. Choose the log file location and name, and click **Start Tracing Now** box.
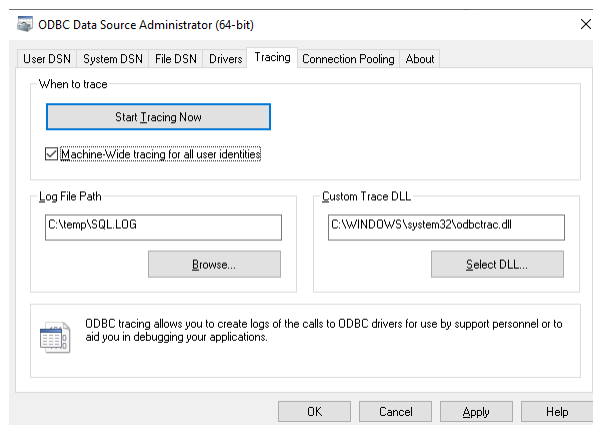
3. Let the application run for some time so you can capture its activities. If you do not wait long enough,

you may miss some of its database access operations.

4. Turn off the ODBC tracing when you are done.

Centralized Log Management Systems

Applications running in containers and cloud-native serverless applications often emit logs to standard output instead of writing them to log files. Their output is usually consumed by a centralized log management system such as Splunk, Humio, NewRelic, SolarWinds, and similar. To analyze their logs, you will need to export historical records from the centralized logging system to log files, and copy them to a location accessible from your computer.

Most centralized log management systems support multiple export methods. Your selection of an export method depends on the data volumes involved. Using search and export from a web browser might be appropriate for a low-volume export. For a higher-volume, use scheduled export, or dump or the Command Line Interface (CLI). You can export records to a server accessible by your centralized log management system, and if required, from that server you can copy them to a location accessible from your computer.

Creating, Customizing, and Testing Parsing Rules

About Parsing Rules

The Application Data Lineage utility uses regular expressions (regex and regexes for plural) for complex and flexible log and source code file processing. We use **rules** and **rule groups** terminology to refer to regexes and sets of regexes in order to avoid confusions with regex groups feature, which we use extensively too. A rule group is typically associated with a specific file format. For example, if you need to parse three types of log files with different log formatting patterns, you may need to create three different rule groups, one for each type of file.

A rule group is like a named container grouping together a set of related parsing rules. Multiple rules get executed against the same file lines in order to identify specific patterns, and to extract different details specific to the identified pattern type. The following example fragment from an application log file demonstrates different elements and messages that may be logged. The text highlighted in the example log needs to be matched by different parsing rules having different instructions for what to extract from the logged messages depending on the message pattern. It also demonstrates that different log lines may have different context details in different log records, and may also feature multi-line messages. For example, logged SQL queries often occupy multiple lines.

```
2022-08-25 09:10:36,004 [44] DEBUG Microsoft.EntityFrameworkCore.Database.Command -
Created DbCommand for 'ExecuteReader' (0ms).
2022-08-25 09:10:36,004 [44] DEBUG Microsoft.EntityFrameworkCore.Database.Connection -
Opening connection to database 'accountgodb' on server 'dev1012host'.
2022-08-25 09:10:36,126 [42] DEBUG Microsoft.EntityFrameworkCore.Database.Command -
Executing DbCommand [Parameters=[], CommandType='Text', CommandTimeout='30']
SELECT [i].[Id], [i].[CostOfGoodsSoldAccountId], [i].[Description],
[i].[InventoryAccountId], [a].[AccountName], [a].[CompanyId], [a].[Description],
[a1].[AccountClassId], [i2].[TotalCost]
FROM [Item] AS [i]
LEFT JOIN [Account] AS [a] ON [i].[CostOfGoodsSoldAccountId] = [a].[Id]
LEFT JOIN [Account] AS [a0] ON [i].[InventoryAccountId] = [a0].[Id]
LEFT JOIN [Account] AS [a1] ON [i].[InventoryAdjustmentAccountId] = [a1].[Id]
LEFT JOIN [Account] AS [a2] ON [i].[SalesAccountId] = [a2].[Id]
LEFT JOIN [ItemTaxGroup] AS [i0] ON [i].[ItemTaxGroupId] = [i0].[Id]
LEFT JOIN [ItemCategory] AS [i1] ON [i].[ItemCategoryId] = [i1].[Id]
LEFT JOIN [Measurement] AS [m] ON [i].[PurchaseMeasurementId] = [m].[Id]
```

```
LEFT JOIN [InventoryControlJournal] AS [i2] ON [i].[Id] = [i2].[ItemId]
ORDER BY [i].[Id], [i2].[Id]
2022-08-25 09:10:36,135 [44] DEBUG Microsoft.EntityFrameworkCore.Infrastructure -
'ApiDbContext' disposed.
```

Creating and Customing Parsing Rules

To create new and to customize the existing parsing rules, use one of the Customize Rules hyperlinks provided in the data lineage project wizard. That will open the **Parsing Rules** dialog. The dialog provides five tabs for different kinds of files that can be ingested by Application Data Lineage utility, and which require different kind of parsing rules and context mapping methods for their data elements.
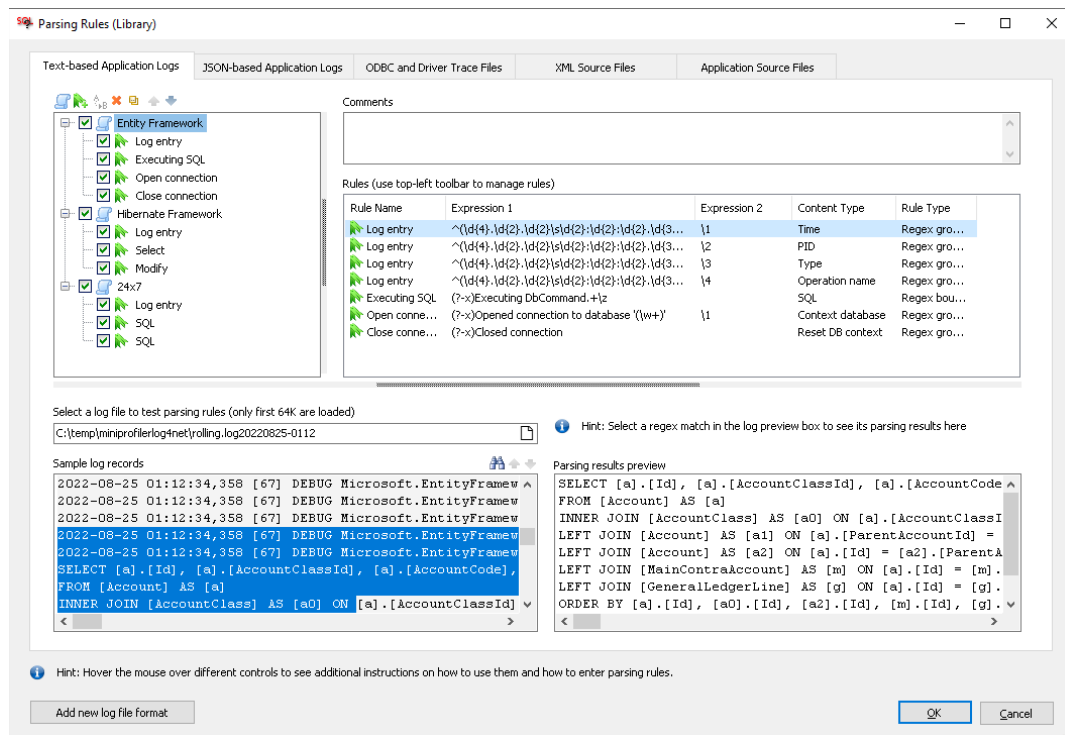
Their specific interfaces are described below

Parsers for Text-based Application Logs

Text-based logs are plain text files containing log records in a structured or semi-structured format that allows logs to be machine-readable and easily parsed. Log files generated by different applications may use different log record formatting patterns. The **Text-based Application Logs** tab is used to define parsing rules for such plain text log files.

Use the left-top rule navigation tree to manage rule groups and their rules. You should create separate rule groups for log files with different record formatting patterns. In each rule group you may need to define one or more parsing rules to parse out different. Please refer to and use as templates predefined parsing rules for logs generated by .NET Entity Framework and Java Hibernate.

Select a rule in the rules navigation tree to see and optionally edit its definition
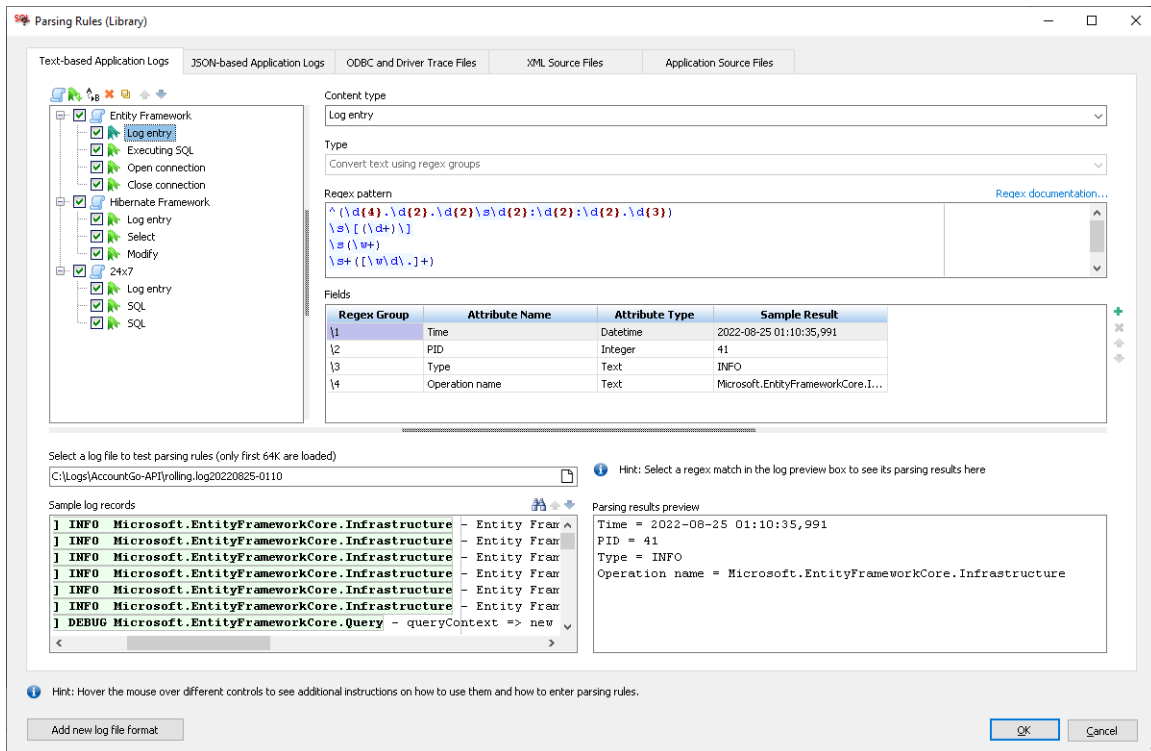
For additional information about regular expressions syntax, see Using Regular Expressions topic in CHAPTER 34, Integrated SQL Editors.

Parsers for JSON-based Application Logs

The JSON (JavaScript Object Notation) is a highly readable data-interchange format that has been adopted as the standard format for structured logging. Yet it may be sometimes challenging to parse too if their log records contain optional elements or feature multi-level structure with flexible optional elements that vary for different event types. The **JSON-based Application Logs** tab is used to define parsing rules for JSON based log files

Use the left-top rule navigation tree to manage rule groups and their rules. You should create separate rule groups for log files with different record structure. In each rule group you may need to define one or more parsing rules to parse out different JSON elements and their content. The parsing process involves two operations

1. Deserialization and mapping of JSON log record fields.

2. Optionally parsing the text queries from free text elements if they are mixed together with operation details and other text

The deserialization of JSON is done automatically for you. You still need to choose the elements providing application context like service name, module name, class name and similar that you want show in the data lineage results, as well as elements containing SQL queries. The following use case example demonstrates single level JSON file featuring SQL queries and an additional text mixed together, and how such file can be parsed.

Step 1: Select your JSON file for preview. Click the **New Parsing Rule** ⟋ icon (above rules tree) to add new rule. This will populate the **Map data** list with available element names. Now drag-and-drop the element containing SQL queries to the **Path** field as demonstrated on the screenshot below
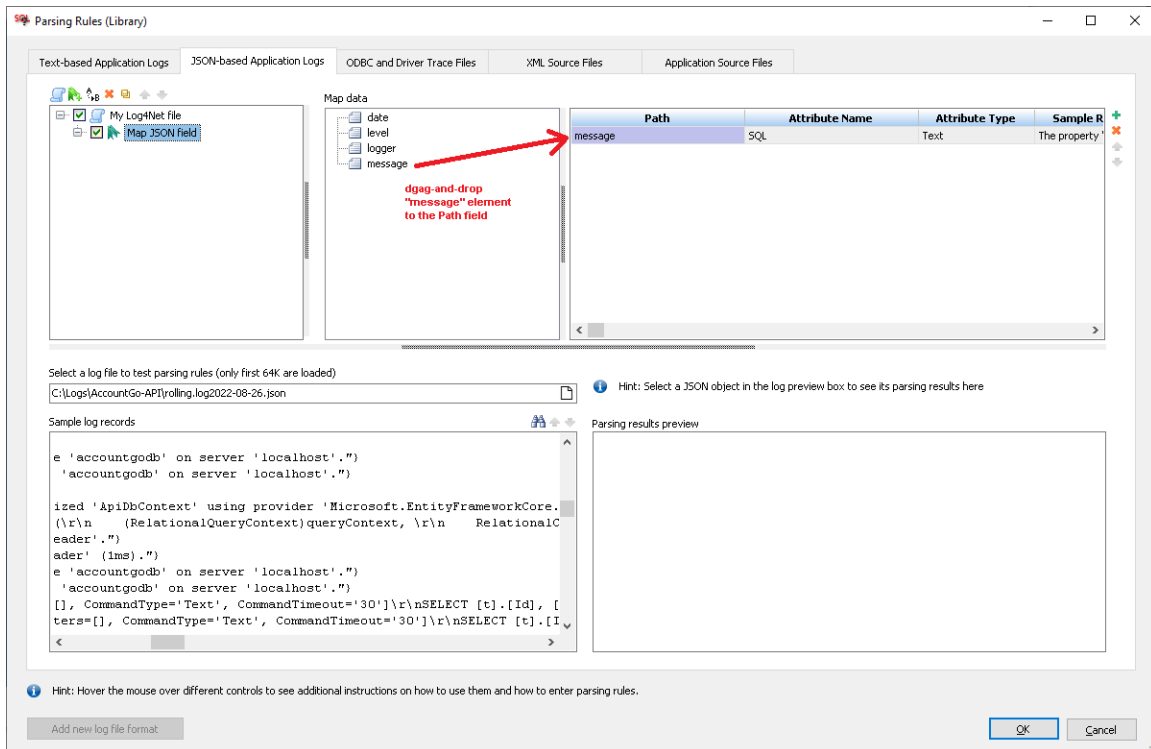
In the **Attribute Name** field use the drop-down list to select **SQL** value.

In the **Attribute Text** field use the drop-down list to select **text** value.

Step 2: Click the Text Replacement ⒜⒝ icon (above rules tree) to add new text extraction operation. Note that the right hand side of the dialog will change. Now you can enter a regex to match and extract only the required part of the message. Note this rule will be applied only to log records that match rule's definition
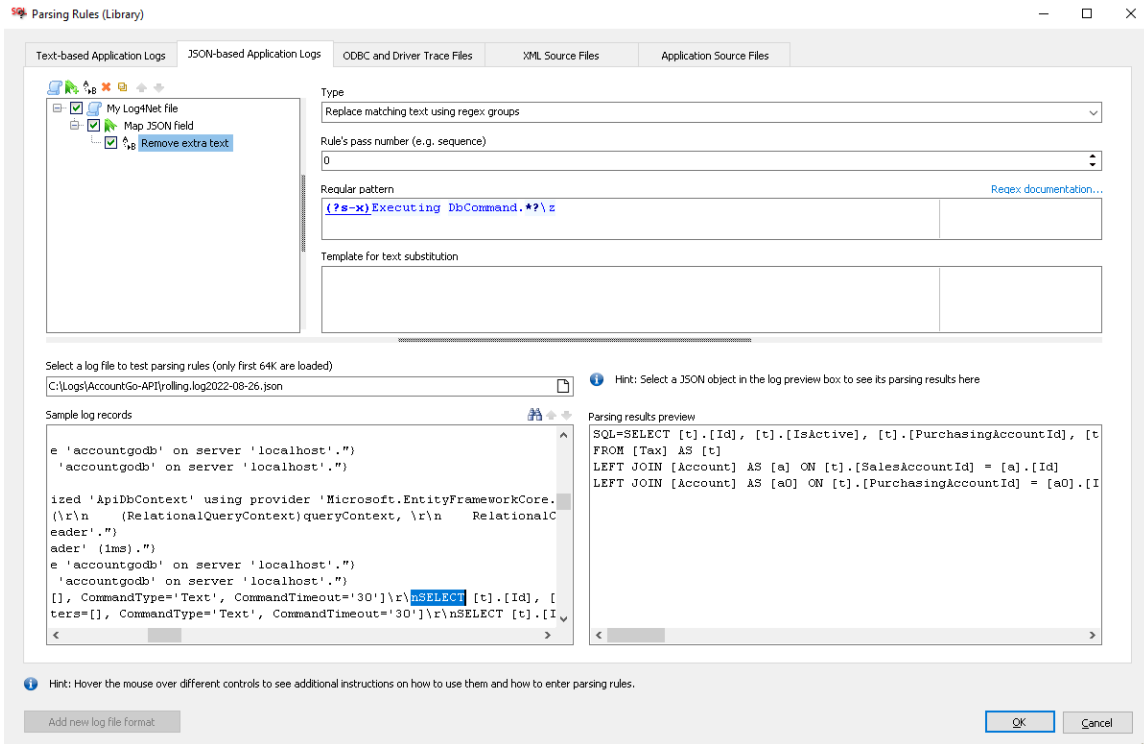
The following example demonstrates how you can map JSON log files with multi-level structure. In this example the logged message contains SQL queries only and doesn't require additional text replacements.
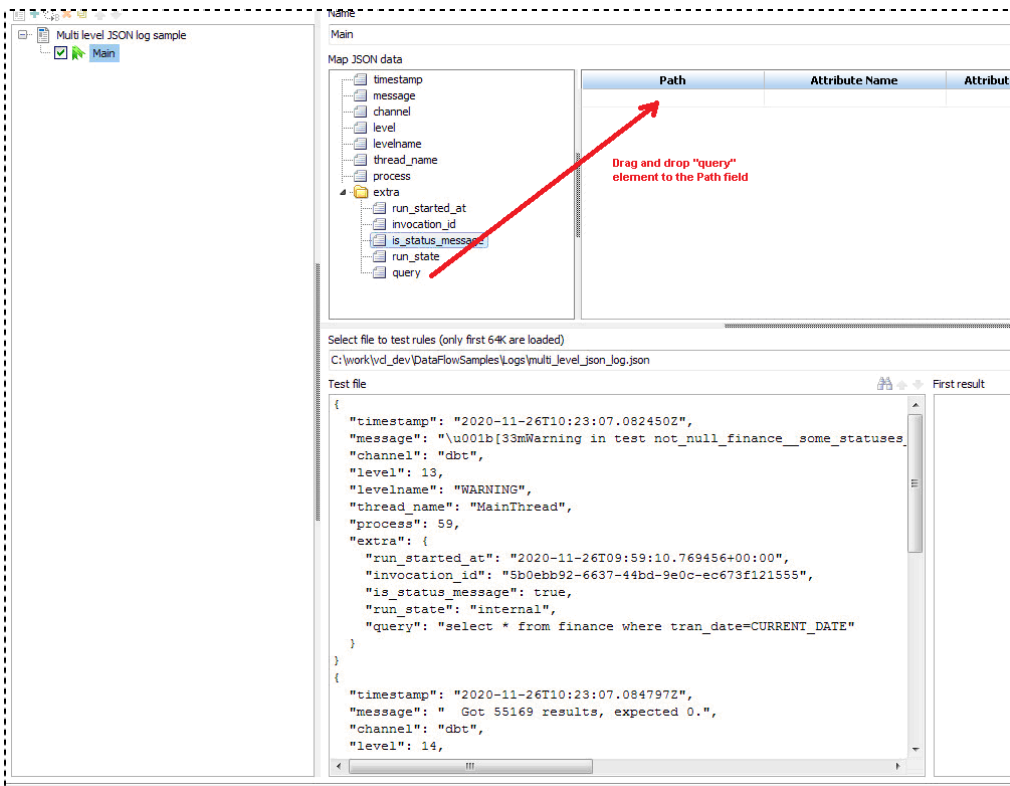
Parsers for ODBC and Driver-trace Logs

The parsing rules for ODBC trace are predefined. Use the **ODBC and Driver-trace Files** tab if you need to customize them.
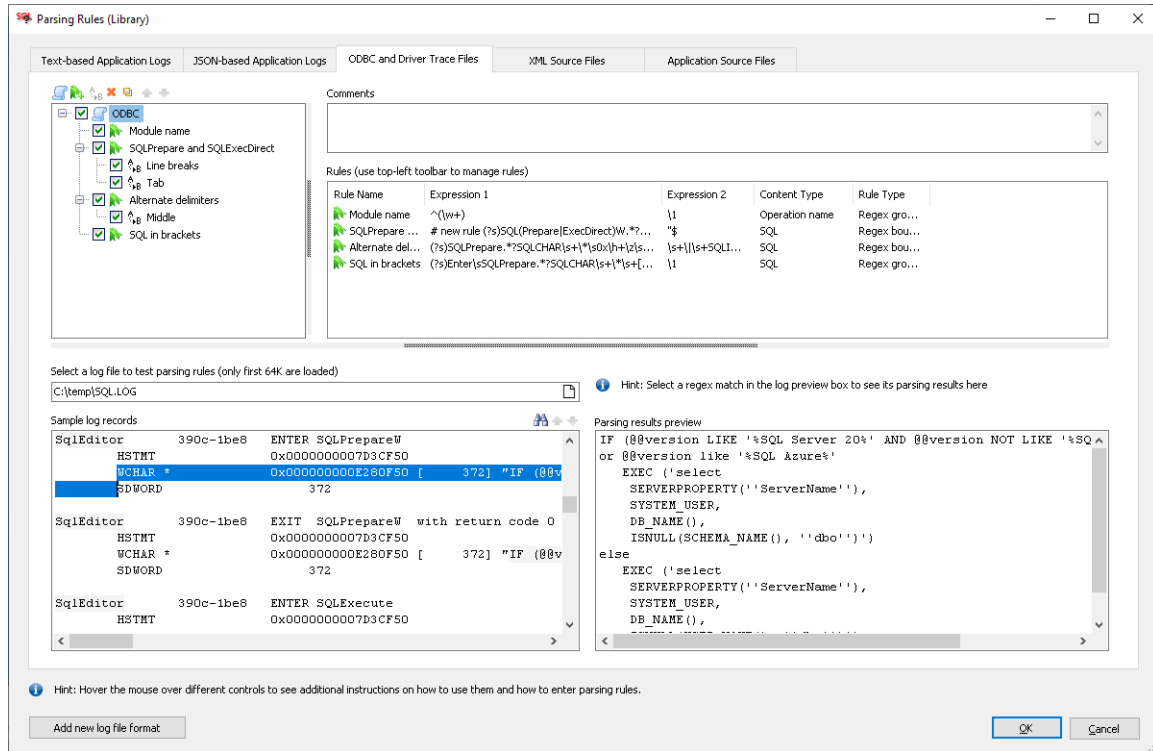
Parsers for XML-Files

A specialized parser is available for XML files which can be used if your application writers XML log files, or stores database queries in XML configuration files,. The **XML Source Files** tab is used to define parsing rules for XML files. Its interface and usage is the same as **JSON-based Application Logs**. Please see Parsers for JSON-based Application Logs topic for additional information.

Parsers for Application Source Files

The **Application Source Files** tab is provided mostly for information purposes only, and also as a testing tool for previewing source code parsing results. The actual code parsing is done performed by customizable programmatic language specific functions. Certain elements including file icons, and descriptions that may appear on the data lineage diagrams can be customized using this tab.

Working with Data Lineage Diagrams

Main Components and Controls

The **Application Data Lineage** workspace has five main parts described below.
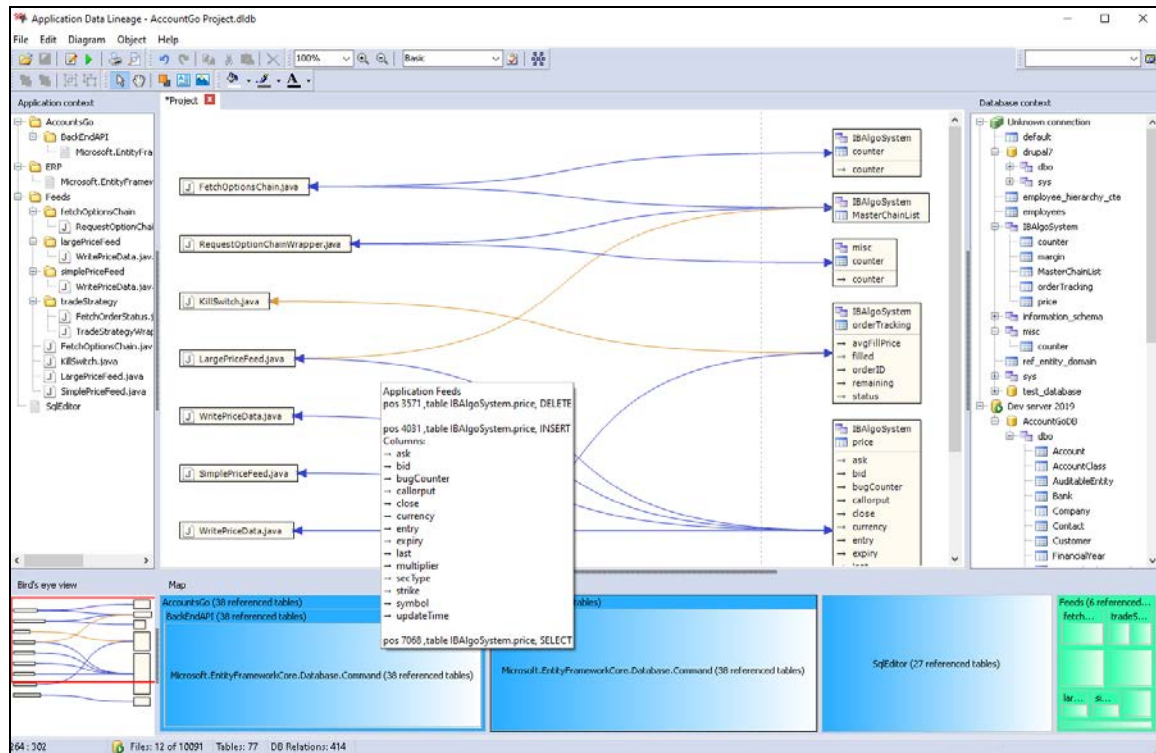
The top level menu and toolbars provide quick access to all main functions supported by the data lineage.

The **Bird's Eye View** window displays a miniature view of the active diagram. The window appears adjacent to the left side of the Application Data Lineage workspace. The sole purpose of the **Bird's Eye View** is to ease the navigation while working with large diagrams. In **Bird's Eye View,** a red rectangle is used to identify the area currently displayed in the active diagram window. Note the red rectangle is not shown if the diagram fully fits in the workspace. . Dragging the red rectangle causes the diagram window to scroll to display the section covered by the rectangle.

The **Application Context Explorer** window appears adjacent to the left side of the Application Data Lineage workspace. The content of that window references services, components, and source files with database dependencies in the scope of the project definition. They are organized in a hierarchical structure. For example, components of the same service appear as subfolders within the service folder. The actual hierarchy depends on how the application context has been defined in the project settings. Clicking a service, a component, or a source file the **Application Context Explorer** opens new diagram in a separate tab scoped to the selected item and their data lineage.

The **Workspace** is the area on the center hosting tabs with the data lineage diagrams. Note that diagrams can be opened and close much like files. See the following topic for detailed instructions on working with the diagrams.

The **Database Context Explorer** window appears adjacent to the right side of the Application Data Lineage workspace. . The content of that window references database tables and columns discovered by the data

lineage analyzer. Clicking a table in **Database Context Explorer** opens new diagram in a separate tab scoped to the selected table and its data lineage.

The colorful interactive **Map** at the bottom of the workspace provides at a glance view of the data dependencies. Application services, components, and files with more dependencies occupy more space on the map. Note the map shows hierarchy of application component nested areas. For example if there are five modules in a service and each module is featuring 2 application classes with database dependencies, eleven regions will appear on the map class regions nested within their modules and all modules nested within the same service region.

Multi-tabbed Multi-document Interface

The Application Data Lineage workspace supports multi-document interface, multiple diagrams can be open in multiple tabs. A new tab is opened for the selected components and tables.

