



# **Embarcadero® PowerSQL™ 1.1 User Guide**

Copyright © 1994-2008 Embarcadero Technologies, Inc.

Embarcadero Technologies, Inc.  
100 California Street, 12th Floor  
San Francisco, CA 94111 U.S.A.  
All rights reserved.

All brands and product names are trademarks or registered trademarks of their respective owners.  
This software/documentation contains proprietary information of Embarcadero Technologies, Inc.; it is provided under a license agreement containing restrictions on use and disclosure and is also protected by copyright law. Reverse engineering of the software is prohibited.

If this software/documentation is delivered to a U.S. Government Agency of the Department of Defense, then it is delivered with Restricted Rights and the following legend is applicable:

Restricted Rights Legend Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of DFARS 252.227-7013, Rights in Technical Data and Computer Software (October 1988).

If this software/documentation is delivered to a U.S. Government Agency not within the Department of Defense, then it is delivered with Restricted Rights, as defined in FAR 552.227-14, Rights in Data-General, including Alternate III (June 1987).

Information in this document is subject to change without notice. Revisions may be issued to advise of such changes and additions. Embarcadero Technologies, Inc. does not warrant that this documentation is error-free.

# Contents

- Welcome to PowerSQL ..... 5
  - Technical Requirements ..... 5
  - Additional Product Resources ..... 6
  - Embarcadero Technologies Technical Support ..... 6
- PowerSQL Overview ..... 9
- Using PowerSQL ..... 10
  - Working with Data Sources ..... 10
    - Register Data Sources ..... 10
    - Browse a Data Source ..... 10
    - View Database Object Properties ..... 11
    - Search for Database Objects ..... 13
    - Filter Database Objects ..... 13
      - Define Data Source-specific Object Filters ..... 14
      - Define Global Database Object Filters ..... 14
    - Drop a Database Object ..... 15
    - Manage Data Source Licenses ..... 15
  - Working with Projects ..... 16
    - Create a New Project ..... 16
    - Open an Existing Project ..... 16
    - Search a Project ..... 17
    - Add Files to a Project ..... 17
    - Delete a Project ..... 18
  - Creating and Editing SQL Files (SQL Editor) ..... 19
    - Create an SQL File ..... 20
    - Open an Existing SQL File ..... 21
    - Working in SQL Editor ..... 21
      - Understanding Automatic Error Detection ..... 22
      - Understanding Code Assist ..... 23
      - Understanding Hyperlinks ..... 26
      - Understanding Code Formatting ..... 26
      - Understanding Code Folding ..... 30
      - Understanding Code Quality Checks ..... 31
    - View Change History ..... 34
    - Revert to an Old Version of a File ..... 35
    - Delete a SQL File ..... 36

Executing SQL Files .....	37
Associate a SQL File with a Data Source .....	37
Execute SQL Code .....	38
Configure a SQL Session .....	39
Troubleshooting .....	39
View Log Details .....	41
Maintain Logs .....	42
Filter Logs .....	43
Import and Export Error Logs .....	45
Find and Fix SQL Code Errors .....	46
Find and Fix Other Problems .....	46
Configuring PowerSQL .....	47
Initial Setup .....	47
Specify a Workspace .....	47
License PowerSQL .....	47
Customizing PowerSQL (Preferences) .....	48
Set Cache Configuration Preferences .....	48
Set SQL Editor Preferences .....	49
Set SQL Execution Preferences .....	49
Set Code Assist Preferences .....	49
Set Code Formatter Preferences .....	49
Set Results View Preferences .....	50
Set Syntax Coloring Preferences .....	50
Set SQL Filter Preferences .....	50
Reference .....	51
Database Objects .....	51
DBMS Connection Parameters by Platform .....	60
IBM DB2 LUW .....	60
Microsoft SQL Server .....	60
JDBC Connection Parameters .....	62
Oracle Connection Parameters .....	62
Sybase Connection Parameters .....	62

# Welcome to PowerSQL

Embarcadero PowerSQL simplifies SQL development for application developers with many features for improving productivity and reducing errors. A rich SQL IDE with code completion, real-time error checking, code formatting and sophisticated object validation tools helps streamline coding tasks. PowerSQL's user interface helps improve overall productivity with database object look up functionality and side-by-side views of application code and SQL code. PowerSQL offers native support for IBM® DB2® for LUW, Oracle®, Microsoft® SQL Server and Sybase® as well as JDBC support for other DBMS . PowerSQL is available as a stand-alone application or as an Eclipse plug-in.

## Key Features

- Migrate data sources from Eclipse DTP (Data Tools Project) or Quest TOAD or automatically detect them on the network.
- Data Source Explorer enables users to easily navigate, search, extract DDL, execute commands, and browse outline views without opening SQL files.
- Code Assist provides context-sensitive database object names to eliminate misspellings and the need to look them up. Also available offline.
- Formatting Profiles ensure consistent, quality code layout for easy review and extension. Profiles can be customized and shared.
- Real-time SQL validation ensures there are no parser violations or object names used that are not found in the database.

## Technical Requirements

Before installing PowerSQL, verify that your environment meets the following requirements.

### Hardware Requirements

The following minimum hardware requirements are required to run PowerSQL:

- Pentium 4-Level Processor
- 1024 MB of memory
- 500 MB of disk space
- 1024 x 768 screen resolution

### Operating System Requirements

PowerSQL supports the following operating systems:

- Microsoft Windows XP (x86-32, Win32)
- Microsoft Vista (x86-32, Win32)
- Microsoft Windows Server 2003
- Red Hat Enterprise Linux 5.0, x86-32, GTK 2
- SuSe Linux Enterprise Server (x86) GTK+ 2.x

## DBMS Support

PowerSQL provides native support for the following platforms (no additional DBMS client software is required):

- Generic JDBC
- IBM DB2 LUW 8.0 - 9.0
- Microsoft SQL Server 2000 and 2005
- Oracle 8i - 11g
- Sybase 12.5 - 15.0

## Installation Notes

PowerSQL can be installed as a standalone application (RCP installation) or as an Eclipse plug-in (Plug-in Installation). Eclipse is an open source development framework that supports application plug-ins to provide additional functionality.

The Eclipse plug-in version of PowerSQL requires Eclipse version 3.3 or higher, and Sun Java Standard Edition 5.0 Update 11 or later in addition to regular system requirements.

Before installing the plug-in version of PowerSQL, ensure that Eclipse and Java is installed on your machine. You can download Eclipse from the following Web site: <http://www.eclipse.org/downloads>.

## Additional Product Resources

The Embarcadero Web site is an excellent source for additional product information, including white papers, articles, FAQs, discussion groups, and the Embarcadero Knowledge Base.

Go to [www.embarcadero.com/support](http://www.embarcadero.com/support), or click any of the links below, to find:

- [Documentation](#)
- [Online Demos](#)
- [Technical Papers](#)
- [Discussion Forums](#)
- [Knowledge Base](#)

## Embarcadero Technologies Technical Support

If you have a valid maintenance contract with Embarcadero Technologies, the Embarcadero Technical Support team is available to assist you with any problems you have with our applications. Our maintenance contract also entitles registered users of Embarcadero Technologies' products to download free software upgrades during the active contract period.

To save you time, Embarcadero Technologies maintains a [Knowledge Base](#) of commonly-encountered issues and hosts [Discussion Forums](#) that allow users to discuss their experiences using our products and any quirks they may have discovered.

To speak directly with Embarcadero Technical Support, see [Contacting Embarcadero Technologies Technical Support](#) below.

**NOTE:** Evaluators receive free technical support for the term of their evaluation (14 days).

## Contacting Embarcadero Technologies Technical Support

When contacting Embarcadero Technologies Technical Support please provide the following to ensure swift and accurate service:

### Personal Information

- Name
- Company name and address
- Telephone number
- Fax number
- Email address

### Product and System Information

- Embarcadero product name and version number. This information is found under Help, About.
- Your client operation system and version number.
- Your database and version number.

### Problem Description

A succinct but complete description of the problem is required. If you are contacting us by telephone, please have the above information, including any error messages, available so that an Embarcadero Technical Support Engineer can reproduce the error and clearly understand the problem.

There are three ways to contact Embarcadero's Technical Support department:

- Via the [Web](#)
- Via [Phone](#) (for Standard and Professional editions only)
- Via [Email](#)

### Via the Web

Embarcadero Technical Support provides an online form that lets you open a Support case via the Web. To access this form, go to [http://www.embarcadero.com/support/open\\_case.jsp](http://www.embarcadero.com/support/open_case.jsp).

We usually acknowledge the receipt of every case on the same day, depending on the time of submission.

### Via Phone

Telephone support is available for the Standard and Professional editions only.

### United States

Embarcadero Technologies Technical Support phone number is (415) 834-3131. Select option 2 and then follow the prompts. The hours are Monday through Friday, 6:00 A.M. to 6:00 P.M. Pacific time.

For licensing issues, including Product Unlock Codes, call (415) 834-3131. Select option 2 and then follow the prompts. The hours are Monday through Friday, 6:00 A.M. to 6:00 P.M. Pacific time.

The Embarcadero Technologies Technical Support fax number is (415) 495-4418.

### EMEA

Embarcadero Technologies Technical Support phone number is +44 (0)1628 684 499. The hours are Monday to Friday, 9 A.M. to 5:30 P.M. U.K. time.

For licensing issues, including Product Unlock Codes, call +44 (0)1628-684 494. The hours are Monday to Friday, 9 A.M. to 5:30 P.M. U.K. time.

The Embarcadero Technologies Technical Support fax number is +44 (0)1628 684 401.

## **Via Email**

### **United States**

Depending on your needs, send your email to one of the following:

- [support@embarcadero.com](mailto:support@embarcadero.com) - Get technical support for users and evaluators.
- [upgrade@embarcadero.com](mailto:upgrade@embarcadero.com) - Request upgrade information.
- [key@embarcadero.com](mailto:key@embarcadero.com) - Request a product key.
- [wish@embarcadero.com](mailto:wish@embarcadero.com) - Make a suggestion about one of our products.

### **EMEA**

Depending on your needs, send your email to one of the following:

- [uk.support@embarcadero.com](mailto:uk.support@embarcadero.com) - Get technical support for users and evaluators.
- [uk.upgrade@embarcadero.com](mailto:uk.upgrade@embarcadero.com) - Request upgrade information.
- [uk.key@embarcadero.com](mailto:uk.key@embarcadero.com) - Request a product key.
- [uk.wish@embarcadero.com](mailto:uk.wish@embarcadero.com) - Make a suggestion about one of our products.



# PowerSQL Overview

Embarcadero PowerSQL simplifies SQL development for application developers, with many features for improving productivity and reducing errors. A rich SQL IDE with code completion, real-time error checking, code formatting, and sophisticated object validation tools helps streamline coding tasks. PowerSQL's user interface helps improve overall productivity with database object look up functionality and side-by-side views of application code and SQL code. PowerSQL offers native support for IBM® DB2® for LUW, Oracle®, Microsoft® SQL Server, and Sybase® as well as JDBC support for other platforms. PowerSQL is available as a stand-alone application or as an Eclipse plug-in.

## Key Features

- Migrates data sources from Eclipse DTP (Data Tools Project) or Quest TOAD or automatically detect them on the network.
- Easily navigate, search, extract DDL, execute commands, and browse outline views without opening SQL files, with Data Source Explorer.
- Provides context-sensitive database object names to eliminate misspellings and the need to look them up using Code Assist, which is also available offline.
- Ensures consistent, quality code layout for easy review and extension using Formatting Profiles. Profiles can be customized and shared.
- Real-time SQL validation ensures there are no parser violations or object names used that are not in the database.

# Using PowerSQL

This section contains detailed instructions for [Working with Data Sources](#), [Working with Projects](#), [Creating and Editing SQL Files \(SQL Editor\)](#), [Executing SQL Files](#), and [Troubleshooting](#).

## Working with Data Sources

The **Data Source Explorer** provides a browseable tree view of all PowerSQL-registered data sources and associated database objects. Data Source Explorer is automatically populated with data sources that have been pre-registered in *Rapid SQL*®, *DBArtisan*®, or previous versions of PowerSQL. If PowerSQL cannot detect a data source, you can register it manually.

### Register Data Sources

When PowerSQL is started, it automatically imports the data source catalog created by any previously installed Embarcadero products or other instances of PowerSQL.

The first time PowerSQL is started, it also scans the local machine for the client software of supported third-party DBMS. These data sources are automatically added to the data source catalog. To manually initiate a scan later, click the **Auto-Discover Data Sources** icon at the top of Data Source Explorer.

To register a data source manually, right-click **Managed Data Sources** in the Data Source Explorer tree, select **New > Data Source**, and enter the connectivity parameters as prompted. For additional information on these parameters, see [DBMS Connection Parameters by Platform](#).

Once registered, the data source appears in the **Data Source Explorer** view. If you have created more than one workspace, they all share the same data source catalog.

Once a data source has been registered, the connection parameters are stored locally. In some cases, a user ID and password are required to connect to a registered data source. PowerSQL can encrypt and save user IDs and passwords to connect automatically.

### Browse a Data Source

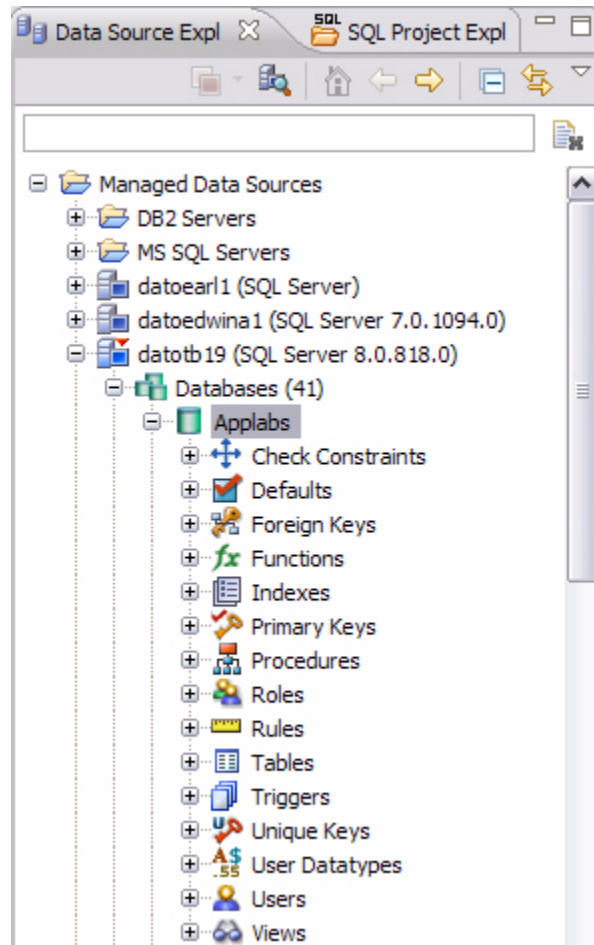
You can drill down in the Data Source Explorer tree to view registered databases on a server, and view tables, and other objects in a database. Additionally, you can view the structure of individual objects such as the columns and indexes of a table. Right-click the object for a menu of available commands, such as **Extract to Project**, which creates a new SQL file containing the object's DDL.

In most cases, whenever you browse a data source, PowerSQL requires login information in order to connect with the data source. Enter a valid user name and password in the fields provided. The **Auto Connect** option retains your login credentials for future connections to the same data source.

