DB Optimizer 18.1

Quick Start Guide

Exported on 03/31/2022

IDERA

Table of Contents

Welcome to DB Optimizer 4
Installation and Configuration 5
Installing DB Optimizer 6
Installation Issues 6
Installing DB Optimizer on Linux 6
Specify a workspace 7
Licensing DB Optimizer 8
Getting Started with DB Optimizer 9
Starting DB Optimizer 10
Overview of the User Interface 11
Working with the Data Source Explorer 13
Adding Data Sources 14
Browsing Data Sources 16
Profiling a Data Source 17
Starting a Profiling Session 18
Analyzing Session Data 19
Saving a Profiling Session 22
Importing Statements to SQL Tuner 23
Tuning Your SQL Statements 24
Creating a New Tuning Job 25
Adding SQL Statements to a Tuning Job 26
Running a Tuning Job 28
Analyzing Tuner Results on the Overview Tab 30
Finding Missing Indexes and SQL Problems 31
Finding Problematic SQL or Schema 39
Implementing Recommendations on the Index Analysis Tab 41
SQL Code Assist and Execution 42
Code Extraction 44
Code Highlighting 45
Automatic Error Detection 46
Code Complete 47

Hyperlinks 48
Code Formatting 49
Code Folding 50
Code Quality Checks 51
SQL Execution 52
Configuring SQL Execution Parameters 54

Welcome to DB Optimizer

IDERA DB Optimizer simplifies SQL optimization and development for application developers with many features for improving productivity and reducing errors. A rich SQL IDE with statement tuning, data source profiling, code completion, real-time error checking, code formatting and sophisticated object validation tools helps streamline coding tasks. DB Optimizer's user interface helps improve overall productivity with integrated development, monitoring, and tuning components.

DB Optimizer has four components that when used together can optimize your database performance.

- **SQL Editor**. A developer can write Java in Eclipse that calls to the database with SQL. The SQL that calls to the database can be written in the SQL Editor with type ahead, code assist and quick fixes to show the users syntax and correct mistakes. For more information, see Creating and editing SQL files (SQL Editor).
- **Load Tester**. The SQL code can be run in the Load Tester to test execution by multiple concurrent users. User load testing is so often done by one single user and then problems don't appear until production with multiple concurrent users. Concurrent user testing is a breeze in DB Optimizer. For more information, see Using SQL Load Editor/Tester.
- **Profiler**. You can run the Profiler while the Load Tester is executing to show clearly the impact on the database. The profiler can also be used by QA on load simulation. Finally, the Profiler can be run on any production database to clearly show load, bottlenecks, and sources of bottlenecks or resource consumption. For more information, see Using profiling.
- **Tuner**. Finally, if a problem SQL is found on the system the Tuner will show if it's correctly optimized by the database or not, and if not it will show the best plan and what hints or optimizer directives can be included in the SQL to force the database to use the optimal plan. For more information, see Using tuning.

Installation and Configuration

This section contains information on configuring DB Optimizer. It includes information on setting up the system directory for project files, as well as licensing information. Additionally, this section contains information on setting preferences within the application for the customization of various features and functionality.

- Installing DB Optimizer
- Initial setup
- Customizing DB Optimizer (Preferences)

Installing DB Optimizer

To install DB Optimizer, run the installer and follow the prompts provided by the Installation Wizard. When the installation is complete, check the Readme file to see if it contains any known issues or advisory notes that will affect your installation of the product.

- Installation Issues
- Installing DB Optimizer on Linux

Installation Issues

Install DB Optimizer in a different directory than previous versions of DB Optimizer. If DB Optimizer is installed in the same directory, the following error appears at application startup: "Unable to read workbench state. Workbench UI layout will be reset."

If a previous version is already installed on your machine, either change the default Workspace or delete all files from the Workspace directory. The Workspace directory location appears whenever you start DB Optimizer, and can be redefined from this dialog box. If the same Workspace is indicated for the new version as was indicated previously, the following error appears: "Cannot restore workbench layout."

If DB Optimizer is already running, you can also modify the Workspace directory by choosing **File > Switch Workspace > Other** from the Main Menu.

Installing DB Optimizer on Linux

To install DB Optimizer in a Linux environment

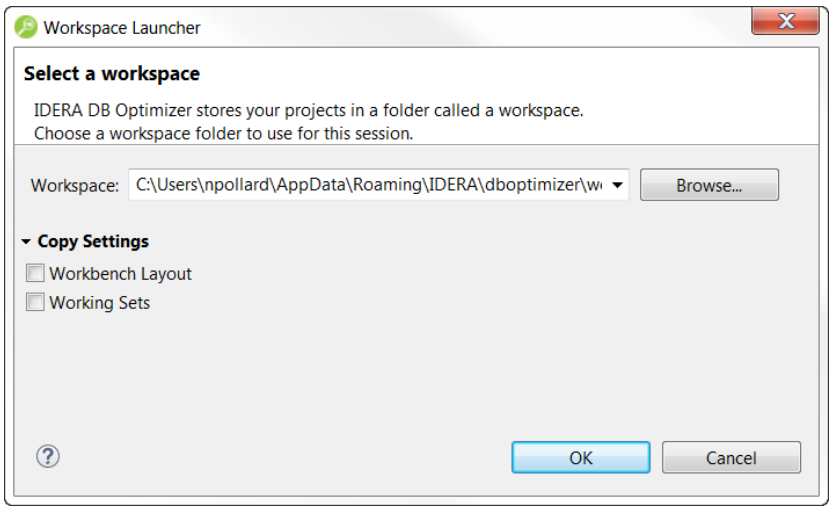
1. Download the DB Optimizer installer and set the execution permissions as follows:
   `chmod 755 dbo_<version>.bin`
2. Launch the installer in GUI mode from the console.
   `./dbo_<version>.bin`
   The DB Optimizer splash screen appears for a short time and afterwards the installer dialog appears.
3. On the Introduction dialog, click **Next**.
4. On the License Agreement dialog, read the License Agreement, select **I accept the terms of the License Agreement**, and then click **Next**.
5. On the Choose Install Set dialog, choose one of the following options, and then click **Next**.
   - **Install as Standalone Application**. This option installs DB Optimizer as a Standalone Application, without any prerequisites.
   - **Install as Eclipse Plug-in**. This option requires Eclipse 3.6 or higher or an Eclipse-based RCP application.
6. On the Choose Instal Folder dialog, depending on the previously selected option, you can select a destination folder where to install the Standalone version or you are required to browse to the folder that contains Eclipse 3.6 (or higher) or an RCP application based on Eclipse 3.6.
7. After selecting the desired folder, click **Next**.
8. On the Choose Link Folder dialog, you can change the location where the Application/ Uninstall links are created. Choose one of the options, and then click **Next**.
9. On the Pre-Installation Summary, review the installation details, and then click **Install**.
   The Installing dialog shows the progress of the installation; it is closed when the installation is complete.
10. When the Install Complete dialog appears, you can select **Start IDERA DB Optimizer**, and then click **Done** in order to start the application for a first time use.

Specify a workspace

When you install DB Optimizer, you are prompted to create a workspace. Then, when you launch DB Optimizer, you have an opportunity to choose your workspace. At any time while running DB Optimizer, you can change your workspace.

To switch your workspace

1. In DB Optimizer, select **File > Switch Workspace**. The Workspace Launcher appears.



2. Browse for the appropriate workspace or create a copy of an existing workspace using the available **Copy** Settings.
3. Click **OK** after making your selection. DB Optimizer opens or creates the appropriate workspace.

Licensing DB Optimizer

The first time you launch DB Optimizer, you are prompted to activate the product. Choose to activate by Internet, and then 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 email alternative.

To continue using DB Optimizer after the evaluation period

1. Select **Help > IDERA Licensing > License Registration**.
2. Follow the prompts.

Getting Started with DB Optimizer

The following tutorial sessions enable you to get started using DB Optimizer. Their purpose is to provide you with the foundation needed to fully utilize the features and benefits of DB Optimizer, and apply them to your own enterprise. This section contains information on how to register data sources from your enterprise, and how to profile, analyze, and tune SQL statements, sessions, and events for the purpose of optimizing the efficiency of your data sources, as wel as identify and prevent system bottlenecks and other waitrelated issues.