Tabs that are no longer required can be closed using the X icon in right-top corner of the tab handle.

New tabs can be opened using the following methods:

- Clicking a service, component, or file in the **Application Context Explorer**

- Clicking a table in the **Database Context Explorer**

- Clicking a region on the **Map**

Single Object and Multi-Object Operations

For operations concerning object position and appearance, you can operation on both single and multiple objects.

Objects added to a layer can be also operated in groups. For example, you can move the entire group by moving its layer.

Selecting Objects

To select a single object, click the object you want to select. Most objects are agnostic to where you click them, but some like layers have a special area in the left-top corner used as a hot-spot for selection. This special area is indicated by a different background color.

To add more objects to the selection: Hold down Shift key and click additional objects to add them to the selection.

To select a group of objects in a specific area: Use mouse lasso effect - while holding down the left mouse button drag a rectangle around the objects you want to select.

To select all objects in a diagram: Press Ctrl+A key to select all objects.

To select a connector line, click the line you want to select.

Grouping and Ungrouping Objects

Grouping objects is typically a temporary operation which you can use to bundle different objects together and then manipulate them in groups while reorganizing the diagrams.

Another use of grouping is keeping related objects together as a single group. However Layers provide a richer set of options that can be used for object grouping.

To group objects, select multiple objects using any appropriate method, then click the **Group** icon on the toolbar.

To ungroup previously grouped objects, select the group, and then click the **Ungroup** icon on the toolbar.

Moving, Rotating, and Resizing Objects

Moving and resizing objects is the same as moving and resizing windows. If multiple objects are selected, they are moved as a group. If objects are grouped using the Group function, all objects in the group are moved together. If objects are placed onto a layer, the entire layer can be moved with all the contained objects.

On contrary, the rotate and resize operations are applied always to individual objects. However if the selected objects have been previous grouped together using the group tool, then resize operations effect all objects in the group.

Note: Selected objects have three dots displayed on each side of the object. Drag the dots to change object's dimensions.

Objects of certain type, for example text notes and images can be rotated, A special handle is shown in the object's area that you can drag sideways to rate the object. The following example demonstrates this technique

Note: The rotation handles are visible only when objects are selected.

Connector Lines and Mouse-over Hints

Two different colors are used for rendering lines connecting application components and database tables. Different colors are used to different data read and data write operations. The actual color depends on the selected theme.

You can mouse-over connector lines to see detailed information about the dependencies described by that data connection. The mouse pointer changes depending on the pointer position. Position mouse pointer directly over a line, makes the pointer shape change to an arrow with small question mark. As moment later balloon is shown with the data lineage details.

Diagram Appearance, Navigation, Pan Mode, Zoom, and Search

SQL Assistant supports shared themed interface for database model diagrams and for data lineage diagrams. See Customizing and Creating Themes topic in CHAPTER 38 for more information on how to customize existing and create new themes.

The Database Modeling Workspace and the Application Data Lineage Workspace share many other elements and user interface functions. See the related Navigation topic in CHAPTER 38 and the topics that directly follow it.

Using Notes

To add notes to a diagram, click **Add new text block** icon on the toolbar and then click in the diagram workspace where you want to add a new note.

To edit note text and change its appearance:

You may edit the note text by double-clicking it to switch to text Edit mode, or simply right click it and choose Edit command from the context menu. Using the context menu you can also change color and font, text block back-to-front relative position, and remove the note.

 Tips:

- The Del key cannot be used when editing notes text. This key is reserved for other uses. If you need to erase part of the text entered, use the Backspace key.

- Note text can have multiple lines. Use the Enter key while entering note text to break text by lines.

Printing Diagrams

The printing functions and options used to create printouts of your diagrams can be found under the **File** menu. The printing options are pretty much the same as in most other Windows programs, and do not require special description.

Saving Diagrams to Images and PDF Files

You can save your diagrams to PDFs and various image file formats so that you can share them with colleagues who do not have the SQL Software installed on their computers.

Saving to image files also enables you to open and edit the diagrams in other tools, as well as use advanced printing functions provided by many graphics editors.

To save your diagram to PDF or image file, select **File -> Export...** menu. Choose the file name and format and then click the **Save** button.

Note: To view a PDF file, you must have a PDF reader installed on your computer such as the Acrobat Reader, available from Adobe Systems.

CHAPTER 46, Automating ETL Operations and Other Processes

Overview

The ETL Orchestrator utility provided with SQL Assistant Professional Edition can be used to automate complex ETL operations and other database activities, and run them on demand, or schedule them using the Windows scheduler or other scheduling utility of your choice.

The ETL Orchestrator supports task grouping and precedence linking for advanced control flow. Tasks can be linked to trigger other tasks upon their completion or error. The linking can be linear described by one-to-one dependency type, or it can be more complex involving multiple tasks with many-to-one and one-to-many dependency types. Task grouping enables linking groups of tasks rather then individual tasks. Tasks with common dependencies start running at the same time. Tasks without dependencies start running first when the project is launched. See Control Flow and Connecting Tasks topics in this chapter for more details. In addition you can use multiple levels of nested subprojects that serve like containers for other automation tasks.

The ETL Orchestrator can be launched from **Data Tools / ETL Orchestrator** top level menu. The ETL Orchestrator workspace has five main parts described below.
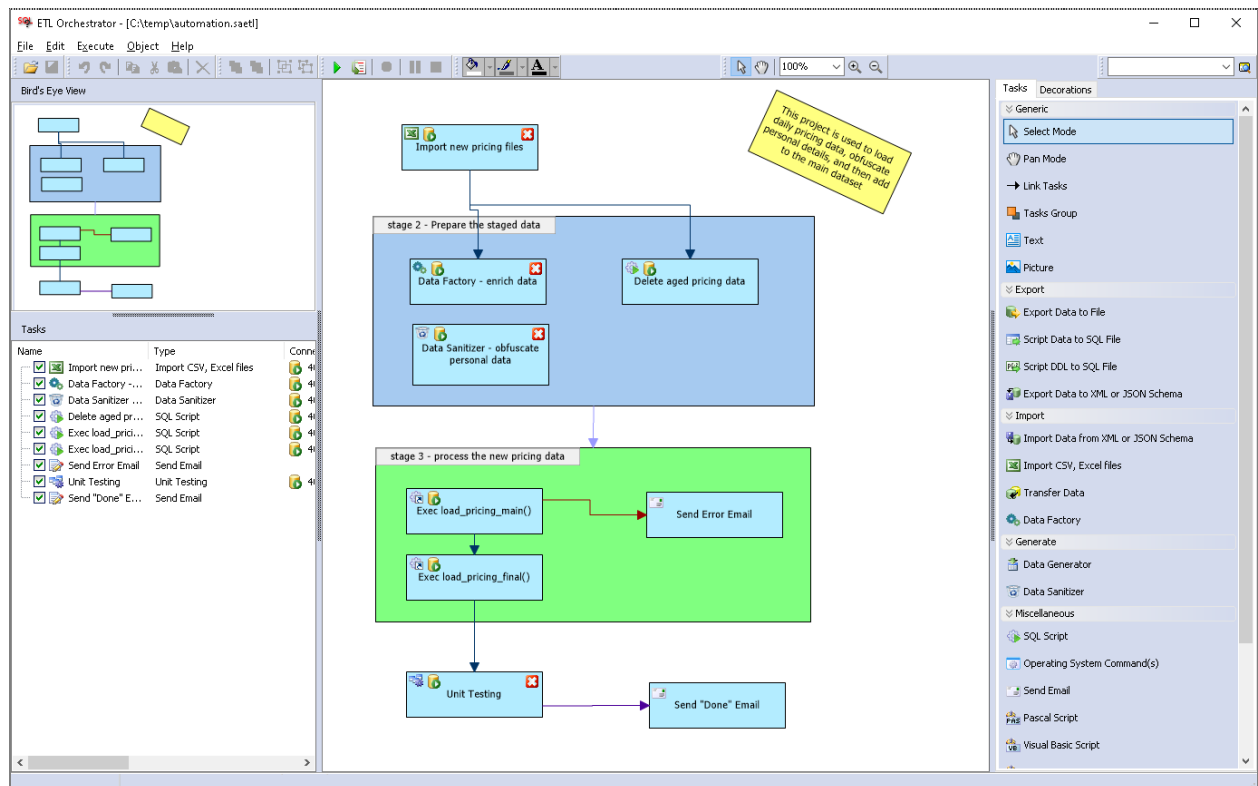
The top level menu and toolbars provide quick access to all main functions supported by the ETL Orchestrator.

The **Bird's Eye View** window displays a miniature view of the automation diagram. The window appears adjacent to the left side of the ETL Orchestrator workspace. The sole purpose of the **Bird's Eye View** is to

ease the navigation while working with large diagrams. In **Bird's Eye View,** a red rectangle is used to identify the area currently displayed in the active diagram window. Note the red rectangle is not shown if the diagram fully fits in the workspace. . Dragging the red rectangle causes the diagram window to scroll to display the section covered by the rectangle.

The **Task Explorer** window appears adjacent to the left side of the ETL Orchestrator workspace. The content of that window references all tasks in the project. Double-clicking a task in the **Task Explorer** opens task-type specific properties editor dialog.

The **Workspace** is the area on the center hosting the automation diagrams. See the following topic for detailed instructions on working with the automation ETL Orchestrator designer.

The **Tasks Toolbox** window appears adjacent to the right side of the ETL Orchestrator workspace. The Tasks Toolbox contains all supported task types that can be used for automating the processes.

Designing Automation Processes

Control Flow

The ETL Orchestrator provides three different types of control flow elements: subprojects that provide modular and reusable design, tasks that provide functionality, and precedence constraints that connect the tasks and task groups into an ordered control flow.

Tasks without preceding dependencies start running first when the project is launched. If there are multiple tasks without preceding dependencies, they all start running in parallel at the same time. Tasks having common preceding dependencies also start running in parallel at the same time when their preceding dependencies successfully complete.

Linear task dependencies

The liner tasks dependencies can be used for sequential task execution. In the following example the pictured tasks run one by one starting from task A, then followed by task B, and finally by task C.
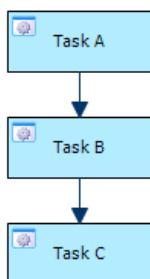
One-to-many task dependencies

In this scenario task A runs first. On successful completion it triggers tasks B, C, and D, all three of them starting at the same time and running in parallel.
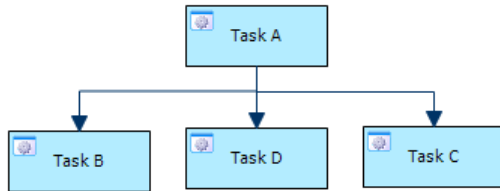
Many-to-one task dependencies

In this scenario tasks A, B, and C start running first, all running in parallel. On successful completion of all three tasks, task D is triggered and starts running too. Please note, that tasks A, B, and C may take different time to run. Starting of task D occurs only when the status of all its them preceding tasks changes to complete.
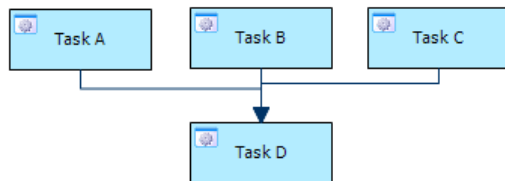
Adding Tasks

1. To add a new task to the automation workflow, click the required task type in the **Tasks Toolbox** and then click the place in the workspace where you want to create that task. Using drag-and-drop you can move tasks in the workspace.

2. Depending on the task type may get a prompt to choose the project or script file. A File Open dialog will be shown. You can use that to select an existing project file and link it to your ETL workflow. For example, if you want to execute data import tasks for which you already have an existing SQL Assistant's Data Import project, simply select that project file. If you don't have an existing project, click the Cancel button. You will be prompted to create a new project, or cancel it if you want to do it later.

3. If you selected an existing project or file and chosen to create a new one, follow instructions provided by the task specific wizard dialog to complete the task properties.

Modifying Tasks

1. Double-click an existing task either in the workspace or in the Tasks List. Or right-click it and select Edit command in the right-click menu.

2. Modify the task properties as required.

Deleting Tasks

1. In the workspace select the task you want to delete.

2. Press the Delete button on your keyboard.

Disabling Tasks

Use the disablement feature to temporarily disable tasks you do not need to run but which you do not want to delete from the project. You can also use disablement when troubleshooting issues and you want to skip some tasks while debugging the project.

1. In the workspace select the task you want to disable.

2. Right-click it and select **Disable** command in the right-click menu.

Connecting Tasks

1. Click the → Link Tasks icon in the Tasks Toolbox. The ETL workflow diagram will change and the tasks and groups will display connecting points that can be connected to link the tasks.
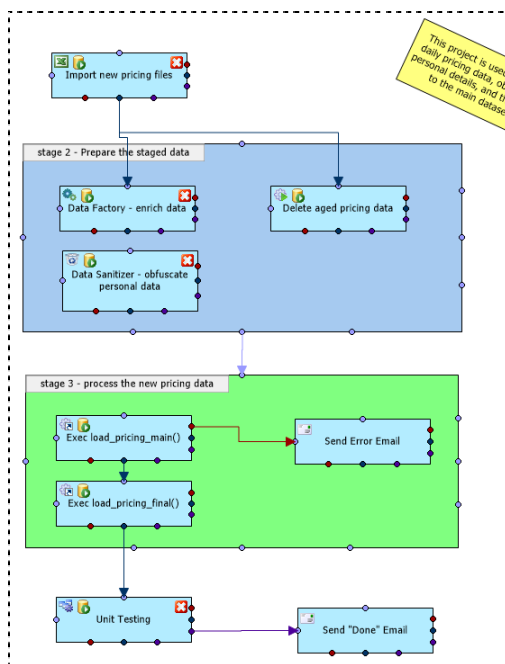
2. Click the red points in the two tasks or two groups if you want to connect them using **On Failure** connection type. Click the blue points in two tasks or groups if you want to connect them using **On Success** connection type, or click the purple points in two tasks or groups if you want to connect them using **On Completion** connection type. Note that **On Completion** connection type makes the downstream tasks to run regardless of completion status of their immediate upstream tasks. When adding connector lines, click the point belonging to the predecessor task first.

3. If you want to add additional comments to the connector lines linking the tasks, right-click it and select the **Edit** command in the right-click menu. Start typing the text you want displayed next to the line, for example

Changing Task Connection Types

Right-click the task connector line whose type you want to change. In the right-click menu choose one of the following commands:

On Success
On Failure
On Completion

Tasks Execution Groups

1. Click the Task Group ![Tasks Group] icon in the **Tasks Toolbox** and then click some free space in the Workspace. Using drag-and-drop move tasks you want to group to the new Task Group you just added.

2. If they do not fit, resize the group box as required by dragging it's edges with the mouse.

Tip: To select and move multiple tasks at once, use the mouse lasso method - hold down the left mouse button and trace around the tasks you want selected. Click any selected task with the left mouse button and without releasing drag the selection to the new group.

Tip: In addition to Execution Groups, task can be grouped into collections to make it easier to move them around and change together. This kind of grouping is completely independent of Execution Groups and does not impact project execution and control of flow. Use the ![icon] toolbar button to group selected tasks. Use the ![icon] toolbar button to ungroup selected task groups.

Common Edit and File Operations

To select multiple tasks and objects in the workspace, hold down Shift key while selecting objects with mouse.

To move selected objects within the workspace using keyboard, select one or more objects, hold down Ctrl key and them use Arrow Left, Right, Up, and Down keys to move them in the required direction.

You can use common edit operations **Copy, Cut, Paste, Delete** to work with objects in the Workspace. Such operations can be performed on all types of objects and their commands are available in the **Edit** menu. In the Edit menu you will also find commands for changing object positions on the screen.

To save project, select **File / Save** command.

To open previously saved project, use **File / Open** menu.

To save project diagram to a PDF file, use **File / Export** menu

To print the diagram, use **File / Print** menu.

Changing Database Connections

The ETL Orchestrator by default assigns the same database connection to all new tasks added to the project. You can assign different connections different tasks. The change the connection, tight-click a task, and select **Change Database Connection** command in the right-click menu.

Adding Annotations and Images

You can use text blocks, images, and predefined objects available on the Decoration tab to annotate your automation projects, to provide descriptions and other types of comments.

Running and Debugging

The ETL Orchestrator includes features that can help you with troubleshooting project execution. In run-time it highlights executed tasks, and their status. It also provides progress reporting with detailed color coded execution logs. In addition it supports breakpoints on tasks that enable you to control and pause task execution.

Progress Reporting

The ETL Orchestrator includes two types of progress reporting: color-coding on the tasks in the diagram based on their execution status, along with their status icons, and also progress messages in the Run Log pane at the bottom of the window.

Note that the Run Log pane is visible only after the project starts running, and hidden in design time.

When you run your project, the ETL Orchestrator depicts execution progress by displaying running tasks in green, and if they fail after execution changes their status to pink. Tasks that are waiting to run have their original colors and icons.

The following colors are used in the Run Log to depict task execution status.

Green   task is running or ran successfully

Pink – task run failed

Grey   task is disabled

Other colors – task is waiting to run. Its run timing depends on its predecessor tasks as defined by the task

dependencies.

Running Single Task

1.   Select the task in the workspace.

2.   Select **Execute / Run Step** command in the top level menu.

Running Entire Project

Select **Execute / Run** command in the top level menu.

Pausing Project Execution

Select **Execute / Pause** command in the top level menu.

Aborting Project Run

Select **Execute / Terminate** command in the top level menu.

Project Execution with Breakpoints

To make the project run pause at certain execution steps and wait for your input

1.   Select the task that you don't want the execute until you select to continue running.

2.   Select **Execute / Toggle Breakpoint** command in the top level menu.

3.   Repeat steps 1 and 2 for other tasks where you want to pause execution.

Use the same steps to remove previously set breakpoints.

During the execution the project run will pause on steps with breakpoints. To continue running use **Execute / Run Step** to run the next pending task or **Execute / Run** command to resume normal project execution until the next breakpoint is hit or until the logical end of the project, whichever occurs first.

Scheduling Unattended Runs

To schedule unattended non interactive project execution use **File / Schedule** menu command in the top level menu. This command opens the **Schedule** dialog. The options and controls are the same for all scheduled operations. See CHAPTER 16, Scheduling SQL Script Execution chapter for more information.

See Managing Scheduled Tasks topic in CHAPTER 13 for information on how to manage previously scheduled SQL Assistant operations.

CHAPTER 47, Querying and Manipulating CSV Files

Overview

SQL Assistant provides advanced data query interface and schema management capabilities for CSV files without a need to use ETL to import/export data from the files before it can be queried and manipulated. This is it. You can query file contents directly, modify file structure, update data values, pivot rows to columns, and perform many other common operations on CSV files. SQL Assistant transparently automates all required internal data management and transformation operations in the background.

The Data File Query Utility

The **Data File Query** wizard is accessible in the Data Tools menu in SQL Assistant Integrated Development Environment. You can use the **Data File Query** utility to execute SQL queries and scripts against CSV files as you would do that against regular database tables. The utility supports both regular CSV having column names in the first line row and headless CSV files. If a file is headless, then the columns are named "c0", "c1", "c2", and so forth.

When you choose a file to work with you have an option to open it in read-only mode. To activate that option, select **Read-only** checkbox in the first step. This option enables SQL Assistant to skip certain internal operations and open the selected file practically instantly. However, in this mode you will be unable to index the file data and modify its content.

Click the **Next** button in the wizard to advance to the second step in which you can run SQL queries against the selected CSV file using SQLite compatible syntax. SQL Assistant uses integrated SQLite engine behind the scenes to execute your query, which is coupled with complete SQL Intellisense, syntax and coding assistance, code snippets and other integrated features.

To utilize data pivot and other advanced data analytics functions, use a SELECT query to retrieve data from the file into a data grid. Then use the data grid toolbar that will appear above the retrieved data. For more information on how to use these functions see CHAPTER 24, Reporting, Data Pivot and Analytics.

For more information on how to use code assistance and SQL Intellisense see CHAPTER 3, Code Assistants and SQL Intellisense and also Working with Data Grid Interface topic in CHAPTER 11.

 Important Note:
Any data change operations are performed on a copy of the data and not directly on the selected file. You can use multiple UPDATE, DELETE, and other DML statements to modify the data and then review the results. If you are satisfied with the results, click the Save icon in the top toolbar to save changes back to the file. In case you are not satisfied with the results, simply reopen the same file without saving changes, or close the **Data File Query** wizard.
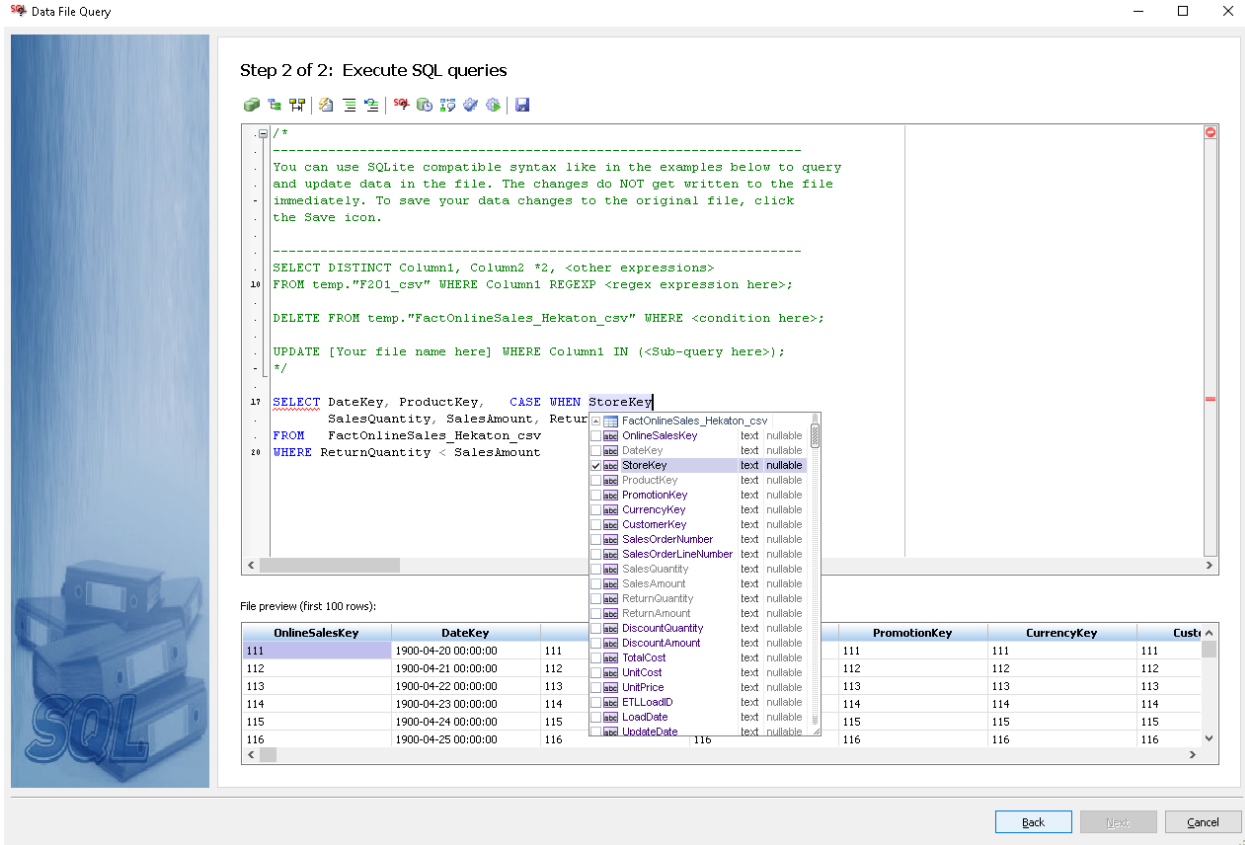
The Data File Transformation Utility

The **Data File Transformation** wizard is accessible in the Data Tools menu in SQL Assistant Integrated Development Environment. You can use the **Data File Transformation** utility to transform data in and alter structure of CSV file, as well as split and merge multiple CSV files .The utility supports both regular CSV having column names in the first line row and headless CSV files. If a file is headless, then the columns are named "c0", "c1", "c2", and so forth.

The **Data File Transformation** wizard provides specific instructions for each type of supported file transformation. Select the required CSV file and operation type and then click the **Next** button to the second step, then choose specific transformation options available for the selected transformation operation type.

Click the **Next** button again to go to the third step in which you can preview the results of the transformation, and if you are satisfied with the results, click the **Finish** button to save them back to the original CSV file or a new CSV file or set of files depending on the selected transformation operation type.
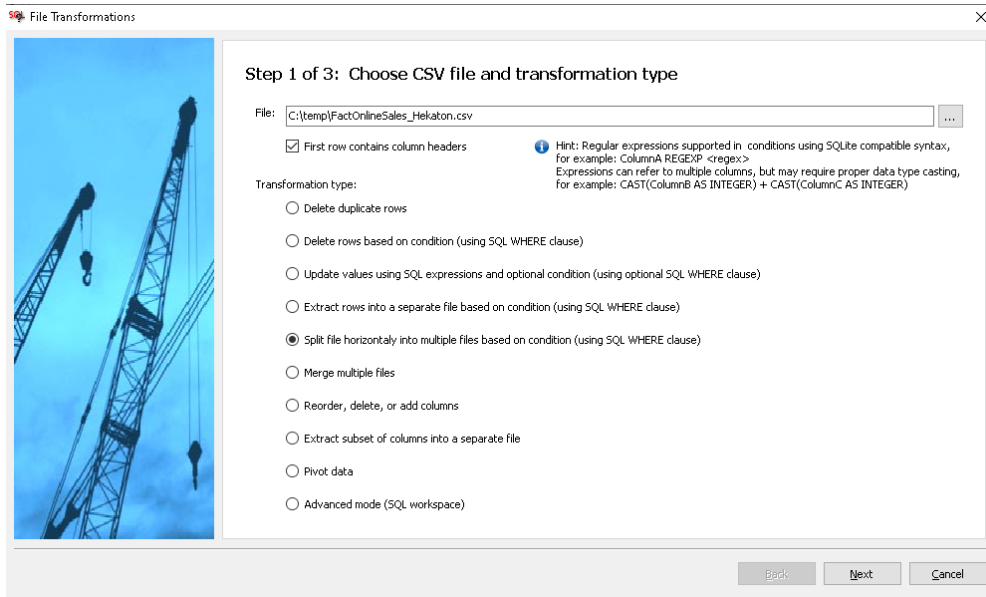
CHAPTER 48, Customizing SQL Assistant's Behavior

Overview

SQL Assistant has been designed from ground up for maximum flexibility and customization. From the very first version a particular attention has been given to enabling users to customize both the behavior and the appearance, to add custom code automation snippets, custom code formatting rules, custom code generation methods, custom templates, and so on. Most of the features can be customized in the main Options dialog which provides graphical means for customizing over 3000 different options and functions for all supported database interfaces. Filtering and searching controls have been added in SQL Assistant versions 10 and later to help users to quickly locate the required options.

To filter all options for a specific database interface, use the "Show options for" drop-down list. To search options by a keyword or substring, type the search text in the "Search options" box.

Note: The following help sections contain many screenshots captured in different historical versions of SQL Assistant. While some screenshots of the Options dialog may appear outdated, the location of the pictured options is not. the positions of different tabs and panels with options have been preserved between the versions.

Customizing Functional Hot Keys

To customize the default global hot keys:

1. Double-click the SQL Assistant icon in the Windows System Tray. The **SQL Assistant - Options** dialog will appear.

2. Click the **Common** section on the left side of the Options dialog screen.
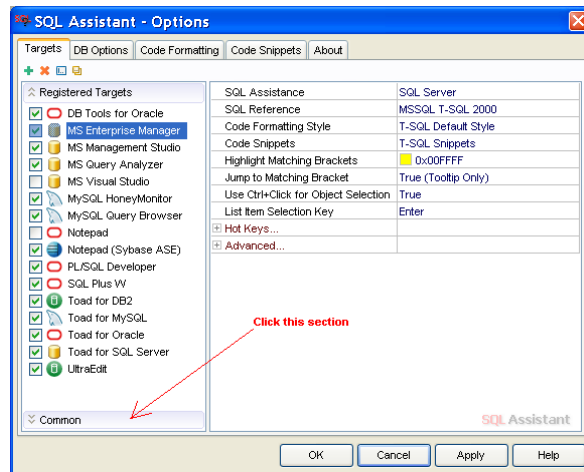
The **Common** section will be displayed.
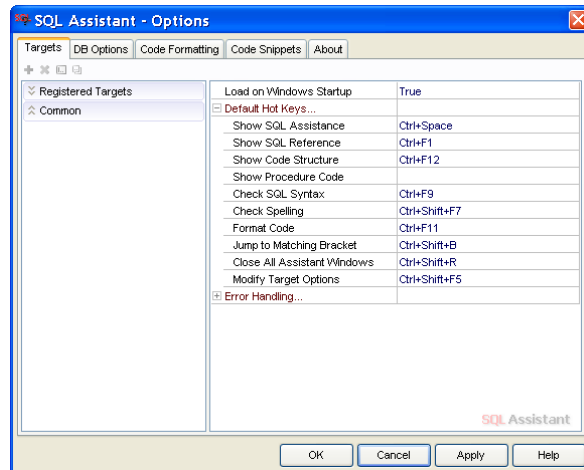
3. To change a hot key that opens a SQL Assistant popup window in a target SQL editor, click the **SQL Assistance** option and then press the keyboard buttons you want to use as the new hot key. For example, if you want to use Shift+Tab as the new hot key, press the Shift key and, while holding it down, press the Tab key.