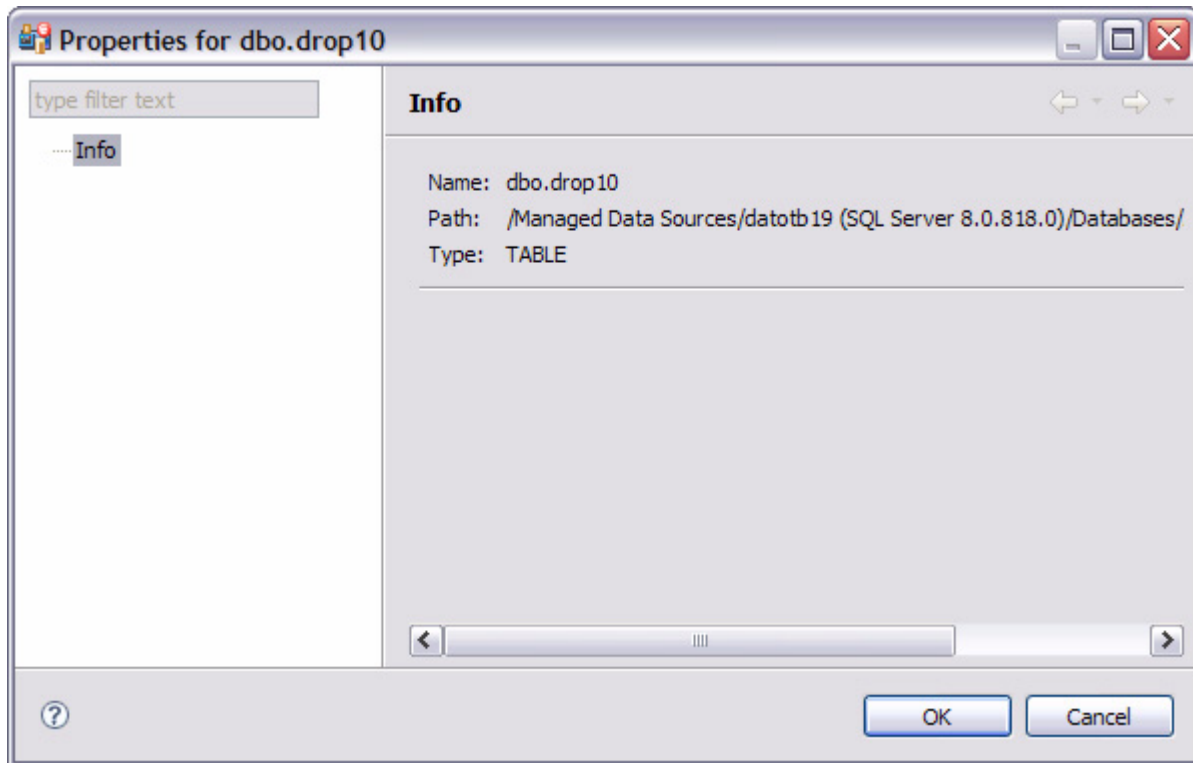
**NOTE:** PowerSQL Personal Edition and PowerSQL Standard Edition can connect to only a limited number of registered data sources. If you run out of licenses, you can reallocate an existing license to a different data source. For more information, see [Manage Data Source Licenses](#).

## View Database Object Properties

All objects in **Data Source Explorer** contain properties as they relate to the PowerSQL application.



PowerSQL Object Properties are viewed via the **Properties** dialog. The dialog is accessed by right-clicking the object in Data Source Explorer.



**To view Data Source Explorer object properties:**

The **Info** properties node is accessed by right-clicking a data source in Data Source Explorer.

The dialog displays the following three properties:

- **Name:** The name of the node as it appears in the Database Explorer. This value cannot be modified.
- **Path:** The folder path and file name where the object is stored on the system. This value cannot be modified.
- **Type:** The type of data source object that the node represents. For example, the **Type** value for a table node would be **TABLE**. This value cannot be modified.

As well, each node representing the actual data source connection (the uppermost parent in a list of data source objects), contains additional properties in addition to the **Info** node and its respective properties. With the exception of the Configuration node, these values can be modified in the **Properties** dialog.

The Configuration node is composed of:

- **Data Source Name**
- **Data Source Type**
- and three subnodes: **Connection Information**, **Data Source Information**, and **Security Parameters**.

These nodes are identical to the parameters used to initially define the data source during the data source registration process. For more information on these values and how to modify them, see [Register Data Sources](#).

The **SQL Filter** node enables a developer to place filters on data source objects that appear in the Database Explorer. For more information, see [Filter Database Objects](#).

## Search for Database Objects

Database object searches rely on the object cache when returning results. By default, caching is set to configure only parts of a database. To configure the object cache to expand object searches, see [Set Cache Configuration Preferences](#).

- 1 Select **Search > Database**. By default, the search scope is all currently connected databases. Under **Specify the scope for the search**, clear any databases or server check boxes you do not want to search.
- 2 Specify the search criteria:
  - Type the value to search for in the **Search String** field. Use the \* character to indicate wildcard string values and the ? character to indicate wildcard character values.
  - Select **Case Sensitive** to indicate to the search function that you want case sensitivity to be a factor when searching for appropriate string matches.
  - Select **Search Cached Data** to indicate that the search function should read the database cache. This increases the performance of the search function and will typically result in faster returns on any hits the search might make.
  - Select **Apply SQL Filters** to apply any relevant database or vendor filters to the search.
  - Choose **Declarations**, **References**, or **All Occurrences** to specify what the search is restricted to in terms of database objects.
  - A declaration is an instance where an object is declared. For example, an object is declared in a CREATE table.
  - A reference is an instance where an object is used or referred to. For example, an object is referred to in a procedure or as a foreign key in a table.
  - Choose **All Occurrences** to return both declarations and references in the search results.
  - Use the check boxes beside the database object panel to select and deselect the specific database objects that you want to be included in the search process.
- 3 Click **Search**.

The results of your search are generated in the **Search** view.

## Filter Database Objects

Filters can be placed on data sources and corresponding data source objects to restrict their display in Data Source Explorer. This feature is useful if you have data sources that contain large numbers of database objects. You can apply filters to view only the schema objects you need for the development process.

There are two types of data source filters available:

- **Global filters** that affect all registered data sources in the PowerSQL development environment.
- **Data Source specific filters** affect only the specified data source for which they are defined.

In both cases, data source object filters are defined via the **Object Filter Manager**, through the development of filter templates. Once defined, filter templates can be activated and deactivated as you need them.

Several filter templates can be combined at a global level or applied to a specific data source.

### See also

[Define Global Database Object Filters](#)

[Define Data Source-specific Object Filters](#)

## Define Data Source-specific Object Filters

Data source-specific object filters affect only the specified data source.

### To define data source-specific filters:

- 1 In **Data Source Explorer**, right-click the data source and select **Properties**.

The **Properties** dialog appears.

- 2 Select the **SQL Filter** node and deselect **Enable Data Source Specific Settings**. The other controls on the dialog become enabled.
- 3 Click **Add**. The **Filter Template** dialog appears.
- 4 Specify the parameters of the filter.
  - In the **Name** field, enter the name of the filter as you want it to appear in the selection window on the **SQL Filter** node.
  - The **Database Type** pane provides a list of data source objects. Deselect the data source objects that this template filters so that they do not appear in **Database Explorer** when displaying data source objects for the data source.
  - Click **New** to add filter parameters for data source object properties. The **New SQL Filter Predicate** dialog appears.
  - Use the **Property** and **Operator** fields to supply the filter criteria. **Property** specifies whether the value is a **Name** or **Schema**, and **Operator** specifies the matching type of the filter syntax. (**Equals**, **Not Equals**, **Like**, **Not Like**, **In**, **Not In**)
  - In the **Value** field, enter the full or partial syntax of the property or properties you want to filter in **Data Source Explorer**.

Click **OK**. The filter property specification is added to the Filter Template.

- 5 When you have finished defining the filter template, click **OK**. The template name is added to the **Properties** dialog. It can be enabled and disabled by selecting or deselecting the check box beside its name, respectively.

## Define Global Database Object Filters

Global filters affect all registered data sources in the PowerSQL development environment. When you create and apply a global filter to a platform vendor in PowerSQL, all databases associated with that vendor are affected by the filter, as defined.

Individual global filter templates are separated, by supported data source platform, on tabs in the **SQL Filter** window. Select the appropriate tab to view existing filter templates or add new ones, as needed.

### To define a global filter:

- 1 Select **Window > Preferences** from the Main Menu. The **Preferences** dialog appears.
- 2 Expand the **PowerSQL** node and select the **SQL Filter** subnode. The **SQL Filter** pane appears.
- 3 Click **New**. The **Filter Template** dialog appears.

- 4 Specify the parameters of the filter template:
  - In the **Name** field, enter the name of the filter as you want it to appear in the selection window on the **SQL Filter** node.
  - The **Database Type** pane provides a list of data source objects. Deselect the data source objects that this template filters so that they do not appear in **Database Explorer** when displaying data source objects for the data source.
  - Click **New** to add filter parameters for data source objects properties. The **New SQL Filter Predicate** dialog appears.
  - Use the **Property** and **Operator** fields to supply the filter criteria. **Property** specifies whether the value is a **Name** or **Schema**, and **Operator** specifies the matching type of the filter syntax. (**Equals, Not Equals, Like, Not Like, In, Not In**)
  - In the **Value** field, enter the full or partial syntax of the property or properties you want the template to filter in **data source Explorer**.
- 5 Click **OK**. The filter property specification is added to the Filter Template.
- 6 When you have finished defining the filter template, click **OK**. The template name is added to the **Properties** dialog. It can be enabled and disabled by selecting or de-selecting the check box beside its name, respectively.

Data Source object filters are added and removed from the development environment by selecting and de-selecting the checkboxes associated with each filter template on both the global and data source-specific dialogs.

## Drop a Database Object

To delete an object permanently from a database, right-click the object in **Data Source Explorer** and choose **Drop** from the menu. The **Drop Wizard** asks you to confirm removal of the object and provides a DDL preview of the deletion code.

## Manage Data Source Licenses

**NOTE:** The following information is not relevant to the Professional edition of PowerSQL.

- You can register any number of data sources.
- The first data sources you connect to are automatically allocated licenses from your pool of data source licenses (five licenses for the Personal Edition, ten for the Standard Edition).
- Once all the licenses in the pool have been assigned, attempting to connect to an unlicensed data source brings up a message that you may purchase additional licenses or reallocate the ones you have.

### To reallocate a license:

- 1 Click **Open License Manager** in the dialog prompt, or select **Help > Embarcadero License Manager > Data Sources**.
- 2 In the **Product** column, double-click **PowerSQL**.
- 3 In the **Data Source Licenses** dialog, clear the check box of a data source to be unlicensed, and select the data source you want to connect to.
- 4 Click **OK**, and then **Close**.

## Working with Projects

You create projects to organize and store SQL development files. The purpose of projects is to keep your work-in-progress files organized, as well as maintain a common directory structure when developing code and executing files on registered data sources. Once a file has been developed and is ready for deployment, that file can then be executed on a registered data source.

**SQL Project Explorer** is used to view and access files. It uses a tree view to display the project as a series of folder directories with a folder labeled with the project name as the parent directory, and with project categories, and associated project files as its children.

All files in a project are organized under the following categories:

- **Connections:** List the connections of any given SQL file of a data source associated with the project.
- **Creation Scripts:** Provide DDL statements and statements that define database objects.
- **General SQL:** Provide a category for all other SQL files that are not used in database object creation. This includes DML files, and so on.
- **Large Scripts:** Contain all files larger than the currently set SQL Editor preference. The file size limit can be modified on the **Preferences** panel by selecting **Window > Preferences** in the Main Menu.

Physically, the projects and files you create as you work in PowerSQL are stored under the Workspace directory you specified at the prompt when PowerSQL was started. The directory and files can be shared, and other tools may be used to work on the files, outside the PowerSQL development environment.

You can move existing files within a project by clicking and dragging the file you want to move in the **Project Explorer** from one node to another, or via the **File > Move** command.

## Create a New Project

- 1 Select **File > New > SQL Project** from the PowerSQL **Main Menu**. The **New Project Wizard** appears.
- 2 Enter the appropriate information in the fields provided:
  - **Name:** Enter the name of the project as you want it to display in the Project Explorer view.
  - **DBMS Platform:** Select the data source platform to which the new project will be associated. This enables PowerSQL to properly parse SQL development code for project files.
  - **Location:** When selected, the **Use Default Location** check box indicates the project is to be created under the currently selected Workspace. Deselect the check box and specify a new folder path if you do not want to create the project in the currently selected Workspace.
- 3 Click **Finish**. The new project icon appears in the **Project Explorer** view under the name that you specified. If you did not select Use Default Location, the project will appear in the appropriate Workspace when you open it in PowerSQL.

**NOTE:** Alternatively, you can select **New > SQL Project** from the Main Menu or click the **New Project** icon in the Tool Bar to create a new project.

## Open an Existing Project

You can open projects by navigating to **SQL Project Explorer** and expanding the node of the project that contains the files you want to access.

Below each project name are a series of nodes that categorize any existing SQL files by development type:

- **Connections:** Lists the connections of any given SQL file of a data source associated with the project.



- **Creation Scripts:** General data source object development scripts. This node contains DDL statements and statements that define database objects.
- **General SQL:** Provides a category for all other SQL files that are not used in database object creation. DML files, etc.
- **Large Scripts:** Contains all files larger than the currently set SQL Editor preference. The file size limit can be modified on the **Preferences** panel. (Choose **Window > Preferences** in the Main Menu to access the panel.)

**NOTE:** Physically, the projects and files you create as you work in PowerSQL are stored under the project directory that you specified at the prompt when the project was created. The directory and files can be shared, and other tools may be used to work on the files, completely exempt from the PowerSQL development environment.

## Search a Project

- 1 Select **Search > File**.
- 2 Specify the search criteria:
  - Type the value to search in the **Containing Text** field. Use the \* character to indicate wildcard string values, the ? character to indicate wildcard character values, and the \ character to indicate an escape character for literals (\* ? \).
  - Select **Case Sensitive** and indicate to the search function that it should take into account case when searching for appropriate string matches.
  - Select **Regular Expression** to indicate to the search function that the string is a regular function.
  - In the **File Name Pattern** field, specify the extension name of the files to search for explicitly. If the value in this field is a \* character, the search function searches all files regardless of extension. Manually type in the extensions to indicate file type (separate multiple file types with commas), or click **Choose** and use the **Select Types** dialog to select the file extensions the process will search for the string by.
  - Select **Consider Derived Resources** to include derived resources in the search.
  - Select **Workspace** or **Working Set** to choose the scope of the search. If you choose **Working Set**, specify the name of the defined working set manually, or click **Choose** and navigate to the working set you want to search for in the provided string.
- 3 Click **Search**. The results of your search are generated in the **Search** view on the Workbench.

## Add Files to a Project

Existing files that reside in directories outside of the workspace can be added to a project via the following methods:

- Dragging and dropping the file set from a system directory to SQL Project Explorer.
- Copying and pasting the file set from a system directory to SQL Project Explorer.
- Executing the **Import** command.

**To drag/drop or copy/paste files from a system directory to SQL Project Explorer:**

- 1 With the SQL Project Explorer view open, navigate to the directory where the files you want to add to the project are located on the system.
- 2 Drag and drop the files you need from Windows Explorer into SQL Project Explorer. The files appear in the tree view under the appropriate categories.

**NOTE:** Alternatively, you can use the **Copy** command on the files you want to add in Windows Explorer, and then right-click the Project Explorer and select **Paste** from the menu. The files appear in the tree view under the appropriate categories.

**To use the Import command:**

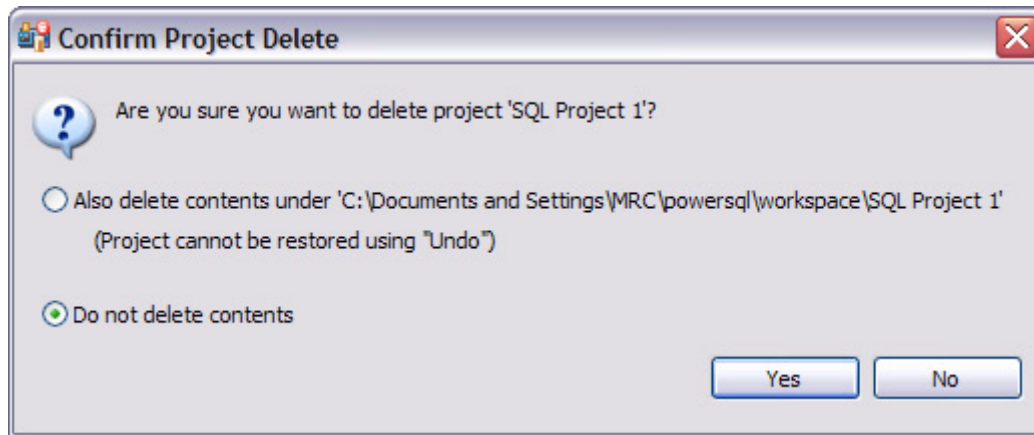
- 1 Right-click anywhere on the Project Explorer and select **Import**. The **Import** dialog appears.
- 2 Expand the **General** node and double-click **File System**. A dialog containing the import specification parameters appears.
  - In the **From directory** field, manually type the directory location of the files you want to import to Project Explorer, or click **Browse** and navigate to the appropriate folder. The panels below the field populate with the folder selection and a list of suitable files contained in that folder. Use the check boxes beside each folder and file to specify what folders/files you want the import function to add in Project Explorer.
  - In the **Into folder** field, manually type the name of the folder within Project Explorer where you want to import the files specified in the panels above, or click **Browse** and navigate to the appropriate folder.
  - Select the **Overwrite existing resources without warning** check box if you do not want to be prompted when the import process overwrites Project Explorer files that contain the same name as the imported files.
  - Choose **Create complete folder structure** or **Create selected folders only**, depending on whether you want the import process to build the folder structure of the imported directory automatically, or only create those folders you selected in the panels above, respectively.
- 3 Click **Finish**. The import process moves all selected folders and files into Project Explorer and thus into the PowerSQL development environment.