This section includes the following topics:

- Starting DB Optimizer
- Overview of the User Interface
- Profiling a Data Source
- Tuning Your SQL Statements
- SQL Code Assist and Execution

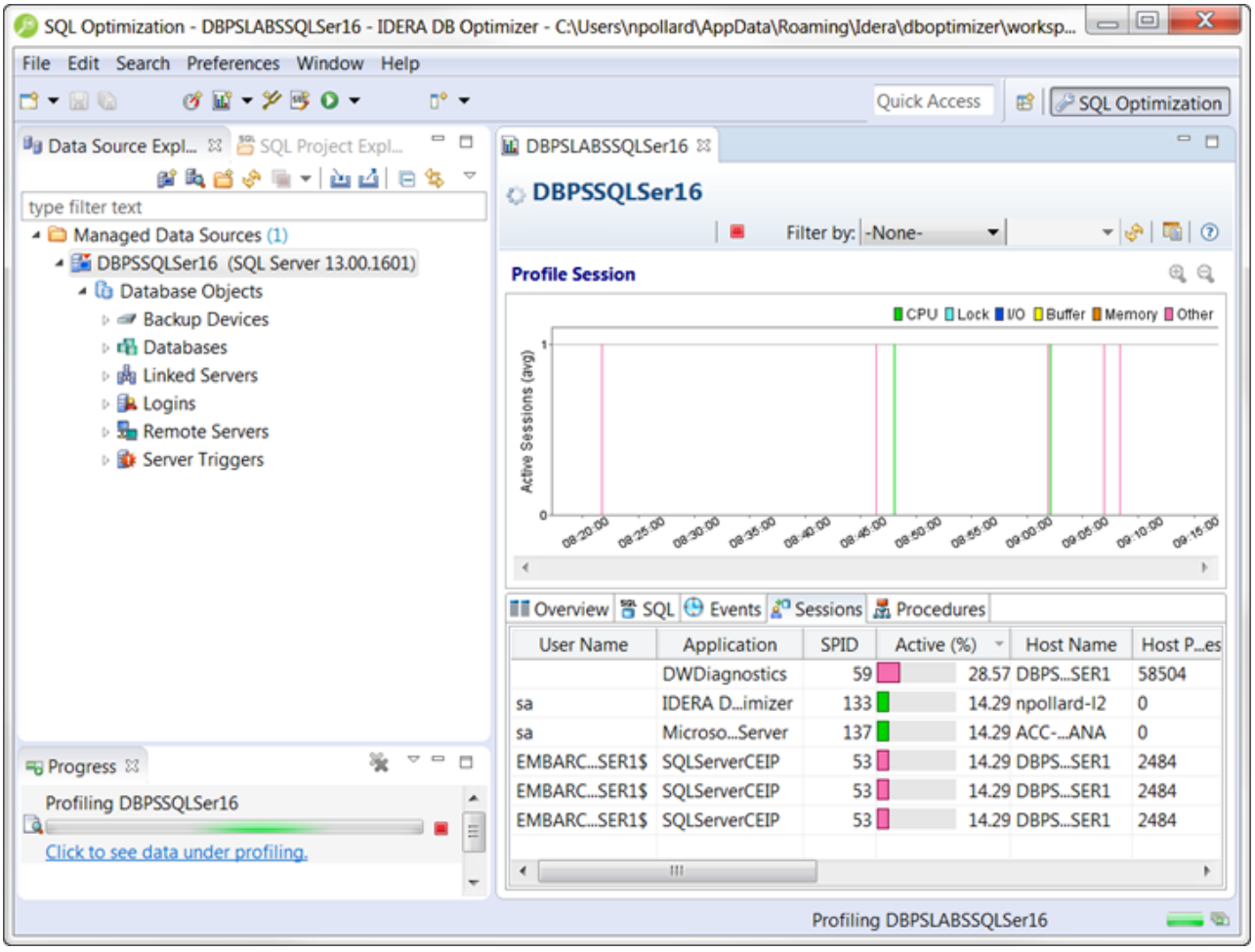
Starting DB Optimizer

From the Desktop, choose the Windows Start menu and then select **Programs > IDERA > IDERA DB Optimizer X.X > DB Optimizer X.X**.

Once you start the application, you can select **Help** from the menu bar to find additional resources that complement and build upon the features and tasks presented in this wiki.

Overview of the User Interface

The DB Optimizer application environment is known as the **Workbench** that provides an interface through which you manage data sources and analyze and tune statements. The Workbench is composed of common interface elements that provide tools to accomplish these tasks. These objects provide a uniform system for working with tuning jobs, profile sessions, and data sources.



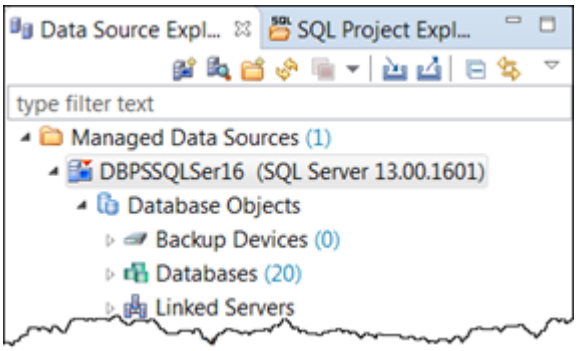
The Workbench provides an environment to build profiling sessions and tune query statements. The **Workbench Space** includes views, editors, and the menu bar and command toolbar. Views and editors enable you to perform DB Optimizer functions and work tasks, as well as manage resources.

- **Views** are used to navigate a hierarchy of information, open Editors or display the properties of various application elements. For example, the Data Source Explorer view provides a tree of all data sources in the environment and the comparison jobs associated with each. You can launch these jobs directly from Data Source Explorer, modify the connection properties of data sources, or create and edit configuration archives from this View.
- **Editors** are used to access DB Optimizer functionality. For example, the SQL Editor provides a means to view, edit, and save SQL code. Editors are differentiated from Views in that they operate on an individual level rather than a supportive one. The SQL Profiler and SQL Tuner are also considered to be editors.

- The **Menu Bar** and **Command Toolbar** contain commands that execute various functional aspects of the application such as launching views and editors, navigation commands, and environment preference settings. The command toolbar contains icons that represent specific menu commands.

Working with the Data Source Explorer

Data Source Explorer provides a tree view of all registered data sources. It breaks down the components of each data source and categorizes databases and corresponding database objects by object type and underlying SQL code. This feature provides you with a view of the contents of data sources in your enterprise in an easily navigated and cataloged interface.



Data Source Explorer sorts databases and database objects by category within DB Optimizer. If data sources are particularly large or complex, or you are only developing specific objects, you can apply database object filters on the view in Data Source Explorer.

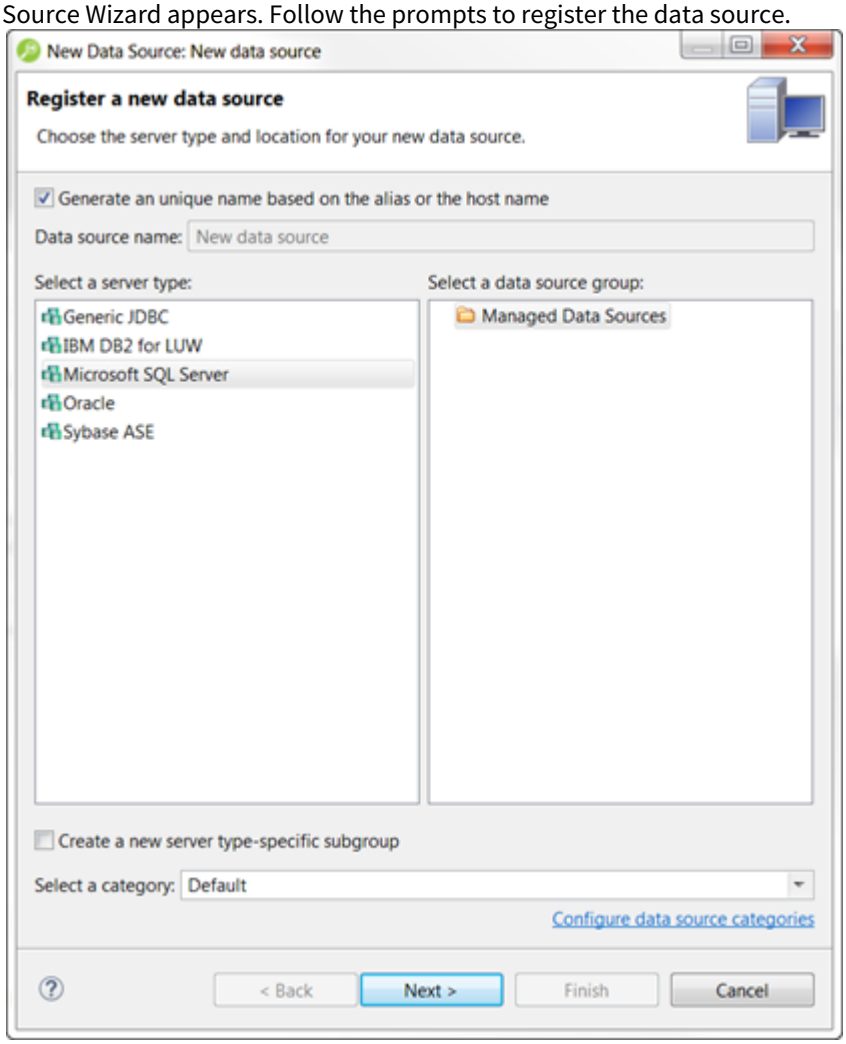
Adding Data Sources

In order to profile and tune statements, you need to register the data sources to be analyzed in the environment by providing connection information and other details to DB Optimizer. Data sources are registered and managed in Data Source Explorer. Each time you register a new data source you need to specify its connection information and organize it in the View, as needed. Once a data source has been registered, it remains stored in a catalog and does not need to be registered again each time you open DB Optimizer. Furthermore, it can be used in multiple jobs, archived, or otherwise "shared" with regards to the general functionality of the application.

The Data Source Explorer view provides an organizational tree of registered data sources and the parameters associated with them.

To add a data source

- Select Data Source Explorer, and then choose **File > New > Data Source** from the Menu bar. The New Data Source Wizard appears. Follow the prompts to register the data source.