    To change the hot key to open SQL Reference, click the **SQL Reference** option and then press the keyboard buttons you want to use as the new hot key. For example, if you want to use Ctrl+F1 as the new hot key, press the Ctrl key and, while holding it down, press the F1 key.

    To change the hot key to invoke SQL editor target registration function, click the **Add / Modify Target** option and then press the keyboard buttons you want to use as a new hot key. For example, if you want to use Ctrl+F5 as the new hot key, press the Ctrl key and, while holding it down, press the F5 key.

Important Note: Hot keys assigned in this way must be unique and must not be used by the SQL editor registered with SQL Assistant. Failure to pick a unique hotkey may lead to incorrect SQL Assistance behavior. If you use several editors you can pick different hot keys for different editors as demonstrated in the following section.

To customize target-specific hot keys:

1. Double-click the SQL Assistant icon in the Windows System Tray. The **SQL Assistant - Options** dialog will appear

2. Click the **Targets** tab page.

3. In the left window of the Targets page, select the target type from the list of targets.
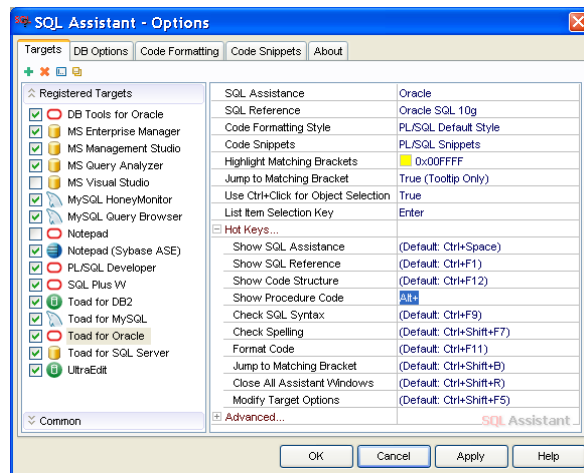
4. To choose an editor-specific hotkey, on the right side of the Options screen select the hot key option you want to change and then press the new hot key. The hot key can be a single key or any combination of Ctrl, Alt, Shift key and other regular keys.

    Tip: The **(default)** text indicates that the default global hot key is used for the selected target for a particular SQL Assistant feature. This is a special option value, which cannot be typed in the hot key field. If you want to undo the custom hot key and switch back to the default, select the feature to undo for which you want to switch back to the default, and then press the Esc key.

To customize list item selection keys used in SQL Assistance popup lists:

1. Double-click the SQL Assistant icon in the Windows System Tray. The **SQL Assistant - Options** dialog will appear.

2. Click the **Targets** tab page.

3.   In Targets page, select the target editor and then expand the **Options** group.
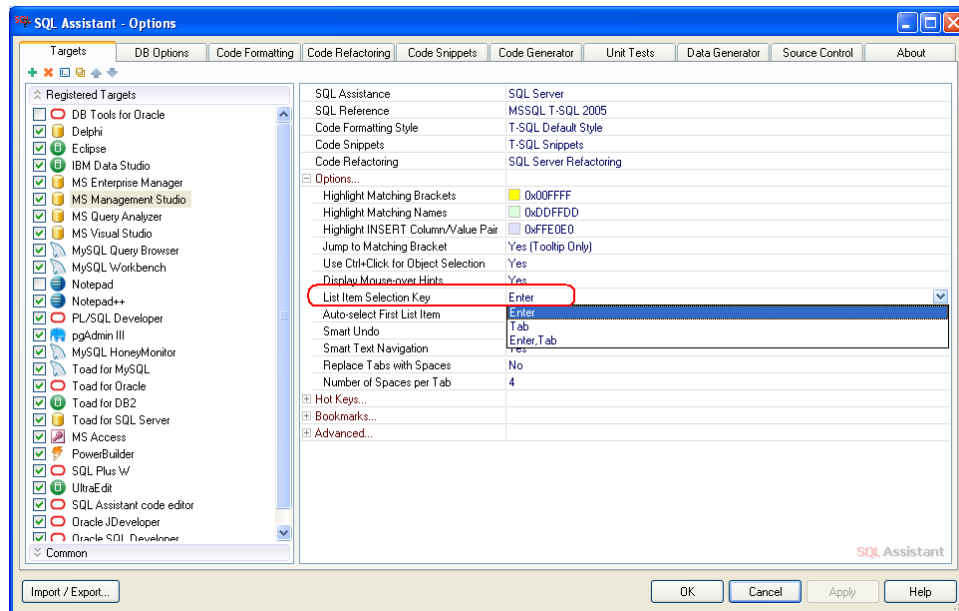
4.   in the **List Item Selection Key** option, choose selection key. Available options are:

Enter – allows using Enter key as item selection key, which is Windows standard item selection method.

Tab – allows using Tab key as item selection key, a non standard method used in Visual Studio based solutions.

Enter, Tab –allows using either Enter or Tab key as item selection key.

Customizing Code Snippet Activation Keys

Code snippet activation keys are used to expand snippet short names and insert snippet generated code into the script. To insert a snippet, place the cursor at the location in the script where you want to insert the snippet generated code, type the snippet name and then press the appropriate code snippet activation keys.

To customize code snippet activation keys:

1.   Double-click the SQL Assistant icon in the Windows System Tray. The **SQL Assistant - Options** dialog will appear.

2.   Click the **Code Snippets** tab page.

3.   In the left window of the **Code Snippets** page, in the top-left list select the SQL dialect whose code snippets you want to customize. The code snippets list appears below the SQL dialect list.

4.   In the snippets list select the snippet name.

5. On the right side of the screen expand the Key Char drop-down list and then select a snippet activation key.
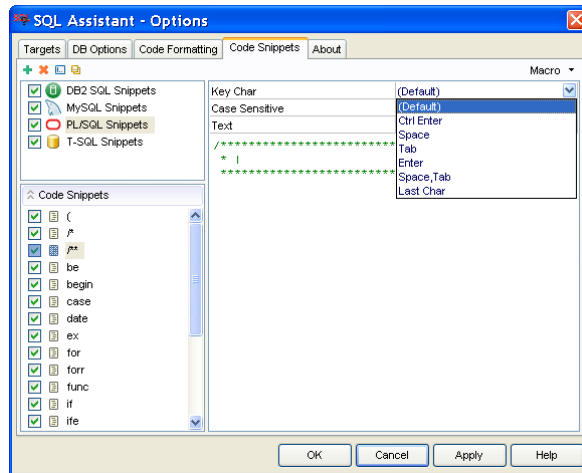
Available options are:

Ctrl+Enter – the Ctrl+Enter key combination (both keys pressed simultaneously) is used as the snippet activation key. This is the default activation key.

Space – the Space key is used as the snippet activation key

Enter – the Enter key is used as the snippet activation key

Tab – the Tab key is used as the snippet activation key

Enter, Tab – either the Enter key or the Tab key may be used as the snippet activation key

Last char – this is a special option that instructs SQL Assistant to automatically insert a code snippet whenever its short name is typed (without the need to enter a snippet activation key). Beware of snippets with similar names. If you define two snippets with a similar name like *snip* and *snip2* and set both to activate automatically using **Last char** option, you will not be able to use *snip2* because the snippet *snip* will always be activated first.

Managing SQL Assistant Load Methods

SQL Assistant supports two different methods for loading SQL Assistant code into the target editor:

- Target Editor Monitoring and Hooking

- Native add-ons.

Target Editor Monitoring and Integration

Target Editor Monitoring is a generic method can be used with many editors. The SQL Assistant system tray application monitors all windows opened within the running applications. it then checks their properties against the list of the registered targets. If the properties do match, it attaches the SQL Assistant client to the new

editor instance. The attached client uses standard Windows Hooks interface to hook the editor's keyboard and mouse event queues so that it can enhance the editor with SQL Intellisense capabilities and provide access to SQL Assistant functions.

Important Notes: The target editor must be a Windows application. The editor must support standard Windows edit control messages including but not limited to EM_GETLINE, EM_GETSEL, EM_GETLINECOUNT, EM_GETFIRSTVISIBLELINE and others. For details on standard edit control messages see MSDN http://msdn.microsoft.com/en-us/library/windows/desktop/ff485923(v=vs.85).aspx

To be able to watch for new editor instances and attach to them, SQL Assistant must be running as a system tray application (see the Starting and Stopping SQL Assistant topic for details).

Native Add-ons

Native Add-ons are currently available for the following several applications.

- Eclipse Integrated Development Environment v3.x and Eclipse-based derivative products, including Eclipse Data Platform, and others utilizing the platform's user interface to the full extent.

- Eclipse Integrated Development Environment v4.x

- DBeaver 20.x

- IBM Data Studio

- Visual Studio 2003, 2005, and 2008

- Visual Studio Professional and Community Editions 2010, 2012, 2013, 2015, 2017, 2019, 2022

- Visual Studio NET

- Visual Studio 2005 and 2008 for Team Database Professionals (formerly Data Dude)

- Microsoft SQL Server Management Studio 2005 and 2008

- Microsoft SQL Server Management Studio 2012, 2014, 2016, 17.x, 18.x, 19.x

- Microsoft SQL Server Management Studio Express

- MySQL Workbench 5.2 and later

- Oracle SQL Developer versions 3.x only. Never versions not supported.

- Delphi versions from 2005 to 2010

- Delphi XE versions from 1 to 10

Add-on Installations

The add-on for Eclipse 3.x must be copied to the *plugins* directory of your Eclipse installation. The *plugins* directory location differs for different Eclipse versions and for derivative Eclipse products. Consult your Eclipse documentation for details on where to find this directory. If copied to the correct directory, the host application automatically loads the add-on on the startup.

To install the add-on for Eclipse 4.x, use **Help -> Install New Software…** menu in your Eclipse IDE. Click the **Add...** button to add new local software repository, and then the **Local…** button. Select *C:\Program Files (x86)\SQL Assistant 12\jar\eclipse-install\eclipse* folder. Select **SQL Assistant Eclipse Plugin** in the list of items. In case Show by Category option is checked, the above item may be appear hidden. Uncheck that option. Click the Next button twice to complete the installation. Restart your Eclipse IDE.

To install the add-on for DBeaver, use **Help -> Install New Software…** menu in your DBeaver IDE. Click the **Add...** button to add new local software repository, and then the **Local…** button. Select *C:\Program Files (x86)\SQL Assistant 12\jar\eclipse-install\dbeaver* folder. Select **SQL Assistant Eclipse Plugin** in the list of items. In case Show by Category option is checked, the above item may be appear hidden. Uncheck that option. Click the Next button twice to complete the installation. Restart your DBeaver IDE.

The add-on for IBM Data Studio must be copied to the *plugins* directory of your IBM Data Studio installation. Consult your IBM Data Studio documentation for details on where to find this directory. If copied to the correct directory, the host application automatically loads the add-on on the startup.

The add-on for Oracle SQL Developer must be copied to the *ide\extensions* directory of your Oracle SQL Developer installation. Consult your SQL Developer documentation for details on where to find this directory. If copied to the correct directory, the host application automatically loads the add-on on the startup.

The add-on for Oracle JDeveloper must be copied to the *ide\extensions* directory of your Oracle JDeveloper installation. Consult your JDeveloper documentation for details on where to find this directory. If copied to the correct directory, the host application automatically loads the add-on on the startup.

The add-on for all Visual Studio and SQL Server Management Studio based products is registered in the system registry in the add-ons registration area. The host application automatically loads registered add-ons into each new instance of the Query Window.

SQL Assistant's system tray application may be used concurrently with these add-ons but theoretically is not required.

See CHAPTER 32, Registering and Unregistering Targets for SQL Assistance for details on how to register targets and modify SQL Assistant add-on loading methods.

Customizing SQL Assistant Menus

You have nearly complete control over composition and appearance of SQL Assistance menus. You can also add your own commands for executing the most frequently used macros, code snippets, unit tests, and external commands.

To change the composition and appearance of SQL Assistance menus:

1. Double-click the SQL Assistant icon in the Windows System Tray. The **SQL Assistant - Options** dialog will appear.

2. Click the **Common** section on the left side of the Options dialog screen. he **Common** section will be displayed.

3.   Click [+] sign in front of the **Menu** options to expand that section.
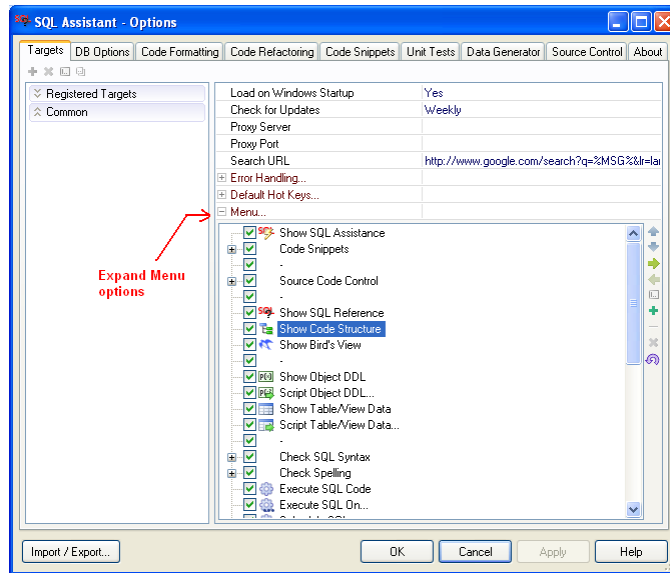
4.   Select the menu items whose appearance you want to modify. Select the checkbox to the left of a menu item to include that item in the menu. Deselect the checkbox to remove an item from the menu.

Use the toolbar buttons on the right side of the dialog to manage menu items in the list. Following is a description of each button:

⬆	Moves the selected item up one line
⬇	Moves the selected item down one line down.
➡	Moves the selected item to the next lower level
⬅	Moves the selected item to the next higher level
▣	Renames the selected item
—	Inserts a horizontal item separator line above the selected item
✚	Inserts a new custom menu item. Clicking this button displays a list of options for the action type to be performed by the new item. The list of available options is described below following step 5.
✖	Deletes the selected custom menu item. The delete operation is available for custom menu items only. SQL Assistant native menu items cannot be deleted but can be hidden using check-boxes to the left of the item name.
↺	Resets the entire menu to its factory-default state.

5.   Click the **OK** button on the bottom of the dialog to save Changes.

Custom Menu Items in the Main Menu

Seven action types are supported for custom menu items added using the Add ✚ toolbar button:

None – creates expandable menu items that group other items in a subordinate list

Execute External Command –creates menu items for executing an external program. With this action type you can select the program name and to enter optional command line parameters.

Execute Test Data Generator Project –creates menu items for executing SQL Assistant's Test-Data-Generator projects. With this action type you can select specific project file that you want to be added to the menu.

Execute Unit Test Project –creates menu items for executing SQL Assistant's Unit Test projects. With this action type you can select specific project file that you want to be added to the menu.

Execute Code Snippet – creates menu items for executing code snippets. With this action type you can select specific code snippet that you want to be added to the menu.

Execute Copy SQL Code – creates menu items for executing SQL Assistant's built-in and user-defined "Copy SQL As…" functions. With this action type you can select specific copy function that you want to be added to the menu.

Execute Script - creates menu items for executing script files using FastScript scripting engine. With this action type you can select specific script file that you want to be added to the menu. For more information about plugins, scripting interfaces, and extensibility see SQL Assistant's Plugins Developer Guide.

Custom Menu Items in Right-click Context Menus

In addition to custom commands that you can add to the SQL Assistant main menu you can also add custom commands to context specific menus that are used in different SQL Assistant windows.
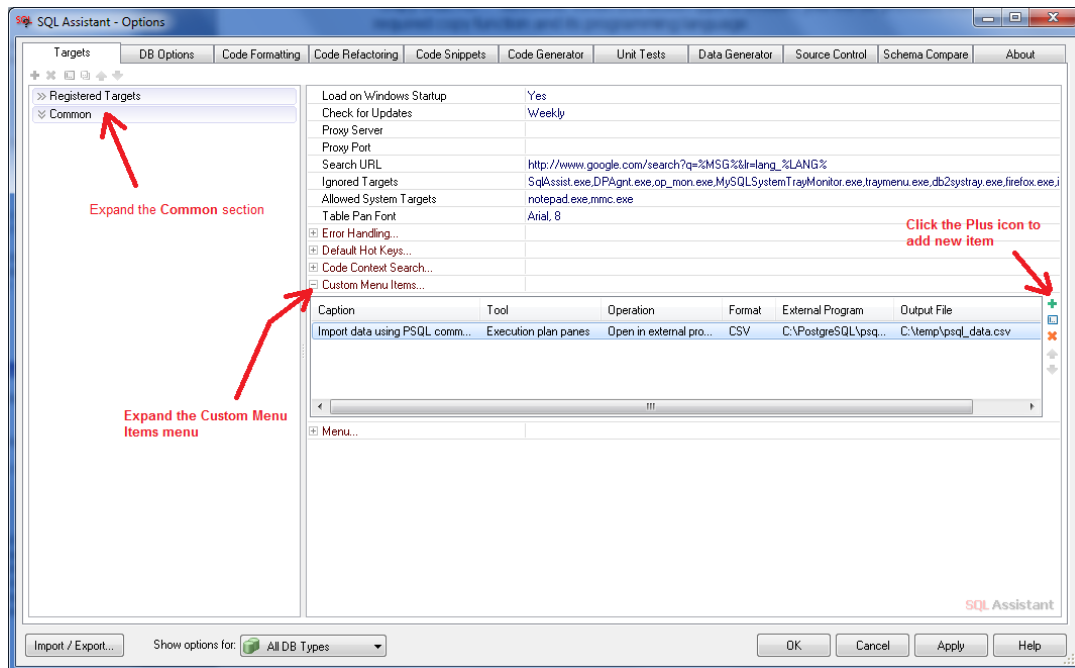
Follow the graphical instructions above to setup new custom menu commands.

To simplify the configuration we separate SQL Assistant windows into four categories:
- **Data-grid panes** –windows with tabular data-grid controls.
- **Execution plan panes** – windows containing SQL execution plan results.

- **Text panes** –windows containing only textual results, for example, DDL View panes.
- **Messages panes** – windows returning information messages after code execution. They are kind of similar to text panes, but have each line starting with a graphical icon and independent.

When you add new custom menu, you have to choose the window type to attach it to.

You can choose four types of actions for your custom commands:

- Launch an external program.
- Save output data from the window to a disk file.
- Copy all data from the window to the system Clipboard and then optionally launch an external program.
- Execute custom plugin. . For more information about plugins, scripting interfaces, and extensibility see SQL Assistant's Plugins Developer Guide.

If you want to pass data from SQL Assistant to a different application, you should choose the last action type. An example is demonstrated on the following screenshot:
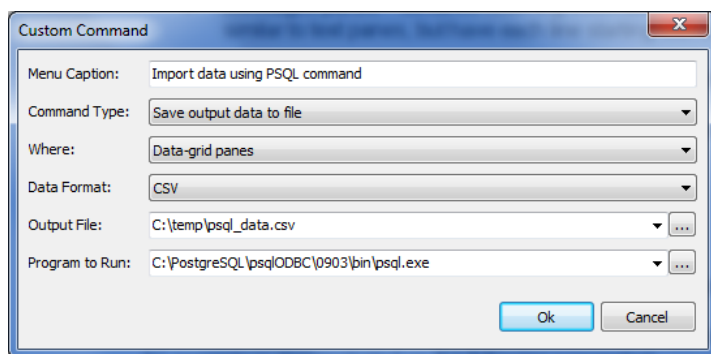
To test your custom command is working.

1. Open SQL Assistant SQL Editor and connect to the database.

2. If your command is supposed to be attached to a data-grid, enter and execute some SQL query returning result set.

3. Right-click the data-grid with the result. In the right-click menu select **Custom Commands** menu branch. Your custom menu command should be shown there. Click it to test it's working as specified.

Tips:

- Data from data-grid windows can be saved to files or clipboard in any of the following formats: TXT, CSV, XML, JSON.
- Data from Execution Plan windows are saved in XML format. This is the only format supported.
- Data from Text and Message windows are saved in TXT format. This is the only format supported.
- When choose Command Type for saving data to a file or clipboard, an external program is started only it is specified in the **Program to Run** value. If that value is empty and processing ends at that point without any errors.
- The **Output File** value is optional. If the output file name is not specified, and the Command Type is Save output data to a file, the system Save dialog is shown when the command is used so that you can dynamically choose the output file.
- If an external program is specified, but specified without any command line parameters, the output file name is passed on the command line as the only parameter. If you need to pass any additional parameters, you should chose static file name for the output (will be overwritten on each use) and specify other parameters as required by the program.

- You can also pass the database connection string used by SQL Assistant to the external program. Use %1 and %2 as predefined command line parameter placeholders for the file name and connection string. %1 placeholder is used to pass the file name without quotes and %2 placeholder is used for the connection string.

Important Notes: The custom menu commands work only in SQL Assistant created windows and SQL Assistant's SQL Editor windows. They will not appear, and will not work in windows created by other target editors. For example, they will not work in SQL Server Management Studio data-grid windows.

Customizing Target Editor Menu Integration

SQL Assistant provides several methods for integrating its menus with target editor menus. By default, SQL Assistant inserts additional menu items into target editor context menus only.

The following steps describe how to choose menus in which to add SQL Assistant commands:

1. Double-click the SQL Assistant icon in the Windows System Tray. The **SQL Assistant - Options** dialog will appear.

2. Click the **Targets** tab page.

3. In the targets list, select the target whose menus you want to modify.

4. Click the **Advanced…** option on the right side of the screen. The [...] button appears on right side of the field. Click this button to show advanced options.
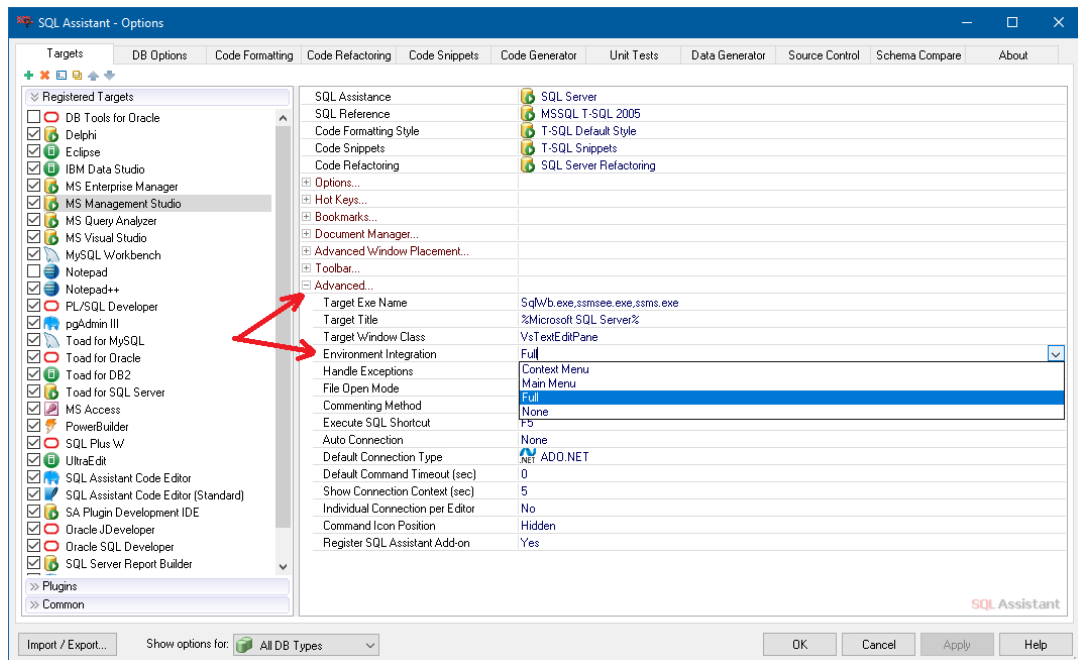
5. Open the drop-down menu for the **Environment Integration** option and choose a menu integration option. Note that the **Full** option means that both the top-level and context editor menus can be modified. The **None** option means that neither menu can be modified.

Tip: Menu integration is not required. It is only provided for your convenience so you won't need to remember all supported hot keys. You can also use SQL Assistant's system tray menu as described in the Using System Tray Icon Menu topic in CHAPTER 3.

Important Note: For target editors such as Eclipse, SQL Server Management Studio, and Visual Studio, when changing the **Environment Integration** options verify SQL Assistant Add-in is registered. The menu integration only works if the **Register SQL Assistant Add-in** option is set to Yes.

Additional Menus in SQL Server Management Studio Grid Controls

Enabling the **Full** menu Integration option activates an additional feature available for SQL Server Management Studio targets only. The context menu for the result set data grids contains additional commands provided by SQL Assistant enhancing user productivity and experience as shown on the following screenshot.
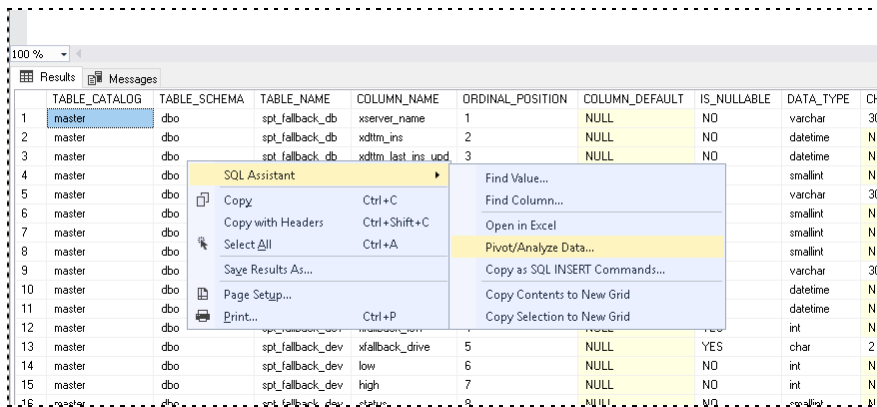
The additional commands expose a subset of the functions available in SQL Assistant data grid controls. To access the complete functionality, use either **Copy Contents to New Grid** of **Copy Selection to New Grid commands**, which will convert the data in the grid to SQL Assistant grid.

Refer to CHAPTER 11, Data Display and Editing for the description of the supported commands and how to use them.

Managing Plugins for SQL Assistant

SQL Assistant supports extensibility via 3rd party and custom plugins. Such plugins extend SQL Assistant's functionality, they are not to be confused with addons and plugins that are part of SQL Assistant software and which are used to integrate SQL Assistant with other development environments. For the later see Native Add-ons topic in this chapter.

Plugins need to be registered in SQL Assistant's options before they can be referenced in custom menus and other places.

To register a new plugin:

1. Make sure the plugin files are copied to C:\Program Files (x86)\SQL Assistant 12\Plugin folder.

2. Double-click the SQL Assistant icon in the Windows System Tray. The **SQL Assistant - Options** dialog will appear.

3. Click the **Targets** tab page.

4.   Click the **Plugins** section.
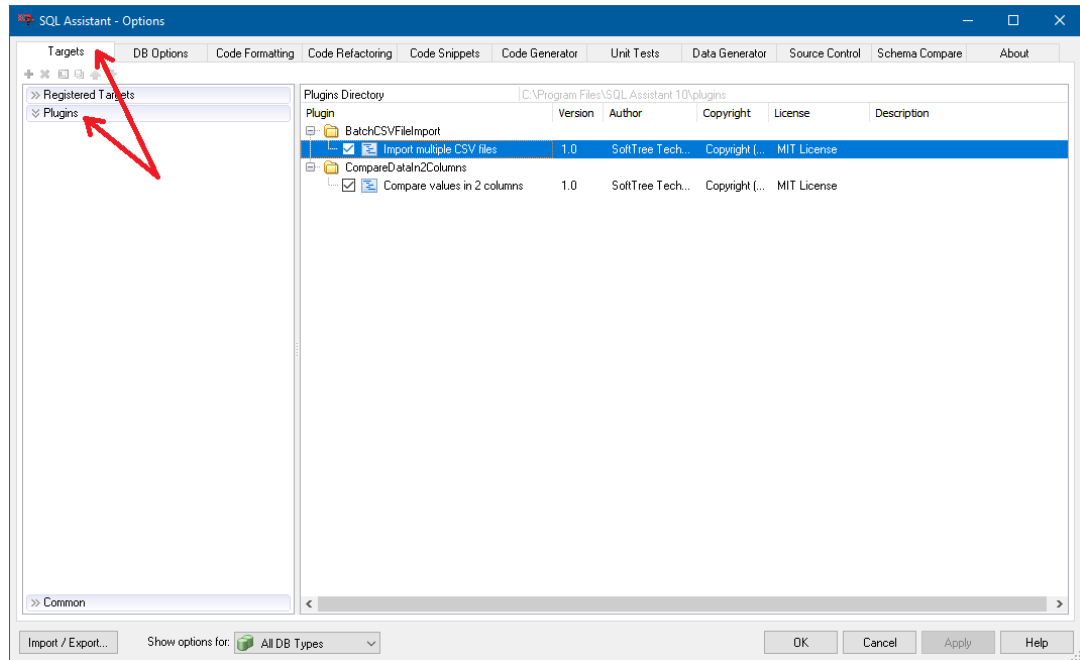
All correctly installed plugins should be listed in this section.