**NOTE:** In addition to accessing the **Import** command via the shortcut menu, you can also access the **Import** dialog by choosing **File > Import ...** from the Main Menu.

## Delete a Project

You can delete a project by right-clicking its folder in the SQL Project Explorer and selecting **Delete**.

When you delete a project, PowerSQL will prompt you with a dialog that asks you to confirm the deletion of the project, and offers you the option of deleting the project from the PowerSQL interface, or deleting the project from the system.



- If you select **Do not delete contents**, the files and directory structure will be removed from SQL Project Explorer, but they will still exist on your machine.
- If you select **Also delete contents** ..., the files and directory structure will be removed from SQL Project Explorer **and** deleted from your machine.

## Creating and Editing SQL Files (SQL Editor)

The **SQL Editor** is a Workbench interface component that enables the development, viewing, and formatting of SQL code.

```

CREATE TABLE dbo.benson
(
    job char(8)          NOT NULL,
    sal numeric(38,0)   NOT NULL,
    loc numeric(38,0)   NOT NULL,
    CONSTRAINT pjob
    PRIMARY KEY CLUSTERED (job)
)
go
IF OBJECT_ID('dbo.benson') IS NOT NULL
    PRINT '<<< CREATED TABLE dbo.benson >>>'
ELSE
    PRINT '<<< FAILED CREATING TABLE dbo.benson >>>'
go
  
```

SQL Editor contains context-sensitive command menus that are tailored with pertinent functionality for the specified file format.

If SQL Editor does not recognize a selected file format, PowerSQL automatically launches the file externally in the system default application. External editors are not embedded in the Workbench. For example, on most machines, the default editor for HTML files is the system Web browser. SQL Editor does not, by default, recognize HTML files, and opening an HTML file from the Workbench launches the file in an instance of the Web browser instead of the Editor.

Any number of instances of SQL Editor can be open on the Workbench at the same time. Multiple instances of SQL Editor displaying different content may be open in the same Workbench. These instances will be stacked by default, but can also be tiled side-by-side so the content of various files can be viewed simultaneously for comparison or multi-tasking purposes. When an instance of SQL Editor is active, the Workbench Main Menu automatically contains commands applicable to the file format. If a view is active, SQL Editor commands are disabled automatically, except when commands are still valid between the selected view and the file displayed in the interface.

When working with code in SQL Editor, the window contains a number of features that provide an increase in the efficiency and accuracy of code development. The following syntax highlighting changes are automatically applied to code as a user adds lines in the interface.

Code	Formatting
Comments	Green font, italics
SQL Commands	Dark blue font
Coding Errors	Red underline
Strings	Red font
Non-Executable Command Line Commands	Aqua font

Single line and multiple line comments appear in different colors.

Furthermore, SQL Editor provides two column bars, one on either side of the code window. The purple change bar in the left-hand column indicates that the line of code has been modified. Hover over the change bar to display the original code text. The red square in the right-hand column indicates that there are errors in the code window. Hover the mouse over the square to view the error count. Click the red bar in this column to navigate directly to the line in which the SQL Editor detects the error. SQL Editor automatically highlights the appropriate code. Non-executable command line commands are displayed in a different formatting style than SQL commands. Syntactic and semantic errors are also highlighted.

SQL Editor also features dynamic error detection, object lookup and suggestion features, code folding, and auto-formatting. SQL Editor is able to identify different areas in a statement, and enables users to retrieve subclauses, resolve table aliases, and dynamically return lists of tables, views, and columns, as needed.

See also:

[Working in SQL Editor](#)

## Create an SQL File

- 1 Create or open a SQL project.
- 2 Select **File > New > SQL File**. A blank instance of SQL Editor appears.

**NOTE:** If you are not in a SQL project when you create a new SQL file, it will not open in SQL Editor.

## Open an Existing SQL File

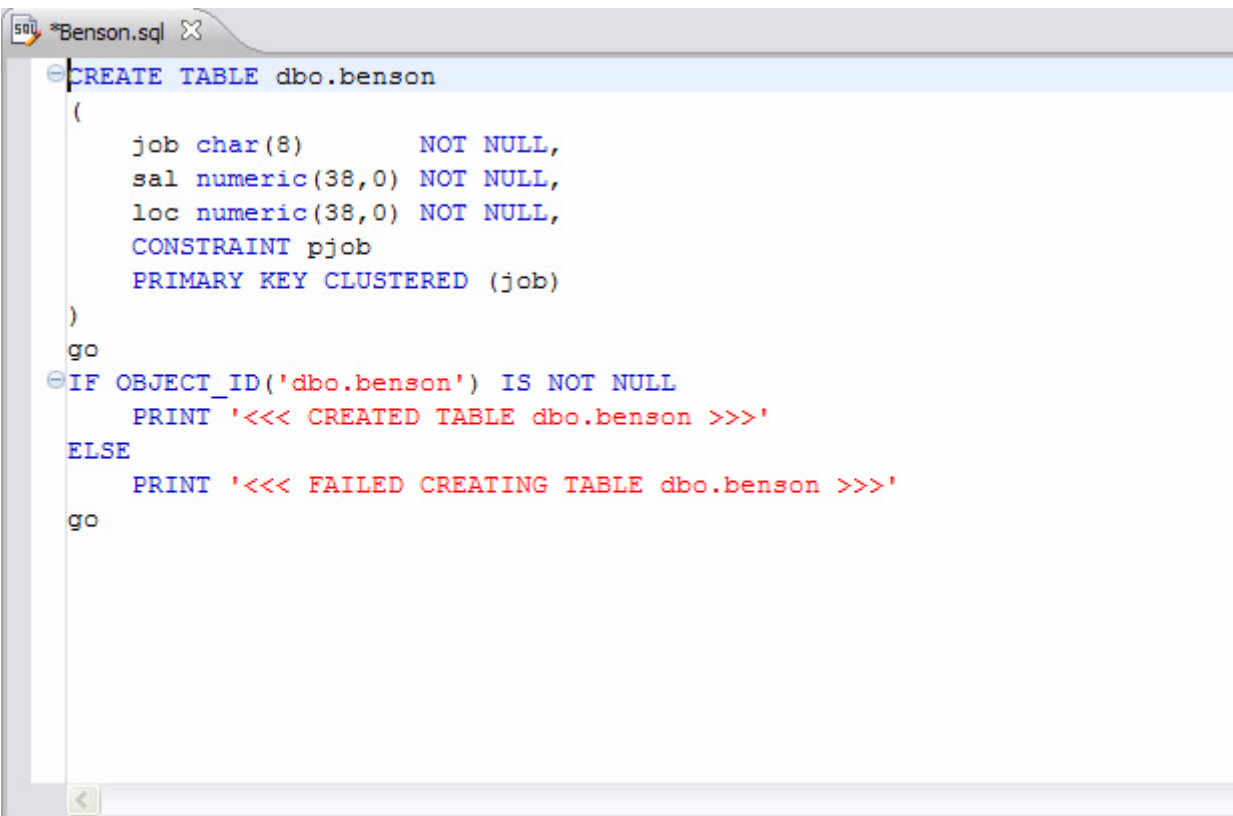
- 1 Open the SQL project containing the file, or that you want to contain the file.
- 2 If necessary, add the file to the project (see [Add Files to a Project](#)).
- 3 In the SQL Project Explorer, double-click the file to open it in SQL Editor.

## Working in SQL Editor

**SQL Editor** handles SQL code formats and contains context-sensitive command menus, tailored with pertinent functionality for development purposes. Other files may be opened in PowerSQL, as well, but these are handled by other editors.

For example, if a text file is opened in the Workbench, PowerSQL detects and opens the contents of that file in a text editor viewer with pertinent commands for that file type.

Any number of instances of **SQL Editor** can be active on the Workbench at the same time. Multiple instances of **SQL Editor** displaying different content may be active on the same Workbench. These instances will be stacked, by default, but can also be tiled side-by-side, so the content of various files can be view simultaneously for comparison or multi-tasking purposes. When an instance of **SQL Editor** is active, the Main Menu contains commands applicable to the file format. If a view is active, SQL Editor commands are disabled automatically, except when commands are still valid between the selected view and the file displayed in the interface.



```

CREATE TABLE dbo.benson
(
    job char(8)          NOT NULL,
    sal numeric(38,0)   NOT NULL,
    loc numeric(38,0)   NOT NULL,
    CONSTRAINT pjob
    PRIMARY KEY CLUSTERED (job)
)
go
IF OBJECT_ID('dbo.benson') IS NOT NULL
    PRINT '<<< CREATED TABLE dbo.benson >>>'
ELSE
    PRINT '<<< FAILED CREATING TABLE dbo.benson >>>'
go
  
```

Among the commands SQL Editor supports via the right-click menu:

- **Revert File:** Automatically restores the working file to the original text as it appeared the last time the **Save** command was issued.
- **Shift Right/Shift Left:** Indents the line of code in the working file to the right or left, respectively.

- **Toggle Comments:** Hides or displays comments in the code of the working file, depending on the current hide/show state.
- **Add Block Comment/Remove Block Comment:** A block comment is used to insert a comment into SQL code that spans multiple lines and begins with a forward slash and asterisk. While block comments are typically used to insert a command that spans multiple lines, some developers find them more useful than line comments, especially if a development team is using different text editors on an individual basis. Moving code from one text editor to another often breaks line comments in the middle of a line and causes errors. Block comments can be broken without causing errors.

**NOTE:** In addition to editing commands, some commands such as extract, drop, and execute can be accessed by right-clicking over statements in SQL code that are performed on specific tables, views, and columns. These commands will appear automatically in the appropriate menu when the code is highlighted. Full information on using these commands is found elsewhere in this documentation, based on the task each executable performs.

- **Explain Plan:** An explain plan details the steps that occur in SELECT, UPDATE, INSERT, and DELETE statements and is primarily used to determine the execution path followed by the database in its SQL execution.

**See also:**

[Understanding Automatic Error Detection](#)

[Understanding Code Assist](#)

[Understanding Hyperlinks](#)

[Understanding Code Formatting](#)

[Understanding Code Folding](#)

[Understanding Code Quality Checks](#)

## Understanding Automatic Error Detection

SQL Editor orders and classifies SQL statements. This enables it to edit code as you work within SQL Editor and highlight errors and typographical errors in “real time”. As you work, SQL Editor examines each clause in a statement and provides error reporting and other features as required.

SQL Editor identifies the following clauses and elements:

- **SELECT:** Specifies the field, constants, and expressions to display in the query results.
- **FROM:** Specifies one or more tables containing the data that the query retrieves from.
- **WHERE:** Specifies join and filter conditions that determine the rows that query returns. Join operations in a WHERE clause function in the same manner as JOIN operations in a FROM clause.
- **GROUP BY:** Specifies one or more columns used to group rows returned by the query. Columns referenced in the SQL SELECT statement list, except for aggregate expressions, must be included in the GROUP BY clause. You cannot group by Memo, General or Blob fields.
- **HAVING:** Specifies conditions that determine the groups included in the query. If the SQL statement does not contain aggregate functions, you can use the SQL SELECT statement containing a HAVING clause without the GROUP BY clause.
- **ORDER BY:** Specifies one or more items used to sort the final query result set and the order for sorting the results.

As you develop code in SQL Editor, it automatically detects semantic errors on a line-by-line basis. Whenever an error is detected, the line is flagged by an icon located in the left-hand column of the editor.

```

CREATE TABLE dbo.benson
(
    job CHAR (8) NOT NULL,
    sal NUMERIC (38, 0) NOT NULL,
    loc NUMERIC (38, 0) NOT NULL,
    CONSTRAINT pjob PRIMARY KEY CLUSTERED (job)
)

go
IF OBJECT_ID('dbo.benson') IS NOT NULL
    PRINT '<<< CREATED TABLE dbo.benson >>>'
ELSE
    PRINT '<<< FAILED CREATING TABLE dbo.benson >>>'

go
SELECT *
FROM tbo.benson;
    
```

Additionally, all semantic errors detected in SQL Editor are displayed in the **Problems** view.

Description	Resource	Path	Location
Errors (3 items)			
An unexpected token "<<< FAILED CR	Benson.sql	SQL Project 1	line 14
An unexpected token "" was found. Exp	File.sql	SQL Project 1	line 6
Table benson cannot be resolved on 'da	Benson.sql	SQL Project 1	line 19

Right-click the an error and select **Go To** in order to find the error. PowerSQL opens and navigates to the specific line of code containing the specified error.

## Understanding Code Assist

When SQL Editor has finished analyzing a partial piece of code, it displays a list of data source objects for you to select from.

SQL Editor takes the following into consideration when analyzing code for a list of possible data source objects for insertion:

- Text to be inserted
- Original text to be replaced
- Content assist request location in original text
- The database object represented by the insertion text

Generally, insertion suggestions use the following format:

```
<insertion_text > - <qualification_information >
```

Content assist is available for SELECT, UPDATE, INSERT, and DELETE statements.

The following table displays a list of all possible object suggestions, and the format in which SQL Editor inserts the suggestions into a statement:

Object Suggestion	Syntax/Example
Table	(TABLE) [catalog].[schema] EMPLOYEE - (TABLE)HR
Alias Table	(TABLE ALIAS) [catalog].[schema]tableName EMPLOYEE-(TABLE ALIAS)HRJOBS
Column	datatype - (Column) [catalog].[schema].tableName JOB_TITLE: varchar(20)-(Column)HRJOBS
Alias Column	datatype - (COLUMN ALIAS) [catalog].[schema].tableName. columnName JOB_TITLE:int-(COLUMN ALIAS)HR.JOBS.JOB_ID
Schema	(SCHEMA) [catalog] dbo-(SCHEMA)NorthWind
Catalog	(CATALOG)

SQL Editor detects incomplete or erroneous code, processes the code fragments, and then attempts to apply the appropriate logic to populate the code.

As code is typed into SQL Editor, the application 'reads' the language and returns suggestions based on full or partial syntax input.

Depending on the exact nature of the code, the automatic object suggestion feature behaves differently; this enables SQL Editor to provide reasonable and 'intelligent' suggestions on coding.

The following chart displays the possible statement fragments that SQL Editor will attempt to suggest/populate with objects:



Statement Fragment Elements	Object Suggestion Behavior
SELECT	A list of tables, when selected automatically, prompts the user to select a column.
UPDATE and DELETE	A list of tables appears in the FROM and/or WHERE clause.
INSERT	A list of tables and views appears in the INSERT INTO and OPEN BRACKET clause prior to values. A list of columns based on the table or view name appears in the OPEN BRACKET or VALUES clause.

In addition to DML statements, SQL Editor also suggests objects based on specific fragmented syntax per line of code:

Statement Syntax	Object Suggestion Behavior
A partial DML statement (for example <b>SEL ...</b> indicates a fragment of the <b>SELECT</b> clause)	The keyword is completed automatically, assuming SQL Editor can match it. Otherwise, a list of suggested keywords is displayed. If the preceding character is a period, and the word prior is a table or view, a list of columns appears. If the word being typed is a part of a table name (denoted by a schema in front of it) the table name is autocompleted. If the word being typed has a part of a column name (denoted by a table in front of it) the column name is autocompleted.
Without typing anything.	A list of keywords appears.
A period is typed.	If the word prior to the period is a name of a table or view, a list of columns is displayed. If the word prior to the period is a schema name, a list of table names is displayed. If the word prior to the period is either a table name or a schema name, then both a list of columns and a list of table names is displayed.

#### To activate object suggestion:

- 1 Click the line that you want SQL Editor to suggest an object for.
- 2 Press **CTRL + Spacebar** on your keyboard. SQL Editor 'reads' the line and presents a list of tables, views or columns as appropriate based on statement elements.

**NOTE:** On a per platform basis, auto-suggestion behavior may vary. (For example, the WITH statement on DB2 platforms.)

To modify object suggestion parameters, see [Set SQL Editor Preferences](#).

## Understanding Hyperlinks

SQL Editor supports hyperlinks that are activated when a user hovers their mouse over a word and presses the **CTRL** key. If a hyperlink can be created, it becomes underlined and changes color. When the hyperlink is selected, the creation script for the hyperlink object is opened in a new editor.

Hyperlinks can be used to link to tables, columns, packages, and other reference objects in development code. Additionally, hovering over a hyperlink on a procedure or function of a call statement will open it. You can also use the hyperlink feature on function calls in DML statements.