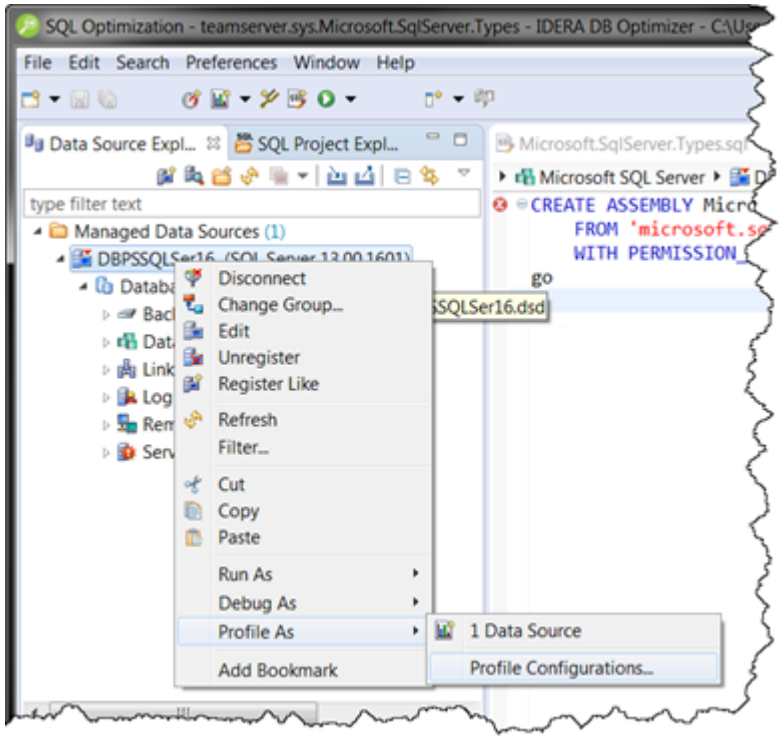


You can categorize data sources in order to color code them in the Data Source Explorer which makes the data source easier to find when you have a long list of data sources. The data source will be decorated with the color associated with the category selected for the data source. A set of preconfigured categories include Development,

Test, QA, and Production. You can also create custom categories. Existing data sources can be categorized using the Data Source Properties page; right-click a disconnected data source, and then select **Properties**.

Browsing Data Sources

To analyze and tune statements on data sources within your enterprise, you must access the data source objects within the application. It is important to view databases and underlying code in an organized manner, especially when maintaining a large system. Data Source Explorer's tree structure can be used to view databases, tables, and other information about data sources. Expand the nodes of each registered data source to view details about each data source. Data Source Explorer is also used to launch profiling sessions, by enabling you to select the data source that you want to execute, and then launching a profiling session from your selection.



The Data Source Explorer tree provides a list of databases, and enables you to launch Optimizer features, such as SQL Profiler, via the context menu. All objects listed in Data Source Explorer are sorted by data source name. Additionally, you can view the underlying SQL code for an individual database object.

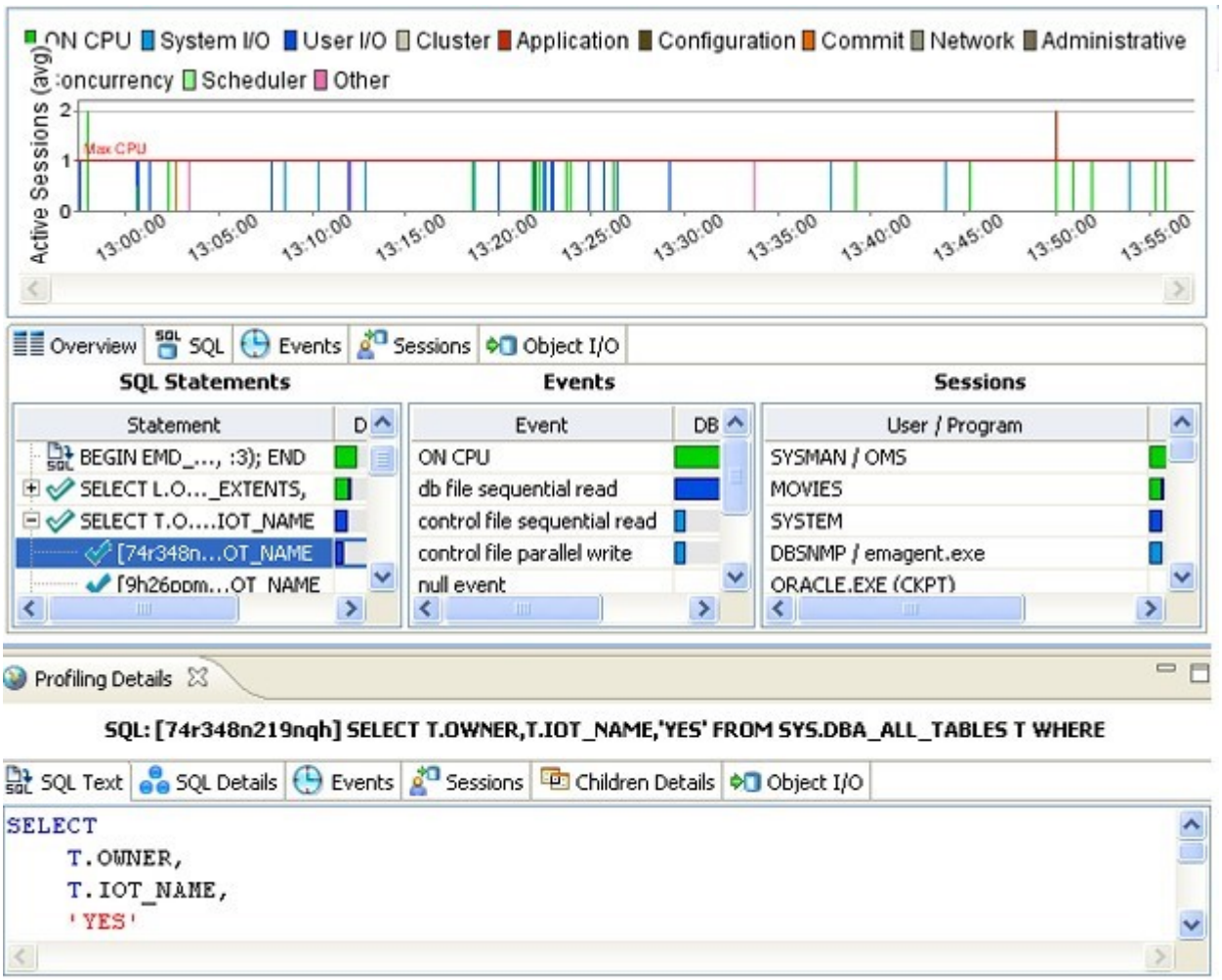
To browse a data source

1. Expand a data source node, providing credentials if prompted.
2. Expand additional subnodes to drill down through object types and then individual objects.
3. Double-click the **Code** subnode for a particular object to view the code in SQL Editor.
   Notice in the previous screenshot, there are Profiling Repositories listed in the Data Source Explorer. For Oracle data sources only, you can save your profiling sessions to the Profiling Repository for later examination. We'll discuss this more later in Saving a Profiling Session.

Profiling a Data Source

> ⚠ The SQL Profiler is not available in the Basic version of DB Optimizer.

SQL Profiler samples a data source and builds a statistical model of the load on the database. Profiling is used to locate and diagnose problematic SQL code and event-based bottlenecks. Profiler also enables you to investigate execution and wait time event details for individual stored routines. Results are presented in the profiling editor, where you can identify problem areas and view individual SQL statements, as needed.



SQL Profiler is composed of the following diagnostic parts:

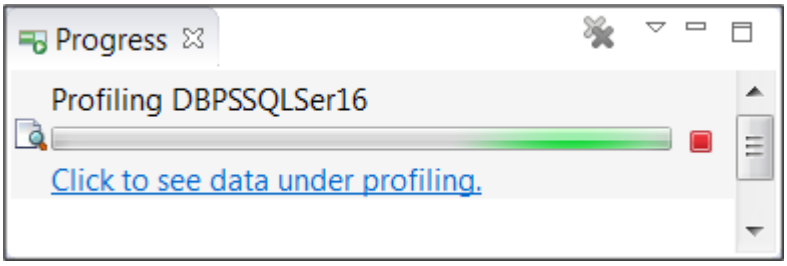
- The **Load Graph** provides a display of the overall load on the system. The bars represent individual aspects of the data source.
- The **Top Activity** section displays where the load originates. Tabs available in this section differ by platform, but typically available are the top SQL statements, the top events the database spends time in, and the top activity sessions that may be causing issues in terms of query time and response.
- The **Profiling Details View** displays detailed information for any item selected from Top Activity. For example, examining a SQL statement from Top Activity displays the identification parameters and the execution statistics of that specific statement selection.

Starting a Profiling Session

In order to access SQL Profiler, you need to run a profiling session on a data source registered in Data Source Explorer.

To run a profiling session

- In Data Source Explorer, right-click on a data source and select **Profile As > Data Source**. The profiling session begins.



Once a profiling session launches, it runs until you stop it or it has run for its specified length of time. You can stop a session by clicking the **Stop** icon on the right-hand side of the Profiling progress bar.

When the profiling session is complete, the first two sections (**Load Chart** and **Top Activity**) are populated with information about the database load. You can then begin analyzing the data and identifying problem areas.