5.   Tick the checkbox to the left of plugin name to enable it or uptick to disable and ignore that plugin.

6.   Click the **OK** button on the bottom of the dialog to save Changes.

Tip: The procedure described above does not install or uninstall plugins. It simply makes them enabled or disabled for loading and use. For more information on plugin development and installation refer to Plugins Developer Guide, which is a separate manual.

Customizing Settings for Eclipse-based Target Editors

Several Eclipse-specific settings can be customized directly in the Eclipse IDE:

1.   In Eclipse, click the **Windows → Preferences** menu command. The **Preferences** dialog will appear.

2. Click the **SQL Assistant Plug-in** item in the Preferences tree.
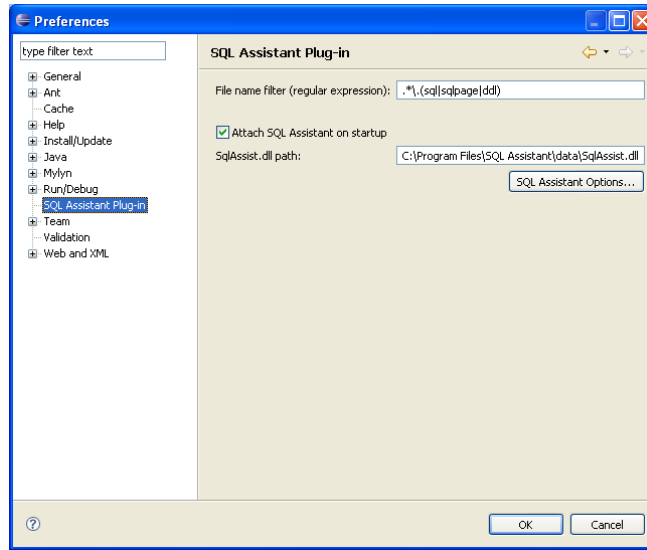
3. On the right side of the dialog, customize the SQL Assistant settings as required.

    Important Note: Make sure the value of "SQLAssist.dll path" option is correct.

4. Click the **OK** button to save Changes.

Tip: If you use specialized SQL editors within the Eclipse environment, it is also recommended that you disable their **Code Assistant auto-activation** feature in the Eclipse Preferences. See the Configuring Eclipse-based Target Editors topic for more information.

Customizing SQL Assistance Types

"SQL Assistant Type" is a collection of various settings and functions defining SQL Assistant behavior. The SQL Assistant comes with several pre-configured SQL assistance types called for clarity "Oracle", "DB2 UDB", "DB2 for iSeries", "SQL Server", "MySQL", "MS Access", "Sybase ASA", "Sybase ASE", "PostgreSQL", "Amazon Redshift" and "SQLite." All pre-configured types are wired to the complete set of pre-defined database catalog queries. code formatting rules, snippets, refactoring methods and a other settings and functions.

SQL Assistance interface allows you to customize the default assistance types as well as create your custom types. The following example demonstrates how to create custom type for SQL Server using a limited subset of catalog queries:

1. Double-click the SQL Assistant icon in the Windows System Tray. The **SQL Assistant - Options** dialog will appear.

2. Click the **DB Options** tab page.
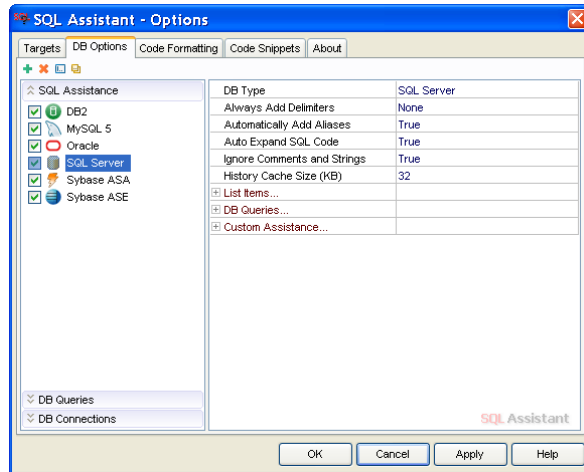
3. Click the Plus Sign icon in the top left corner of the **DB Options** tab page to add a new SQL Assistance type. Type a name for the new row type, for example *My Custom Type*.
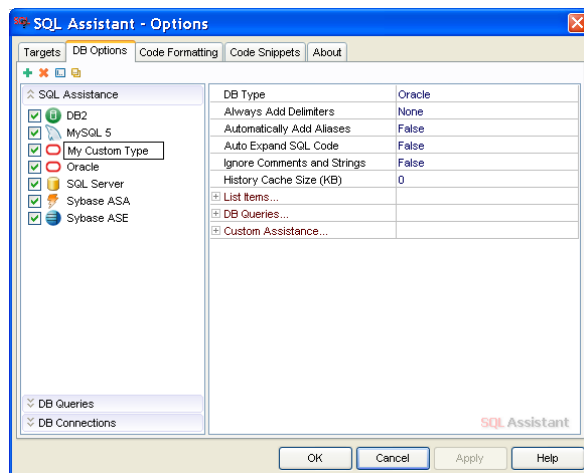
4. In the **DB Type** drop-down, select of the database type for which the new SQL Assistance type will be used.

5. Expand the **DB Queries** option to show DB Query associations and their properties.
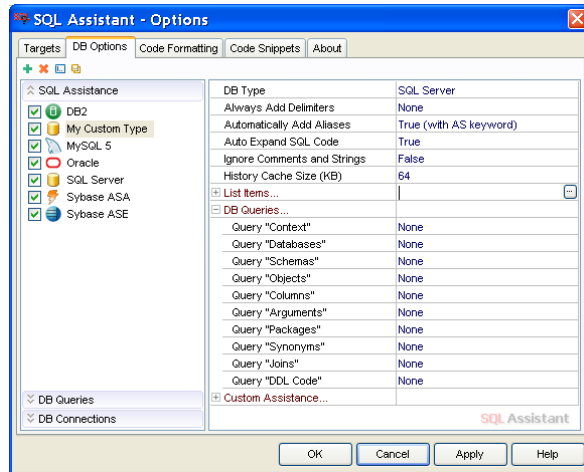
6. Choose query associations as needed.
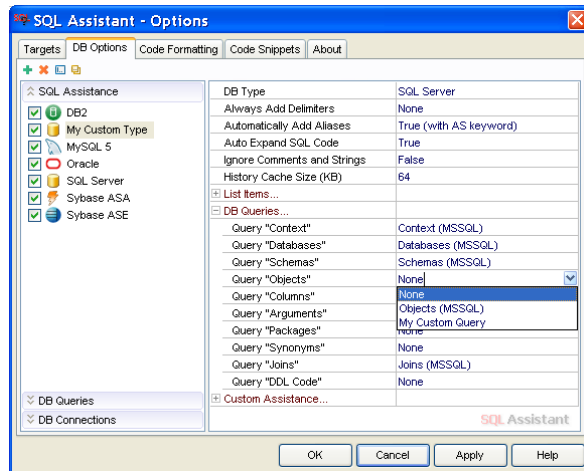
For example, in the DB Type option select "SQL Server", for "Objects", "Columns" and "Joins" select items as on the following sample screenshot. Note that if you leave "Databases" option associated with the "None" value, SQL Assistant will not look for database names in the database catalog so no database names will appear in SQL Assistant popups.

Tip: You can define your own database catalog queries in the DB Queries page and then associate them with SQL Assistance types. See the Customizing Database Catalog Queries topic for more information.

7. The new SQL Assistance Type can be then associated with a specific development environment or a standalone editor. Use the following steps to accomplish this:

8. Click the **Targets** tab page on the **SQL Assistant - Options** dialog.
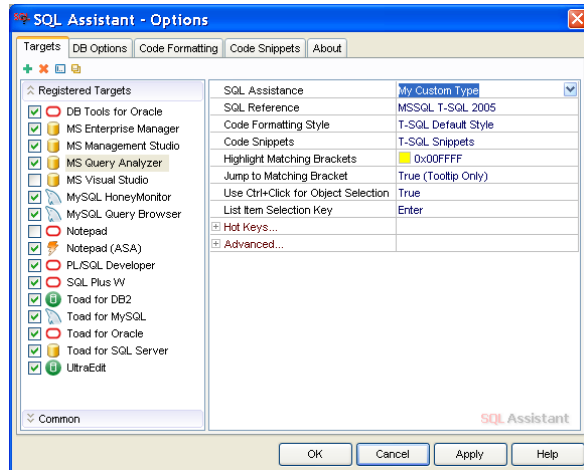
9. On the left side of the screen, select the target editor for which you want to use the new custom configuration. Then for the **SQL Assistance** option on the right side of the screen, select a custom type (such as the one you created earlier in this procedure) from the **SQL Assistance** option drop-down list.

10. Choose the type and version of **SQL Reference** you want to associate with this target and also the associated type of **SQL Snippets**. Choose "None" as a value for items you do not want to use.

11. Click the OK button to save all changes and close the Options dialog.

Customizing Database Catalog Queries

SQL Assistant uses several types of pre-defined database queries that are tuned for working with different versions of Oracle and SQL Server databases. If necessary, you can modify these queries and tune them for the specific version of the database you are working with. You can also enter new queries and assign them to your own custom SQL Assistance configuration.

To modify existing queries for reading database catalog information:

1. Double-click the SQL Assistant icon in the Windows System Tray. The **SQL Assistant - Options** dialog will appear.

2. Click the **DB Options** tab page.
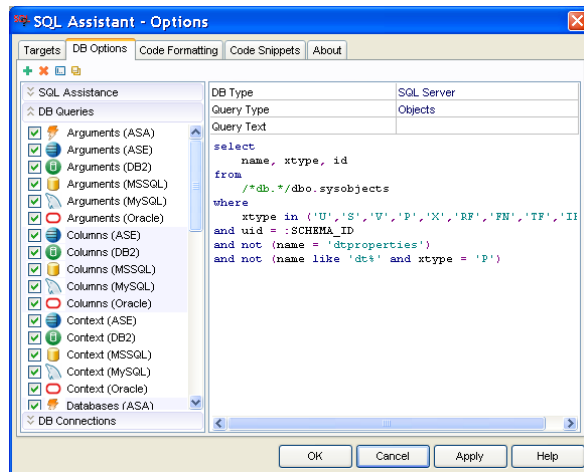
3. Select the **DB Queries** section on the left-side of the Options screen.

4. Select the catalog query you want to modify. The query text will appear on the right side of the screen.

5. Change the query type an database target type as needed, and make any desired edits to the query text.

6. If you want to modify other queries, repeat steps 4 and 5 for each query you want to change. When done, click the OK button to save your changes and close the Options dialog.

To create a new query for reading database catalog information:

Use steps 1 to 6 as described above. In step 4 instead of selecting an existing query, click the Plus Sign icon ![plus] in the top left corner of the **DB Options** tab page to add a new query.

To quickly create a new query from an existing query:

Use steps 1 to 6 as described above. In step 4, select the existing query you want to copy and modify. Click the Copy Sign icon ![copy] in the top left corner of the **DB Options** tab page to create a copy of the query. Enter a name for the new query and edit as necessary.

To delete an existing query:

Use steps 1 to 6 as described above. In step 4, instead of editing query text, click the Delete icon ![delete] in the top left corner of the **DB Options** tab page to delete the selected query.

![tip icon] **Tip:** You can associate your custom catalog queries with both predefined and custom SQL Assistance types. This feature provides you with nearly complete control over SQL Assistant behavior and appearance. See the Customizing SQL Assistance Types topic for more information.

Two Queries for Retrieving Table Column Information

The default SQL Assistant configuration includes two versions of database catalog queries for each database type that are used for describing database table and view columns: The returned information is displayed in various SQL Assistant popups and is also used by the Performance Analyzer. For example, for SQL Server systems the queries are

- **Columns (MSSQL)** – this query returns basic column information

- **Columns (MSSQL) + Keys** - this query returns extended column information including column indexes and constraints but it is more performance expensive and may make SQL Assistant slow when working with very rage database systems.

Two versions of the "Columns" queries are provided for all other database systems. You can choose which query to use with your database system. To customize the selection:

1. Double-click the SQL Assistant icon in the Windows System Tray. The **SQL Assistant - Options** dialog will appear.

2. Click the **DB Options** tab page.

3. Select the **SQL Assistance** type for your database system and editor.

4. Expand the **DB Queries** section on the right side.

5. In the **Query "Columns"** option select the version of the 'Columns" query to use for your environment. Alternatively, to switch the query version, you can change the **Show Keys and Indexed Columns** option with simple Yes/No value.
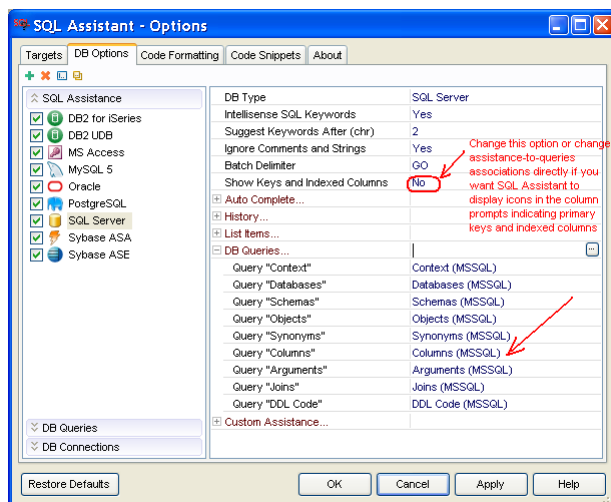
Before change   **After change**
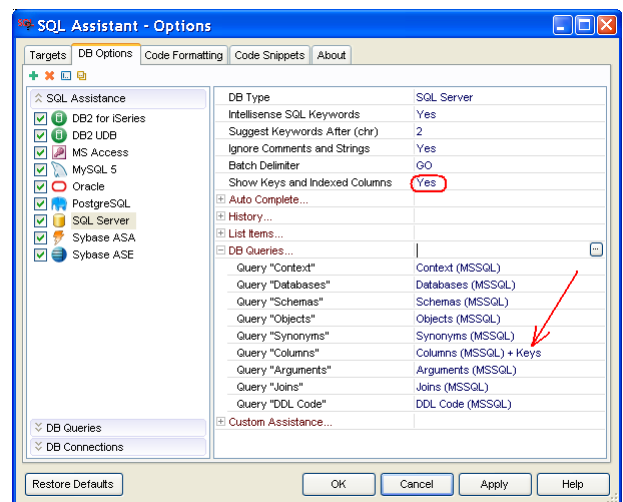
See the Enabling Display of Key Columns and Indexed Columns topic in CHAPTER 3 for more information. This topic also explains why the simple version of the "Columns" query has been chosen as a default pre-configured option.

Special Macro Variables Allowed in Database Catalog Queries

A set of special macro variables can be used the catalog queries to chain their results. This set of special macro-variables cannot be used anywhere else. The available macro variables are database type specific and described in the following table.

Variable	Meaning
DB_NAME	The database name for the query context.
SCHEMA_NAME	The schema name for the query context.
SCHEMA_ID	The unique schema id if such is supported by the database type.
OBJECT_NAME	The schema object name for the query context
OBJECT_ID	The unique schema object id if such is supported by the database type.
OBJECT_TYPE	The type of the object for the query context.
PACKAGE_NAME	The Oracle package name for the query context

Tip: The special macro variables can be used in two different formats

- As text substitution variables using $macro$ notation, for example,
  `SELECT ... FROM [$DB_NAME$].sys.objects WHERE ...`

- As query bind variables using colon notation :macro, for example,
  `SELECT ... FROM syscat.tables WHERE  tabschema = :SCHEMA_NAME AND ...`

Using Advanced Filtering for Fast Database Catalog Data Access

The previous topic describes how you can manage and customize database catalog queries. You can use the same technique for tuning database catalog query performance and filtering the returned results. This is a very handy feature when you work with very large databases running various ERP applications with tens of thousands of objects, columns, procedures and so on. To improve the performance and limit query results only to specific object types or schemas, you can use the following techniques:

- Edit a slow catalog query and add the required performance hints. For example, if your Oracle database server is tuned for cost-based optimization and you want to use rule-based optimization for catalog queries, you can add the /*+ RULE */ hint after each SELECT keyword in text of each catalog query.

- Filter out unused databases and schemas. For example, if you never use databases msdb and model in SQL Server, you can filter them out and add a WHERE clause to the "Databases" catalog query

with text like WHERE name NOT IN ('msdb', 'model')

- Filter in used schemas only. For example, if you only work with schemas SHIPPING and RECEIVING in your Oracle database, you can add a WHERE clause to the "Schemas" catalog query with text like WHERE username IN (' SHIPPING', 'RECEIVING')

Using the techniques described above, you can tune SQL Assistant to fit your requirements and maintain top performance.

Using Object Type Filtering

There are several tools for controlling which objects are displayed in SQL Assistant popups. For example, you can use the **List Items** filter to display only certain types of objects.

To use this filter type:

1. Activate the **DB Options** tab.

2. Select the **SQL Assistant type** whose behavior you want to modify.
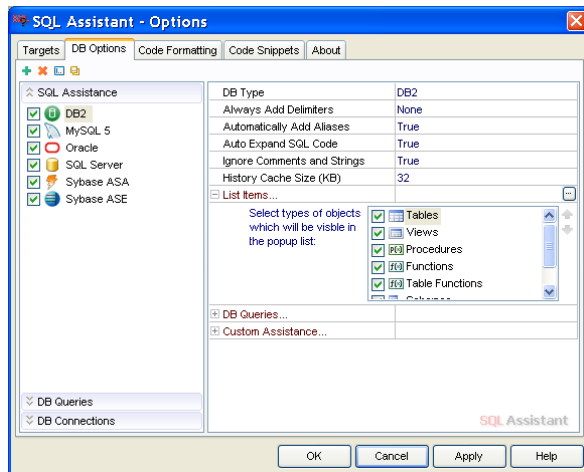
3. Expand the **List Items** option group as shown on the example image.

4. Select the required list item types and press the **Ok** button to apply changes and close the Options dialog.

Note: Different types of items can be configured for different types of supported database systems. If you are working with multiple database system types, you may need to customize each type individually.

Note: Some types of items support 3 states:

- **Enabled** (checkbox is checked) – items of this type are always shown in the popups.

- **Disabled** (checkbox is unchecked) – items of this type are hidden in the popups.

- **Enabled after first letter match** (checkbox is in a third state with the box shaded) – items of this type are initially hidden in the popups but appear when you type first letter matching their names
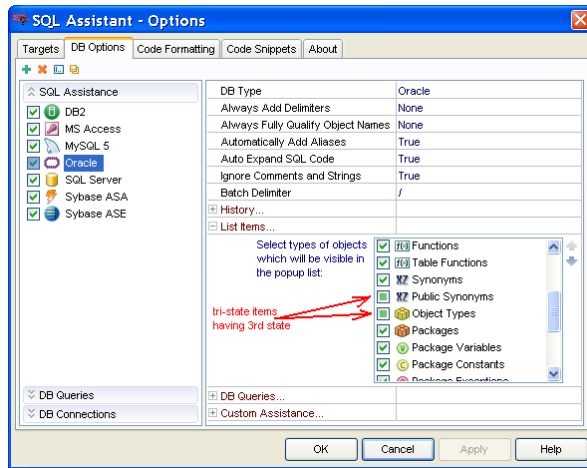
This special state allows you to hide items that would cause the popup list to be too cluttered. For example, Oracle aliases by default are marked with the third state. If they were set to always appear, thousands of alias names would appear at the beginning of most popup lists.

Changing the Order of Objects in Common Object Names Popup

Certain object types can appear in different positions within the Object Names Popup list. To modify the default order:

1. Activate the object type filter as described in the previous topic.

2. in the **List Items** list click the item type whose display position you want to change. Be careful not to remove the checkmark when selecting the item type.

3. Click the Up Arrow or Down Arrow ⬆⬇ icon next to the item type list to move the selected item type up or down.

Tip: Note that not every type can be repositioned. This dependency is a result of internal constraints that cannot be modified. The Up Arrow and Down Arrow icons become enabled only for types whose order in the popup list can be modified.

Changing the Order of Tables in Context Popups

You can customize order in which tables and views are listed in context-based popups appearing within SQL DML statements. Do not confuse this type of popup with the common list of objects and their order. Context

popups are displayed after certain keywords such as WHERE, ORDER BY, GROUP BY, AND, OR, etc. as well as after certain syntax elements such as commas, equal signs, etc. Context popups list only the objects referenced in the same SQL query. The default SQL Assistant behavior for context based popups is to list object names either in the order they are referenced in the code or alphabetically depending on the context. To choose a different order:

1. Open the SQL Assistant **Options** dialog.

2. Activate the **DB Options** tab.

3. Select your database type in the **SQL Assistance** group of options on the left.

4. On the right, expand the **Auto Complete…** option group.

5. Change the **Context Tables Order** option to the desired value and press the **Ok** button to apply changes and close the Options dialog.

The following ordering options are supported:

Default – SQL Assistant decides the best order of objects in the list based on the list context. For example, in the ORDER BY clause, SQL Assistant lists object names in alphabetical order (database and schema names are ignored), while in the JOIN… ON clause, it uses Last to First order.

First to Last – The first object referenced in the current SQL statement appears first in the list, the second object appears second, and so on.

Last to First – The last object referenced in the current SQL statement appears first in the list, the one before the last object appears second, and so on.

Alphabetically – Objects are always listed alphabetically by their names (database names and schema names are ignored).

Alphabetically, Ignore Prefixes – Objects are always listed alphabetically by their names, ignoring name prefixes listed in the "Ignore Name Prefixes" list of recognizable prefixes. Database names and schema names are ignored. See the Customizing Code Auto-completion Options topic in this chapter for instructions on how to customize the list if recognizable prefixes.

Changing the Appearance of JOIN Suggestions

Following a JOIN keyword, SQL Assistant, by default, displays a list of join suggestions based on the available referential constraints for the tables referenced in the query. To improve code entry efficiency, the suggestions are expanded automatically to allow single-click selection of joined columns. The resulting suggestion may look like the following example:
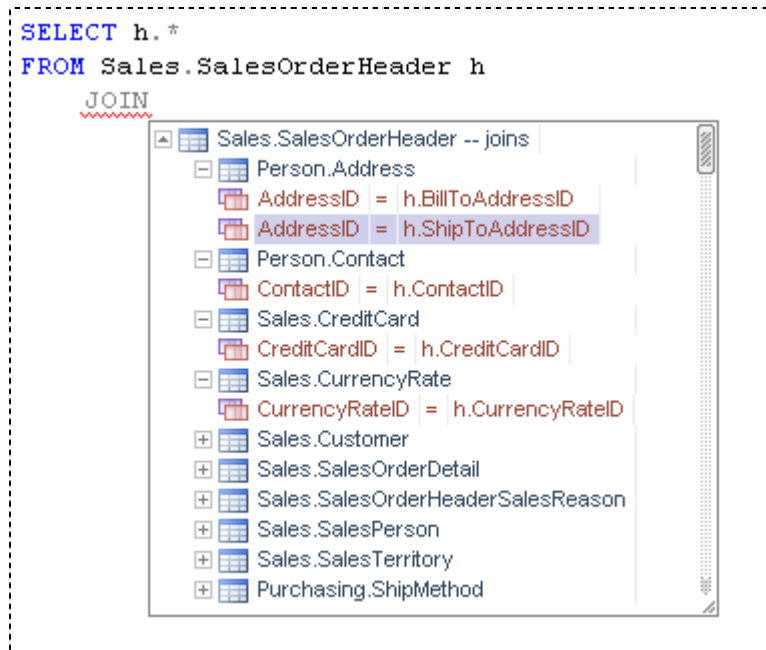
To shorten the list, the column-level is auto-expanded for the first 4 tables only. You can use keyboard navigation keys and/or the mouse to expand column-level in the remaining tables as needed and to select columns for the join. .

Tip: Different code is inserted into the editor depending on which type of item you select in the JOIN suggestions. The most efficient method requiring less input on your part is to select specific columns to be joined and letting SQL Assistant generate the complete JOIN clause, including the ON part containing column matches. For more information, see the Example 2: Building complete SELECT starting with joins and Using JOIN Clause Completion Features topics in CHAPTER 3.

You can control how many tables are auto-expanded using the **Auto expand Join Column-Level** configuration option.

1. Open SQL Assistant's Options dialog.

2. Activate the **DB Options** tab.

3.   Select the **SQL Assistant type** whose behavior you want to modify.
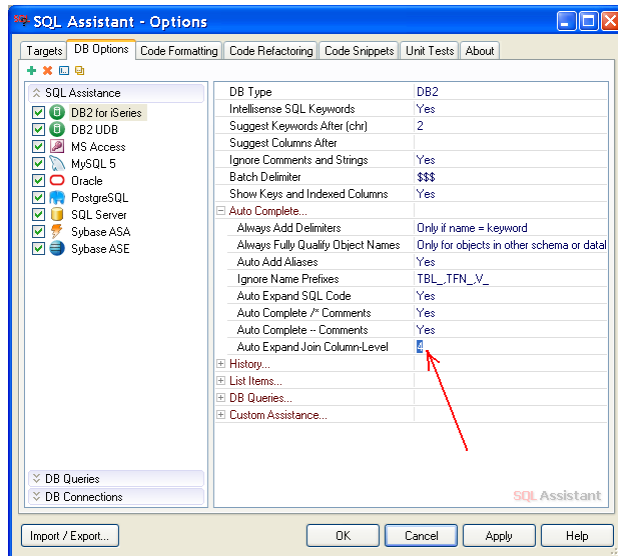
4.   Expand the **Auto-complete** option group as shown on the example image.

5.   Modify the **Auto Expand Join Column-Level** option. The number in this option represents the number of tables that will be automatically auto-expanded. Press the **Ok** button to apply changes and close the Options dialog.

Note: To disable automatic column-level expansion, set this option value to zero.

Note: Different types of items can be configured for different types of supported database systems. If you are working with multiple database system types, you may need to customize each type individually.

Customizing Performance Analysis Options

Performance analysis behavior can be customized in SQL Assistant's Options dialog on the **DB Options** tab.
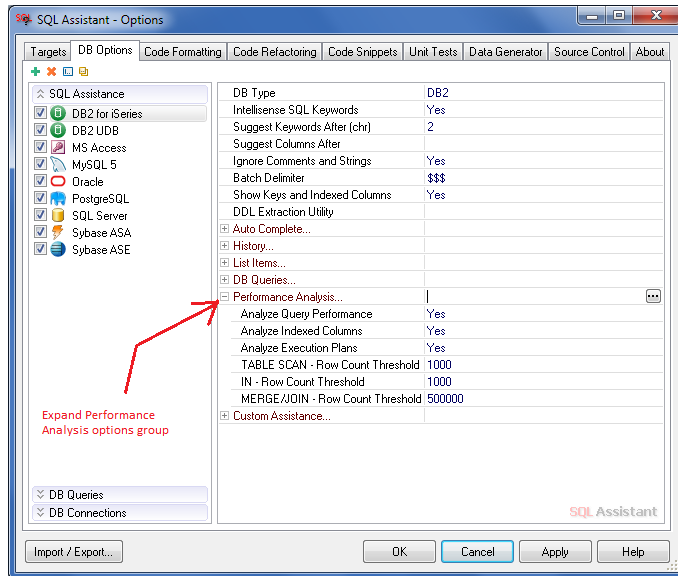
To change performance analysis options, use the following method:

1. Double-click the SQL Assistant icon in the Windows System Tray. The **SQL Assistant - Options** dialog will appear.

2. Click the **DB Options** tab page.

3. In the **SQL Assistance** list of assistance types, select the type whose behavior you want to modify.

4. Expand the **Performance Analysis** option group on right side of the dialog and click the [+] sign in front of the group name to expand the next level.

5. Modify options as required.

6. Click **Ok** button to save changes and close the **SQL Assistant - Options** dialog.

The following performance analysis options can be customized:

Analyze Query Performance – Turns SQL Assistant's code performance analyzer on or off. The default behavior is "**Yes**".

Analyze Indexed Columns –Controls whether SQL Assistant's code performance analyzer checks columns referenced in the code in JOIN and WHERE clauses and checks for missing indexes and / or indexed columns referenced in such a way that their indexes cannot be used. The default value is "**Yes.**"

Analyze Execution Plans – Controls whether SQL Assistant's code performance analyzer collects estimated execution plans for queries in the editor. It also checks them for full table scans performed in large tables and for other poorly performing operations. The default value is "**Yes.**"

TABLE SCAN – Row Count Threshold – Specifies the threshold value for full table scan operations raising performance alerts. The default value is 1000 records. Typically if this value is exceeded, it is an indication that the query has not been optimized query and/or that an index is missing.