Clicking a hyperlink performs an action. The text editor provides a default hyperlink capability. It allows a user to click on a URL (for example, [www.embarcadero.com](http://www.embarcadero.com)) and database object links.

Hyperlink options (look and feel) can be modified via the **Hyperlinking** subnode in the **Editors > Text Editors** node of the **Preferences** panel.

**NOTE:** Hyperlink functionality relies on certain objects being cached in the Object Cache. If the cache is turned off, or has been restricted in what information it captures, users will be unable to link them (as they are non-existent within the cache.) To specify object cache types, see [Set Cache Configuration Preferences](#).

## Understanding Code Formatting

Code formatting provides automatic code formatting in SQL Editor while you are developing code.

To access the code formatter, select the open editor you want to format and select **Ctrl+Shift+F**. The code is formatted automatically based on formatting parameters specified in the **Code Formatter** subnode of the **SQL Editor** node in the **Preferences** panel. (See Set [Code Formatter Preferences](#) for more information on these formatting options.)

You can also format an entire group of files from **Project Explorer**. To do so, select the directory or file and execute the **Format** command via the shortcut menu. The files will be formatted automatically based on your formatting preferences.

The following examples display a list of code formatting parameters and the resultant output in SQL Editor, based on the same set of SQL statements.

### Custom Code Formatting Example 1

The following chart indicates a list of custom code formatting parameters and their corresponding values. The chart is followed by the actual syntax as it would appear in SQL Editor, based on the formatting parameter values. Compare the parameters and formatted code in Example 2 with this example for a concept of how custom formatting works.

Custom Code Formatting Parameter	Value (if applicable)
Stack commas separated by lists?	Yes
Stack Lists with ___ or more items.	3
Indent Size?	2
Preceding commas?	Yes
Spaces after comma?	1
Trailing commas?	--
Spaces before comma?	--
Right align FROM and WHERE clauses with SELECT statement?	Yes
Align initial values for FROM and WHERE clauses with SELECT list?	Yes

Custom Code Formatting Parameter	Value (if applicable)
Place SQL keywords on their own line?	No
Indent size?	--
Indent batch blocks?	Yes
Number of new lines to insert	1
Indent Size	5
Right Margin?	80
Stacked parentheses when they contain multiple items?	No
Stack parentheses when list contains ___ or more items.	--
Indent Size?	5
New line after first parentheses?	No
Indent content of conditional and looping constructs?	Yes
Number of new lines to insert?	1
Indent size?	5

The screenshot shows a SQL editor window titled 'File.sql'. The code is formatted according to the settings in the table above. The query is as follows:

```

Begin

  If x=5

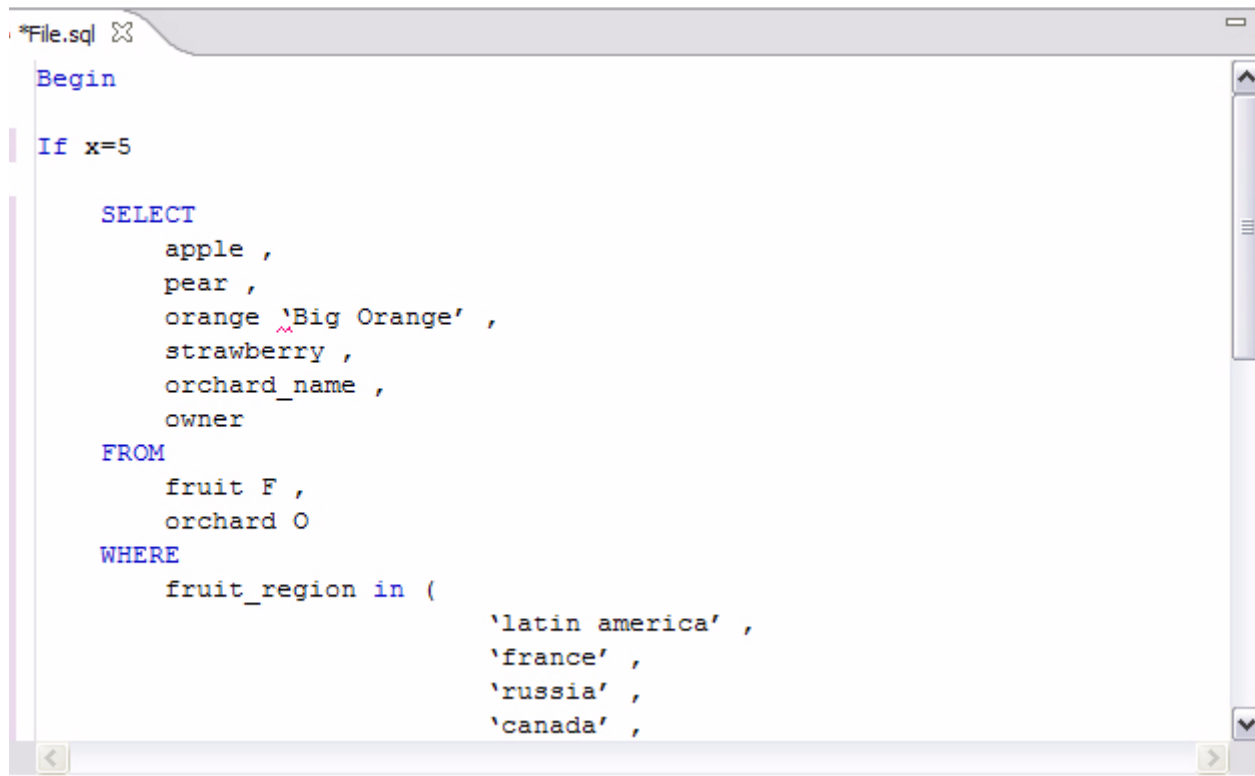
    SELECT apple
           \ pear
           \ orange 'Big Orange'
           \ strawberry
           \ orchard_name
           \ owner
    FROM fruit F, orchard O
 WHERE fruit_region in ('latin america'
           \ 'france'
           \ 'russia'
           \ 'canada'
           \ 'hawaii')
 and orchard not in (select region
                    from bad_growers bg, (select orchard
                                         from hybrid_growers
                                         where us_approved in

```

### **Custom Code Formatting Example 2**

The following chart indicates a list of custom code formatting parameters and corresponding values. The chart is followed by the actual syntax as it would appear in SQL Editor based on the formatting parameter values. Compare the parameters and formatted code in Example 1 with this example for a concept of how custom formatting works.

<b>Custom Code Formatting Parameter</b>	<b>Value (if applicable)</b>
Stack commas separated by lists?	Yes
Stack Lists with ___ or more items.	2
Indent Size?	0
Preceding commas?	--
Spaces after comma?	Yes
Trailing commas?	Yes
Spaces before comma?	2
Right align FROM and WHERE clauses with SELECT statement?	No
Align initial values for FROM and WHERE clauses with SELECT list?	--
Place SQL keywords on their own line?	Yes
Indent size?	4
Indent batch blocks?	No
Number of new lines to insert	1
Indent Size	5
Right Margin?	80
Stacked parentheses when they contain multiple items?	Yes
Stack parentheses when list contains ___ or more items.	2
Indent Size?	2
New line after first parentheses?	Yes
Indent content of conditional and looping constructs?	--
Number of new lines to insert?	1
Indent size?	5

The image shows a screenshot of the SQL Editor window. The window title is '\*File.sql'. The code is as follows:

```
Begin
If x=5
    SELECT
        apple ,
        pear ,
        orange 'Big Orange' ,
        strawberry ,
        orchard_name ,
        owner
    FROM
        fruit F ,
        orchard O
    WHERE
        fruit_region in (
            'latin america' ,
            'france' ,
            'russia' ,
            'canada' ,
```

The code is formatted with color-coding: 'Begin' is blue, 'If x=5' is purple, 'SELECT' is blue, 'FROM' is blue, 'WHERE' is blue, and the list of regions is in black. The editor has a scrollbar on the right and navigation arrows at the bottom.

## Understanding Code Folding

SQL Editor features code folding that automatically sorts code into an outline-like structure within the editor window for easy navigation and clarity while developing code.

```

CREATE DEFAULT ET_FALSE AS 0
go
+ IF OBJECT_ID('PERFCNTR_KS.ET_FALSE') IS NOT NULL
go
- CREATE TABLE PERFCNTR_KS.ADDRESS_ROLE
(
    ADDRESS_ROLE_ID    numeric(10,0) IDENTITY(20000,1),
    NAME               varchar(128) NOT NULL,
    DESCRIPTION        varchar(255) NULL,
    LIST_ORDER         int NOT NULL,
    IS_DEFAULT         bit NOT NULL,
    IS_SYSTEM_REQUIRED bit NOT NULL,
    ROWTIMESTAMP       datetime CONSTRAINT DF__ADDRESS_R__ROWTI__1BFI
    CONSTRAINT ADDRESSROLEPK
    PRIMARY KEY CLUSTERED (ADDRESS_ROLE_ID)
)
go
EXEC sp_bindefault 'PERFCNTR_KS.ET_FALSE','ADDRESS_ROLE.IS_DEFAULT'
go
EX sp_bindrule 'PERFCNTR_KS.TRUEORFALSE','ADDRESS_ROLE.IS_DEFAULT'

```

The editor window automatically inserts collapsible nodes in the appropriate lines of code for organizational purposes. This enables you to expand and collapse statements, as needed, while developing code in particularly large or complicated files.

## Understanding Code Quality Checks

Code quality markers provide annotations that prevent and fix common mistakes in the code.

These notes appear in a window on any line of code where the editor detects an error, and are activated by clicking the light bulb icon in the margin or by pressing **Ctrl + I**.

For example, if a statement reads **select \* from SCOTT.EMP, SCOTT.DEPT**, when you click the light bulb icon or press **Ctrl + I**, a window appears beneath the line of code that suggests **Add join criteria**.

When you click on a proposed fix, the statement is automatically updated to reflect your change.

**NOTE:** Code quality checks are only applicable to Oracle data sources.

The following common errors are detected by the code quality check function in the editor:

Code Quality Check Type	Definition
Statement is missing valid JOIN criteria	<p>If a SELECT statement contains missing join criteria, when it is executed, it can produce a Cartesian product between the rows in the referenced tables. This can be problematic because the statement will return a large number of rows without returning the proper results.</p> <p>The code quality check detects missing join criteria between tables in a statement and suggests join conditions based on existing foreign keys, indexes, and column name/type compatibility.</p> <p><b>Example</b></p> <p>The following statement is missing a valid JOIN criteria:</p> <pre>SELECT * FROM employee e, customer c, sales_order s WHERE e.employee_id = c.salesperson_id</pre> <p>The code quality check fixes the above statement by adding an AND clause:</p> <pre>SELECT * FROM employee e, customer c, sales_order s WHERE e.employee_id = c.salesperson_id AND s.customer_id = c.customer_id</pre>
Invalid or missing outer join operator	<p>When an invalid outer join operator exists in a SELECT statement, (or the outer join operator is missing altogether), the statement can return incorrect results.</p> <p>The code quality check detects invalid or missing join operators in the code and suggests fixes with regards to using the proper join operators.</p> <p><b>Example</b></p> <p>The following statement is missing an outer join operator:</p> <pre>SELECT * FROM employee e, customer c WHERE e.employee_id = c.salesperson_id (+) AND c.state = 'CA'</pre> <p>The code quality check fixes the above statement by providing the missing outer join operator to the statement:</p> <pre>SELECT * FROM employee e, customer c WHERE e.employee_id = c.salesperson_id(+) AND c.state(+) = 'CA'</pre>
Transitivity issues	<p>The performance of statements can sometimes be improved by adding join criteria, even if a join is fully defined. If this alternate join criteria is missing in a statement, it can restrict the selection of an index in Oracle's optimizer and cause performance problems.</p> <p>The code quality check detects possible join conditions by analyzing the existing conditions in a statement and calculating the missing, alternative join criteria.</p> <p><b>Example</b></p> <p>The following statement contains a transitivity issue with an index problem:</p> <pre>SELECT * FROM item i, product p, price pr WHERE i.product_id = p.product_id AND p.product_id = pr.product_id</pre> <p>The code quality check fixes the above statement with a transitivity issue by adding the missing join condition:</p> <pre>SELECT * FROM item i, product p, price pr WHERE i.product_id = p.product_id AND p.product_id = pr.product_id AND i.product_id = pr.product_id</pre>



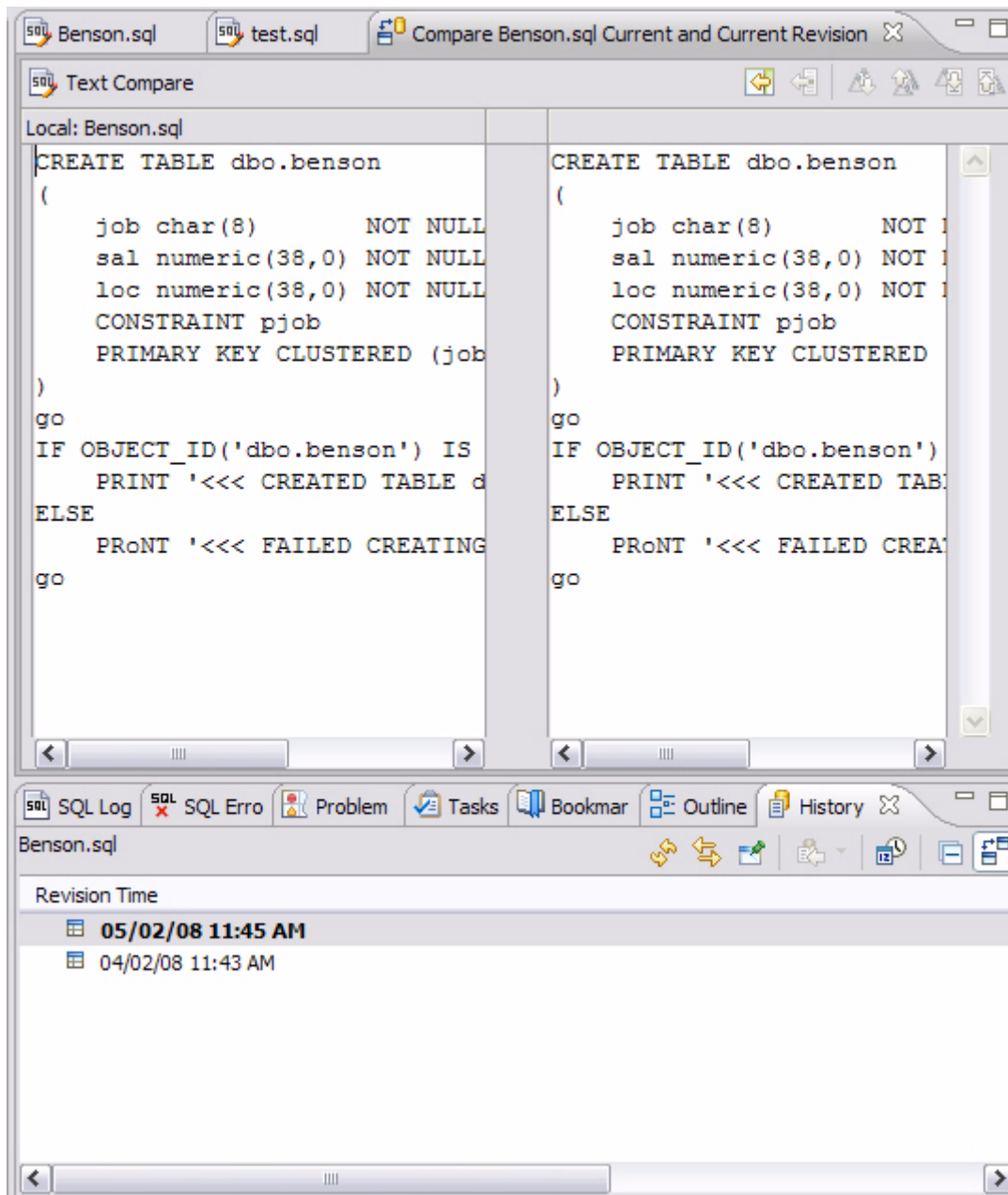
Code Quality Check Type	Definition
<p>Nested query in WHERE clause</p>	<p>It is considered bad format to place sub-queries in the WHERE clause of a statement, and such clauses can typically be corrected by moving the sub-query to the FROM clause instead, which preserves the meaning of the statement while providing more efficient code.</p> <p>The code quality check fixes the placement of sub-queries in a statement, which can affect performance. It detects the possibility of moving sub-queries from the FROM clause of the statement.</p> <p><b>Example</b></p> <p>The following statement contains a sub-query that contains an incorrect placement of a WHERE statement:</p> <pre>SELECT * FROM employee WHERE employee_id = (SELECT MAX(salary) FROM employee)</pre> <p>The code quality check fixes the above statement by correcting the sub-query issue:</p> <pre>SELECT employee.* FROM employee (SELECT DISTINCT MAX(salary) col1 FROM employee) t1 WHERE employee_id = t1.col1</pre>
<p>Wrong place for conditions in a HAVING clause</p>	<p>When utilizing the HAVING clause in a statement</p> <p>It is recommended to include as few conditions as possible while utilizing the HAVING clause in a statement. PowerSQL detects all conditions in a given HAVING statement and suggests equivalent expressions that can benefit from existing indexes.</p> <p><b>Example</b></p> <p>The following statement contains a HAVING clause that is in the wrong place:</p> <pre>SELECT col_a, SUM(col_b) FROM table_a GROUP BY col_a HAVING col_a &gt; 100</pre> <p>The code check fixes the above statement by replacing the HAVING clause with equivalent expressions:</p> <pre>SELECT col_a, SUM(col_b) FROM table_a WHERE col_a &gt; 100 GROUP BY col_a</pre>
<p>Index suppressed by a function or an arithmetic operator</p>	<p>In a SELECT statement, if an arithmetic operator is used on an indexed column in the WHERE clause, the operator can suppress the index and result in a FULL TABLE SCAN that can hinder performance.</p> <p>The code quality check detects these conditions and suggests equivalent expressions that benefit from existing indexes.</p> <p><b>Example</b></p> <p>The following statement includes an indexed column as part of an arithmetic operator:</p> <pre>SELECT * FROM employee WHERE 1 = employee_id - 5</pre> <p>The code quality check fixes the above statement by reconstructing the WHERE clause:</p> <pre>SELECT * FROM employee WHERE 6 = employee_id</pre>