Analyzing Session Data

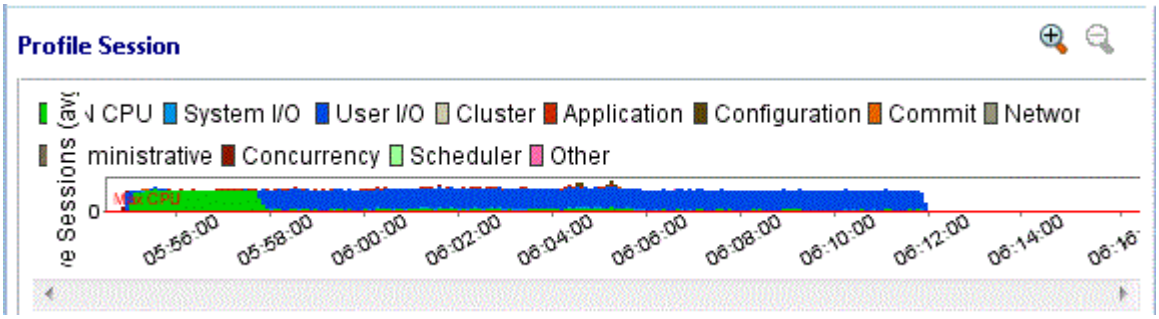
The SQL Profiler is composed of three essential diagnostic views that provide information about load on a particular database in the system. These views enable you to identify bottlenecks and view details about specific queries that execute and wait over the course of a profiling session.

The following describe the three components of SQL Profiler, in descending order of granularity:

- Load Chart
- Top Activity
- Profiling Details

Load Chart

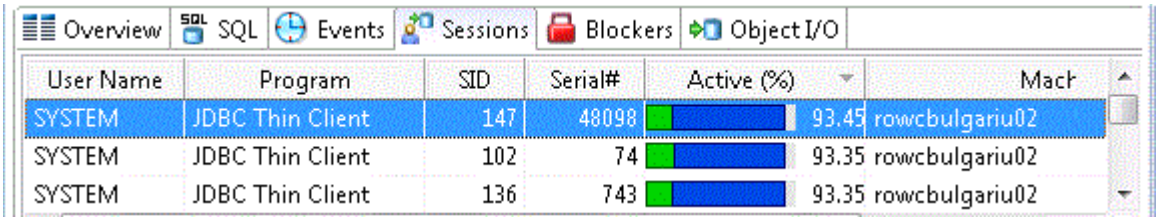
The Load Chart provides an overview of the load on the system, and is designed as a high level entry point to reading session results. The colored bars represent individual aspects of the database, and the graph can be used to discover bottlenecks.



Time is displayed on the X axis, and the Y axis shows the average number of sessions waiting or executing. Each support platform type has a specific set of wait event times. For example, Sybase platforms will display CPU, Lock, Memory, I/O, Network, and Other categories. Use the chart legend to understand the graph. It displays a color and code scheme for executing and waiting session categories in the upper right-hand corner of the chart.

Top Activity

Below the Load Graph, the Top Activity section displays where the load originates and outlines items such as the top SQL statements, top events, and the top activity sessions on the database. It is composed of a series of DBMS platform-specific tabs that provide detailed statistics on individual SQL statements and sessions that are waiting or executing over the length of the profiling session.



While available tabs differ by platform, the most common are:

- The **Overview** tab provides summary information about SQL statements and events and their activity levels, and sessions, their system process IDs and their activity level. You can reorder the rows in any of the three sections of this tab. For example, clicking the **Event** column in the **Events** section changes the alphabetical order to ascending or descending.

- The **SQL** tab provides information about SQL statements and procedures. This includes all INSERT, SELECT, DELETE, and UPDATE statements that are executing or waiting to execute over the length of the profiling session.
- The **Events** tab displays information about wait events, and should be used to tune at the application or database configuration level. For example, if the top events are locks, then application logic needs to be examined. If the top events are related to database configuration, then the database setup should be investigated.
- The **Sessions** tab displays information about sessions, and can be used to discover sessions that are very active or bottlenecked.
- The **Object I/O** tab provides information about I/O when profiling an Oracle data source.
- The **Procedures** tab provides information about the procedures running in the session when profiling a SQL Server or Sybase data source.

Profiling Details

When you select any item from Top Activity, details are displayed on the **Profiling Details** view.

> You may have to select **Windows > Show View > Profiling Details** to display the Profiling Details view.

The tabs that compose the **Profiling Details** view are dependent on the nature of the object selected in Top Activity, in order to reflect that item's specific information. The tabs are also dependent on the data source platform. For example, the **Object I/O** tab is available only for the Oracle platform whereas the **Procedures** tab is available only for the SQL Server, Sybase, and Oracle platforms.

Depending on the data source platform you have specified, the tabs that appear in the view will be different, in order to accommodate the parameter specifics of the statement you have selected.

Depending on the top activity selected and the profiled platform types, some tabs may not be available.

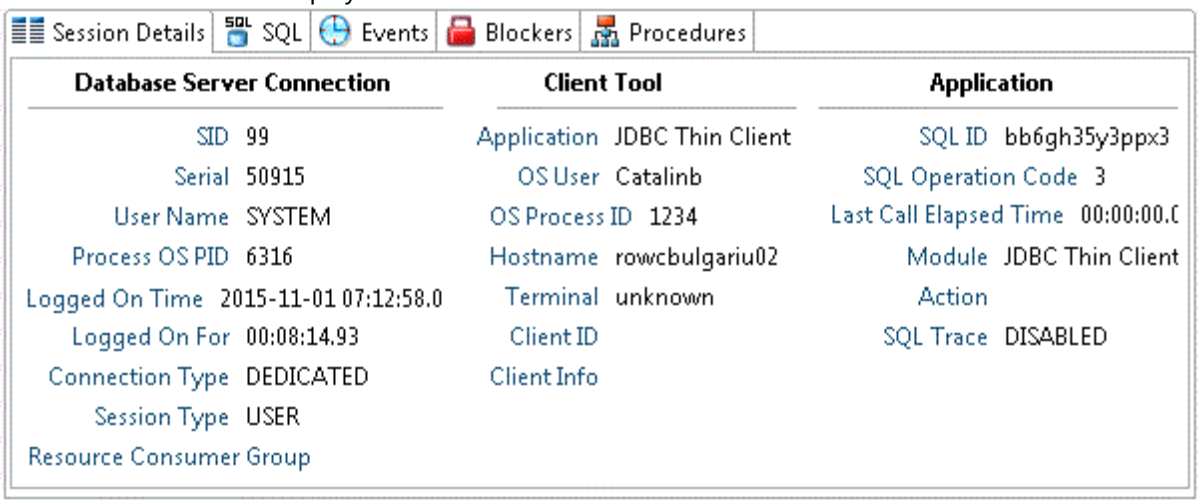
When right-clicking on a SQL statement in the Top Activity section in the SQL Profiler, if the SQL statement is run by a different user than the user who is running DB Optimizer, then the User Mismatch dialog appears, with an example of the following message: "This query was executed by [SOE] and you are currently connected as [system]. We recommend you reconnect as [SOE] to tune the SQL. Would you like to continue anyway?" This message indicates that the statement is being tuned by a user other than the user who originally ran the query, and tables may be missing based on the different schemas. Click **OK** to run the query, or click **Cancel** and run the tuning job under the original user.

View Session Details

To view session details

1. In the **Profile Session** area of the SQL Profiler, in the **Sessions** column, click anywhere in the row of an application that ran during the profiling session.

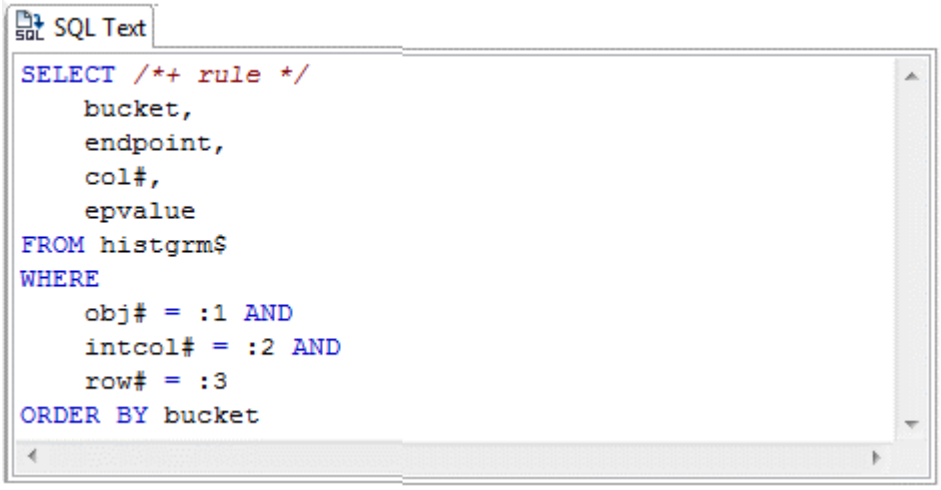
2. In the **Profiling Details** area, click the Sessions tab.
   Details of the session are displayed.



View SQL

To view SQL

1. In the **Profile Session** area of the SQL Profiler, click anywhere in the row of an application that ran during the profiling session.
2. In the **Profiling Details** area, click the SQL Text or SQL tab.
   The associated SQL text is displayed.



From the SQL tab you can easily tune a statement by right-clicking a statement on the SQL tab to initiate the tuner, which then opens with the selected statement in the Ad hoc SQL tab of the Tuner Input.