IN – Row Count Threshold – Specifies the threshold value for records returned from subqueries used for column comparisons and JOINs. A query may look like the following:

SELECT ... columns here ...
FROM table1
WHERE col1 IN (SELECT lots_of_records FROM table2 WHERE ... some condition ...)

The default value is 1000 records. If this value is exceeded, it is typically an indication that the query has not been optimized.

MERGE JOIN – Row Count Threshold – Specifies the threshold value for JOIN operations resulting in data merges using intermediate tables and raising performance alerts. The default value is 50000 records. Typically if this value is exceeded, it is an indication that the query has not been optimized query and/or that an index is missing.

Managing Database Connections

Use the following steps to manage database connections saved in SQL Assistant configuration files.

To modify an existing connection:

1. Double-click the SQL Assistant icon in the Windows System Tray. The **SQL Assistant - Options** dialog will appear.

2. Click the **DB Options** tab page.
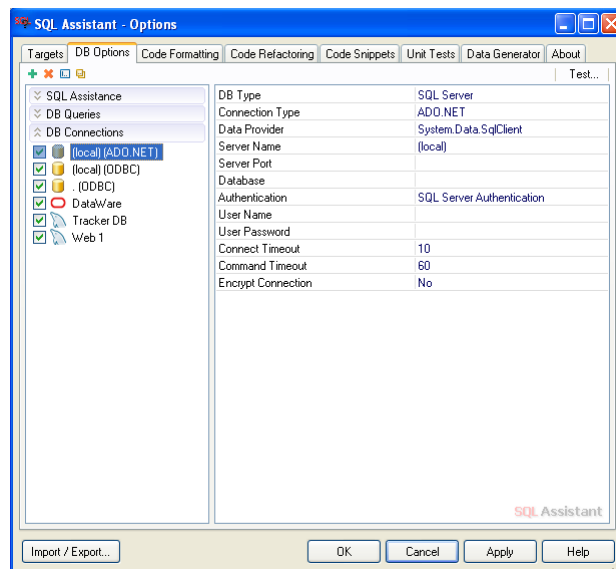
3. Select the **DB Connections** section on the left side of the dialog.

4. Select the connection you want to modify. The connection properties will appear on the right side of the dialog.

5. Edit properties as required.

    Note: "Server," "user," "password," "connection type" and "database name" are optional properties. If they are not specified, they can be specified later in the Connection dialog displayed if SQL Assistant needs to establish a new connection to the database not shared with the target editor.

    Also, note that a different set of properties is available for different connection types. For example, for Oracle connections, you may need to enter the "Oracle TNS name", "connection type" (Normal, SYSDBA or SYOPER) and/or "path to Oracle's OCI.DLL".

    Additional information on supported connection types and their properties is available in CHAPTER 2, Connecting to Your Database.

6. [Optional step] To test that the connection is working properly, click the Test... button in the top right corner of the Options dialog.

7. To modify other connections, repeat steps 4 and 5 for the connections you want to modify. When done, click the OK button to save your changes and close the Options dialog.

To create a new connection:

Use steps 1 to 6 as described above. In step 4, instead of selecting an existing connection, click the Plus Sign icon ✚ in the top left corner of the **DB Options** tab page to add a new connection.

To quickly create a new connection from an existing connection:

Use steps 1 to 3 as described above. In step 4, select an existing connection to copy and modify. Click the Copy icon 🗗 in the top left corner of the **DB Options** tab page to add the new connection, then modify it as needed. If adding SQL Server connection, use the SQL Server instance name as the connection name. If adding an Oracle connection, use the Oracle TNS connection name as you would enter it in SQL*Plus and other Oracle programs.

To delete an existing connection:

Use steps 1 to 3 as described above. After step 3, click the Delete icon ✖ in the top left corner of the **DB Options** tab page.

Customizing Brackets and SQL Code Matching and Navigation

Matching Brackets and Matching Block Delimiters

SQL Assistant supports automatic bracket matching which gives you immediate feedback on misplaced brackets or open-ended SQL code blocks such as BEGIN without END or IF without END IF, and other SQL structures requiring beginning and ending keywords. It also provides quick code navigation methods using matching bracket jumping so you can quickly navigate from the start of a SQL block to the end or visa versa.

The following example screenshot demonstrates the bracket matching feature in action. In this example the cursor is positioned over the **BEGIN** keyword. If there is a matching **END** keyword in the same procedure code, both keywords are highlighted using the chosen bracket highlighting color.

```
ALTER PROCEDURE [dbo].[uspLogError]
    @ErrorLogID [int] = 0 OUTPUT -- contains the ErrorLogID of the row inserted
AS                              -- by uspLogError in the ErrorLog table
BEGIN
    SET NOCOUNT ON;

    -- Output parameter value of 0 indicates that error
    -- information was not logged
    SET @ErrorLogID = 0;

    BEGIN TRY
        -- [Jump to closing END] ere is no error information to log
        IF ERROR_NUMBER() IS NULL
            RETURN;

        -- Return if inside an uncommittable transaction.
        -- Data insertion/modification is not allowed when
        -- a transaction is in an uncommittable state.
        IF XACT_STATE() = -1
        BEGIN
            PRINT 'Cannot log error since the current transaction is in an uncommittable s
                + 'Rollback the transaction before executing uspLogError in order to succe
            RETURN;
        END

        INSERT [dbo].[ErrorLog]
            (
            [UserName],
            [ErrorNumber],
            [ErrorSeverity],
```

You can change the behavior of this feature by modifying the following options:

- **Highlight Matching Bracket** – Specifies the color used to highlight matching brackets and block delimiters or can be used to turn highlighting completely off. Choose **No** value in the drop-down to disable the matching brackets highlighting feature. Choose **Yes (Select Color…)** value in the drop-down to choose desired highlighting color using the standard color-picker control.

- **Jump to Matching Bracket** – Controls behavior of the mouse-over processing for matching brackets and quick navigation or can be used to turn this feature completely off. The available options are:

    o **No** – Do not display mouse-over hints for matching brackets

    o **Yes (Tooltip Only)** – Show small tooltips when the mouse is rested over a bracket for 2 or more seconds.

- o **Yes (Tooltip & Text)** – Show small tooltips when the mouse is rested over a bracket for 2 or more seconds. The tooltip also contains a hyperlink that can be used to jump to the code line containing the matching opening or closing bracket.

Both options are available in the Targets options group and can be configured differently for different targets. To modify these options:

1. Double-click the SQL Assistant icon in the Windows System Tray. The **SQL Assistant - Options** dialog will appear.

2. Click the **Targets** tab page.
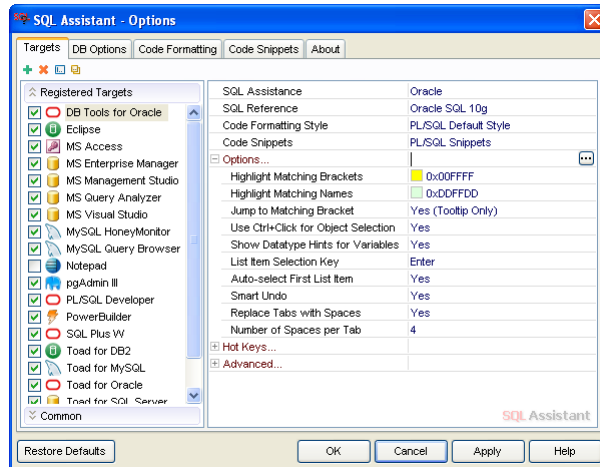
3. On the left side of the screen, select the target whose options you want to modify.

4. On the right side of the screen, change bracket-related options as desired.

5. To modify options for other targets, repeat steps 3 and 4 for each target. When done, click the OK button to save your changes and close the Options dialog.

SQL Assistant supports matching and highlighting of the following types of brackets and block delimiters:

( )
[ ]
{ }
BEGIN…END
BEGIN TRY…END TRY
BEGIN CATCH… END CATCH
CASE…END
IF...END IF
LOOP...END LOOP
FOR...LOOP… END LOOP
WHILE…LOOP...END LOOP
FOR… END FOR
WHILE…END WHILE
REPEAT...END REPEAT

The set of brackets and block delimiters supported by this feature depends on the current type of SQL Assistance selected for the active editor which, in turn, is controlled by the database server type attribute.

Matching Names Highlighting

SQL Assistant supports automatic name matching and highlighting for variables, functions, object names, and so on, which gives you immediate feedback on the use of these names and their references.

The following sample screenshot demonstrates the name matching feature in action. In this sample, the cursor is rested on the **@CheckDate** variable. All occurrences of this variable within the procedure code are highlighted using the selected name highlighting color.

```sql
ALTER PROCEDURE [dbo].[uspGetWhereUsedProductID]
    @StartProductID [int],
    @CheckDate [datetime]
AS
BEGIN
    SET NOCOUNT ON;

    --Use recursive query to generate a multi-level Bill of Material (i.e. all level 1 component
    WITH [BOM_cte]([ProductAssemblyID], [ComponentID], [ComponentDesc], [PerAssemblyQty], [Stand
    AS (
        SELECT b.[ProductAssemblyID], b.[ComponentID], p.[Name], b.[PerAssemblyQty], p.[Standar
        FROM [Production].[BillOfMaterials] b
            INNER JOIN [Production].[Product] p
            ON b.[ProductAssemblyID] = p.[ProductID]
        WHERE b.[ComponentID] = @StartProductID
            AND @CheckDate >= b.[StartDate]
            AND @CheckDate <= ISNULL(b.[EndDate], @CheckDate)
        UNION ALL
        SELECT b.[ProductAssemblyID], b.[ComponentID], p.[Name], b.[PerAssemblyQty], p.[Standar
        FROM [BOM_cte] cte
            INNER JOIN [Production].[BillOfMaterials] b
            ON cte.[ProductAssemblyID] = b.[ComponentID]
            INNER JOIN [Production].[Product] p
            ON b.[ProductAssemblyID] = p.[ProductID]
        WHERE @CheckDate >= b.[StartDate]
            AND @CheckDate <= ISNULL(b.[EndDate], @CheckDate)
        )
    -- Outer select from the CTE
    SELECT b.[ProductAssemblyID], b.[ComponentID], b.[ComponentDesc], SUM(b.[PerAssemblyQty]) AS
    FROM [BOM_cte] b
    GROUP BY b.[ComponentID], b.[ComponentDesc], b.[ProductAssemblyID], b.[BOMLevel], b.[Recurs
    ORDER BY b.[BOMLevel], b.[ProductAssemblyID], b.[ComponentID]
    OPTION (MAXRECURSION 25)
END;
```

You can change the behavior of this feature by modifying the following option:

- **Highlight Matching Names** – Specifies the color used to highlight matching brackets and block delimiters or can be used to turn the highlighting completely off. Choose **No** value in the drop-down to disable the matching names highlighting feature. Choose **Yes (Select Color…)** value in the drop-down to choose desired highlighting color using the standard color-picker control.

This option is available in the Targets options group and can be configured differently for different targets. To modify this option:

1. Double-click the SQL Assistant icon in the Windows System Tray. The **SQL Assistant - Options** dialog will appear.

2. Click the **Targets** tab page.

3. On the left side of the screen, select the target whose options you want to modify.

4. On the right side of the screen, change name match highlighting options as required.

5. To modify options for other targets, repeat steps 3 and 4 for each target. When done, click the OK button to save your changes and close the Options dialog.

Customizing Existing and Creating New Code Snippets

Any changes you make to code snippets on the **Code Snippets** tab page apply only to the SQL dialect selected in the SQL dialects list in the top left corner of the tab page. Before you change a code snippet, make sure to select the SQL dialect you want to modify.

To modify an existing code snippets:

1. Double-click the SQL Assistant icon in the Windows System Tray. The **SQL Assistant - Options** dialog will appear.

2. Click the **Code Snippets** tab page.

3. On the left side of the screen, select the SQL dialect for which you want to modify the snippet.
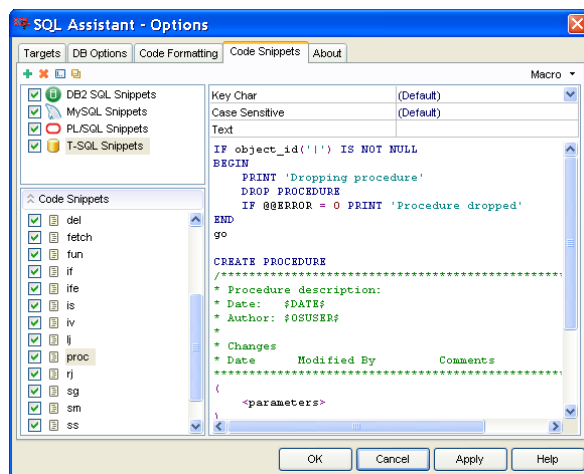
4. Select name of the snippet you want to modify. The snippet code will appear on the right side of the Options dialog.

5. Change snippet options and edit the snippet text as needed.

6. To modify other code snippets, repeat steps 4 and 5 for each snippet. When done, click the OK button to save your changes and close the Options dialog.

To create a new snippet:

Use steps 1 to 6 as described above. In step 4, instead of selecting an existing snippet, click the Plus Sign icon ✚ in the top left corner of the **Code Snippets** tab page to add a new snippet.

To create a new snippet from an existing snippet:

Use steps 1 to 6 as described above. In step 4, select an existing snippet Click the Copy Sign icon in the top left corner of the **Code Snippets** tab page to add a new snippet, modify the snippet copy as desired, and then rename the new snippet with a unique name.

To delete an existing snippet:

Use steps 1 to 6 as described above. In step 5, instead of editing snippet text, click the Delete icon in the top left corner of the **Code Snippets** tab page.

Tip: By default, when you select the **Code Snippets** tab, the first available SQL dialect is selected from the list in the top left corner of the page. If you make changes in code snippets often, it is a good idea to move the most frequently used SQL dialect to in the top of the list. You can use drag-and-drop to reorder the list. You can use drag-and-drop to reorder the list. Your changes will be remembered the next time you open the Options dialog.

Customizing Keywords Used With Keyword Prompts and the Capitalization Feature

Keyword prompts and the keywords capitalization features are part of the SQL Intellisense described in detail in CHAPTER 3, Code Assistants and SQL Intellisense

Important Note: Any changes you make to the keywords and code formatting rules on the **Code Formatting** tab page apply only to the SQL dialect selected in the SQL dialects list in the top left corner of the tab page. Before you make any changes in formatting rules, make sure to select the SQL dialect you want to modify.

To enable automatic keyword reformatting and/or customize keywords list:

1. Double-click the SQL Assistant icon in the Windows System Tray. The **SQL Assistant - Options** dialog will appear.

2. Click the **Code Formatting** tab page.

3. On the left side of the screen, select the code formatting style whose keyword list you want to modify.
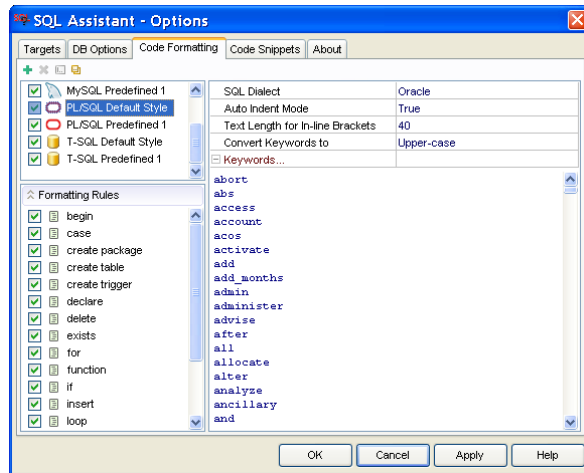
4. Click the empty field next to the **Keywords** option. The small ⊡ button appears on right side of the field. Click this button to expand the list of keywords and the formatting functions applied to these keywords.

   In the **Convert Keywords to** drop-down list, choose the **Uppercase** function to automatically convert keywords to upper case. Choose the **Lowercase** function to automatically convert keywords to lower case. Choose the **Initcaps** function to automatically convert keywords to lower case with first letters in upper case. Choose the **Custom-case** option to have the keywords formatted exactly as they are entered in the keywords list.

   Choosing the **None** option effectively disables the automatic keyword formatting feature.

5. Edit the keyword list as needed.

6. Click the OK button to save all changes and close the Options dialog.

To disable automatic keyword reformatting:

1. Double-click the SQL Assistant icon in the Windows System Tray. The **SQL Assistant - Options** dialog will appear.

2. Click the **Code Formatting** tab page.

3. On the left side of the screen, select the SQL dialect whose keyword list you want to modify.

4. In the **Convert Keywords to** drop-down list, choose **None** to disable automatic keyword formatting.

5. Click the OK button to save all changes and close the Options dialog.

 Tip: By default, when you select the **Code Formatting** tab, the first available SQL dialect is selected from the list in the top left corner of the page. If you make changes in code formatting often, it is a good idea to move the most frequently used SQL dialect to the top of the list. Use drag-and-drop method to rearrange order in which SQL dialect names appear in the list and place the required SQL dialect first. You can use drag-and-drop to reorder the list. Your changes will be remembered the next time you open the Options dialog

Customizing List of Preferred Keywords in Keyword Prompts

SQL Assistant keyword prompts typically consist of two sections separated by a horizontal line. The top-section lists preferred keywords and code structures beginning with the entered letters. The bottom section lists all other keywords beginning with the same letters. If no keywords are listed in the top section, only one section is displayed.

For greater efficiency, the preferred keywords section should include keywords and entire code structures that you type most often. The SQL Assistant default configuration includes a number of predefined choices for preferred keywords and code structures that you can easily customize using the following procedure:

1. Double-click the SQL Assistant icon in the Windows System Tray. The **SQL Assistant - Options** dialog will appear.

2. Click the **Code Formatting** tab page.

3. On the left side of the screen, select the code formatting style whose preferred keyword list you want to modify.
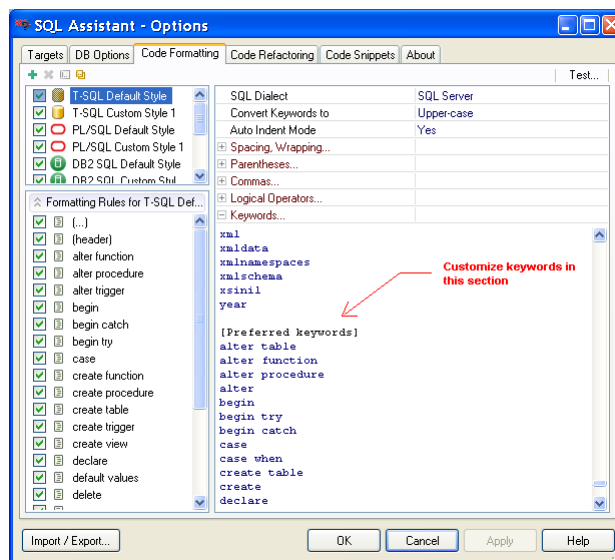
4. On the right side of the screen, click the empty field next to the **Keywords** option. The small button appears on right side of the field. Click this button to expand the list of keywords.

5. Scroll down the list of keywords and locate the **[Preferred Keywords]** section which is close to the bottom of the list.

6. Customize the list of preferred keywords as you see the fit. Note that you may enter separate keywords and entire code structures; for example, SELECT TOP 100 * FROM

    🔖 **Tip**: The display order of keywords in the preferred keywords section in keyword prompts depends on the order of keywords entered in the **[Preferred Keywords]** section in the Options. For example, if you enter CREATE TABLE, CREATE PROCEDURE, and CREATE INDEX in the order specified in this example, you will see them in the same order when you type CR letters in the editor. This example assumes use of SQL Assistant default settings, in which keyword prompts are enabled and configured to appear after first 2 characters are typed.

    🔖 **Tip**: Preferred keywords and code structures specified in the **[Preferred Keywords]** section do not need to be listed in the list of keywords for code formatting. But in order for your preferred keywords to appear in keyword prompts, at least one keyword must be defined in the main section with the same starting letters.

7. Click the OK button to save all changes and close the Options dialog.

Customizing Symbols Triggering Column Name Popups

SQL Assistant automatically displays context-sensitive column name popups. The popups are displayed in the following cases:

- After certain keywords such as WHERE, GROUP BY, ORDER BY
- In JOIN sections of SQL statements
- After dots following table names or table name aliases
- After commas in parts of SQL statement where column names are expected
- After AND and OR logical operations

You can use the **Suggest Columns After** option to specify additional symbols and keywords after which column name popups should be displayed automatically.

To customize options for column name popup handling:

1. Double-click the SQL Assistant icon in the Windows System Tray. The **SQL Assistant - Options** dialog will appear.

2. Click the **DB Options** tab page.

3. On the left side of the screen, select the required **SQL Assistance** style whose popup display options you want to modify.

4. Change value of the **Suggest Columns After** option as needed. You must enter additional symbols and/or keywords as a comma separated list, for example:

    <,>,<>,!=,BETWEEN

5. Click the OK button to save all changes and close the Options dialog.

Customizing Handling of Object and Column Names in Case of Keyword Name Conflicts

By default, SQL Assistant uses the "**Only if name = keyword**" code auto-completion rule when pasting object and column names into the code editor. This rule instructs SQL Assistant to enclose pasted names in brackets (or quotes) only if they match SQL keywords. This rule is provided as a way to avoid creating code syntax errors. However, database engines may allow certain names to be used without brackets even if they match predefined keywords. For example, frequently used column names like ID and NAME are allowed and may be coded without any special treatment in most databases. This kind of special name can be excluded from the "**Only if name = keyword**" rule by switching to the "**Only if name = keyword (limited)**" rule and customizing the list of exception names.

To change the default rule for adding name delimiters:

1. Double-click the SQL Assistant icon in the Windows System Tray. The **SQL Assistant - Options** dialog will appear.

2. Click the **DB Options** tab page.

3. On the left side of the screen, select the **SQL Assistance** style whose name/keywords treatment you want to modify.

4. On the right side of the screen, click the [+] sign in front of the **Auto-Complete** section to expand that section.

5. Modify the value of the **Always Add Delimiters** option and set it to **Only if name = keyword (limited).**

6. Click the OK button to save all changes and close the Options dialog.

To customize the list of exception names in SQL Assistant Options:
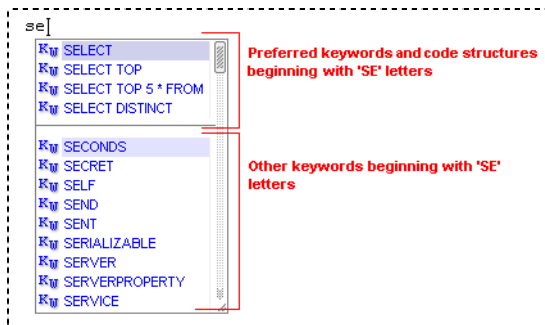
1. Double-click the SQL Assistant icon in the Windows System Tray. The **SQL Assistant - Options** dialog will appear.

2. Click the **Code Formatting** tab page.

3. On the left side of the screen, select the code formatting style whose keyword name exclusions list you want to modify.

4. On the right side of the screen, click the empty field next to the **Keywords** option. The small button appears on right side of the field. Click this button to expand the list of keywords.

5. Scroll down the list of keywords and locate the **[Do not add delimiters]** section located near the bottom of the list.

6. Customize the list of names you want to exclude.

7. Click the OK button to save all changes and close the Options dialog.

Customizing Code Auto-completion Options

Code auto-completion behavior can be customized in SQL Assistant's Options dialog on the **DB Options** tab.
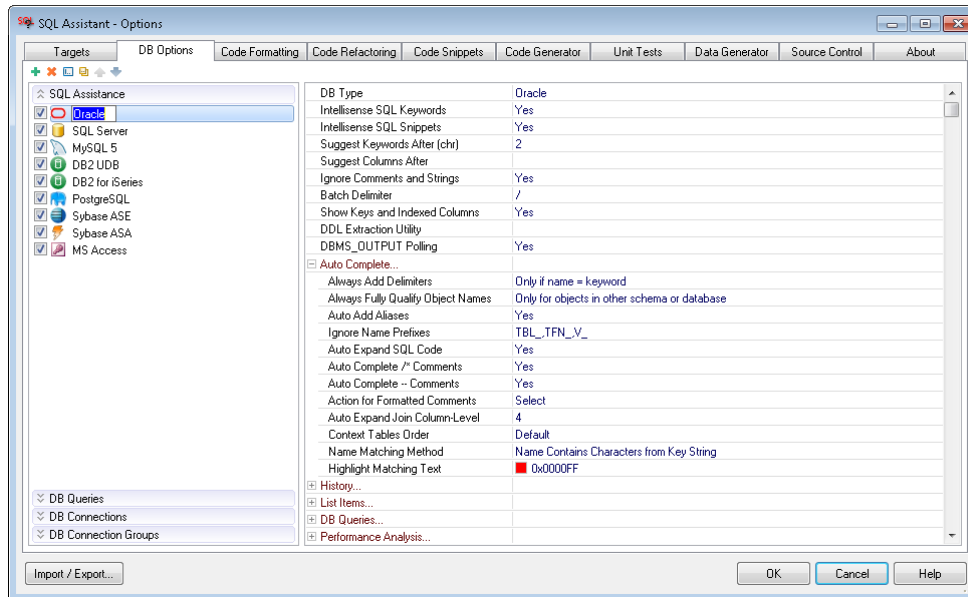
To change auto-completion behavior options, use the following method:

1. Double-click the SQL Assistant icon in the Windows System Tray. The **SQL Assistant - Options** dialog will appear.

2. Click the **DB Options** tab page.

3. In the **SQL Assistance** list of assistance types, select the type whose behavior you want to modify.

4. On right side of the dialog, expand the **Auto Complete** option group, then click the [+] sign in front of the group name to expand the next lower level.

5. Modify options as required.

6. Click the **Ok** button to save changes and close the **SQL Assistant - Options** dialog.

The following auto-completion features can be customized:

Always Add Delimiters – Controls how SQL Assistant inserts object and column names into the code. Options are:

- **Only if name = keyword** - Causes names to be enclosed in brackets (or quotes) only if they match SQL keywords

- **Only if name = keyword (limited)** - the same as the "Only if name = keyword" rule except that certain keyword names commonly recognized by databases, such as ID and NAME, are not automatically delimited. You can customize the list of exclusions. This list is maintained in the **Keywords** section in

SQL Assistant Options, on the **Code Formatting** tab. See the Customizing Handling of Object and Column Names Matching Keywords topic for more details.

- **Always** - Delimiters are added to all inserted names

- **Never** - Delimiters are never added to inserted names

Tip: The state of the **Always Add Delimiters** option has no impact on how SQL Assistant inserts names containing spaces and special symbols. These names are always enclosed in delimiters.

Always Fully Qualify Object Names – Controls how SQL Assistant inserts object names into the code. The following choices are supported:

Only for objects in another schema or database – if the selected name is for an object located in a different database or in a schema different from the current user schema, the name is automatically expanded to include the database and schema parts using standard dot notation.

With schema name – if the selected name is for an object located in the current database, the name is always automatically expanded to include schema part. If the selected name is for an object located in a different database, the name is automatically expanded to include the database and schema parts.

With schema name (functions only) – this option is specific to SQL Server requirements for user-defined function calls to include schema names. If the selected name is for a user-defined function located in the current database, the name is always automatically expanded to include schema part. If the selected name is for a user-defined function located in a different database, the name is automatically expanded to include the database and schema parts.

With database and schema names –the selected name is always automatically expanded to include database and schema parts, even if the referenced object is located in the current database schema.

Auto Add Aliases – Controls the generation of automatic aliases after table and view names and inserting them before referenced column names.