Code Quality Check Type	Definition
<p>Mismatched or incompatible column types</p>	<p>When the data types of join or parameter declaration columns are mismatched, the optimizer is limited in its ability to consider all indexes. This can cause a query to be less efficient as the system might select the wrong index or perform a table scan, which affects performance.</p> <p>The code quality check flags mismatched or incompatible column types and warns that it is not valid code.</p> <p><b>Example</b></p> <p>Consider the following statement if Table A contains the column col int and Table B contains the column col 2 varchar(3):</p> <pre>SELECT * FROM a, b WHERE a.col = b.col;</pre> <p>In the above scenario, the code quality check flags the 'a.col = b.col' part of the statement and warns that it is not valid code.</p>
<p>Null column comparison</p>	<p>When comparing a column with NULL, the !=NULL condition may return a result that is different from the intended command, because col=NULL will always return a result of false. Instead, the NULL/IS NOT NULL operators should be used in its place.</p> <p>The code quality check flags occurrences of the !=NULL condition and replaces them with the IS NULL operator.</p> <p><b>Example</b></p> <p>The following statement includes an incorrect col = NULL expression:</p> <pre>SELECT * FROM employee WHERE manager_id = NULL</pre> <p>The code quality check replaces the incorrect expression with an IS NULL clause:</p> <pre>SELECT * FROM employee WHERE manager_id IS NULL</pre>

## View Change History

Each time an SQL file is saved, the local history of that file is recorded (changes made).

Using the **Local History** command, you can view all changes made to the file. **Local History** is accessed via the shortcut menu of SQL Editor and selecting **Compare With > Local History**.



- The **History** view displays all recorded times the file was changed since its inception/introduction into the workspace.
- Double-click a time in the **History** view to access the **Text Compare** panel. It displays the text of the file after the change occurred at the time indicated in the History view.

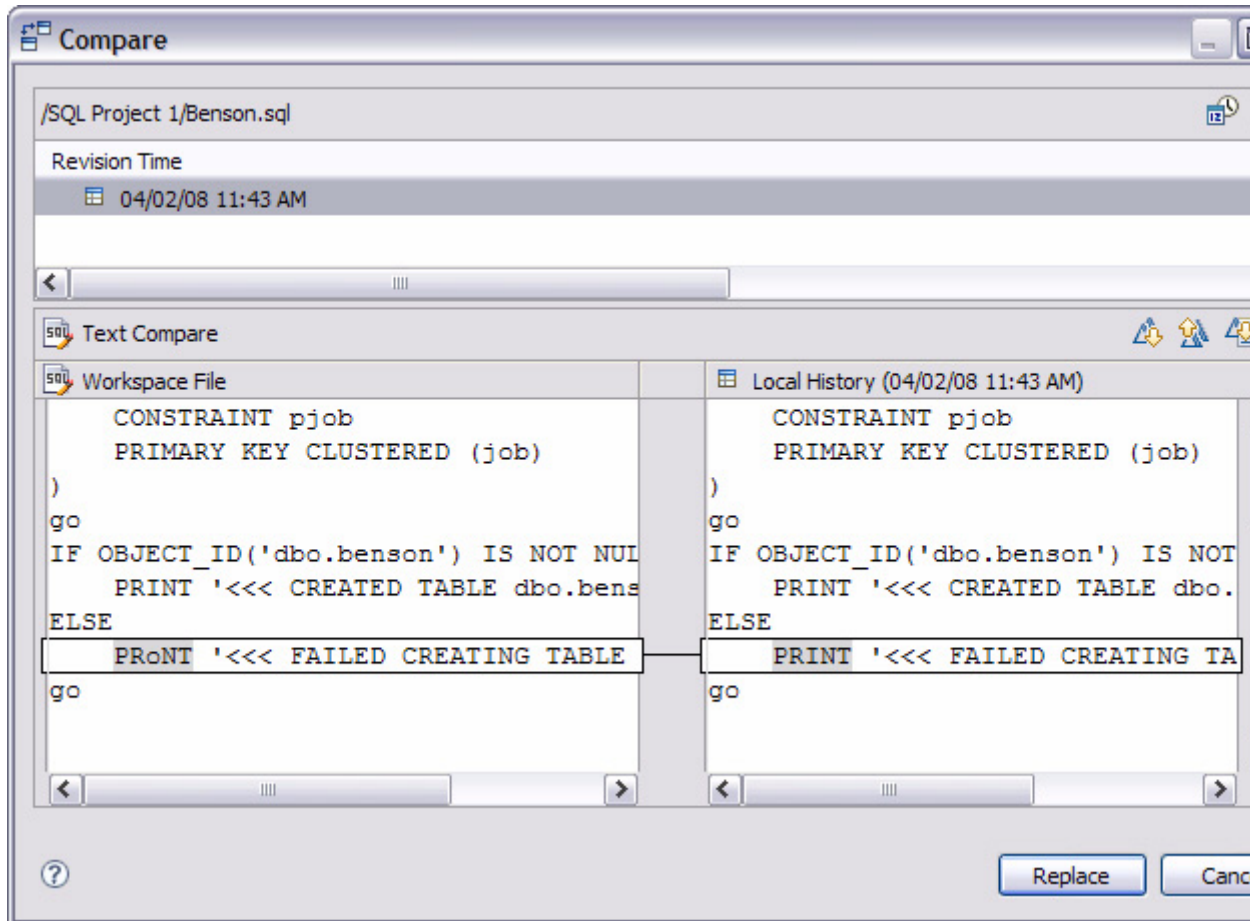
## Revert to an Old Version of a File

The **Replace With > Local History** command provides you with the ability to revert a SQL file back to a previously recorded local history.

**To replace the contents of a file with the contents of a previously saved version via local history:**

- 1 Right-click the SQL Editor and select **Replace With > Local History** from the shortcut menu.

The **Replace from Local History** dialog appears.



- 2 In the **Local History of ...** panel, select a previously recorded version of the file by clicking the appropriate timestamp.
- 3 Click **Replace**.

The contents of the currently-opened file revert to the contents of the file at the history point you selected in the dialog.

Alternatively, from the shortcut menu, select **Replace With > Previous from Local History** to replace the contents of the file with PowerSQL's last recorded history point.

## Delete a SQL File

To delete a file, right-click its icon in the **SQL Project Explorer** and select **Delete**. This will remove the file from both the SQL project and the file system.

## Executing SQL Files

PowerSQL can execute SQL code directly on registered data sources.

Files are executed via the **Execute SQL** command in the **Run** menu, or by clicking the green arrow button on the toolbar.

When an SQL file is open in the Workspace, select it and choose a database and an associated catalog on which you want to execute the file via the lists in the Toolbar.

You can click the execute icon to execute code on the specified database and catalog, start a transaction or commit a transaction, or modify SQL session options prior to execution.

### To execute a file:

Open the SQL file you want to run, ensure it is associated with the correct database, and click **Execute**. PowerSQL executes the code on the data source you specified. Results are displayed in the Results view and can be exported into a file via the **Data Export** wizard.

### To execute a transaction:

To execute transactions, you need to ensure that the auto commit feature is turned off. See [Set SQL Execution Preferences](#) for more information on how to turn off auto commit.

Open the transaction file you want to run, ensure it is associated with the correct database, and click **Start Transaction**. PowerSQL executes the transaction on the data source you specified.

Once the transaction runs, you can execute the file as normal.

**NOTE:** Click **Commit** or **Rollback** to finish or cancel a transaction.

### To commit a transaction:

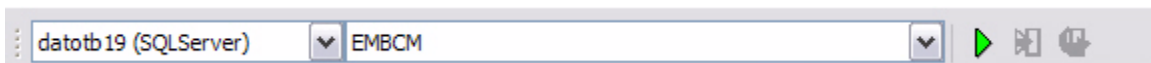
Open the transaction file you want to commit, ensure it is associated with the correct database, and click **Commit Transaction**. PowerSQL commits the transaction on the data source you specified.

**TIP:** You can set transactions to auto-commit prior to execution on the **SQL Execution** node of the **Preferences** panel.

## Associate a SQL File with a Data Source

When working with files, SQL Editor enables developers to view and change the data source to which they are connected.

The lists on the Toolbar are used to display and specify a data source in relation to the specified SQL Editor file. The menus contain a list of all registered data sources. Additionally, on platforms that support catalogs, files can be associated with these as well.



Changing a catalog via the drop down lists is the equivalent of issuing a **USE DATABASE** command on SQL Server, Sybase, and MySQL platforms. Any change will not affect the current connection, and the list automatically updates to display the name of the newly selected data source.

If no registered database is associated with a SQL file (as would be the case if a user started a new, unsaved file), the list is empty. This indicates that the file is not connected to a registered data source.

### To change or associate a registered data source with a SQL file:

Click the database list and select the name of a registered database from the list provided. Depending on the state of the code in SQL Editor, PowerSQL's behavior differs when the connection is made:

**TIP:** If you are receiving multiple syntax errors, always check that the file is associated with the correct data source and corresponding database/catalog before troubleshooting further.

## Execute SQL Code

Files can be launched from within the PowerSQL development environment for execution on a registered data source. Files are executed via the commands in the **Run** menu.

When a SQL file is open in the Workspace, select it and choose a database and an associated catalog on which you want to execute file using the drop down menus in the Toolbar. You can click the execute icon to execute the code on the specified database and catalog, start a transaction or commit a transaction, or modify the SQL session options prior to execution.

### To execute code:

Open the SQL file you want to run, ensure it is associated with the correct database and click the **Execute** icon. PowerSQL executes the code on the data source you specified. Results are displayed in the same tab or in a new tab.

### To execute a transaction:

Open the transaction file you want to run and ensure it is associated with the correct database, and then click the **Start Transaction** icon. PowerSQL executes the transaction on the data source you specified.

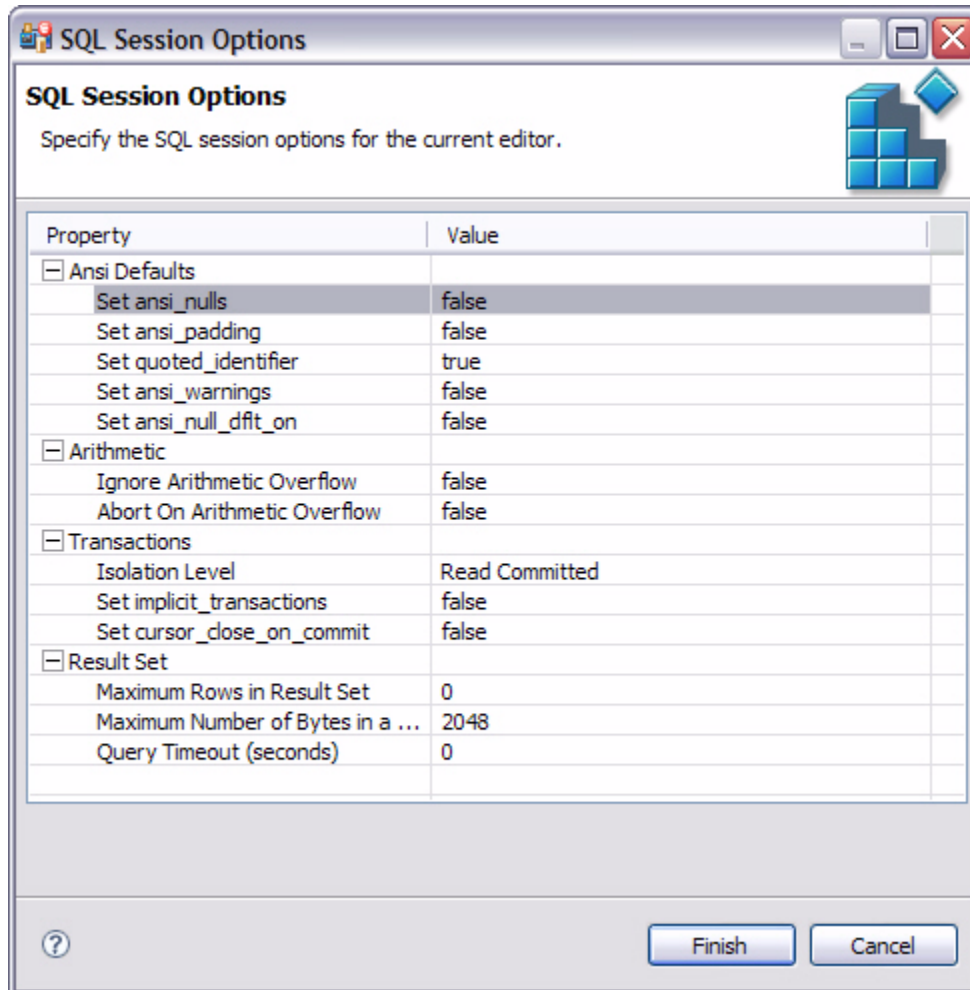
### To commit a transaction:

Open the transaction file you want to commit, ensure it is associated with the correct database, and then click the **Commit Transaction** icon. PowerSQL commits the transaction on the data source you specified.

**TIP:** You can set transactions to auto-commit prior to execution on the **SQL Execution** node of the **Preferences** panel in PowerSQL.

## Configure a SQL Session

The SQL Session Options dialog provides configuration parameters that indicate to PowerSQL how to execute code in the development environment.



### To modify SQL session options:

- 1 Click the SQL Session Options icon in the Toolbar.

The **SQL Session Options** dialog appears.

- 2 Click on individual parameters in the **Value** column to change the configuration of each property, as specified.
- 3 Click **Finish**.

The session options will be changed and PowerSQL will execute the code as specified when you execute it.

**NOTE:** Session options only apply to the corresponding editor and are not retained when executing multiple SQL files.

## Troubleshooting

PowerSQL contains a number of views used exclusively to log and monitor the SQL development process.

- The **SQL Log** captures all SQL commands executed by SQL Editor and the system. **SQL Log** entries are listed by **SQL Statement** name, **Date** issued, **Host/Server**, **Service**, **User**, **Source**, and the **Time** (in milliseconds) it took to execute the command.