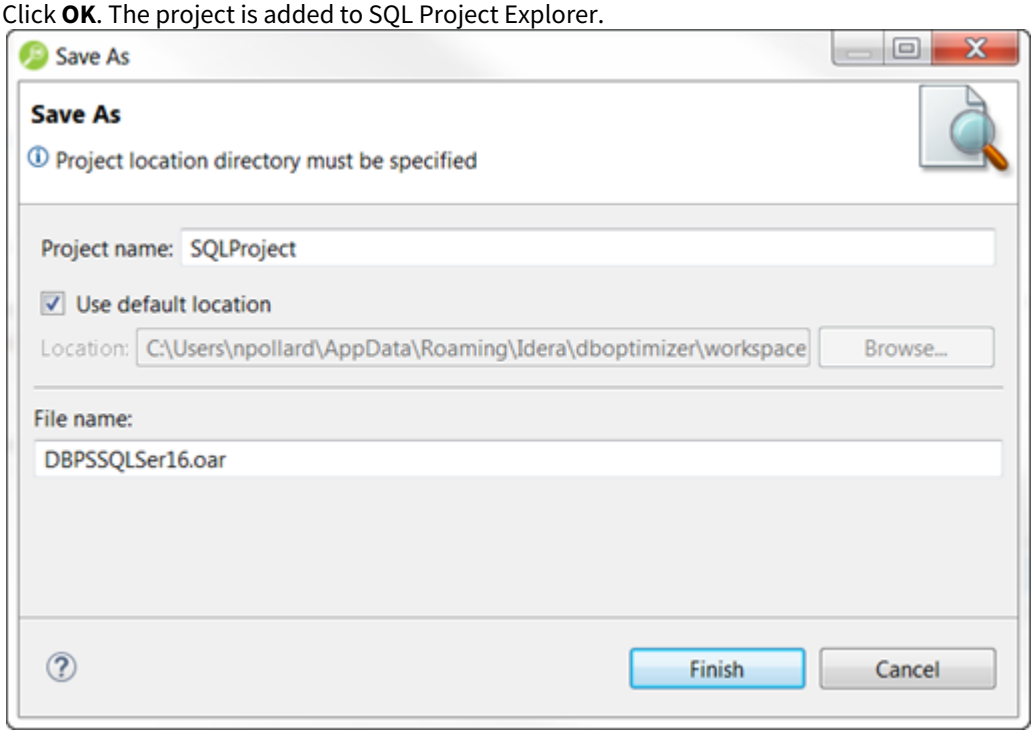
Saving a Profiling Session

A profiling session can be saved to a file with a .oar suffix that contains the name of the data source. When profiling Oracle data sources, you can alternatively save the session data to the Profiling Repository. This enables you to open the file at a later time for analysis.

Save a Profiling Session as a .OAR File

To save a profiling session as a .OAR file

1. Select the profiling session and then choose **File > Save As**.
2. Specify the project location where you want to save the file, and then modify the file name as needed.
3. Click **OK**. The project is added to SQL Project Explorer.



Once you save a profiling session, it appears in the SQL Project Explorer under the name you saved it as. It can be opened again by double-clicking the project name.

Saving a Profiling Session in the Profiling Repository

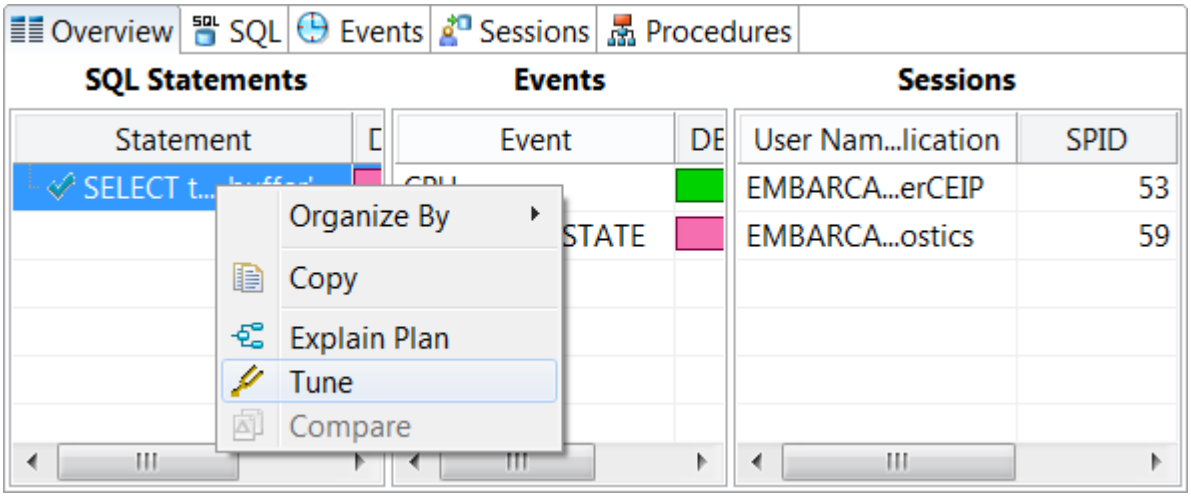
You can also have the session saved automatically as a Profiling Repository file in an Oracle data source. This allows you to profile your data source for a longer period of time, for days or even weeks.

1. Right-click the data source you want to profile and select **Profile As > Profile Configurations**.
2. In the Profile Configurations dialog, select **Save to data source**, and then select the data source where you want to save the profile sessions.

The saved session file, named using the date and time when the profiling session was stopped, can be viewed from the Data Source Explorer and thus available to share with other DB Optimizer users.

Importing Statements to SQL Tuner

SQL Profiler enables you to submit one or more statements into SQL Tuner. This enables you to take advantage of Tuner's hint-based and transformation-based suggestions if you want to tune a problem statement that you detected over the course of a profiling session.



Context menu commands in SQL Profiler enable you to import statements to a tuning job directly from the Profiler interface.

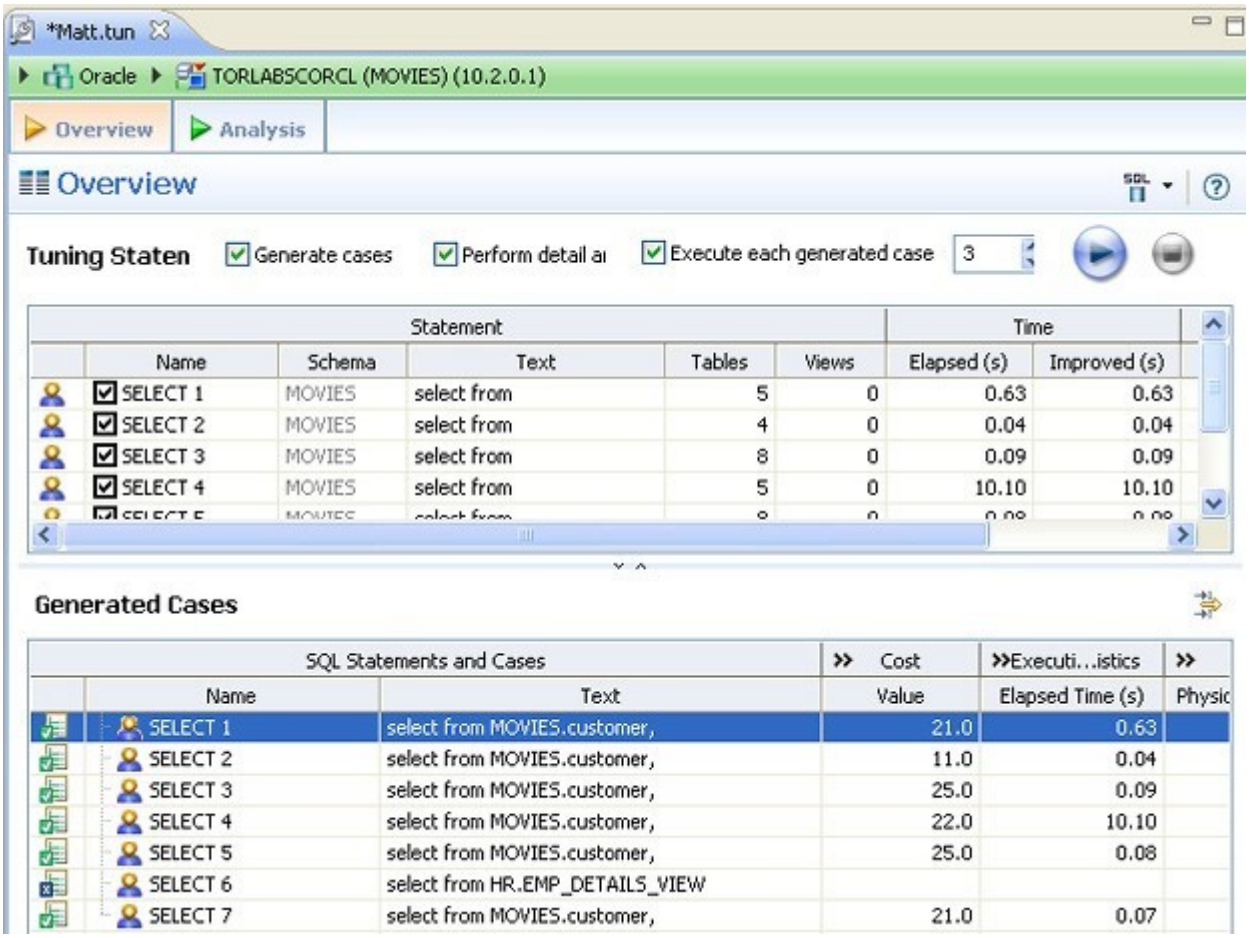
Import a statement from SQL Profiler into SQL Tuner

To import a statement from SQL Profiler into SQL Tuner

- Select one or more statements in Profiler, right-click, and then select **Tune** from the context menu. SQL Tuner opens and contains the selected statements in a new tuning job. You can now proceed to tune the problematic statements.