Example code containing automatically added aliases:	Example code without aliases:

<pre>SELECT
    e.EmployeeID,
    e.LastName,
    e.FirstName,
    e.Title,
    e.TitleOfCourtesy,
    e.BirthDate,
    e.HireDate,
    e.[Address],
    e.City,
    e.Region,
    e.PostalCode,
    e.Country,
    e.HomePhone,
    e.Extension,
    e.Photo,
    e.Notes,
    e.ReportsTo,
    e.PhotoPath
FROM
    Employees AS e</pre>
	<pre>SELECT
	    EmployeeID,
	    LastName,
	    FirstName,
	    Title,
	    TitleOfCourtesy,
	    BirthDate,
	    HireDate,
	    [Address],
	    City,
	    Region,
	    PostalCode,
	    Country,
	    HomePhone,
	    Extension,
	    Photo,
	    Notes,
	    ReportsTo,
	    PhotoPath
	FROM
	    Employees</pre>

Options are:

- **No** – Do not add aliases

- **Yes (without AS keyword)** – automatically add aliases after table names, as in the following example:
  ```
  FROM Employees e
  ```

- **Yes (with AS keyword)** – automatically add aliases after table names, as in the following example:
  ```
  FROM Employees AS e
  ```

Ignore Name Prefixes – Specifies name prefixes that SQL Assistant should ignore when calculating automatic aliases. This option is used only if the **Auto Add Aliases** options is enabled. Name prefixes to ignore must be entered as a comma separated list; for example:

```
TBL_,TFN_,V_,T_,MV_
```

Aliases Character Case – Specifies character case for generated aliases

`Upper Case` – generates aliases consisting of upepr case letters only, for example: "PO"

`Lower Case` (this is the default) - generates aliases consisting of upepr case letters only, for example: "po"

`Title Case` – Generates alises with firts letter in upper case and the rest in lower case, for example, "Po"

Custom Aliases – List of custom preferred aliases for specific database objects

`Regular Names` – this method allows you to enter a list of tables and the aliases that you want to use for them in a simple tabular format. Enter the data as a tab separated list of names and alises, for example:

```
SALES.Orders              SO
Purchasing.orders         PO
FinancialStatements.tblBalanceSheetQuarterly    qbs
HumanResources.Employee                  HR_emp
...
Address                   adrs
```

Tip: Each table name may or may not be fully qialfied. If it's not fully qualified, then the alias can be used in any database context .
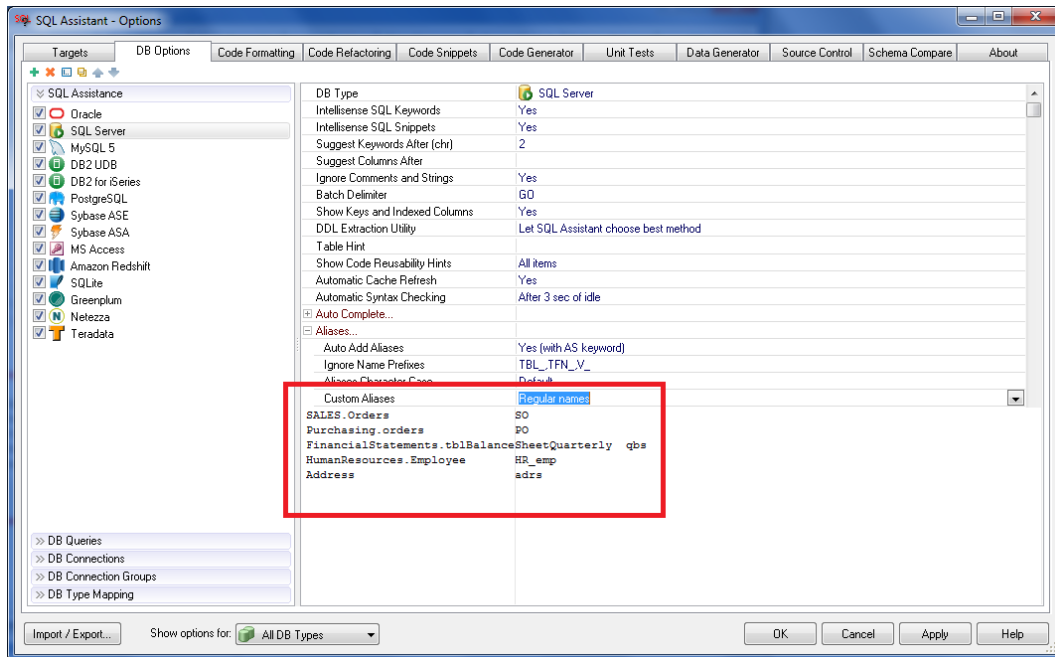
`Regular Expressions –` this method allows you to specify a regular expression that will be used to calculate automatic aliases. For example, the following expression can be used to compose aliases from first letter in the schema name and the first letter in the object name

(\w).*?\.(\w).* \1\2

For *HumanResources.Employee* table name as entered, the resulting alias would be " he"

Auto Expand SQL Code – Controls SQL Assistant behavior and the performed activities for object names selected in <u>object names popup</u> displayed after the first DML statement keyword. For example, if this option is enabled and table Employee is selected in the popup following the SELECT keyword, SQL Assistant generates the complete expanded SELECT statement including all required keywords, columns names, aliases, and so on, as in the following example:

```
SELECT
    e.EmployeeID,
    e.LastName,
    e.FirstName,
    e.Title,
    e.TitleOfCourtesy,
    e.BirthDate,
    e.HireDate,
    e.[Address],
    e.City,
    e.Region,
    e.PostalCode,
    e.Country,
    e.HomePhone,
    e.Extension,
    e.Photo,
    e.Notes,
    e.ReportsTo,
    e.PhotoPath
FROM
    Employees AS e
```

If **Auto Expand SQL Code** is disabled, only the object name is inserted after the keyword, as in the following example:

```
SELECT Employees
```

Auto-complete / * */ Comments – Controls automatic expansion of multi-line comments. If enabled, pressing the Enter key after the opening /* comment tag, causes SQL Assistant to automatically generate the entire comment block, adding the closing tag */ and additional asterisk symbols along the left comment edge. In the simplest example, an empty comment will appear as in the following example: The edit caret is placed in the beginning of middle line.

```
/**
 * |
 **/
```

Tips: All asterisk symbols you enter before pressing the Enter key are automatically copied to the last line, for example:

```
/*******************
 * |
 *******************/
```

Pressing the Enter key inside the comment block causes SQL Assistant to automatically insert a new line and add the left edge asterisk. If the Enter key is pressed in the middle of comment text, the text is split at that point, and the trailing part is moved to the line following the asterisk.

Before Enter key is pressed (black vertical bar denotes cursor position):	Before Enter key is pressed (black vertical bar denotes cursor position):
```/*********************************** * test line having |several words ***********************************/```	```/*********************************** * test line having * |several words ***********************************/```

Auto-complete – – Comments – Controls the automatic continuation of single-line comments. If enabled, pressing the Enter key in the middle of a comment line beginning with the double dash (– –) comment tag, causes SQL Assistant to automatically split the text at that point. The trailing part is moved to the line following the following line where it begins with the double dash (– –) comment tag.

Before Enter key is pressed (black vertical bar denotes cursor position):	Before Enter key is pressed (black vertical bar denotes cursor position):
```-- test line having |several words```	```-- test line having -- |several words```

Item Name Matching Methods - Specifies the method that governs the way SQL Intellisense name matching responds to names you type in a SQL editor. Note that text matching is **case insensitive**. The supported methods are:

- **Name Starts with Key String** – the name must begin with the text you typed into the editor. For example, if you typed "Order", names like "OrderHeader", "OrderDetail" would be listed in the context popup.

- **Name Contains Key String, Order Alphabetically** – the name must contain the text you typed into the editor. The text string could be anywhere within the name. For example, if you typed "Order", names like "OrderHeader", "OrderDetail", "fnOrderData", "prDeleteOrder" would be listed in the

context popup. The matching names are filtered and then sorted alphabetically. In the example here, the resulting order is going to be "fnOrderData", "prDeleteOrder", "OrderDetail", "OrderHeader".

- **Name Contains Key String, Order by Best Match** – the name must contain the text you typed into the editor. The text string could be anywhere within the name. For example, if you typed "Order", names like "OrderHeader", "OrderDetail", "fnOrderData", "prDeleteOrder" would be listed in the context popup. The matching names are filtered and then sorted in order of the best match. In the example here, the resulting order is going to be "OrderDetail", "OrderHeader", "fnOrderData", "prDeleteOrder".

- **Name Contains Characters from Key String, Order Alphabetically** –the name must contain the characters from the text you typed into the editor. The characters appear in the same order but do not need to be sequential. For example, if you typed "Ordr", names like "OrderHeader", "OrderDetail", "fnOrderData", "prDeleteOrder" , "vwOrdrWklyReport", as well as "vwOrdYearlyReport" will be listed in the context popup. The matching names are filtered and then sorted alphabetically before they are displayed in the popup.

- **Name Contains Characters from Key String, by Best Match** –the name must contain the characters from the text you typed into the editor. The characters appear in the same order but do not need to be sequential. For example, if you typed "Ordr", names like "OrderHeader", "OrderDetail", "fnOrderData", "prDeleteOrder" , "vwOrdrWklyReport", as well as "vwOrdYearlyReport" will be listed in the context popup.. The matching names are filtered and then sorted in order of the best match before they are displayed in the popup.

Highlight Matching Text Color – Controls the color SQL Assistant uses to highlight matching text in the popups. By default, the color red is used. Choose **No** value in the drop-down to disable the highlighting feature. Choose **Yes (Select Color…)** value in the drop-down to choose desired highlighting color using the standard color-picker control.

Customizing Code Formatting Patterns

Code formatting patterns can be built using these elements:

- Keywords
- Ellipses (triple dots indicating text between keywords)
- Parentheses and commas indicating code flow
- White spaces consisting of space and tab characters
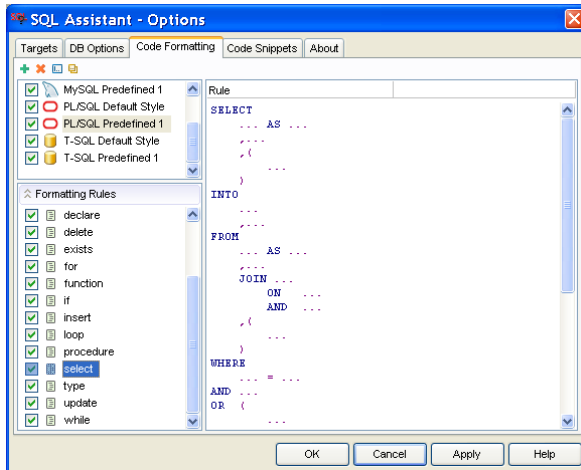
It is important to understand that code-formatting patterns may include more keywords than the actual SQL statement formatted using the pattern. Keywords included in the pattern that do not appear in the SQL statement are ignored during code processing and do not affect the results.

Spacing and positions of commas, parentheses and logical operators are very important. To get an idea how this affects code formatting, consider the following T-SQL DECLARE statement with multiple variables:

DECLARE @var1 int, @var2 datetime, @var3 char(5), @var4 int

If you specify the formatting pattern for DECLARE as

DECLARE   …,
*   …*

The code formatter will produce the following code:

DECLARE   @var1 int,
*   @var2 datetime,*
*   @var3 char(5),*
*   @var4 int*

However, if you specify the formatting pattern for DECLARE as the following (note the position of the comma character):

DECLARE   …
*   ,…*

The code formatter will produce the following code

DECLARE   @var1 int
*   , @var2 datetime*
*   , @var3 char(5)*
*   , @var4 int*

Similarly, if a formatting pattern for the WHERE clause in a SELECT statement is defined as:

WHERE …
    AND …

The code formatter will produce the following code

WHERE col_a = @col_a
   AND col_b = @col_b

However, if you specify the following formatting pattern for the WHERE clause (note the position of the AND logical operator):

WHERE … AND
   …

The code formatter will produce the following code:

WHERE col_a = @col_a AND
   col_b = @col_b

Tip: The best way to learn how code formatting works is to modify the default options and to apply code formatting to your SQL file so you can see the results of your changes.

Customizing Bookmark Handling Options

Bookmark activation and behavior can be configured individually for each target editor on the **Targets** tab in **SQL Assistant - Options** dialog.

To change bookmark handling options, use the following method:

1. Double-click SQL Assistant icon in the Windows System Tray. The **SQL Assistant - Options** dialog will appear.

2. Click the **Targets** tab page.

3. In the targets list select the target whose options you want to modify.

4. On right side of the dialog, expand **Bookmarks** option group, click the [+] sign in front of the group name to expand the next level.

5. Modify options as required.

6. Click **Ok** button to save changes and close the **SQL Assistant - Options** dialog.

The following options are available for customizing bookmark handling:

- **Enable Bookmarks** - Enables the Visual Bookmarks feature in the selected target editor. If this option is disabled, all other bookmark-related options are disabled as well.

- **Show Docked Icons** – Enables the use of small docked icons along the edges (top or bottom) of the editor window. If enabled, docked icons are displayed for bookmarks located outside of the visible area. If

disabled, bookmarks are highlighted using colored lines and no bookmark icons are used.

- **Show Screen Shots** – Enables capturing and caching screenshots. if enabled, editor screenshots captured near bookmark locations are displayed on mouse-over event for docked bookmark icons.

- **Highlight Lines –** Enables editor text highlighting in bookmarked lines. If enabled, bookmarked lines are indicated by colored highlighting. Highlight color is unique to each bookmark.

- **Allow One-click Create –** If enabled, adding new bookmarks can be done with a single mouse click when the mouse is positioned near the right edge of edit window. If this option is disabled, bookmarks may still be added using SQL Assistant menus.

- **Allow One-click Remove –** If enabled, removing bookmark can be done with a single mouse click. If this option is disabled, bookmarks may still be removed using SQL Assistant menus.

Customizing Error Handling Options

SQL Assistant supports 4 error-handling modes. To choose which mode is right for you, open SQL Assistant options and activate the **Targets** page. On the left side of the Targets page, expand the **Common** section and then, on the right side of the page, expand the **Error Handling** option group.

The following error handling modes and options are supported:

- **Show system tray notifications –** In this mode, if an error occurs, SQL Assistant displays a small notification message in the system tray area above the SQL Assistant icon. The message contains the first 60 characters of the error message text. Click on the notification message to open the full Error Message dialog. The Error Message dialog describes the error and gives you the option of reporting the error to SoftTree Technologies technical support. If you choose to report the error, you will be given a tracking number for the error. Other error handling activities in this mode are the same as those described below for the **Display error messages**.

- **Display error messages –** In this mode, if an error occurs, SQL Assistant displays the Error Message dialog describing the error and asks if you want to send an error report to SoftTree Technologies technical support. If you click the **Send** button, the error message, SQL Assistant version, name and version of the target editor, type and version of the current database system, as well as your Windows system type and version will be sent to SoftTree Technologies technical support.

  The error dialog contains two optional edit fields you can use to describe the steps taken before the error occurred and, optionally, your contact email. **Please note that the error data contains no confidential or personal information unless you explicitly enter such data yourself.** Submitted error reports are processed by an online support system, and a unique tracking number is assigned to each report. The tracking number can be used later to monitor the status of the issue. By specifying a valid contact email, you are giving SoftTree Technologies permission to contact you for additional information about the error if necessary. The email address you provide can also be used to notify you of status changes. For example, if a bug in SQL Assistant program code is causing an error, SoftTree Technologies may contact you to notify you that the bug has been fixed. **The contact email will be used for communications regarding the reported error. It will not be used for any other purposes**.

  Tip: The error dialog is resizable. If you cannot see the entire error text displayed in the top white area of the dialog, drag the edge of the dialog to make it bigger.

- **Ignore all errors –** In this mode, SQL Assistant silently ignores all errors that might occur.

- **Silently report errors to support –** This is similar to the first option, except that errors are automatically reported to SoftTree Technologies technical support, and SQL Assistant does not ask you to describe the steps. This option may be less disruptive to code processing while still helping to improve future SQL Assistant versions; however, you do not get a chance to enter related error information or comments. Selecting this option diminishes the value of reported errors and makes it more difficult to analyze and troubleshoot the cause of the problem.

In the Options dialog SQL Assistant also allows you to specify a default contact email for the error reporting service to use in the **Display error messages** and the **Silently report errors to support** error-handling modes. Use **Your email address (optional)** option to specify your default contact email. Note that this is an optional value and it can be left blank.

Customizing Data Type Mapping for Data Transformations

The data type mapping for cross databases data conversions are used by Data Factory, Data Transfer, and a few other tools and features supported by SQL Assistant. The mapping can be customized in SQL Assistant options.

To review and customize data type mappings:

1. Open the **Options** dialog.

2. Select the **DB Options** tab.

3. Expand the **Data Type Mapping** section in the left-bottom corner.

4. In the database type list select the type of the database whose data type mapping you want to customize.

5. The data grid with the data type conversion matrix will appear in the main area of the Options dialog. You can customize the values as a required.

Tips:

- The mapping supports formulas for dynamic translation of data type attributes, for example, NUMERIC(p,s) stands for NUMERIC source data type with precision p and scale s. During data conversion a comparable data type will be used for the destination database with the same precision and scale.

- Depending on the source and destination database type features and limitations, the same data type can be converted differently based on its attributes. The data types with multiple mappings display [+] icon in front of the data type name. Click that icon to expand the data type specific section. Simple expressions like VARCHAR(n..2000) -> VARCHAR2(n) and VARCHAR(2001..n) -> CLOB are used for attribute value ranges. In this example, if the source SQL Server data type is VARCHAR and its length is defined within 1 to 2000 characters, it gets converted to Oracle VARCHAR2 data type with the same length, but if it's 2001 and longer, the conversion requires using Oracle's CLOB data type.

CHAPTER 49, Registering and Configuring Targets for SQL Assistance

Overwiew of Target Editor Registration Modes

SQL Assistant supports two methods for hooking SQL Assistant code into the target editor: **add-ons** and **dynamic monitoring and hooking**. For several pre-configured targets, SQL Assistant provides specially created add-ons that allow full and easy SQL Assistant integration with the target environment. The add-ons must be properly installed and configured with the target environment.

The dynamic application monitoring and hooking method is generic and it can be used with many Windows-based programs, including new and unknown programs. You can register new programs with SQL Assistant and configure SQL Assistant settings for use with the newly registered programs.

See the Managing SQL Assistant Load Methods topic in CHAPTER 48 for more information about attaching SQL Assistant to the target editor environment.

If you experience application conflicts when attempting to use the dynamic application monitoring and hooking method, see the Troubleshooting Application Conflicts topic in this chapter for setting application exceptions.

The following topics describe how to register and unregister SQL Assistant add-ons and new target editors.

Installing and Enabling SQL Assistant Add-ons

SQL Assistant provides a number of add-ons for better integration with popular application development and database management systems such as Visual Studio, Microsoft SQL Server Management Studio, Eclipse, Delphi, Oracle SQL Developer, and other. These add-on files are located in the SQL Assistant installation directory, but they are not necessarily enabled. Do the following to enable an add-on:

1. Double-click the SQL Assistant icon in the Windows System Tray. The **SQL Assistant - Options** dialog will appear.

2. On the **Targets** tab page, on the left side of the screen, select the target for which you want to enable the add-on.

3. On the right side of the screen, expand the **Advanced…** option group.

4. Use the drop-down list or double-click the value in the **Register SQL Assistant Add-on** option to change the value from False to True.

5. Click the OK button to save all changes and close the Options dialog.

6. To verify the registration, close the SQL Assistant system tray application (see the SQL Assistant systray icon topic for details) then restart the target editor. The SQL Assistant menu should appear in

the target, and SQL Assistant should be active in the Query Editor windows.

The SQL Assistant add-on for Eclipse IDE and derivative products such as Easy-Eclipse, and a number of others software products is not installed by default. These products can be installed anywhere on the disk. They typically do not register themselves with the system and SQL Assistant does not know where to find them so a simple manual installation step is required to install the add-on for Eclipse. Do the following to install and enable this add-on:

1. Open Windows Explorer and locate the SQL Assistant installation directory. The default path is
    `C:\Program Files (x86)\SQL Assistant 12`

2. Expand the **Data** subfolder and right-click the `com.softtree.sa.eclipse.jar` file. Right-click the file and select the **Copy** command from the right-click menu.

3. Locate the **plugins** subfolder of your Eclipse installation. If you have Eclipse installed in the root folder of the drive C, the subfolder for plugins typically would be `C:\eclipse\plugins.` Some Eclipse-based products use
    `C:\Documents and Settings\{user name}\workspace\metadata\plugins`
    as the startup folder for their plugins. For the exact plugins subfolder name and location consult your Eclipse documentation

4. Right-click the plugins subfolder and select the **Paste** command.

5. To verify the registration, close the SQL Assistant system tray application (see the SQL Assistant systray icon topic for details) then restart the Eclipse IDE. Click the **Help → About Eclipse Platform** menu to display Eclipse's About dialog box. In this dialog, click the **Plug-in Details…** button. The About Eclipse Platform Plug-ins dialog box will appear. Scroll the list to the bottom. The **SQL Assistant Plug-in** name should appear at the end of the list as demonstrated on the following screenshot.
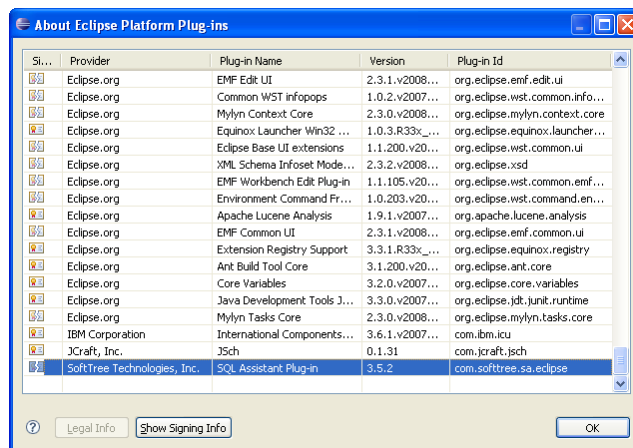
The SQL Assistant add-on for IBM Data Studio also is not installed by default. Like most other Eclipse derivative products, the Data Studio can be installed anywhere on the disk and does not register itself with the system. SQL Assistant does not know where to find this application so a simple manual installation step is required to install the add-on for Data Studio. Do the following to install and enable this add-on:

1. Open Windows Explorer and locate the SQL Assistant installation directory. The default path is
    `C:\Program Files (x86)\SQL Assistant 12`

2. Expand the **Data** subfolder and right-click `com.softtree.sa.datastudio.jar` file. In the

right-click menu, click the Copy command.

3. Locate the **plugins** subfolder of your Data Studio installation. If you have Data Studio installed in the Program Files subfolder of drive C, the subfolder for plugins typically would be `C:\Program Files\IBM\IBM Data Studio\plugins`

4. Right-click the plugins subfolder and click the **Paste** command in the right-click menu.

5. To verify the registration, close the SQL Assistant system tray application (see the SQL Assistant systray icon topic for details) then restart the Data Studio. Click the **Help → About IBM Data Studio** menu to display the Data Studio's About dialog box. In this dialog, click the **Plug-in Details…** button. The About IBM Data Studio Plug-ins dialog box will appear. Sort the list by Plug-in Name. The **SQL Assistant DataStudio Plug-in** name should appear in the list.

Registering New Targets for SQL Assistance

The simplest way to register a new editing target is to use the Ctrl+Shift+F5 hot key. The following example demonstrates how Windows WordPad application can be registered with SQL Assistant:

1. Start WordPad from the Windows Start menu.

2. Click the editor area of the WordPad window, to the set the input focus in the editor. Press the Ctrl+Shift+F5 hot key. The SQL Assistant dialog will appear on the screen as shown in the following screenshot.
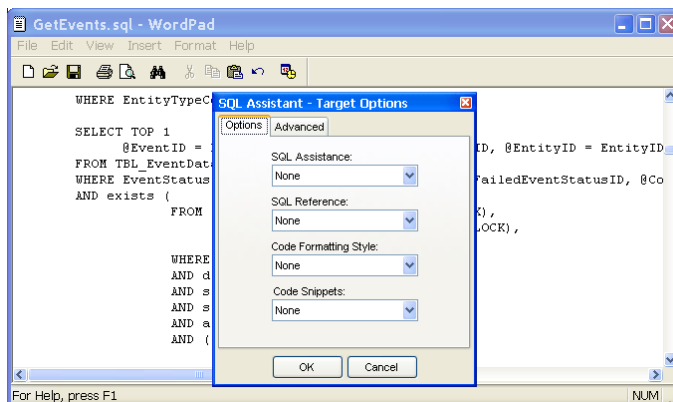
3. Customize the target options.

If you would like to use WordPad as an editor for SQL Server files, choose "SQL Server" in the **SQL Assistance** drop-down. If you want to use code snippet shortcuts with this editor, choose T-SQL in the **Code Snippets** drop-down.

Before:

After:
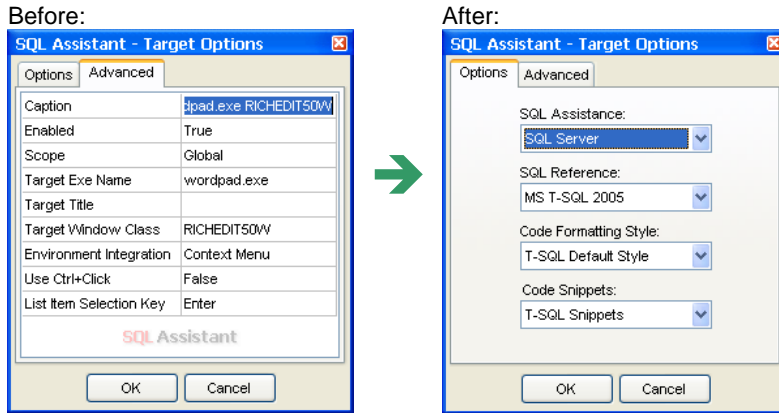
Change **SQL Assistance** option to "SQL Server."

Choose the version of **SQL Reference** you want to use with the new target. For Microsoft SQL Server 2000, choose "MS T-SQL 2000." For Microsoft SQL Server 2000, choose "MS T-SQL 2005".

To use code snippet shortcuts, change the **Code snippets** option to "T-SQL."

Customize the **Advanced** options.

For example, you can change the target name in the **Caption** option to "WordPad" or other descriptive name that will be used to reference this target in SQL Assistant Options.

Leave the default **Enabled** option as "True."

Use "Global" for the **Scope** option if you want to register WordPad permanently and make SQL Assistant attach to current and future WordPad instances. To apply options only to the current instance of the editor, choose either "Current process" or "Current window." For WordPad, both options have the same effect because the WordPad editor does not have a multiple document interface and can edit only one file at a time.

Before:

After:
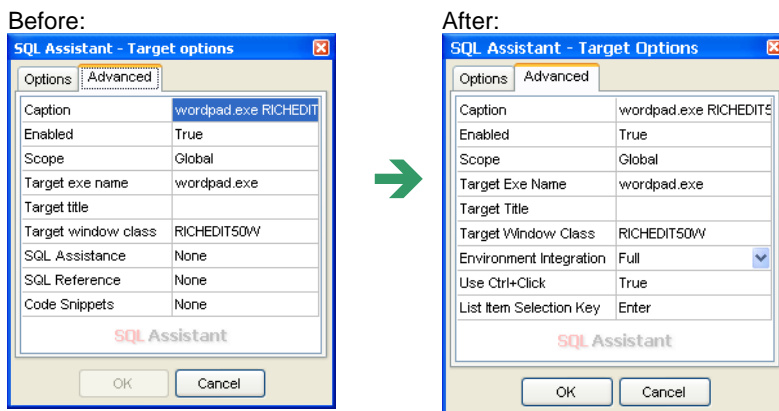
Don't change the **Target exe name** and **Target window class** options unless you are instructed to do so by technical support.

If the same executable file (as specified in **Target exe name**) is used to start different programs and you don't want to use SQL Assistant with all of these programs, specify an optional **Target Title** filter, which forces SQL Assistant to compare program titles as well as program names. For example, Microsoft SQL Server Manager for SQL Server 2000 is started using MMC.EXE, which is a general purpose Microsoft

tool for launching various server management applications. With the "Microsoft SQL Server Manager %" **Target Title** you can limit SQL Assistant to the Microsoft SQL Server Manager application only.

Tip: The percent sign can be used in **Target Title** filter property as a wildcard symbol much like a LIKE operator in SQL queries. Using this wildcard, you can target applications whose title changes depending on the database connection type or name of the opened object or the file being edited.

Unregistering Previously Registered and Preconfigured Targets

To unregister already registered targets, including those that come pre-configured with the default installation, use the procedure described below. Unregistering a target permanently deletes its settings in SQL Assistant.

1. Double-click the SQL Assistant icon in the Windows system tray. The Options dialog will appear.

2. Click the **Targets** tab page.

3. Select the editing target you want to unregister, then click the Delete button in the top left corner of the screen.
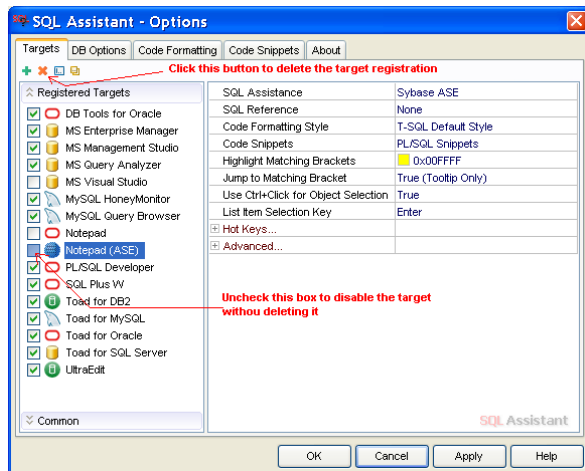
Disabling Targets Without Deleting Their Registrations

If you do not want to delete an existing target permanently but want to temporarily disable SQL Assistance, deselect the **Target Enabled** checkbox as demonstrated in the screenshot above.

Troubleshooting Application Conflicts

Starting with version 6.3, SQL Assistant provides two options for setting application registration exceptions.