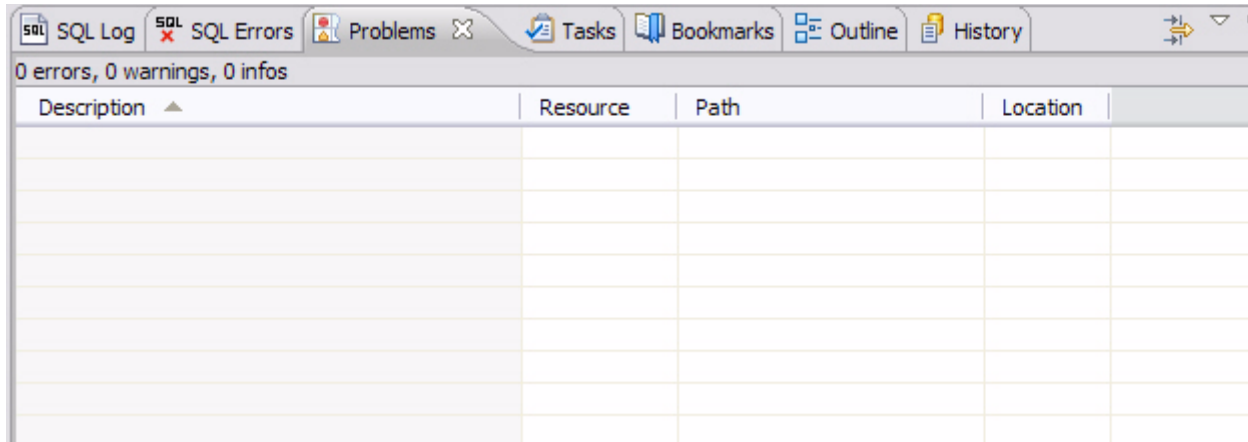
SQL Statement	Date	Host/Server	DBMS
<input type="checkbox"/> ALTER TABLE dbo.testapps ADD CONSTRAINT CK_123 CHECK	2008-02-04 11:06:12.656	datotb19	SQLSe
<input type="checkbox"/> CREATE TABLE dbo.benson ( job char(8) NOT NULL, sal nu	2008-02-04 11:06:53.00	datotb19	SQLSe
<input checked="" type="checkbox"/> IF OBJECT_ID('dbo.benson') IS NOT NULL PRINT '<<< CREATEL	2008-02-04 11:06:53.171	datotb19	SQLSe

- The **SQL Errors** log automatically logs all SQL errors encountered when SQL commands are executed through PowerSQL. Errors are listed by **Error Code**, **SQL State**, error **Details**, **Resource**, and the **Location** of the error in the SQL file.

Error Code	SQL State	Details	Resource	Location
170	37000	Line 4: Incorrect syntax near 'PRoNT'.	Benson.sql	line 13
170	37000	Line 8: Incorrect syntax near 'sdasd'.	Benson.sql	line 8



- The **Problems** view captures syntactic and semantic errors and warnings in the files of the workspace. These entries typically take the form of error messages or warnings issued by the system over the course of a procedure execution. Problems are organized by **Description** (which indicates the type of problem logged), **Resource**, file **Path**, and **Location**.



See also:

[View Log Details](#)

[Maintain Logs](#)

[Filter Logs](#)

[Import and Export Error Logs](#)

[Find and Fix SQL Code Errors](#)

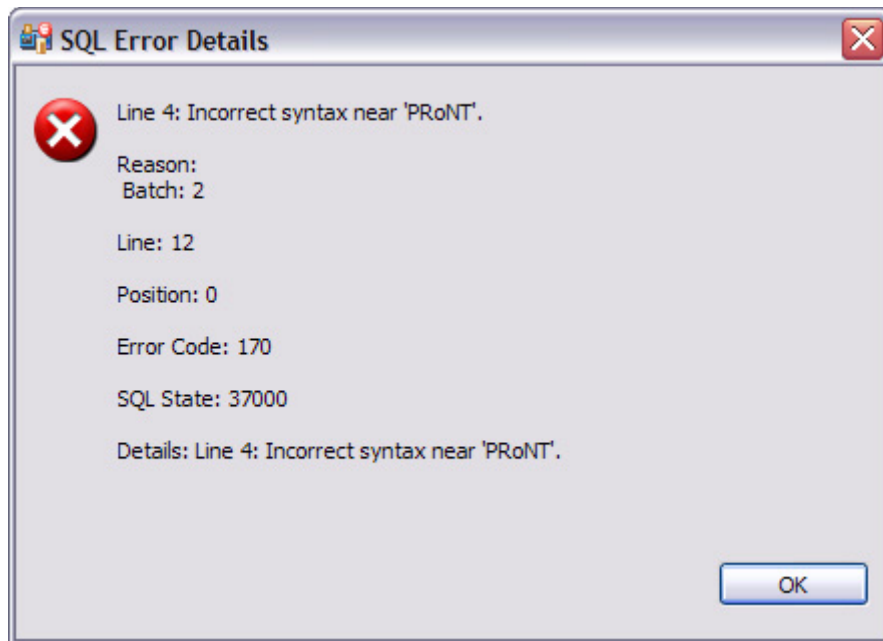
[Find and Fix Other Problems](#)

## View Log Details

The **SQL Error Log** and **Problems** views contain functionality that enable you to view details regarding individual log entries, and in some cases, locate or fix those issues automatically.

### To view details about SQL Errors entries:

Right-click the error whose details you want to view and select **SQL Error Details**.



The SQL Error Details dialog provides information about the specified SQL error.

Additionally, you can double-click the error to view the problem code in **SQL Editor**.

#### To view details about Problems

- Right-click the entry whose details you want to view and select **Properties**. The **Properties** dialog appears, summarizing the issue.

## Maintain Logs

The **SQL Log** and **SQL Errors** views both contain commands that enable you to save, restore, or otherwise move log entries into files outside of PowerSQL. Additionally, both views also contain commands that enable the clearing of the view.

The current editor option will only show users statements as generated by the active editor.

#### To maintain log entries:

All entries automatically captured by the Error Log are written to a file (.log suffix) that resides in the Workspace .metadata folder.

- From PowerSQL, right-click in the **SQL Log** and select **Clear Log Viewer** to remove all messages.
- In the shortcut menu, select **Delete Log** to delete the .log file. If entries are created after the **Delete Log** command is issued, PowerSQL will automatically generate a new .log file in the .metadata subfolder.

**NOTE:** Old Error Log entries cannot be recovered once the .log file is deleted. To prevent data loss, archive the .log file via the **Export** command prior to deletion.

- To clear the **Error Log** view without deleting the .log file, select **Clear Log Viewer** from the shortcut menu. The View will be cleared of entries, but these entries will still be contained in the .log file.

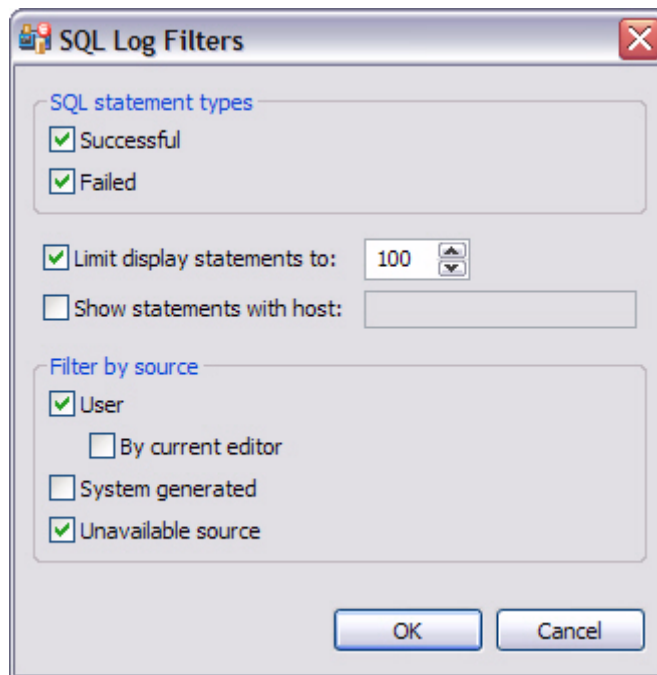
- To restore the **Error Log** view based on the entries contained in the **.log** file, select **Restore Log** from the shortcut menu. The View is restored based on the entries in the **.log** file.

## Filter Logs

Filters can be applied to **Problems**, **SQL Log**, and the **SQL Error Log** to limit searches when troubleshooting and pinpointing specific processes within the system.

### To filter the SQL Log:

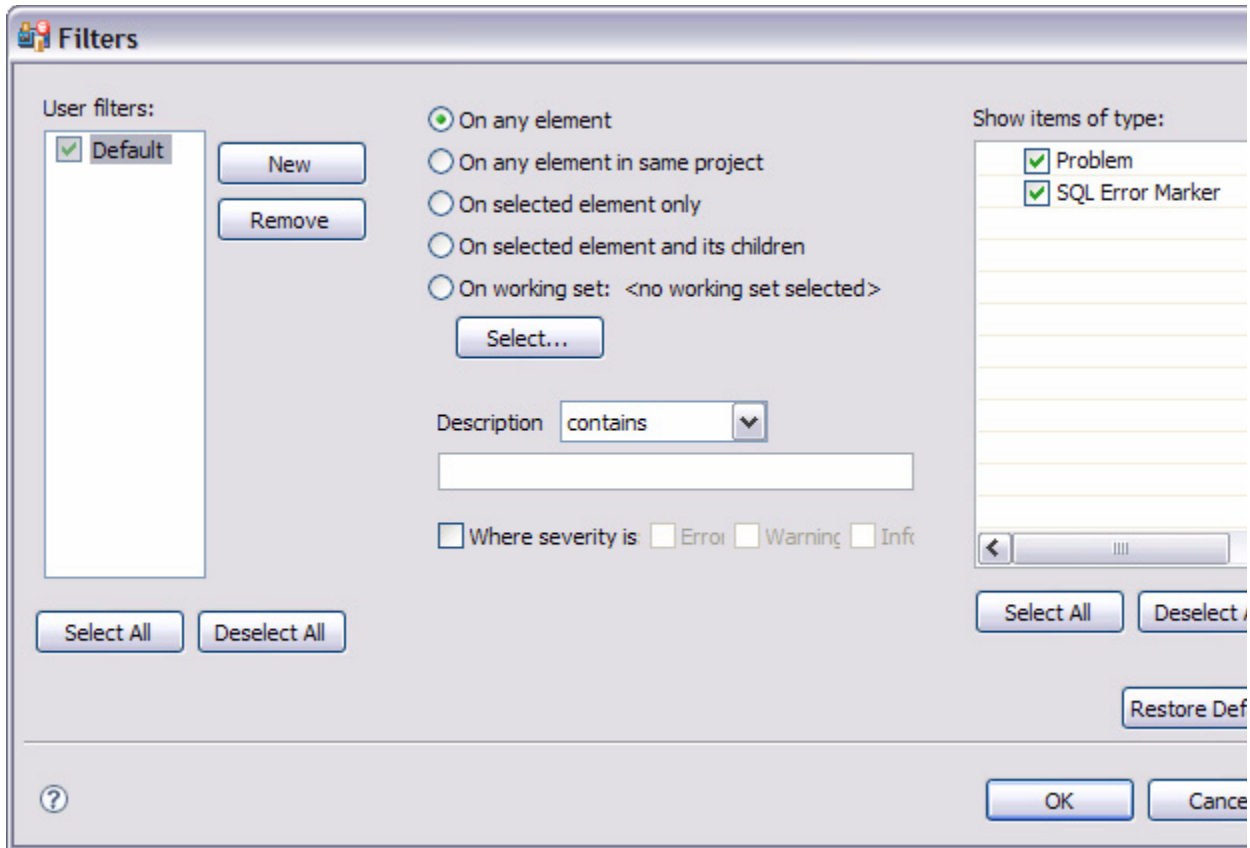
- Select the **Toolbar Menu** icon (the downward-pointing arrow in the right-hand corner of the view) and choose **Filters**. The **SQL Log Filters** dialog appears.



- In the **SQL Statement Types** frame, select **Successful** or **Failed** to filter by the type of Error Log entries.
- Select **Limit display statements** to indicate a maximum limit of the number of entries displayed in the **Error Log**, and enter the maximum entry value in the corresponding field.
- Select **Show statements with host** to indicate that only entries from a specific data source are to be displayed, then type the name of the data source (as it appears in the **Database Explorer**) in the corresponding field.
- In the **Filter by Source** pane, specify **User**, **System Generated**, or **Unavailable Source** to filter statements by the type of source from where they originated.

### To filter the Problems log:

Select the **Toolbar Menu** icon and choose **Configure Filters**. The **Filters** dialog appears:



The **Filters** dialog enables the creation of multiple filter profiles that can be applied to the log via the **Toolbar Menu**. The **User Filters** panel on the left-hand side of the dialog displays all existing filter profiles stored in the Workspace. Initially, the Workspace only contains the **Default** filter profile. Selecting it displays its filter parameters, and selecting the check box associated with its name enables the filter in the **Problems** view (only problems that match the criteria defined in the **Filters** dialog will appear in the view).

The ability to define different profiles enables the selection of multiple filter profiles. For each profile selected, the profile criteria is applied to the View, concurrently. You can filter problems by:

- Working Set
- Character String
- Problem Severity
- Problem Type
- A combination of the above four filter options

Profile Criteria	Description
<b>Working Set</b>	<p>The options located in the center of the dialog enable you to filter problems based on defined Working Sets. A Working Set is a collection of user-defined Project files that you can organized, as needed, in PowerSQL. Select an option, and then click <b>Select</b> to define a Working Set to which the parameters apply. If no Working Sets exist, you need to define one or more via the <b>New</b> button on the <b>Select Working Set</b> dialog.</p> <p>Select one or more Working Sets to which you want the criteria to apply. If no Working Sets exist, or none suitably match the current filter criteria, click <b>New</b> or <b>Edit</b> to define a new Working Set, or edit an exist Working Set, respectively.</p>
<b>Character String</b>	<p>Use the <b>Description</b> list to select <b>contains</b> or <b>doesn't contain</b>, as needed, and type the character string in the field below the list. The Problems view is filtered to only contain, or omit, problem descriptions that fully or partially match the string value.</p>
<b>Problem Severity</b>	<p>Select the <b>Where severity is</b> check box and choose <b>Error</b>, <b>Warning</b>, <b>Info</b>, or some combination of the three check boxes. Only entries whose severity matches the check boxes you have selected remain visible in the Problems view.</p>
<b>Problem Type</b>	<p>The options in the <b>Show items of type</b> list on the right-hand side of the dialog enable you to filter problems by type. Deselect <b>Problem</b> to remove any system entries from the view, or deselect <b>SQL Error Marker</b> to remove any SQL code entries from the view.</p>

Once you have defined and/or selected the appropriate filter profiles, the profiles will appear in the **Filters** submenu in the **Toolbar Menu** of the **Problems** view. Select or deselect the profiles from the submenu, as needed.

## Import and Export Error Logs

Error messages are written to a file named **.log** located in the Workspace directory **.metadata** folder. This file can (and should) be cleared periodically via the **Delete Log** command to prevent performance issues with regards to system memory and file size. However, the **Export** command enables you to archive log files prior to deletion. The files created by the **Export** command can then be imported back into the **Error Log** as needed at a later point in time.

### To export the SQL Log:

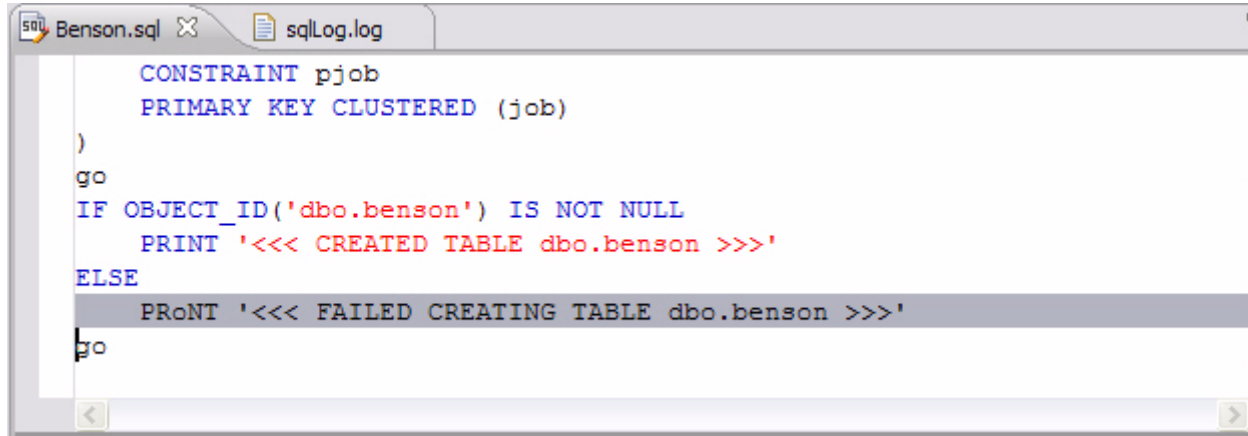
Right-click the **SQL Log** view and choose **Export Log**. The log is saved in the specified directory path with a **.log** extension.

### To import the Error Log:

Right-click the **SQL Log** view and choose **Import Log**. Select the previously exported **.log** file. The **Error Log** view is restored with the entries from the specified export file.

## Find and Fix SQL Code Errors

The **SQL Errors** view contains an option that enables you to navigate directly to the resource associated with an error entry.