Tuning Your SQL Statements

SQL Tuner provides an easy and optimal way to discover efficient paths for queries that may not be performing as quickly or as efficiently as they could be. Tuner enables the optimization of poorly-performing SQL code through the detection and modification of execution paths used in data retrieval. This is primarily performed through hint injection, and index and statistics analysis.

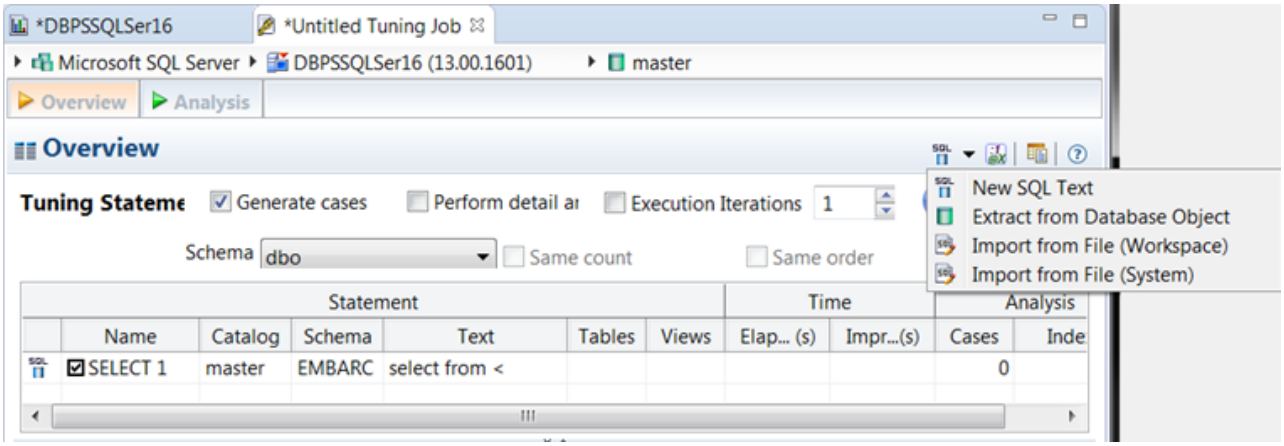


For example, different joining methods such as nested loops or hash joins can be used and will be tested, as appropriate. Tuning will select alternate paths, and enable you to change the original path to one of the alternates. Execution paths slower than the original path are eliminated, which enables you to select the quickest of the returned selections and improve query times, overall.

This enables the DBA to correctly optimize queries in the cases where the native optimizer failed.

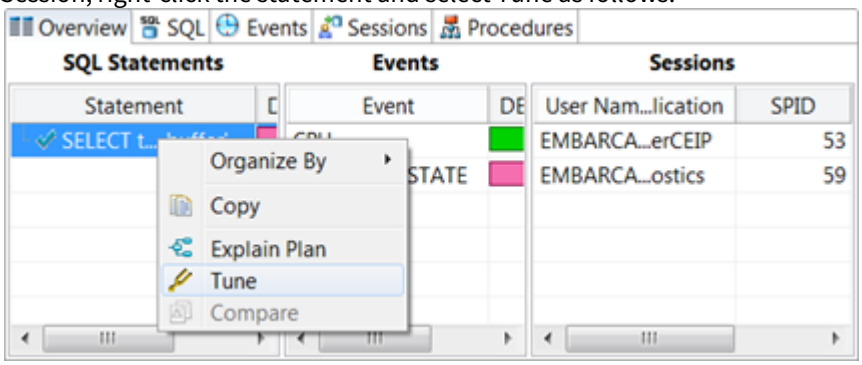
Creating a New Tuning Job

New tuning jobs are created from scratch where you can specify the statements to be tuned from a variety of sources, or statements can be directly imported from existing profiling sessions on data sources currently registered in the environment.



To create a new tuning job

- If you determined from a profiling session that a specific SQL statement should be tuned, in the Profile Session, right-click the statement and select Tune as follows:



OR

- Select **File > New > Tuning Job**, or click the **New Tuning Job** icon on the Toolbar. SQL Tuner opens and you can proceed to set up the parameters of the new job.

Once you define the input of the tuning job, you can save the file with a .tun suffix via the **Save As** command. The job is added to the SQL Project Explorer and can be re- opened and re-run at any time once it is saved to the system.

Adding SQL Statements to a Tuning Job

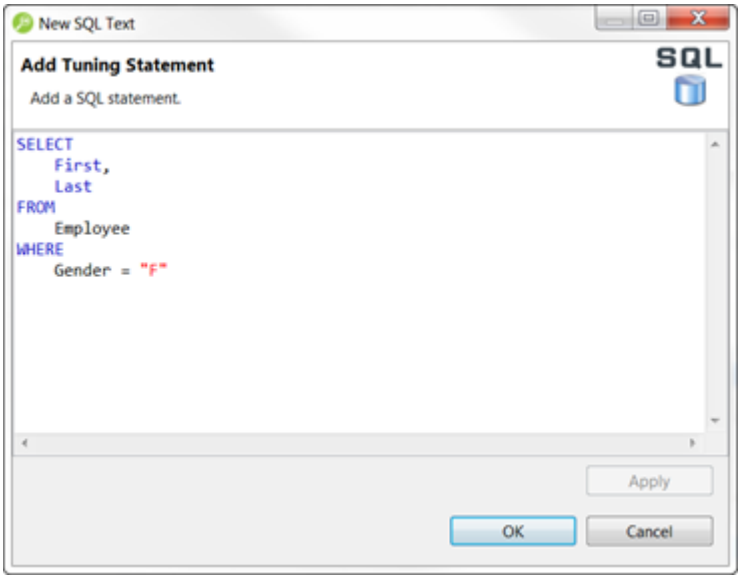
Once you have created a name for the tuning job and indicated its source, you can add SQL statements that you want the job to tune. Statements are added to a job via the options available in the SQL list on the Overview tab. All standard DML statements (SELECT, INSERT, DELETE, UPDATE) are viable for the tuning procedure. On SQL Server 2008 and higher, MERGE statements can also be tuned.

There are four or five different ways to add SQL statements to a job, as reflected by the options available in the SQL list as shown previously:

- **New SQL Text**. Copy/paste SQL statements to the New SQL Text window or write queries by hand. You can also paste a statement directly into the **Overview** statement grid by copying the statement, right-clicking anywhere on that grid, and then selecting **Paste**.
- **Extract from Database Object**. Drag and drop database objects from the Data Source Explorer to the Statements grid on the Overview tab.
- **Import from File (Workspace) and Import from File (System)**. Browse the workspace or file system and select SQL files.
- **Scan Oracle SGA**. For the Oracle platform only, you can also scan the System Global Area (SGA) for statements to tune.

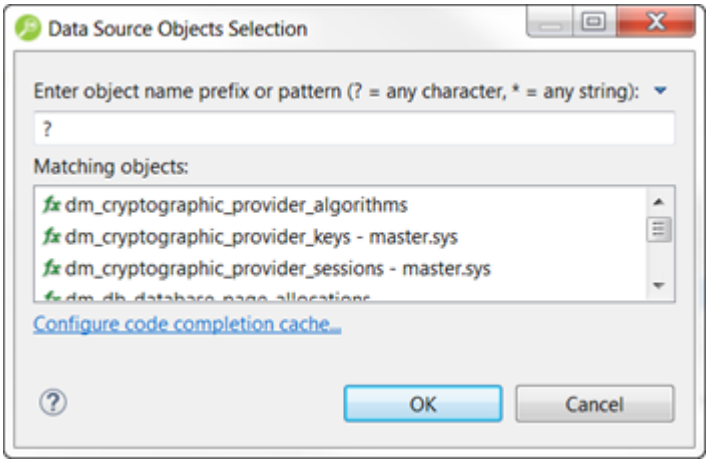
Add New SQL Text

From the **SQL** list, select **New SQL Text**, and then manually type an SQL statement in the window. Alternatively, copy/paste the statement from another source.



Add a Database Object

From the **SQL** list, select **Extract from Database Object**. The Data Source Objects Selection dialog appears. Type an object name prefix or pattern in the field provided, and then choose a statement from the window as it populates to match what you typed.



You can also drag an object, such as a Packages, Package Bodies, Views, and Functions, from the Data Source Explorer to the grid on the Overview tab. In order to drag a database object on the Overview tab, the database object in the Data Source Explorer must be in the same database that displays in the bread crumbs at the top of the Tuning window.

Add a Saved SQL File

From the **SQL** list, select **Import from File (Workspace)** or **Import from File (File System)**, depending on where the file you want to add is stored. Select a file from the dialog that appears and it will be added to the tuning job.