Both options are specified as comma separate lists of file names and file masks using the standard * and ? wildcards. If you enter a complete file name, make sure to enter it exactly as it appears in the Windows Task Manager on the Processes tab.
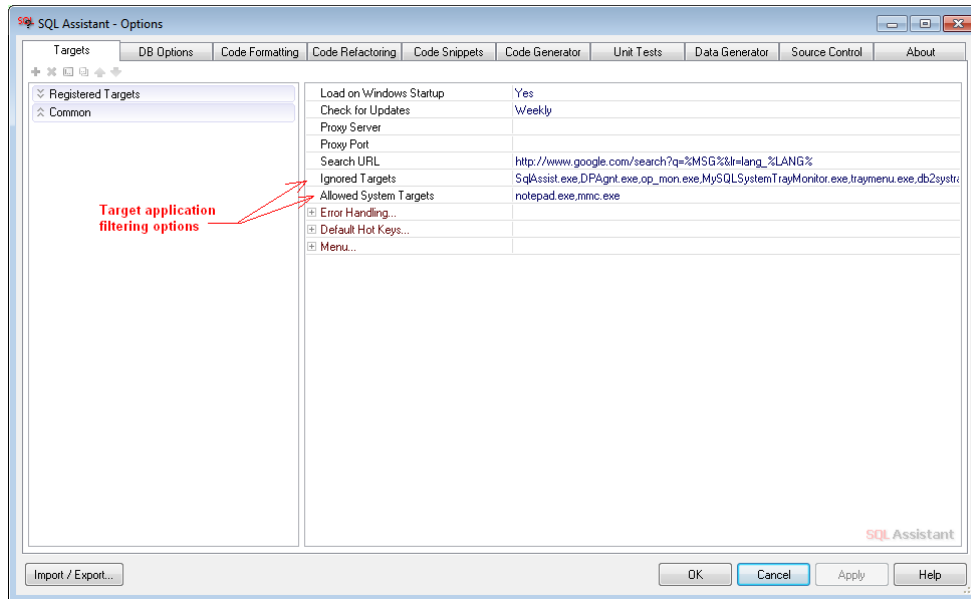
Ignored Targets – Specifies a list of application program files that should be ignored and not registered in the SQL Assistant's process list. If you have a program that does not run successfully when SQL Assistant is started before the program, you can add that program's executable file (or files) to the **Ignored Targets** list. Add only executable (EXE) files, do not add any other file types.

Tip: If you are not sure which file name you need to add, look it up in the Windows Task Manager on the Processes tab.

Allowed System Targets – By default, SQL Assistant ignores all applications started from the Windows home folder and subfolders except for Windows Notepad and MMC.EXE which is used by SQL Server Enterprise Manager. If your editor or development environment is started using another program in one of the Windows system folders, add it to the **Allowed System Targets** list.

Configuring Eclipse-based Target Editors

Configuring the SQL Assistant Add-on for Use with Eclipse:

1. In the target IDE, click the **Windows → Preferences** menu. The **Preferences** dialog will appear.

2. Click **SQL Assistant Plug-in** in the Preferences tree.
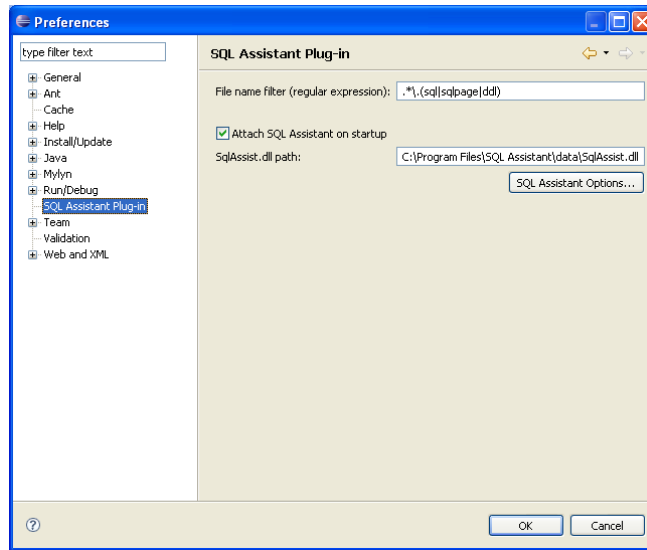
3. **File name filter:** Enter a regular expression that Eclipse can use to decide for which types of files the SQL Assistant add-on should be activated. The default value **.*\.(sql|sqlpage|ddl)** enables SQL Assistant for files having SQL, SQLPAGTE, and DDL .file extensions.

4. **Attach SQL Assistant on startup:** If this option is selected, Eclipse will load SQL the Assistant add-on on startup.

5. **SqlAssist.dll path:** Enter the full path to the SQLAssist.dll file. This file implements SQL Assistant integration interface for Eclipse. If this value is incorrect, Eclipse will not load the SQL Assistant add-on.

Configuring Other Options

If you use specialized SQL editors within the Eclipse environment, it is recommended that you disable the **Code Assistant auto-activation** features in Eclipse Preferences. The SQL IntelliSense features available in SQL Assistant supersede the Code Assistant features of these editors. If left with the default settings, **Code Assistant auto-activation** may interfere with the editor's code completion features which, in turn, might cause various code entry anomalies.

Different types of Eclipse-compatible SQL editors can use different names for this feature and can have their settings located in different places. The following screenshot demonstrate where and how to disable this feature in the default SQL Editor that comes with Eclipse Data Tools Platform.

The following screenshot demonstrates which options should be modified.
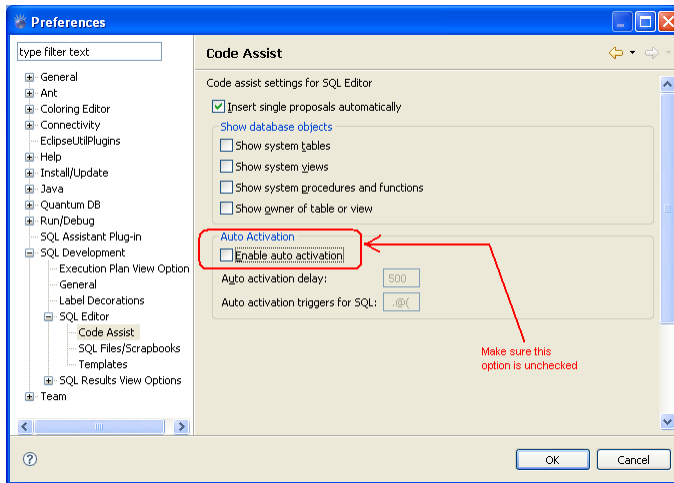
The Preferences dialog can be accessed using the **Window / Preferences** menu in the in the Eclipse IDE.

Configuring IBM Data Studio Targets

Configuring SQL Assistant Add-on for Use with IBM Data Studio

1. In the Data Studio IDE, click the **Windows → Preferences** menu. The **Preferences** dialog will appear.

2. Click **SQL Assistant Plug-in** in the Preferences tree.
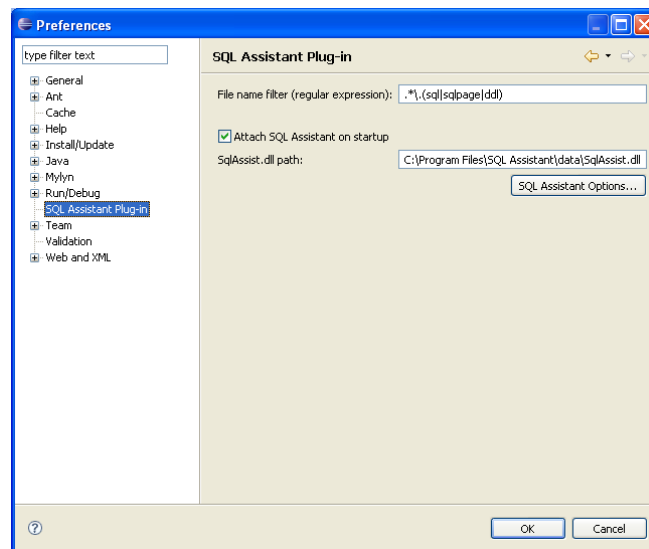
3. **File names filter:** Enter a regular expression that Data Studio can use to decide which types of files should be activated for the SQL Assistant add-on. The default value **.*\.(sql|sqlpage|ddl)** enables SQL Assistant for files having SQL, SQLPAGTE, and DDL .file extensions.

4. **Attach SQL Assistant on startup:** If this option is selected, Data Studio loads the SQL Assistant add-on at startup.

5. **SqlAssist.dll path:** Enter the full path to the SQLAssist.dll file. This file implements SQL Assistant communication interface for Data Studio. If this value is incorrect, Data Studio will be unable to load SQL Assistant add-on.

The Preferences dialog can be accessed using the **Window / Preferences** menu in the in the Data Studio IDE.

Configuring Native Tools Provided with SQL Server 2000 and SQL Server 2005

No changes or customization are required in the settings of SQL Query Analyzer, SQL Server Management Studio 2005 or Management Studio Express 2005. SQL Assistant can be used with these tools using their default settings.

However, in order for SQL Server Management Studio to find and load the SQL Assistant add-on, the add-on must be properly registered.
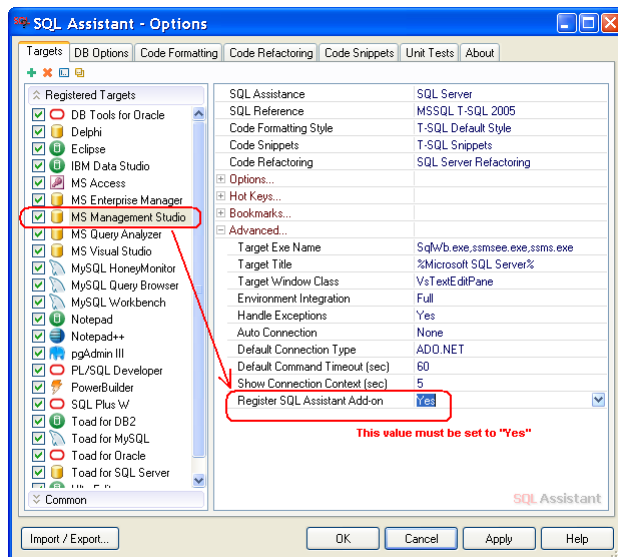
Add-on registration can be enabled and disabled in SQL Assistant Options as demonstrated on the screenshot above.

For best experience, it is recommended that you disable automatic text wrapping options to ensure that code formatting can perform accurately in accordance with your code formatting rules.

Important Notes for Windows Vista, 7, 8, 8.1, 2008, 2010, 2012: To register the SQL Assistant add-on in Windows Vista, 7, 8, 8.1, 2008, 2010, and 2012 systems, start SQL Assistant from a Windows Administrator account. Right-click the SQL Assistant shortcut in the Windows Start Menu and choose **Run as Administrator** from the right-click menu. After SQL Assistant starts, open SQL Assistant's Options and set the Add-on registration state to **Yes**. Right-click the SQL Server Management Studio shortcut in the Windows Start Menu

and choose **Run as Administrator** mode. Open SQL editor and type SELECT then space. The SQL Assistant add-on self-registration should be complete. From this point, you can start SQL Assistant and SQL Server Management Studio from a regular user account.

If multiple users run SQL Assistant on the same system, the add-on registration procedure must be completed for each user.

Configuring Native Tools Provided with SQL Server 2008

It is recommended that you disable the **Statement completion** option in SQL Server Management Studio 2008 Options. SQL IntelliSense features available in SQL Assistant supersede the **Statement completion** features of SQL Server Management Studio. If left with default settings, **Statement completion** will interfere with SQL IntelliSense causing significant inconveniences. The following screenshot demonstrates options that should be modified.

The Options dialog can be accessed using the **Tools → Options** menu in SQL Server Management Studio 2008.

For best experience, it is recommended that you disable automatic text wrapping options to ensure that code formatting can perform accurately in accordance with your code formatting rules.

See the Configuring Native Tools Provided with SQL Server 2000 and SQL Server 2005 topic for details on how to enable or disable SQL Assistant add-on for SQL Server Management Studio.

Important Notes for Windows Vista, 7, 8, 8.1, 2008, 2010, and 2012: To register the SQL Assistant add-on in Windows Vista, 7, 8, 8.1, 2008, 2010, and 2012 systems, start SQL Assistant from a Windows Administrator account. Right-click the SQL Assistant shortcut in the Windows Start Menu and choose **Run as Administrator** from the right-click menu. After SQL Assistant starts, open SQL Assistant's Options and set the Add-on registration state to **Yes**. Right-click the SQL Server Management Studio shortcut in the Windows Start Menu and choose **Run as Administrator** mode. Open SQL editor and type SELECT then space. The SQL Assistant add-on self-registration should be complete. From this point, you can start SQL Assistant and SQL Server Management Studio from a regular user account.

If multiple users run SQL Assistant on the same system, the add-on registration procedure must be completed for each user.

Configuring Native Tools Provided with SQL Server 2012, 2014, 2016, 17.x, 18.x, 19.x, 20.x

It is recommended that you disable the **Transact SQL IntelliSense** option in SQL Server Management Studio Options. SQL IntelliSense features available in SQL Assistant supersede the **Transact SQL IntelliSense** features of SQL Server Management Studio. If left with default settings, **Transact SQL IntelliSense** will interfere with SQL IntelliSense causing significant inconveniences. The following screenshot demonstrates the option that should be modified.
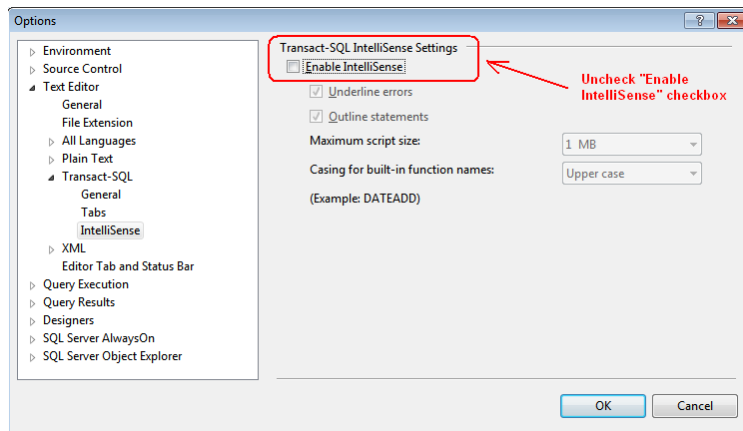
The Options dialog can be accessed using the **Tools → Options** menu in SQL Server Management Studio 2012.

For best experience, it is recommended that you disable automatic text wrapping options to ensure that code formatting can perform accurately and match your code formatting rules.

See the Configuring Native Tools Provided with SQL Server 2000 and SQL Server 2005 topic for details on how to enable or disable SQL Assistant add-on for SQL Server Management Studio.
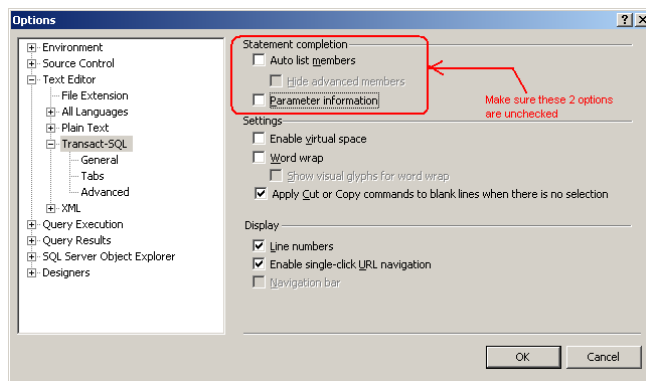
Important Notes for Windows Vista, 7, 8, 8.1, 2008, 2010, 2012: To register the SQL Assistant add-on in Windows Vista, 7, 8, 8.1, 2008, 2010, 2012 systems, start SQL Assistant from a Windows Administrator account. Right-click the SQL Assistant shortcut in the Windows Start Menu and choose **Run as Administrator** from the right-click menu. After SQL Assistant starts, open SQL Assistant's Options and set the Add-on registration state to **Yes**. Right-click the SQL Server Management Studio shortcut in the Windows Start Menu and choose **Run as Administrator** mode. Open SQL editor and type SELECT then space. The SQL Assistant add-on self-registration should be complete. From this point, you can start SQL Assistant and SQL Server Management Studio from a regular user account.

If multiple users run SQL Assistant on the same system, the add-on registration procedure must be completed for each user.

Configuring Toad Targets

If you use SQL Assistant with recent versions of Quest Software Toad tools, it is recommended that you disable the **Statement Completion** and **Casing** features in Toad Options. SQL IntelliSense features in SQL Assistant

supersede **Statement Completion** features of Toad Editors. If left with default settings, **Statement Completion** will interfere with SQL IntelliSense causing significant inconveniences. The following screenshot demonstrates options that should be modified.
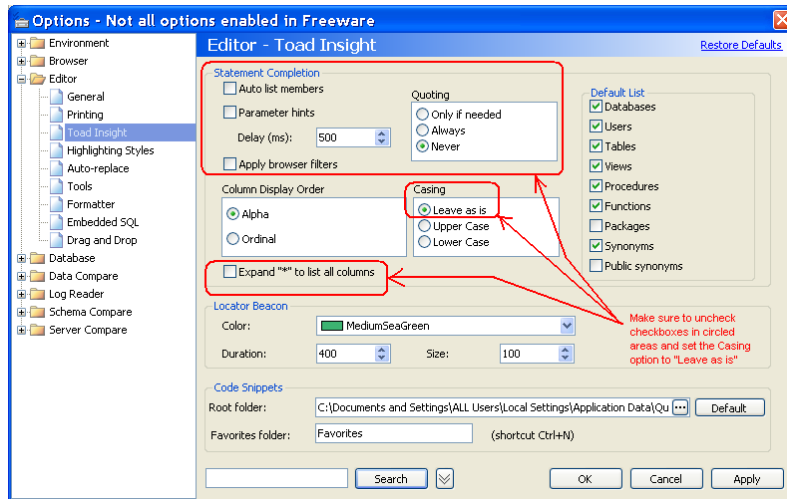
The Options dialog can be accessed using the **Tools → Options** menu in Toad.

Configuring UltraEdit Targets

Different versions of UltraEdit editors render the same text differently. As far as SQL Assistant interface for UltraEdit is concerted, the important differences affect tab spacing and the width of characters. For best experience, it is recommended that you set the Tab Stop option value to 2. If that option is set to another value, certain SQL Assistant functions may appear misaligned on the screen, such as syntax check results highlighting errors, highlighting of matching brackets, and so on. Note that a Tab Stop value of 2 has been tested with UltraEdit versions 15 and 16. If you run an older version, you may need to choose a different Tab Stop value.
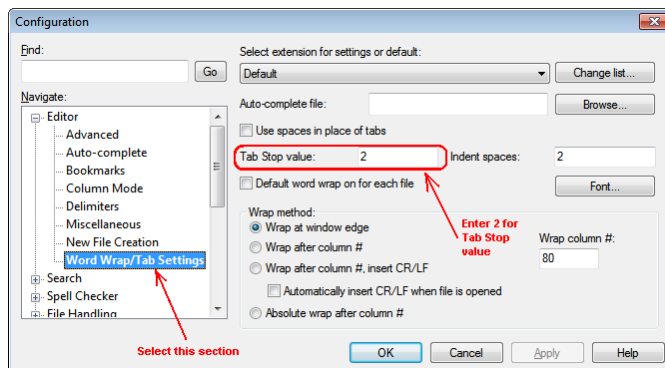
The Configuration dialog can be accessed using the **Advanced →Configuration** menu in UltraEdit.

Configuring Visual Studio .NET, 2003, 2005 and 2008 Targets

No changes or customization are required in the settings of Visual Studio. SQL Assistant can be used with these tools using their default settings. However, for Visual Studio to find and load the SQL Assistant add-on, the add-on must be properly registered.

The add-on registration can be enabled and disabled in SQL Assistant Options as shown in the screenshot above.

For best experience, it is recommended that you disable automatic text wrapping options to ensure that code formatting can perform accurately and match your code formatting rules.

Important Notes for Windows Vista, 7, 8, 8.1, 2008, 2010, 2012: To register the SQL Assistant add-on in Windows Vista, 7, 8, 8.1, 2008, 2010, 2012 systems, start SQL Assistant from a Windows Administrator account. Right-click the SQL Assistant shortcut in the Windows Start Menu and choose **Run as Administrator** from the right-click menu. After SQL Assistant starts, open SQL Assistant's Options and set the Add-on registration state to **Yes**. Right-click the SQL Server Management Studio shortcut in the Windows Start Menu and choose **Run as Administrator** mode. Open SQL editor and type SELECT then space. The SQL Assistant add-on self-registration should be complete. From this point, you can start SQL Assistant and SQL Server Management Studio from a regular user account.

If multiple users run SQL Assistant on the same system, the add-on registration procedure must be completed for each user.

# Configuring Visual Studio 2010, 2012, 2013, 2015, 2017, and 2019 Targets

No changes or customization are required in the settings of Visual Studio. SQL Assistant can be used with Visual Studio 2010 and Visual Studio 2013 using the development environment's default settings. However, for Visual Studio to find and load the SQL Assistant add-on, the add-on must be properly registered.
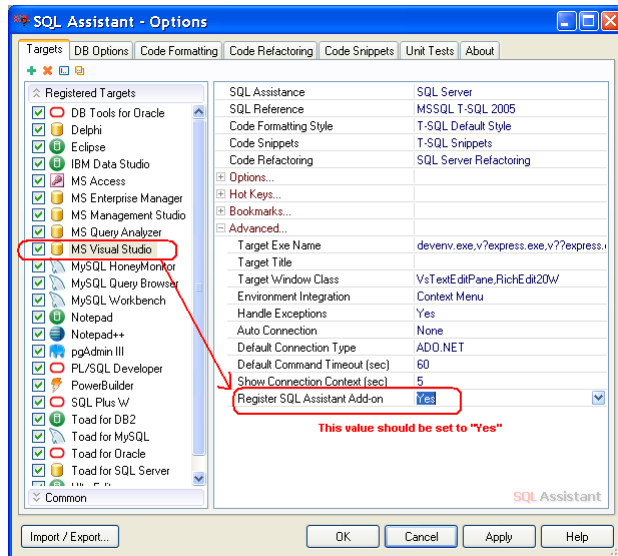
The add-on registration can be enabled and disabled in SQL Assistant Options as shown in the screenshot above.

For best experience, it is recommended that you disable automatic text wrapping options to ensure that code formatting can perform accurately in accordance with your code formatting rules.

Important Notes for Windows Vista, 7, 8, 8.1, 2008, 2010, 2012: To register Visual Studio add-ons, start SQL Assistant from a Windows Administrator account. Right-click SQL Assistant shortcut in the Windows Start Menu and choose **Run as Administrator** from the right-click menu**.** After SQL Assistant starts, open SQL Assistant's Options and set the Add-on registration state to **Yes**. From this point, you can start SQL Assistant and SQL Server Management Studio from a regular user account.

If multiple users run SQL Assistant on the same system, the add-on registration procedure must be completed for each user.

Troubleshooting add-on registration

1. Check the required SQL Assistant manifest file exists:
    Windows Vista, 7, 8, 8.1, 2008, 2010, and 2012
    ```
    C:\Users\[your user name]\AppData\Local\Microsoft\VisualStudio\[VS
    version]\Extensions\SoftTree Technologies\SqlAssist.vs\1.0\extension.vsixmanifest
    ```

    Windows XP, 2000, 2003
    ```
    C:\Documents and Settings\[your user name]\Application Data\
    Microsoft\VisualStudio\10.0\Extensions\SoftTree
    Technologies\SqlAssist.vs\1.0\extension.vsixmanifest
    ```

2. Open the SQL Assistant manifest file in Notepad and verify that the path to SQL Assistant files is fully qualified and points to the correct installation location. This section of the manifest file should look like the following:

```
<Content>
    <MefComponent>C:\Program Files (x86)\SQL Assistant
12\data\SqlAssist.vs.dll</MefComponent>
</Content>
```

3. Start Visual Studio. Click the Tools → Extensions menu. Verify that the SQL Assistant add-on is listed and enabled. If it is not enabled, click the Enable button and restart Visual Studio. Click the Tools > Extensions menu again and verify that the SQL Assistant add-on is now enabled.

Configuring Oracle SQL Developer Targets

Configuring SQL Assistant Plug-in for Use with Oracle SQL Developer

No special configuration is required. After the installation, the SQL Assistant plug-in for Oracle SQL Developer should be able to operate normally. Simply ensure that the plug-on is enabled.

Click the **Tools → Preferences** menu in SQL Developer, then select **Extensions** in the Preferences tree.
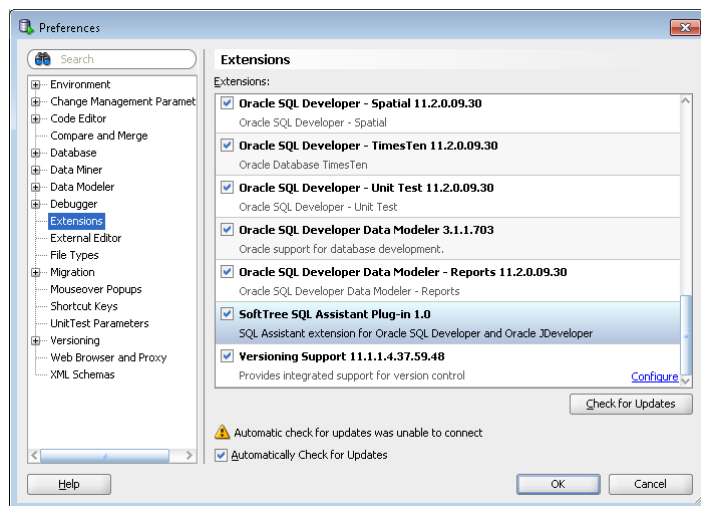
Verify that the SoftTree SQL Assistant plug-in is enabled. If it is not, enable it and click the OK button.

Configuring Oracle JDeveloper Targets

Configuring SQL Assistant Plug-in for Use with Oracle JDeveloper

No special configuration is required. After the installation, the SQL Assistant plug-in for Oracle JDeveloper should be able to operate normally. Simply ensure that the plug-in is enabled.

Click the **Tools → Preferences** menu in JDeveloper, then select **Extensions** in the Preferences tree.
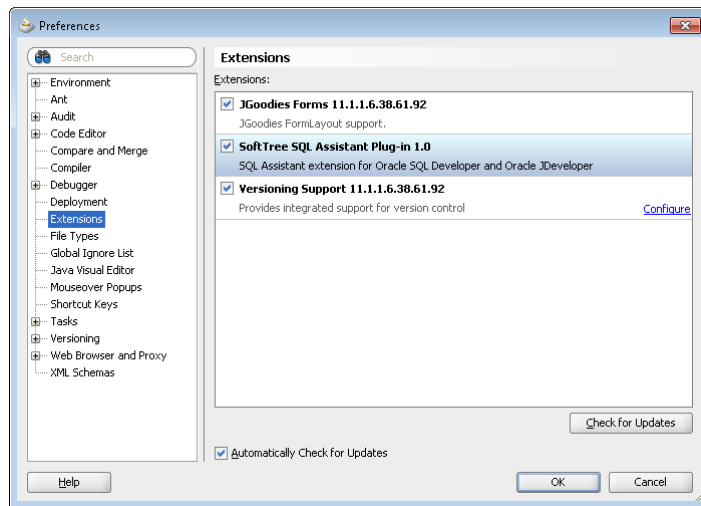
Verify that the SoftTree SQL Assistant plug-in is enabled. If it is not, enabled it and click the OK button.

Other Target Environments

In most situations, no changes are required in the settings of the target development environments and editors listed in SQL Assistant options.

Warning: Please be aware that SQL Assistant software has been tested with the most recent English versions of the pre-configured target editors available at the time of the latest major SQL Assistant software release. The target editors were tested using their default setups. The pre-configured settings are not guaranteed to be fully compatible with older versions or with versions that are more recent than the tested versions. The pre-configured settings are also not guaranteed to work with localized versions. You may need to tweak SQL Assistant settings to get SQL Assistant to work seamlessly with your SQL editor.

Most Common Compatibility Issues and How to Resolve Them

- **SQL Assistant does not appear in the target editor**
  You can try the following: Click the editor window and set the focus to the text part of the editor. Press the Ctrl+Shift+F5 hotkey to re-register the editor with SQL Assistant. If that works, you should follow the instructions in the Registering New Targets for SQL Assistance topic in CHAPTER 49. If that does not work and you get an error such as "Incompatible Window Class," it is likely that the maker of the editor developed a version that is incompatible with SQL Assistant and is also incompatible with standard Windows edit controls. In this case, SQL Assistant will be unable to communicate with the editor. Please Contact SoftTree Technologies and check to see if the issue can be resolved some other way.

- **Database connectivity issues**
  Strictly speaking, this is not an editor-specific compatibility issue. The first thing to check is to ensure that the required database client software has been installed on your system. Without the database client software, SQL Assistant cannot establish a database connection. To resolve this issue,

download and install the missing client software. For software requirements, please see APPENDIX A, Hardware and Software Requirements. If the required software has been already installed and you are still unable to connect to the database, follow the instructions in CHAPTER 2, Connecting to Your Database.