The screenshot shows a SQL Editor window with two tabs: 'Benson.sql' and 'sqlLog.log'. The SQL script in the editor is as follows:

```

CONSTRAINT pjob
PRIMARY KEY CLUSTERED (job)
)
go
IF OBJECT_ID('dbo.benson') IS NOT NULL
PRINT '<<< CREATED TABLE dbo.benson >>>'
ELSE
PRINT '<<< FAILED CREATING TABLE dbo.benson >>>'
go
  
```

The line containing the error message, `PRINT '<<< FAILED CREATING TABLE dbo.benson >>>'`, is highlighted in a grey background.

### To navigate to the source of a SQL error entry:

right-click the the entry to which you want to navigate and select **Go To**. The file to which the error applies automatically opens in a new instance of **SQL Editor**, and the line is highlighted in the window.

## Find and Fix Other Problems

By default, the Problems view organizes problems by severity. You can also group problems by type, or leave them ungrouped.

The first column of the Problems view displays an icon that denotes the type of line item, the category, and the description. Click the problem and PowerSQL will open the SQL file and automatically highlight the line that triggered the issue.

You can filter Problems to view only warnings and errors associated with a particular resource or group of resources. You can add multiple filters to the view, as well as enable/disable them as required. Filters are additive, so any problem that satisfies at least one of the filters will appear.

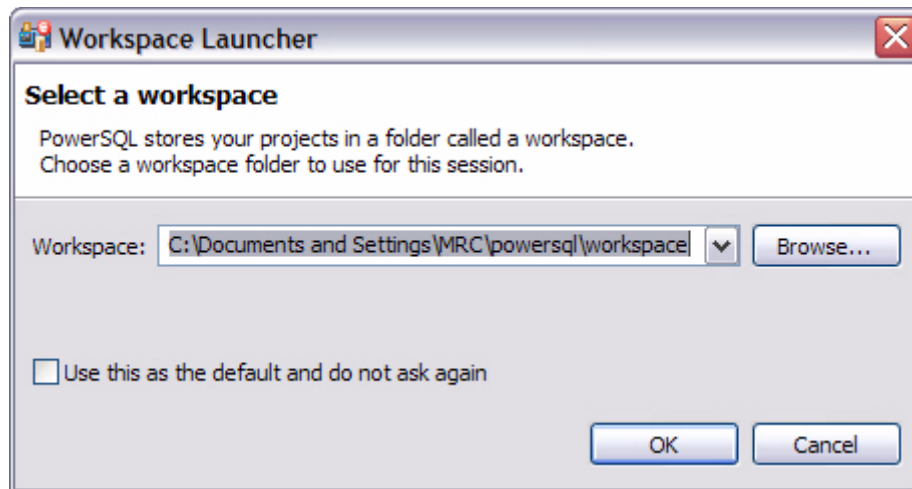
Problems can sometimes be fixed via the **Quick Fix** command in the shortcut menu. The list of possible resolutions is displayed. To fix other similar problems, use the **Find Similar Problems** command and PowerSQL will locate other problems in the view to which the same fix can be applied.

# Configuring PowerSQL

## Initial Setup

### Specify a Workspace

When you start Eclipse or the PowerSQL standalone application for the first time, you are prompted to create a workspace.



Click **Use this as the default and do not ask again** to set the specified folder as the permanent default workspace. For more information about workspaces, see **Help > Help Contents > Workbench User Guide**.

### License PowerSQL

When you first install PowerSQL, you will be prompted to activate the product. Choose to activate by Internet and follow the prompts. During the activation process you will receive an email with an activation key; after you enter that key into the License Setup dialog, you will receive a free 14-day evaluation license.

If due to firewall or other restrictions you cannot use Internet activation, select the E-mail alternative. If that does not work either, select the Phone alternative.

To continue using PowerSQL after the evaluation period, select **Help > Embarcadero License Manager** and follow the prompts, or visit the Embarcadero online store at <http://www.embarcadero.com/store.html>.

## Customizing PowerSQL (Preferences)

To customize various aspects of PowerSQL, select **Window > Preferences > SQL Development**. For information on the other preference categories, see **Help > Help Contents > Workbench User Guide**.

**NOTE:** PowerSQL does not support the Eclipse platform Debug or Team Synchronizing features. Debug and Team appear in the preferences of the standalone version because they are tied to the Run and Local History features, which are supported.

### Set Cache Configuration Preferences

The **Data Source Object Cache** is a local repository that stores the schema of registered data sources in PowerSQL. It is automatically set to cache information about data sources registered in the development environment.

By default, the **Data Source Object Cache** caches all catalogs, functions, procedures, tables, and views. Additionally, after the initial cache, the object cache performs incremental caches.

However, there is a definitive trade-off when caching a full database schema. The time it takes to fully cache a large schema and logical space considerations on local workstations, often makes it inefficient for PowerSQL to perform this task each time a new data source is registered in PowerSQL. Thus, the Object Cache can be configured via the PowerSQL **Preferences** dialog to accommodate machine processing ability and speed.

By default, when a data source connects to PowerSQL, the Object Cache automatically begins indexing its schema.

**Cache Configuration** parameters enable you to indicate how schema caching behaves by specifying at what level data source objects will be cached, the specific catalogs, schemas, and data source objects to cache, and other factors that speed up the caching process at a cost of slower retrieval for those objects not cached by the process.

Additionally, over the course of a PowerSQL session, cache information is periodically updated by PowerSQL. The cache refresh process uses the same specified parameters as the initial cache process and therefore can cause application slowdown and performance issues if the cache behavior has not been configured in an efficient manner.

#### To configure the cache:

- 1 Select **Window > Preferences > SQL Development > Cache Configuration**. Change the settings as appropriate:
  - The tree view displays a list of database objects as they are organized in the Database Explorer view. Use the check boxes beside each object to specify the data sources that are to be included in the cache process.
  - Select **Clear Cache on Startup** to delete the Object Cache each time the application is started.
  - Select **Apply SQL Filters** if you want to apply any pre-defined filters to the cache.
  - If you are having performance problems due to a caching issue (such as a configuration error), the **Stop Caching**, **Clear Cache**, and **Start Caching** buttons enable you to stop, clear, and/or restart the cache process, respectively.
  - The **Max. number of objects to cache** field indicates how many logical data source objects can be cached before the Object Cache has reached maximum cache size.
  - The options in the **When the maximum cache size is reached** pane specify how the Object Cache should handle the caching process once it has reached the maximum. Select **Grow the cache automatically until all data is cached** to override the maximum cache setting, or select **Stop caching** to terminate the cache process when the maximum cache size is reached.
  - The **Objects to include in cache** pane contains a list of data source objects. Select or clear the check boxes beside each data source object to indicate the specific data source objects that are included and excluded, respectively, from the cache process.
  - The **Cache Expiration Time (hours)** setting indicates that a cache job will not start automatically until the specified number of hours have passed. The cache can also be started manually via **Start Caching**.



- 2 When you are finished configuring the Object Cache, click **Apply** to save your changes.

## Set SQL Editor Preferences

- 1 Select **Window > Preferences > SQL Development > SQL Editor**.
- 2 Change the settings as appropriate in each section and then click **Apply**.
  - **Enable SQL Parser** indicates that the parser that provides code development features in SQL Editor is enabled. If this option is disabled, functions such as code formatting, auto completion, semantic validation, and hyperlinks, will not be available. The options below the check box enable you to limit (by file size) when the parser is enabled when dealing with code on a file-by-file basis.
  - **Parsing Delay** specifies the amount of time that the parser delays prior to activating features after a keystroke.
  - **Severity Level for Semantic Validation Problems** determines how semantic code errors are flagged in the editor and the **Problems** view.

**NOTE:** Clearing **Enable SQL Parser** will disable many of the “smart” SQL editor features, including code formatting, auto completion, semantic validation, and hyperlinks. For better performance, you may disable the parser for files above a specified size.

## Set SQL Execution Preferences

Select **Window > Preferences > SQL Development > SQL Editor**.

**NOTE:** If you disable auto-commit for a platform, you must use SQL Editor’s transaction features to execute code on that platform.

## Set Code Assist Preferences

The **Code Assist** panel is used to specify configuration parameters that determine how code completion features in SQL Editor behave.

Select **Window > Preferences > SQL Development > Code Assist**.

- **Enable Auto Activation** enables or disables code assist functionality.
- **Insert Single Proposals Automatically** specifies if only a single code completion suggestion is returned, it is inserted automatically.
- **Fully Qualified Completions Automatically** specifies if code completion results are returned specific (fully qualified), rather than the minimum required to identify the object.
- **Code Assist Color Options** specifies the color formatting of code completion proposals. Select background or foreground options from the menu and modify them as appropriate.

## Set Code Formatter Preferences

The **Code Formatter** pane provides configuration options for code formatting functionality in SQL Editor.

Select **Window > Preferences > SQL Development > Code Formatter**.

The panel provides a drop down list of formatting profiles and a preview window that displays how each profile formats code.

- Click **New** to define additional code formatting profiles.
- Click **Edit** to modify existing profiles. You can modify how code characters appear in the interface and how SQL Editor determines line breaks.
- Click **Rename** to change the name of an existing profile. The new name cannot be the same as another existing profile.

**NOTE:** If you create a new profile with a name that already exists in the system, a prompt will appear asking you to change the name of the new code formatting template.

## Set Results View Preferences

The **Results Viewer** pane provides configuration options that specify how the **Results** view displays results.

Select **Window > Preferences > SQL Development > Results Viewer**.

- **Grid Refresh Interval** indicates the speed in milliseconds that the Results view refreshes.
- **Stripe the Rows of the Results Table** adds intermittent highlighted bars in the Results view.
- **Display Results in Separate Tab in SQL Editor** opens the Results view in a separate window on the Workbench.

## Set Syntax Coloring Preferences

The **Syntax Coloring** panel provides configuration options that change the look and feel of code syntax in SQL Editor.

Select **Window > Preferences > SQL Development > Syntax Coloring**.

Use the tree view provided in the **Element** window to select the comment type or code element you want to modify. Select the options to the right-hand side of the window to modify it. The **Preview** window shows a piece of sample code that updates according to the changes you made.

## Set SQL Filter Preferences

See [Filter Database Objects](#) for instructions.

# Reference

## Database Objects

The following table describes the database objects displayed in PowerSQL and contains information regarding each one, including object name, DBMS platform, and any notes pertaining to the specified object.

In PowerSQL, database objects are stored in **Data Source Explorer** as subnodes of individual, pertinent databases.

Database Object	DBMS Platforms	Notes
Aliases	DB2	<p>An alias is an alternate name that references a table, view, and other database objects. An alias can also reference another alias as long as the aliases do not reference one another in a circular or repetitive manner.</p> <p>Aliases are used in view or trigger definitions in any SQL statements except for table check-constraint definitions. (The table or view name must be referenced in these cases.)</p> <p>Once defined, an alias is used in query and development statements to provide greater control when specifying the referenced object. Aliases can be defined for objects that do not exist, but the referenced object must exist when a statement containing the alias is compiled.</p> <p>Aliases can be specified for tables, views, existing aliases, or other objects. Create Alias is a command available on the shortcut menu.</p>
Check Constraints	All	<p>A check constraint is a search condition applied to a table. When a check constraint is in place, Insert and Update statements issued against the table will only complete if the statements pass the constraint rules.</p> <p>Check constraints are used to enforce data integrity when it cannot be defined by key uniqueness or referential integrity restraints.</p> <p>A check condition is a logical expression that defines valid data values for a column.</p>
Clusters	Oracle	<p>A cluster is a collection of interconnected, physical machines used as a single resource for failover, scalability, and availability purposes.</p> <p>Individual machines in the cluster maintain a physical host name, but a cluster host name must be specified to define the collective as a whole.</p> <p>To create a cluster, you need the CREATE CLUSTER or CREATE ANY CLUSTER system privilege.</p>
Database Links	Oracle	<p>A database link is a network path stored locally, that provides the database with the ability to communicate with a remote database.</p> <p>A database link is composed of the name of the remote database, a communication path to the database, and a user ID and password (if required).</p> <p>Database links cannot be edited or altered. To make changes, drop and re-create.</p>

Database Object	DBMS Platforms	Notes
Foreign Keys	All	<p>A foreign key references a primary or unique key of a table (the same table the foreign key is defined on, or another table and is created as a result of an established relationship). Its purpose is to indicate that referential integrity is maintained according to the constraints.</p> <p>The number of columns in a foreign key must be equal to the number of columns in the corresponding primary or unique key. Additionally, the column definitions of the foreign key must have the same data types and lengths.</p> <p>Foreign key names are automatically assigned if one is not specified.</p>
Functions	DB2, Oracle	<p>A function is a relationship between a set of input data values and a set of result values.</p> <p>For example, the <code>TIMESTAMP</code> function passes input data values of type <code>DATE</code> and <code>TIME</code>, and the result is <code>TIMESTAMP</code>.</p> <p>Functions can be built-in or user-defined. Built-in functions are provided with the database. They return a single value and are part of the default database schema. User-defined functions extend the capabilities of the database system by adding function definitions (provided by users or third-party vendors) that can be applied in the database engine itself.</p> <p>A function is identified by its schema, a function name, the number of parameters, and the data types of its parameters.</p> <p>Access to functions is controlled through the <code>EXECUTE</code> privilege. <code>GRANT</code> and <code>REVOKE</code> statements are used to specify who can or cannot execute a specific function or set of functions.</p>
Groups	All	<p>Groups are units that contain items. Typically, groups contain the result of a single business transaction where several items are involved.</p> <p>For example, a group is the set of articles bought by a customer during a visit to the supermarket.</p>

Database Object	DBMS Platforms	Notes
Indexes	All	<p>An index is an ordered set of pointers to rows in a base table.</p> <p>Each index is based on the values of data in one or more table columns. An index is an object that is separate from the data in the table. When an index is created, the database builds and maintains it automatically.</p> <p>Indexes are used to improve performance. In most cases, access to data is faster with an index. Although an index cannot be created for a view, an index created for the table on which a view is based can improve the performance of operations on that view.</p> <p>Indexes are also used to ensure uniqueness. A table with a unique index cannot have rows with identical keys.</p> <p>DB2: Allow Reverse Scans, Percent Free (Lets you type or select the percentage of each index page to leave as free space when building the index, from 0 to 99), Min Pct Used (Lets you type or select the minimum percentage of space used on an index leaf page. If, after a key is removed from an index leaf page, the percentage of space used on the page is at or below integer percent, an attempt is made to merge the remaining keys on this page with those of a neighboring page. If there is sufficient space on one of these pages, the merge is performed and one of the pages is deleted. The value of integer can be from 0 to 99).</p> <p>Oracle: The Logging, No Sort, Degrees, and Instances properties are documented in the editor.</p>
Java Classes	Oracle	<p>A model or template, written in Java language, used to create objects with a common definition and common properties, operations and behavior.</p> <p>Java classes can be developed in Eclipse (or another Java development environment such as Oracle JDeveloper) and moved into an Oracle database to be used as stored procedures.</p> <p>Java classes must be public and static if they are to be used in this manner.</p> <p>When writing a class to be executed within the database, you can take advantage of a special server-side JDBC driver. This driver uses the user's default connection and provides the fastest access to the database.</p> <p>Java classes become full-fledged database objects once migrated into the database via the loadjava command-line utility or the SQL CREATE JAVA statement.</p> <p>A Java class is published by creating and compiling a call specification for it. The call spec maps a Java method's parameters and return type to Oracle SQL types.</p> <p>Once a Java class is developed, loaded, and published -- the final step is to execute it.</p>
Java Resources	Oracle	<p>A Java resource is a collection of files compressed in a .jar file.</p>

Database Object	DBMS Platforms	Notes
Libraries	Oracle	<p>A library is a configurable folder for storing and sharing content with an allocated quota. Multiple libraries may exist in the same database environment.</p> <p>A library is a special type of folder in Oracle Content Services. Unlike Containers and regular folders, each library has a Trash Folder and an allocated amount of disk space.</p> <p>A library is composed of a name (mandatory), description, quota, path, and library members.</p> <p>The library service allows you to create folders, list quotas, and manage categories, workflow, trash folders, and versioning. The Library service does not allow you to create or upload files.</p>
Materialized Views	Oracle	<p>A database object that contains the results of a query. They are local copies of data located remotely, or are used to create summary tables based on aggregations of table data. Materialized views are also known as snapshots.</p> <p>A materialized view can query tables, views, and other materialized views. Collectively, these are called master tables (a replication term) or detail tables (a data warehouse term).</p> <p>For replication purposes, materialized views allow you to maintain copies of remote data on your local node. These copies are read-only. If you want to update the local copies, you need to use the Advanced Replication feature. You can select data from a materialized view as you would from a table or view.</p> <p>For data warehousing purposes, the materialized views commonly created are aggregate views, single-table aggregate views, and join views.</p>
Materialized View Logs	Oracle	<p>Because Materialized Views are used to return faster queries (a query against a materialized view is faster than a query against a base table because querying the materialized view does not query the source table), the Materialized View often returns the data at the time the view was created, not the current table data.</p> <p>There are two ways to refresh data in Materialized Views, manually or automatically. In a manual refresh, the Materialized View is completely wiped clean and then repopulated with data from the source tables (this is known as a complete refresh). If source tables have changed very little, however, it is possible to refresh the Materialized View only for changed records -- this is known as a fast refresh.</p> <p>In the case of Materialized Views that are updated via fast refresh, it is necessary to create Materialized View Logs on the base tables that compose the Materialized View to reflect the changes.</p> <p>If the number of entries in this table is too high, it is an indication that you might need to refresh the Materialized Views more frequently to ensure that each update does not take longer than it needs.</p> <p>Select owner, then select from tables with Materialized Views, etc.</p>