Add Active SQL in SGA

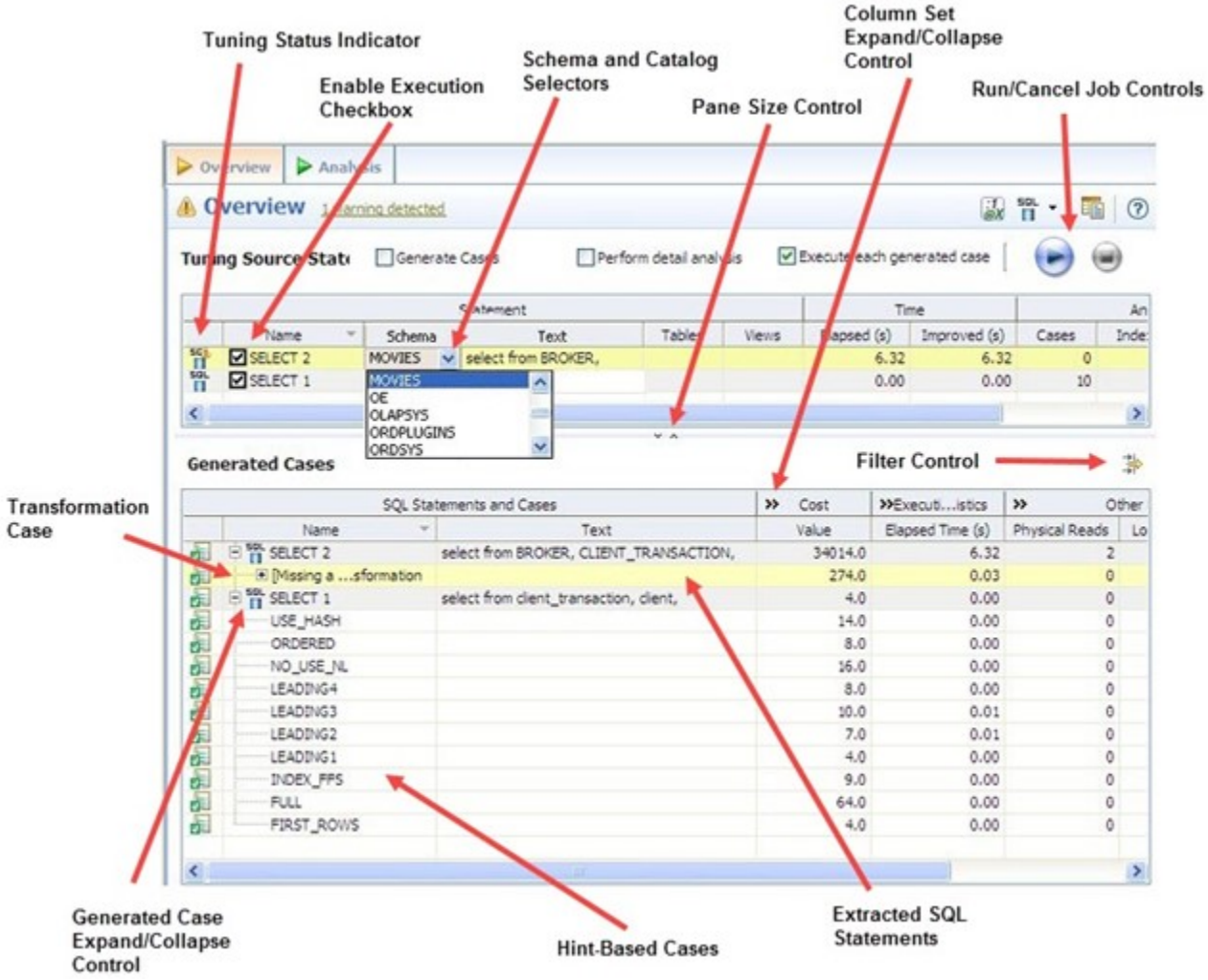
From the **SQL** list, select **Scan Oracle SGA**. Specify any filters as required, and then click **Next**. From the list of SQL statements retrieved from the SGA, select those you want to optimize, and then click **Finish**. The selected statements are copied to the tuning job.

Running a Tuning Job

After you add SQL statements to the job, click the Overview tab. Once you choose your tuning options and click the **Run Job** icon, the DML is parsed from the statements and added to the **Generated Cases** area. The Generated Cases are alternative execution paths or explain paths that could be better or worse than the default path the database uses. When these cases are executed, you can use the execution statistics to determine which case would optimize performance.

Each extracted statement is listed by **Name** and **Text**. Additionally, each statement has a **Cost**, **Elapsed Time**, and **Other Execution Statistics** value that provide information on how effectively each case executes on the specified data source. These parameters let you compare the efficiency of the original statements to the cases generated by the tuning process when it is executed.

You can double-click a generated case to view or edit the SQL source of the statement.



The **Tuning Status Indicator** provides the status of each statement or case, and indicates if they are ready for execution. In some cases, SQL code may need to be corrected or bind variables may need to be set prior to

executing statements. When you try to tune a statement containing a bind variable, you are now warned that either the type is not set or the value is not set.

Use the check boxes to select which statements and cases you want to run, and then click the **Run** icon in the upper right-hand corner of the screen. The Execute each generated case field enables you to execute each selected statement or case.

Use the **Schema and Catalog Selectors** to select a schema and catalog for the tuning job. The catalog selector is available only for SQL Server and Sybase data sources. By specifying the schema and catalog, the tuner can use the paths of the schema and catalog selected to find the tables queried in the job rather that use the paths of the schema and catalog used to connect to the data source. If you change the schema or catalog used in a tuning statement, you will need to refresh the tuning statements in order for new cases to be generated, which take into consideration the schema used. Right-click a tuning statement, and then select **Refresh Tuning Statements**.

Once you execute a tuning job, the Generated Cases tab reflects SQL Tuner's analysis of the specified statements. Once analyzed, you can proceed to modifying the Tuner results and applying specified cases on the data source to optimize its performance.

Run a Tuning Job

To run a Tuning Job

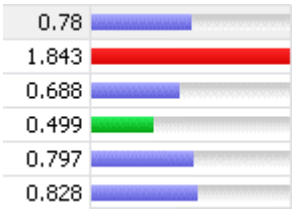
1. Once you have a SQL statement that is a tuning candidate, navigate to the Overview tab.
2. In the **Tuning Statements** area, select the checkbox next to the Statement name that you want to analyze and then:
    - *To analyze the SQL statements*, click **Generate cases**.
    - *To perform the analysis that populates the Analysis tab now*, click **Perform detail analysis**. Otherwise, this analysis is performed when you click the Analysis tab.
    - *To have the system generate execution statistics*, click **Execute each generated case**, and then select the number of times the system should execute each generated case. Multiple executions can verify that the case results are not skewed by caching. For example, the first time a query is run, data might be read off of disk, which is slow, and the second time the data might be in cache and run faster. Thus, one case might seem faster than another but it could be just benefiting from the effects of caching. Generally, you only need to execute the cases once, but it may be beneficial to execute the cases multiple times to see if the response times and statistics stay the same.
3. Click the **Run Job** icon at the top right-hand side of the window. The tuning job runs, analyzing each statement and case, and providing values in the appropriate columns.

Analyzing Tuner Results on the Overview Tab

When you have executed a tuning job, the **Generated Cases** area of the Overview tab reflects Tuner's analysis of the specified statements and cases. The Generated cases create alternative execution or explain paths that could be more or less efficient than the default path the databases uses. Executing these cases provides the statistics necessary to optimize performance.

Once a tuning job has executed, use the **Cost** and **Execution Statistics** value columns to determine the fastest execution path for each statement. The **Cost** column shows the performance cost of an execution path as determined by the database. The **Execution Statistics** are the actual results of running the SQL statement using the generated case. This is where DB Optimizer can help you find where the database default path is actually not the optimal path. The **Elapsed Time(s)** and **Results** columns can more accurately show the most efficient execution path.

In the **Cost** and **Execution Statistics** columns, the values of the original statement are considered to be the baseline values. A **Cost** column can be expanded to provide a graphical representation of the values for statements and cases. Similarly, the **Execution Statistics** column can also be expanded to display a graphical representation of values as well. The bar length and colors are intended as an aid in comparing values, particularly among cases.



Case query times based on the original statement can be represented as colored bars on the Generated Cases area of the Overview tab to help you determine the fastest execution path for the given selections. The baseline value of the original statement spans half the width of the column, in terms of bar length. For cases of the original statement, if one or more cases show a degradation value, the largest value will span the width of the column. Bar lengths for all other cases will then be displayed in comparison length to the highest degradation value.

The cost and execution results are color-coded as follows:

- **Light blue**. These cases are within the degradation and improvement threshold. Applying these changes may marginally improve or degrade the efficiency of the SQL statement.
- **Green**. These cases have values less than the improvement threshold. There is a high probability that changing the SQL statement with this alternative execution path will improve efficiency.
- **Red**. These cases have values higher than the improvement threshold. Implementing these changes will degrade the efficiency of the SQL statement.