- **Certain hotkeys stop working correctly after a SQL Assistant installation or upgrade**
  There might be a keyboard conflict. A hotkey reserved in your target editor might be intercepted and processed by SQL Assistant or visa versa. See the next topic, Resolving Keyboard Hotkey Conflicts for information on resolving this problem.

Resolving Keyboard Hotkey Conflicts

It is possible that certain keyboard hotkeys used by SQL Assistant are also used by your target editor. In this case, pressing a conflicting hotkey may lead to a double action or might suppress one of the functions associated with that hotkey.

It is easy to resolve hotkey conflicts. Every pre-defined default hotkey in SQL Assistant can be changed to any other key sequence of your choice. This customization can be done either globally across all registered targets in the Common section of SQL Assistant Options, or individually for each target. If hotkeys are defined individually, you can assign different hotkey sequences for use with different targets. For more information about customizing hotkeys, see the Customizing Hot Keys topic in CHAPTER 48, Customizing SQL Assistant's Behavior.

CHAPTER 50, Managing Scheduled Tasks

Overview

SQL Assistant provides built-in functions for scheduling recurring and delayed SQL Assistant operations using your computer's system scheduler. Do not confuse them with the graphical management interfaces that SQL Assistant provides for managing scheduled tasks running on your database servers. They are unrelated.

SQL Assistant enables you to schedule many types of recurring operations and unattended tasks, including:

- SQL code execution - see CHAPTER 16, Scheduling SQL Script Execution.
- Source Code Control repository, database, and workspace automatic updates and synchronization - see Scheduling Automated Source Control Operations topic in CHAPTER 23, Database Source Code Control Interface.
- Context Search and Code Reusability FTS repository updates - see Managing FTS Code Repository topic in CHAPTER 37, Improving Code Reusability.
- Unit testing – see Scheduling Unit Test Project Runs topic in CHAPTER 19, Unit-testing Database Code.
- Test data generation – see Working With Test Data Generator topic in CHAPTER 18, Generating Test Data.
- Data import/export – see Scheduling Automated Data Import, Export, and Transfer Operations in CHAPTER 13, Scripting, Exporting, Importing, and Copying Data.
- Data transfer – see Scheduling Automated Data Import, Export, and Transfer Operations in CHAPTER 13, Scripting, Exporting, Importing, and Copying Data.
- Schema comparison – see Command Line Interface in CHAPTER 27, Schema Compare Utility.
- Job comparison – same scheduling method as for schema comparison, – see Command Line Interface in CHAPTER 27, Schema Compare Utility.
- Data comparison – see Command Line Interface in CHAPTER 26, Data Compare Utility.
- Database load generation – see Scheduling Database Benchmark Project Runs in CHAPTER 36, Testing Database Performance Under Heavy Load.

You can also manage generic properties of scheduled tasks using Windows Scheduled Tasks applet in the Windows Control Panel, such as task's enabled/disabled state, run-time account, task schedule, and so on.

You can use the **Scheduled Tasks** dialog for modifying already scheduled tasks and customizing SQL Assistant's specific task properties. The **Scheduled Tasks** dialog can be opened using SQL Assistant's right-click or top-level menu in the target SQL editor. Alternatively, you can right-click the SQL Assistant icon in the Window system tray, and in the right-click menu select **SQL Assistant** submenu, and then select **Execute/Schedule SQL → Scheduled Tasks...** menu command.

The **Scheduled Tasks** dialog lists only scheduled tasks that are known to SQL Assistant. These are the tasks for which SQL Assistant has created task definitions in the `Application Data\SQL Assistant\Schedule` folder. On Windows XP and Windows 2003 systems, the Application Data folder is typically found in "C:\Documents and Settings\<username>\Application Data\", where "username" is your login name. The location of this folder differs in Windows Vista, Windows 7, 8, and other Windows versions. You can learn the actual location of this folder by executing the following command from the command prompt:

echo %APPDATA%

Using Scheduled Tasks Dialog

To disable a task – un-tick the checkbox in front of the task name. This will disable the associated scheduled Windows task.

To enable a task – tick the checkbox in front of the task name. This will enable the associated scheduled Windows task.

To change task's schedule and/or associated database connection – select the task that you want to modify, and then click the **Schedule** button. This will open the **SQL Assistant - Task Schedule** dialog. Fill the required schedule properties and then click the Ok button to save changes.

To open task for editing – select the task that you want to modify, and then click the **Edit** button. This will open the task's script or project file in its associated graphical tool which can be used to edit the project settings. For example, the scheduled test-data generation projects are opened in the Test Data Generator dialog. Note that not all scheduled tasks are associated with a script or project. Only tasks associated with a script or project can be edited. Project-less tasks can be edited directly in the task schedule properties by changing task's command line parameters.

To delete a task – select the task that you want to modify, and then click the **Edit** button. This will delete the associated scheduled Windows task and also delete the task properties file from `Application Data\SQL Assistant\Schedule` folder.

CHAPTER 51, Backing Up and Sharing SQL Assistant Settings

Overview

All SQL Assistant settings are stored in the SQLAssist.sas configuration file which is located in the user's `AppData\SQL Assistant\<version number>` subfolder. The folder is typically found in "C:\Users\<username>\AppData\Roaming\", where "username" is your login name. The location of this folder differs in Windows Vista, Windows 7, 8, and other Windows versions. You can learn the actual location of this folder by executing the following command from the command prompt:

echo %APPDATA%

Backing up SQL Assistant Settings Using the File System

One way to create a backup of your SQL Assistant settings is to simply create a backup copy of the SQLAssist.sas file described in the Overview topic.

Warning: Note that this type of backup contains the complete set of SQL Assistant settings you have on your system, including all database connections and credentials that you may have saved in the file. Be careful not give this file to other users. If you have saved automatic connections and connection passwords in the configuration file, even though passwords are stored in encrypted form, other users will theoretically be able to use your database credentials if given your configuration file. To avoid getting into this situation, you can use the Export/Import utilities described in the following sections in this chapter

Backing up SQL Assistant Settings Using the Import/Export Utilities

The Export/Import utilities can be accessed from the SQL Assistant graphical Options dialog or from the command line interface. The Export/Import utilities provide complete control over which settings are exported from SQL Assistant configuration. You can choose to export all settings or selected groups of settings.

To export SQL Assistant settings to a file:

1. Double click the SQL Assistant icon in the system tray. The Options dialog will appear on the screen.

2. In the bottom left corner of the Options dialog, click the **Import/Export** button. The SQL Assistant Import / Export Options dialog will appear on the screen.

3.   Select the Export tab.
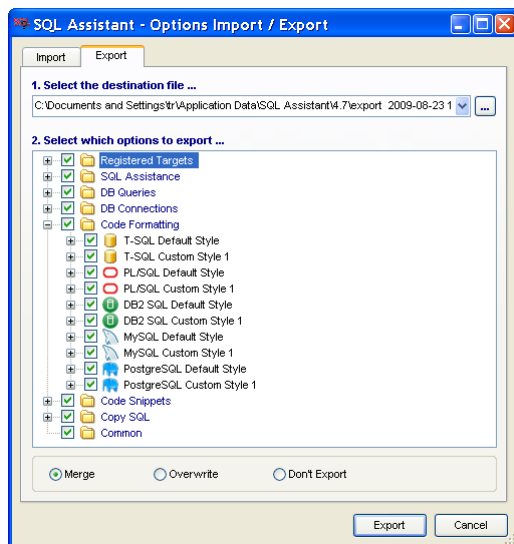
Choose the export file name. You can type a new file name or use the browse button to open the Select Export File dialog or you can pick one of the files you used previously from the **Select the destination file** list.

 Tip: Two types of export file formats are supported by the export, native SQL Assistant's configuration file type and generic XML file type. The type of the export format is determined by file extension. Native format files must have .SAS extension, while XML files must have .XML extension.

4.   In the **Select which options to export** window, choose the options you want to export. Click the [+] sign in front of option groups and subgroups to drilldown to specific settings.

5.   Select the type of the export file to create. The following choices are available:

- **Merge** – Instructs the Import utility to compare settings saved in the export file against the settings in your active SQL Assistant configuration file and import only settings that are new to your file. For example, if the export file contains formatting rules for SELECT, MERGE and DELETE statements and your active file already has formatting rules for SELECT and DELETE (which you may have customized locally), the Import utility will skip rules for SELECT and DELETE and import rules for MERGE statement only.

- **Overwrite** – Instructs the Import utility to compare settings saved in the export file against the settings in your active SQL Assistant configuration file and overwrite matching settings in your active SQL Assistant configuration file. If the export file contains new settings that cannot be found in your active configuration file, they will be added to your file too.

- **Don't Export** –Instructs the Export utility to export all settings except those that are selected in the **Select which options to export** window. Use this option to create export files that do not include customized, user-sensitive settings such as database connections, keyboard shortcuts and so on. This option is provided for your convenience to allow exporting and later importing most of the settings, skipping only a few selected options.

7.   Click the **Export** button.

 Warning: Do not edit settings export files directly using external programs. Any issues arising from incorrectly modified files will not be supported and may lead to a data loss.

Command Line Backup to XML Files

You can use SaCmd.exe utility to export SQL Assistant's settings to XML files from the command line.

"C:\Program Files (x86)\SQL Assistant 12\SACmd.exe" "sasxml:<XML export file name>"

Replace <XML export file name> including brackets with the name of the file you want the settings to be exported to.

Warning: If you specify an existing file, its contents will be overwritten

Restoring SQL Assistant Settings from a Backup File

If you have created a backup file within the File System as described in the topic Backup SQL Assistant Settings Using File System you can restore the settings by simply restoring this file to its original location.

Warning: Make sure to shutdown SQL Assistant and all targets before restoring this file. If you fail to close all targets, the restored configuration file could be overwritten by one of the running SQL Assistant add-ons.

Restoring SQL Assistant Settings Using the Import/Export Utilities

To restore default SQL Assistant settings:

1. Double click SQL Assistant's icon in the system tray. The Options dialog will appear on the screen.

2. In the bottom left corner of the Options dialog, click the Import/Export button. The SQL Assistant Import / Export Options dialog will appear on the screen.

3. Make sure the **Load Default Options** checkbox is checked and that all options are selected. If you want to restore only a subset of options, customize the selection and deselect the options, option groups and subgroups you do not want to restore.

4. Click the **Import** button.

To import SQL Assistant settings from an export file:

1. Double click SQL Assistant's icon in the system tray. The Options dialog will appear on the screen.

2. In the left bottom corner of the Options dialog, click the **Import/Export** button. The SQL Assistant Import / Export Options dialog will appear on the screen.

3. Deselect the **Load Default Options** checkbox.

4. Select the previously created export file from which you want to import SQL Assistant settings. You can type the file name or use the […] browse button to open Select Export File dialog or you can pick one of the files you used previously from the **Select the source file** list.

5. Choose the options you want to import. If you want to import only a subset of options, customize the selection and uncheck options, option groups and subgroups you do not want to restore.

6. Click the **Import** button.

Sharing SQL Assistant Settings Between Team Members

A person responsible for maintaining enterprise-wide coding standards can use the Import/Export utilities described above to create export files created using selected, standardized groups of settings. The standardized export files can be then distributed to other team members and imported on their systems.

Automating Distribution and Sharing of SQL Assistant Settings

The steps for exporting SQL Assistant settings, sharing the export file, and then importing it on a different systems can be fully automated. Following is a suggested procedure for creating an automated process:

1. Create an export file with selected settings as described in [Backing up SQL Assistant Settings Using the Import/Export Utilities](#) topic or using the command line interaface.

2. Copy the export file to a shared network drive accessible to all team members.

3. Add the following command to each SQL Assistant user's logon script:

    "C:\Program Files (x86)\SQL Assistant 12\SQLAssist.exe" /merge:< export file name>

    Replace the <export file name> placeholder with the full file path and file name of the export file. If the export file name or path includes space characters, you must enclose the entire **/merge:< export file name>** parameter value in double-quotes.

Note that if you maintain team member computers within an Active Directory environment, the described 3rd step is required just once. You can use Active Directory tools to set up and modify users' logon scripts and include commands to import SQL Assistant settings into users copies of SQLAssist.sas files.. To ensure that this method works reliably, always use the same file name and path for the shared export file.

Customizing Add-on Behavior

Options in the **Document Manager** section for the target editor control behavior of the **Document Manager and Code History** add-on.

To customize target-specific options

1. Double-click the SQL Assistant icon in the Windows System Tray. The **SQL Assistant - Options** dialog will appear

2. Click the **Targets** tab page.

3. In the left window of the Targets page, select the target editor type, for example, SQL Server Management Studio from the list of targets.
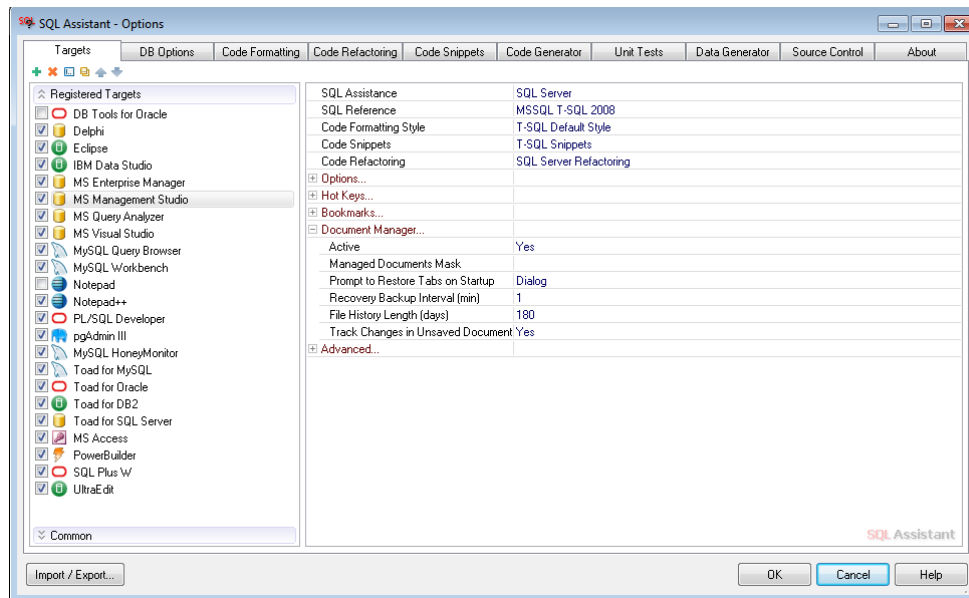
4. Expand the **Document Manager** section on the right side of the Options dialog

5. Customize options as required

The **Active** option enables or disables the Document Manager and Code History add-on. the default value is Yes.

The **Managed Documents Mask** option allows you to customize which file types should be tracked. If no value is specified, files of all types are tracked. To enter one or more types, specify file extensions as a pipe-separated list, for example SQL|DDL|TXT.

The **Prompt to Restore Tabs on Startup** option enables or disables the **Restore Tabs** feature. Three choices are available:

None - all tabs are restored automatically on the editor startup without displaying any prompts
Message - a simple "Restore - Yes/No" message box is displayed, you can choose Yes to restore all tabs, or choose No to not to restore anything.
Dialog - the "Restore Tabs" dialog is displayed. You can review and choose which tabs to restore and which not.

The **Track Changes un Unsaved Documents** option enables or disables tracking changes in tabs containing unnamed scripts that have not been saved to file system files yet. The default value is Yes. If you set this value to No, SQL Assistant will track changes in named files only and ignore all unnamed scripts, such as scripts opened directly from the database or pasted into a new editor tab from the clipboard.

The **Recovery Backup Interval (min)** option controls how often SQL Assistant checks for script changes and saves new script version. The interval value is in minutes. The default value is 15 minutes. To disable the Code History Feature, set this value to zero.

The **File History Length (days)** option controls how long SQL Assistant stores file references in the history list and the associated revision files. By default it deletes files having all revisions that are 180 days or older.

CHAPTER 52, Installation and Uninstallation

Installation

The SQL Assistant setup program provides a simple, intuitive installation method. Simply run the setup program and follow prompts displayed on the screen.

Important Notes for Windows Vista and later version of Windows: When installing the software on these systems, make sure you run the setup in the Windows Administrator mode. When logged in as a local system administrator, you must right-click the setup program and choose **Run as Administrator** in the context menu to run it with elevated privileges**.** Failure to do so will enable Windows to virtualize all registry and file changes made by the installer and remove them after you logoff from the system. As a result, next time you login and try to run SQL Assistant, the program could behave incorrectly if it is unable to locate and load the required settings. This might also affect loading of SQL Assistant add-ons from other programs such as SQL Server Management Studio, Visual Studio, and others.

Important Notes for using Remote Desktop and Terminal Servers: Remote installations of SQL Assistant software are not directly supported at this time. You should install the software directly on the system. After installation, you can access and run SQL Assistant software remotely.

Uninstallation

SQL Assistant supports standard uninstallation mechanism for removing program files from the computer.

To uninstall the SQL Assistant:

1. Click the Windows **Start** button.

2. From the Start Menu, select **Settings**, then **Control Panel**.

3. Double-click the **Add/Remove Programs** icon in the Control Panel.

4. Select SQL Assistant in the programs list, then click the **Add/Remove** button.

Checking and Installing Updates

Manual Mode

1. Right-click SQL Assistant's icon in the system tray and select the **Check for Updates** command from the right-click menu.

2. If new updates are available, you will receive a message indicating the available version as well as a prompt to download and install the update.

- To initiate the update, click the **OK** button. The update installation process begins immediately.

- If you click the **Cancel** button, the message will disappear and no updates will be installed. You will be prompted to install the update at a later time.

Automatic Mode

In automatic mode, SQL Assistant runs in the background, periodically checking for new updates When a new update becomes available, SQL Assistant asks your permission to download and install it. If you click the **OK** button, the automatic update process begins immediately. If you click the **Cancel** button, the message disappear and no updates are installed.

The frequency of automatic checks for updates is controlled by SQL Assistant Options. The default frequency is set to "Weekly". You can change this value to "Monthly" using this procedure:

1. Double-click the SQL Assistant icon in the system tray.

2. On the **Options** dialog, choose the **Targets** tab page.

3. On the left side of the Options dialog, click the **Common** section.

4. On the right side of the Options dialog, change the value of the **Check for Updates** option to "Monthly."

5. Click the **Ok** button to save changes and close the dialog.

To turn off Automatic Mode:

1. Double-click the SQL Assistant icon in the system tray.

2. On the **Options** dialog, choose the **Targets** tab page.

3. On the left side of the Options dialog, click the **Common** section.

4. On the right side of the Options dialog, change the value of the **Check for Updates** option to "Never".

5. Click the **Ok** button to save changes and close the dialog.

Tip: By default, SQL Assistant uses the Internet connection settings configured in the Internet Explorer web browser. If the browser is configured to connect to the Internet via a proxy server, SQL Assistant uses that connection type; otherwise, it attempts to connect to the Internet directly. If you do use proxy server for Internet connections but do not use Internet Explorer as your browser, you can specify proxy server settings directly in SQL Assistant's options. Use the steps described above to open the **Common** section in SQL Assistant options. In that section, specify the proxy server's network name or IP address and the server's port number.

APPENDIX A, Hardware and Software Requirements

Minimum Requirements

1   x86 or x64-based workstation or server running Windows 7 or later Windows version.

2   2 GB RAM

3   478 MB free disk space for full installation

4   Required database client software (consult your database system documentation for details)

5   If ODBC database interface is used, ODBC 3.0 and compatible database connectivity driver

6   If spell check feature is used, Microsoft Word 97 or later must be installed on the system.

7   If Open in Excel or Export from Excel features are used, Microsoft Excel 97 or later must be installed on the system.

<u>**Database Server Software**</u>

Any of the supported database servers:

- Oracle 8i, 9i, 10g, 11g, 12c, 18c, 19c, 21c, 23c
- Microsoft SQL Server 2000, 2005, 2008, 2012, 2014, 2016, 2017, 2019, 2022
- Windows Azure SQL Database 11.x, 12.x (formerly SQL Azure)
- Apache Hive 2.x, 3.x
- Apache SparkSQL 2.x, 3.x
- Apache Cassandra 3.x, 4.x
- DataStax Enterprise 6.x
- DB2 UDB for LUW 7.x, 8.x, 9.x, 10.x, 11x
- DB2 UDB for iSeries 5.x, 6.x, 7.x
- MySQL 5.x, 8.x
- MariaDB 5.x, 10.x, 11.x
- SAP Sybase SQL Anywhere 9.x to 16.x (formerly Adaptive Server Anywhere)
- SAP Sybase ASE 12.x, 15.x, 16.x
- PostgreSQL 8.x, 9.x, 10.x, 11.x, 12.x, 13.x, 14.x, 15.x, 16.x
- Microsoft Access 2003 to Office 365 (11.x to 16.x)
- Amazon Redshift
- Teradata 14.x, 15.x, 16.x, 17,x
- IBM Netezza 7.x
- Pivotal Greenplum 4.3.x, 5.x, 6.x
- Snowflake 6.x, 7.x
- SQLite 2.x, 3.x
- MongoDB via Connector for BI 2.x

<u>**Database Client Software**</u>

Depending on the database type and selected target editor, you may need to have one or more of the following database client software packages installed on the computer running the target editor:

- **Oracle client** software provided by Oracle Corporation.

- **Oracle ODBC Driver** provided by Oracle Corporation.
- **Microsoft SQL Server ODBC driver** for SQL Server provided by Microsoft Corporation.
- **Microsoft Access ODBC driver** provided by Microsoft Corporation.
- **Sybase ASE ODBC driver** for ASE 12 and later provided by Sybase Corporation
- **Adaptive Server Enterprise ODBC driver** for ASE 15 provided by Sybase Corporation
- **Adaptive Server Anywhere (a.k.a. SQL Anywhere) ODBC drivers** for ASA 9 or later provided by Sybase Corporation
- **IBM DB2 ODBC driver** for DB2 7 and later, which is part of IBM DB2 ODBC drivers software provided by IBM Corporation.
- **MySQL ODBC 3.51 driver** provided by Oracle Corporation.
- **MySQL Connector** v5.0 or later which is part of MySQL client software provided by Oracle Corporation. MySQL native client **libmysql** is available as a shared library libmysql.dll.
- **MariaDB native client** which is part of MariaDB client software provided by MariaDB Foundation. Part of that is MariaDB native client **libmariadb** is available as shared library libmariadb.dll.
- **PostgreSQL LibPQ** library version 9.0 or later is included with SQL Assistant software.
- **PostgreSQL ODBC driver (ANSI)** provided by PostgreSQL Global Development Group.
- **PostgreSQL ODBC driver (UNICODE)** provided by PostgreSQL Global Development Group.
- **Amazon Redshift ODBC driver** provided by Amazon LLC.
- **Teradata ODBC driver** provided by Teradata Corporation.
- **IBM Netezza ODBC driver** provided by provided by IBM Corporation.
- **SQLite ODBC driver** provided by Christian Werner
- **ADO.NET** software and database connectivity drivers, which are part of Microsoft NET Framework v2.0 or later provided by Microsoft Corporation
- **Apache Hive and Apache SparkSQL ODBC drives** available from various vendors.

Tip: Note that the text in bold indicates actual driver names available directly from database vendors. SQL Assistant currently does not provide support for connectivity drivers provided by third party vendors. You can try using them at your own risk.

APPENDIX B, License Agreement

SOFTWARE PRODUCT USER LICENSE

Copyright laws and international copyright treaties, as well as other intellectual property laws and treaties protect this SOFTWARE PRODUCT. The SOFTWARE PRODUCT is licensed, not sold.

CAUTION: Loading this software onto a computer indicates your acceptance of the following terms. Please read them carefully.

GRANT OF LICENSE: SoftTree Technologies, Inc. ("SoftTree Technologies") grants you a license to use the software ("Software"). One licensed copy of the Software may either be used by a single person who uses the software personally on one or more computers, or installed on a single workstation used non-simultaneously by multiple people, but not both. One licensed copy of the Software can be used with any number of database servers.

You may make other copies of the Software for backup and archival purposes only. You may permanently transfer all of your rights under this Software LICENSE only in conjunction with a permanent transfer of your validly licensed copy of the product(s).

The Software and associated add-in components are licensed on a RUN-TIME basis, which means, that for each computer on which the Software is installed, a valid run-time license must exist.

RESTRICTIONS: Unregistered versions (shareware licensed copies) of the Software may be used for a period of not more than 14 days. After 14 days, you must either stop using the Software, or obtain a validly licensed copy.

You must maintain all copyright notices on all copies of the Software. You may not sell copies of the Software to third parties without express written consent of SoftTree Technologies and under SoftTree Technologies' instruction.

EVALUATION copies may be distributed freely without charge so long as the Software remains whole including but not limited to existing copyright notices, installation and setup utilities, help files, licensing agreement, In executing such an act as distributing without the similar copyright or license violation, to the maximum extent permitted by applicable law you may be held liable for loss of revenue to SoftTree Technologies or SoftTree Technologies' representatives due to loss of sales or devaluation of the Software or both.

You must comply with all applicable laws regarding the use of the Software.

COPYRIGHT: The Software is the proprietary product of SoftTree Technologies and is protected by copyright law. You acquire only the right to use the Software and do not acquire any rights of ownership.

For your convenience, SoftTree Technologies provides certain Software components in the source code format. You may customize this code for your environment, but you agree not to publish, transfer, or redistribute in any other form both the original code and the modified code.

You agree not to remove any product identification, copyright notices, or other notices or proprietary restrictions from the Software.

You agree not to cause or permit the reverse engineering, disassembly, or decompilation of the Software. You shall not disclose the results of any benchmark tests of the Software to any third party without SoftTree Technologies' prior written approval.

DESCRIPTION OF OTHER RIGHTS AND LIMITATIONS: You may not rent, lease or transfer the Software except as outlined under GRANT OF LICENSE - use and copy.

Without prejudice to any other rights, SoftTree Technologies may terminate this Software LICENSE if you fail to comply with the terms and conditions of this Software LICENSE. In such event, you must destroy all copies of the Software and all of its component parts.

WARRANTY DISCLAIMER: SoftTree Technologies provides this license on an "as is" basis without warranty of any kind; SoftTree Technologies disclaims all express and implied warranties, including the implied warranties of merchantability or fitness for a particular purpose.

LIMITATION OF LIABILITY: SoftTree Technologies shall not be liable for any damages, including direct, indirect, incidental, special or consequential damages, or damages for loss of profits, revenue, data or data use, incurred by you or any third party, whether in an action in contract or tort, even if you or any other person has been advised of the possibility of such damages.

GOVERNING LAW: This Agreement is governed by the laws of the State of New York and the intellectual property laws of the United States of America.

SoftTree Technologies, Inc.
USA

Copyright (C) 2006-2024 SoftTree Technologies, Inc. All Rights Reserved.

UniCodeEditor portions of the Software - Copyright (C) 1999-2001 digital publishing AG, Copyright (c) 1999-2005 Mike Lischke

VirtualTree portions of the Software - Copyright (C) 1999-2005 Mike Lischke

EControl Editor portions of the Software - Copyright (C) 2004-2015 Mikhail Zakharov.

pg_dump, pg_restore, pqLib - Copyright (c) 1996-2021, PostgreSQL Global Development Group. Portions Copyright (c) 1994, Regents of the University of California.

libmariadb - Copyright (C) 2012, 2020, MariaDB Corporation. Portions copyright (C) 2000 MySQL AB & MySQL Finland AB & TCX DataKonsult AB.

FastScript portions of the Software - Copyright (c) 1998-2005 Fast Reports Inc.

TrStringGridEd portions of the Software - Copyright (c) 2005-2017 Rosinský Software

FastReport and FastCube portions of the Software - Copyright (C) 2013, 2017 Fast Reports Inc.

TeeChart portions of the Software - Copyright (C) 2013 Steema Software.

Git Credential Manager for Windows - Copyright (C) 2015 Microsoft Corporation.

SQL Server Shared Management Objects (SMO) - Copyright (C) 2020 Microsoft Corporation

madExcept portions of the software - Copyright (C) 1999 - 2018 Mathias Rauen

AlphaSkins portions of the software – Copyright © 2002-2019 AC Team

SynPdf and Synopse framework - Copyright (C) 2016 Arnaud Bouchez

Apache Cassandra native driver - portions Copyright (c) 2020 DataStax, Inc.

libssh2 - Copyright (C) 2004-2007 Sara Golemon <sarag@libssh2.org>, Copyright (C) 2005,2006 Mikhail Gusarov <dottedmag@dottedmag.net>, Copyright (C) 2006-2007 The Written Word, Inc., Copyright (C) 2007 Eli Fant <elifantu@mail.ru>, Copyright (C) 2009-2023 Daniel Stenberg, Copyright (C) 2008, 2009 Simon Josefsson, Copyright (C) 2000 Markus Friedl, Copyright (C) 2015 Microsoft Corp.

 libssh2 wrapper portions of the software - Copyright (C) 2020 Kiriakos Vlahos (PyScripter)