Database Object	DBMS Platforms	Notes
Oracle Job Queue	Oracle	<p>The Oracle Job Queue allows for the scheduling and execution of PL/SQL stored procedures at predefined times and/or repeated job execution at regular intervals, as background processes.</p> <p>For example, you could create a job in the Oracle Job Queue that processed end-of-day accounting -- a job that must run every weekday, but can be run unattended, or you could create a series of jobs that must be run sequentially -- such as jobs that might be so large, that in order to reduce CPU usage, only one is run at a time.</p> <p>Runs PL/SQL code at specified time or on specified schedule, can enable/disable.</p>
Outlines	Oracle	<p>Oracle preserves the execution plans of "frozen" access paths to data so that it remains constant despite data changes, schema changes, and upgrades of the database or application software through objects named stored outlines.</p> <p>Outlines are useful for providing stable application performance and benefit high-end OLTP sites by having SQL execute without having to invoke the cost-based optimizer at each SQL execution. This allows complex SQL to be executed without the additional overhead added by the optimizer when it performs the calculations necessary to determine the optimal access path to the data.</p>
Packages	All	<p>A package is a procedural schema object classified as a PL/SQL program unit that allows the access and manipulation of database information.</p> <p>A package is a group of related procedures and functions, together with the cursors and variables they use, stored together in the database for continued use as a unit. Similar to standalone procedures and functions, packaged procedures and functions can be called explicitly by applications or users.</p> <p>DB applications explicitly call packaged procedures as necessary with privileges granted, a user can explicitly execute any of the procedures contained in it.</p> <p>Packages provide a method of encapsulating related procedures, functions, and associated cursors and variables together as a unit in the database. For example, a single package might contain two statements that contain several procedures and functions used to process banking transactions.</p> <p>Packages allow the database administrator or application developer to organize similar routines as well as offering increased functionality and database performance.</p> <p>Packages provide advantages in the following areas:                      encapsulation of related procedures and variables, declaration of public and private procedures, variables, constraints and cursors, separation of the package specification and package body, and better performance.</p> <p>Encapsulation of procedural constructs in a package also makes privilege management easier. Granting the privilege to use a package makes all constructs of the package assessable to the grantee.</p> <p>The methods of package definition allow you to specify which variables, cursors, and procedures are: public, directly accessible to the users of a package, private, or hidden from the user of the package.</p>

Database Object	DBMS Platforms	Notes
Package Bodies	Oracle	<p>A package body is a package definition file that states how a package specification will function.</p> <p>In contrast to the entities declared in the visible part of a package, the entities declared in the package body are only visible within the package body itself. As a consequence, a package with a package body can be used for the construction of a group of related subprograms in which the logical operations available to clients are clearly isolated from the internal entities.</p>
Primary Keys	All	<p>A key is a set of columns used to identify or access a row or rows. The key is identified in the description of a table, index, or referential constraint. The same column can be part of more than one key.</p> <p>A unique key is a key that is constrained so that no two of its values are equal. The columns of a unique key cannot contain NULL values.</p> <p>The primary key is one of the unique keys defined on a table, but is selected to be the key of the first importance. There can only be one primary key on a table.</p> <p>Oracle: If an index constraint has been defined for a table, the constraint status for the table's primary key cannot be set to Disabled.</p>
Procedures	All	<p>A procedure is an application program that can be started through the SQL CALL statement. The procedure is specified by a procedure name, which may be followed by arguments enclosed within parenthesis.</p> <p>The argument or arguments of a procedure are individual scalar values, which can be of different types and can have different meanings. The arguments can be used to pass values into the procedure, receive return values from the procedure, or both.</p> <p>A procedure, also called a stored procedure, is a database object created via the CREATE PROCEDURE statement that can encapsulate logic and SQL statements. Procedures are used as subroutine extensions to applications, and other database objects that can contain logic.</p> <p>When a procedure is invoked in SQL and logic within a procedure is executed on the server, data is only transferred between the client and the database server in the procedure call and in the procedure return. If you have a series of SQL statements to execute within a client application, and the application does not need to do any processing in between the statements, then this series of statements would benefit from being included in a procedure.</p>
Profiles	Oracle	<p>Profiles are a means to limit resources a user can use by specifying limits on kernel and password elements. Additionally, Profiles can be used to track password histories and the settings of specific profiles may be queried.</p> <p>The following kernel limits may be set: maximum concurrent sessions for a user, CPU time limit per session, maximum connect time, maximum idle time, maximum blocks read per session, maximum blocks read per call, and maximum amount of SGA.</p>



Database Object	DBMS Platforms	Notes
Roles	Oracle	<p>A role is a set or group of privileges that can be granted to users to another role.</p> <p>A privilege is a right to execute a particular type of SQL statement or to access another user's object. For example: the right to connect to a database, the right to create a table, the right to select rows from another user's table, the right to execute another user's stored procedure.</p> <p>System privileges are rights to enable the performance of a particular action, or to perform a particular action on a particular type of object.</p> <p>Roles are named groups of related privileges that you grant users or other roles. Roles are designed to ease the administration of end user system and object privileges. However, roles are not meant to be used for application developers, because the privileges to access objects within stored programmatic constructs needs to be granted directly.</p>
Sequences	DB2, Oracle	<p>A sequence generates unique numbers.</p> <p>Sequences are special database objects that provide numbers in sequence for input into a table. They are useful for providing generated primary key values and for the input of number type columns such as purchase order, employee number, sample number, and sales order number.</p> <p>Sequences are created by use of the CREATE SEQUENCE command.</p>
Structured Types	DB2	<p>Structured Types are useful for modeling objects that have a well-defined structure that consists of attributes. Attributes are properties that describe an instance of the type.</p> <p>A geometric shape, for example, might have as attributes its list of Cartesian coordinates. A person might have attributes of name, address, and so on. A department might have a name or some other attribute.</p>
Synonyms	Oracle	<p>A synonym is an alternate name for objects such as tables, views, sequences, stored procedures, and other database objects.</p> <p>A synonym is an alias for one of the following objects: table, object table, view, object view, sequence, stored procedure, stored function, package, materialized view, java class, user-defined object type or another synonym.</p>
Tables	All	<p>Tables are logical structures maintained by the database manager. Tables are composed of columns and rows. The rows are not necessarily ordered within a table.</p> <p>A base table is used to hold persistent user data.</p> <p>A result table is a set of rows that the database manager selects or generates from one or more base tables to satisfy a query.</p> <p>A summary table is a table defined by a query that is also used to determine the data in the table.</p>
Tablespaces	DB2, Oracle	<p>A tablespace is a storage structure containing tables, indexes, large objects, and long data. Tablespaces reside in database partition groups. They allow you to assign the location of database and table data directly onto containers. (A container can be a directory name, a device name, or a file name.) This can provide improved performance and more flexible configuration.</p>

Database Object	DBMS Platforms	Notes
Triggers	All	<p>A trigger defines a set of actions that are performed when a specified SQL operation (such as delete, insert, or update) occurs on a specified table. When the specified SQL operation occurs, the trigger is activated and starts the defined actions.</p> <p>Triggers can be used, along with referential constraints and check constraints, to enforce data integrity rules. Triggers can also be used to cause updates to other tables, automatically generate or transform values for inserted or updated rows, or invoke functions to perform tasks such as issuing alerts.</p>
Undo Segments	Oracle	<p>In an Oracle database, Undo tablespace data is an image or snapshot of the original contents of a row (or rows) in a table. The data is stored in Undo segments in the Undo table space.</p> <p>When a user begins to make a change to the data in a row in an Oracle table, the original data is first written to Undo segments in the Undo tablespace. The entire process (including the creation of the Undo data is recorded in Redo logs before the change is completed and written in the Database Buffer Cache, and then the data files via the database writer (DBW) process.)</p>
Unique Keys	All	<p>A unique key is a key that is constrained so that no two of its values are equal. The columns of a unique key cannot contain null values. The constraint is enforced by the database manager during the execution of any operation that changes data values, such as INSERT or UPDATE. The mechanism used to enforce the constraint is called a unique index. Thus, every unique key is a key of a unique index. Such an index is said to have the UNIQUE attribute.</p> <p>A primary key is a special case of a unique key. A table cannot have more than one primary key.</p> <p>A foreign key is a key that is specified in the definition of a referential constraint.</p> <p>A partitioning key is a key that is part of the definition of a table in a partitioned database. The partitioning key is used to determine the partition on which the row of data is stored. If a partitioning key is defined, unique keys and primary keys must include the same columns as the partitioning key, but can have additional columns. A table cannot have more than one partitioning key.</p> <p>Oracle: You cannot drop a unique key constraint that is part of a referential integrity constraint without also dropping the foreign key. To drop the referenced key and the foreign key together, check the Delete Cascade option for the foreign key.</p> <p>Clustered: A cluster composes of a group of tables that share the same data blocks, and are grouped together because they share common columns and are often used together.</p> <p>Filegroup: Lets you select the filegroup within the database where the constraint is stored.</p> <p>Fill Factor: Lets you specify a percentage of how large each constraint can become.</p>

Database Object	DBMS Platforms	Notes
Views	All	<p>A view provides an alternate way of looking at the data in one or more tables.</p> <p>A view is a named specification of a result table and can be thought of as having columns and rows just like a base table. For retrieval purposes, all views can be used just like base tables.</p> <p>You can use views to select certain elements of a table and can present an existing table in a customized table format without having to create a new table.</p>

## DBMS Connection Parameters by Platform

### IBM DB2 LUW

Connection Parameter	Description
Use Alias from IBM Client or Generic JDBC Configuration	If you choose to use the alias from the IBM client, select the appropriate alias name. Otherwise, choose Generic JDBC Configuration and enter the connection parameters, as specified.
Schema ID (Optional)	The name of the database schema.
Function Path	Optional. Enter an ordered list of schema names to restrict the search scope for unqualified function invocations.
Security Credentials	The log on information required by PowerSQL to connect to the data source.
Auto Connect	Automatically attempts to connect to the data source when selected in Data Source Explorer, without prompting the user for connection information.
JDBC Driver (Advanced)	The name of the JDBC Driver utilized by PowerSQL to initiate a JDBC standard access connection.
Connection URL (Advanced)	Used by the JDBC Driver to connect with a data source. Often contains host and port numbers, as well as the name of the data source to which it connects. For example: <b>jdbc:postgresql://host:port/database</b> <b>jdbc:derby://host:port/database</b>
Custom JDBC Driver Properties (Advanced)	The name and property value of any custom JDBC drivers associated with the data source.

### Microsoft SQL Server

Connection Parameter	Description
Use Network Library Configuration	If the data source utilizes a network library, select this parameter. The corresponding connection parameter fields become available. Otherwise, choose Generic JDBC Configuration and enter the connection parameters, as specified.
Host/Instance (JDBC Configuration)	The name of the data source.
Port (JDBC Configuration) (optional)	The listening port used in TCP/IP communications between PowerSQL and the data source.
Protocol (JDBC Configuration)	The communication mechanism between PowerSQL and the data source. Choose <b>TCP/IP</b> or <b>Named Pipes</b> .
Default Database (Optional)	The default SQL database name, as defined by the schema.
Security Credentials	The log on information required by PowerSQL to connect to the data source.
Auto Connect	Automatically attempts to connect to the data source when selected in Data Source Explorer, without prompting the user for connection information.
Allow Trusted Connections	Enables trusted connections to the data source from PowerSQL.
JDBC Driver (Advanced)	The name of the JDBC Driver utilized by PowerSQL to connect and communicate with the database.

Connection Parameter	Description
Connection URL (Advanced)	Used by the JDBC Driver to connect with a database. Often contains host and port numbers, as well as the name of the database to which it connects. For example: <b><code>jdbc:postgresql://host:port/database</code></b> <b><code>jdbc:derby://host:port/database</code></b>
Custom JDBC Driver Properties (Advanced)	The name and property value of any custom JDBC drivers associated with the data source.

## JDBC Connection Parameters

Connection Parameter	Description
Connect String	Used by the JDBC Driver to connect with a database. Often contains host and port numbers, as well as the name of the database to which it connects. For example: <b>jdbc:postgresql://host:port/database</b> <b>jdbc:derby://host:port/database</b>
Datasource Name	The name of the data source to which you want PowerSQL to connect.

## Oracle Connection Parameters

Connection Parameter	Description
Use TNS Alias	If the data source is mapped to a net service name via <b>tnsnames.ora</b> , select this parameter. Otherwise, choose Generic JDBC Configuration and enter the connection parameters, as specified.
Host/Instance (JDBC Configuration)	The name of the host machine on which the data source resides.
Port (JDBC Connection)	The listening port used in TCP/IP communications between PowerSQL and the data source.
Type (JDBC Configuration)	Indicates if the data source is defined via a system identifier (SID) or a service name.
Service/SID Name (JDBC Configuration)	The name of the system identifier (SID) or service name that identifies the data source.
Security Credentials	The log on information required by PowerSQL to connect to the data source.
Auto Connect	Automatically attempts to connect to the data source when selected in data source Explorer, without prompting the user for connection information.
Allow Trusted Connections	Enables trusted connections to the data source from PowerSQL.
JDBC Driver (Advanced)	The name of the JDBC Driver utilized by PowerSQL to connect and communicate with the database.
Connection URL (Advanced)	Used by the JDBC Driver to connect with a database. Often contains host and port numbers, as well as the name of the database to which it connects. For example: <b>jdbc:postgresql://host:port/database</b> <b>jdbc:derby://host:port/database</b>
Custom JDBC Driver Properties (Advanced)	The name and property value of any custom JDBC drivers associated with the data source.

## Sybase Connection Parameters

Connection Parameter	Description
Use Alias Information from your SQL.INI File	If the data source is mapped to a name via <b>SQL.INI</b> , select this parameter to use that name for connection. Otherwise, choose Generic JDBC Configuration and enter the connection parameters, as specified.

Connection Parameter	Description
Host/Instance (JDBC Connection)	The name of the host machine on which the data source resides.
Port (JDBC Connection)	The listening port used in TCP/IP communications between PowerSQL and the data source.
Default Database (JDBC Connection) (Optional)	The default database name, as defined by the schema.
JDBC Driver (Advanced)	The name of the JDBC Driver utilized by PowerSQL to connect and communicate with the database.
Connection URL (Advanced)	Used by the JDBC Driver to connect with a database. Often contains host and port numbers, as well as the name of the database to which it connects. For example: <b>jdbc:postgresql://host:port/database</b> <b>jdbc:derby://host:port/database</b>
Custom JDBC Driver Properties (Advanced)	The name and property value of any custom JDBC drivers associated with the data source.

# Index

## A

Additional resources 6  
associating 37

## C

Cache configuration 48  
change history 34  
code folding 30  
code formatting 26

## D

data source 10  
database objects 13, 51  
DB2 LUW 60  
delete 18  
Discussion Groups 6  
DMBS 6  
Documentation 6

## E

editing 21  
EMEA  
    Technical Support 6  
error 46  
error detection 22  
error logs 45  
execute 38

## F

FAQs 6  
files 17  
filters 13  
Forums 6

## G

global filters 14

## H

Hardware 5  
hyperlinks 26

## I

IBM DB2 LUW 60  
Import 18

## J

JDBC connection 62

## K

Knowledge Base 6

## L

license 47  
local history 35  
log 45

## N

nection 62  
new\_project\_wizard\_page 16  
new\_sql\_file\_wizard\_page 20

## O

object properties 11  
objects 11  
opening 16  
Operating System 5  
Oracle 62

## P

PowerSQL 5  
project\_info\_page 16  
Projects 16

## S

SQL file 21  
SQL Server 60  
Sybase 62

## T

technical Requirements 5  
Technical Support 6

## W

workspace 47