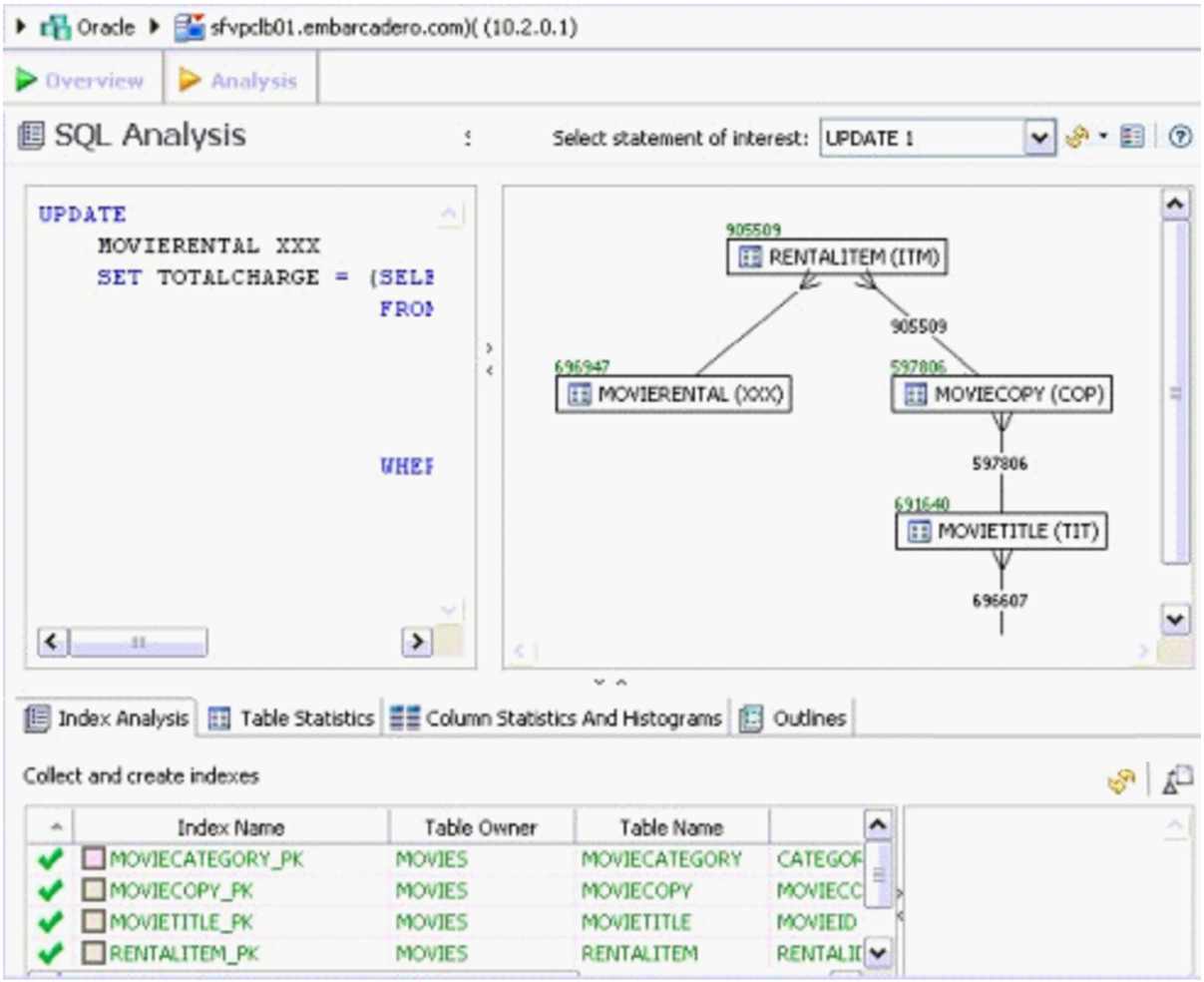
Determine the Best Cases for Statement Execution Path Time

Once the tuning job has executed, view the **Generated Cases** area of the Overview tab and determine the best possible case in terms of the **Execution Statistics** column values. This will indicate the most optimized query path for a given statement. Once you have determined the best case, you can execute that case on the specified data source and alter the database code to run the statement as that case on the native environment.

If you don't find an acceptably fast path, go to the Analysis tab. The Analysis tab can identify missing indexes and by examining the diagram you may be able to determine if there is something wrong with the SQL or schema.

Finding Missing Indexes and SQL Problems

The tuner performs the index and SQL analysis as part of the tuner run job performed on the Overview tab if **Perform Detail Analysis** is selected. Otherwise, the analysis is performed when you click the Analysis tab.



DB Optimizer can parse an SQL query and analyze the indexes and constraints on the tables in the query and display the query in graphical format on the Visual SQL Tuning (VST) diagram. The VST diagram can be displayed in either Summary Mode or Detail Mode. The VST diagram helps developers, designers, and DBAs see flaws in the schema design such as Cartesians joins, implied Cartesians joins and many-to-many relationships. The VST diagram also helps the user to more quickly understand the components of an SQL query, thus accelerating trouble-shooting and analysis.

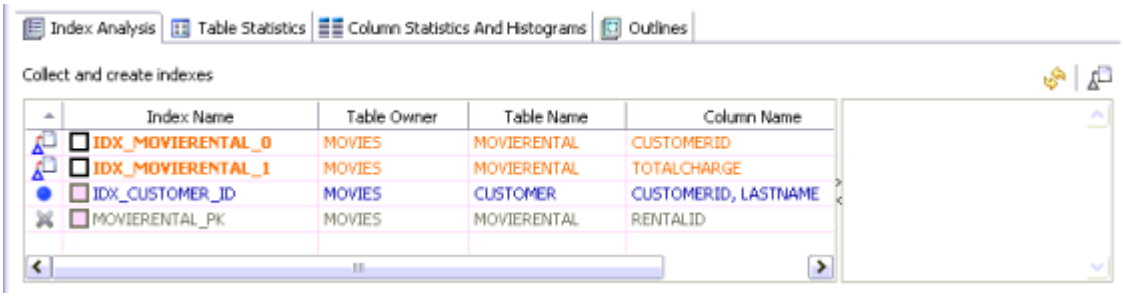
This section is comprised of the following topics:

- Finding Missing Indexes
- Exploring the VST Diagram

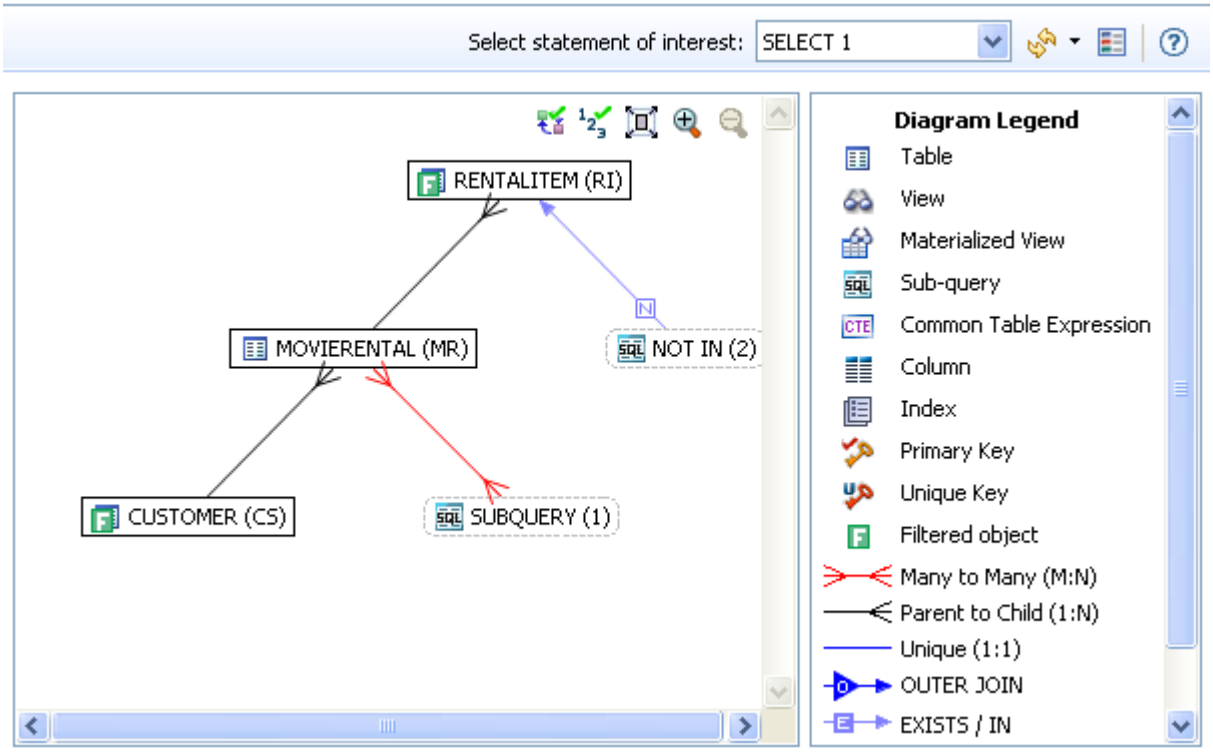
Finding Missing Indexes

Missing indexes are color-coded **orange** in the **Collect and create indexes** area of the Overview tab. Creating a missing index can improve the execution path of the SQL statement being analyzed.

Indexes that are used are color-coded **green**. Indexes that exist in the table but are not used in this execution path are color-coded **grey**. Indexes that are usable but not used by the current execution path are color-coded **blue**.



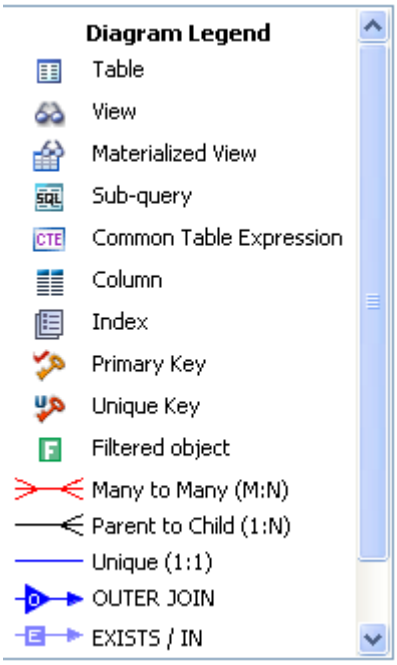
Exploring the VST Diagram



- Displaying or Hiding the Diagram Legend
- Displaying or Hiding the Explain Plan
- Displaying or Hiding Table Counts and Ratios
- Displaying or Hiding Table Columns and Indexes
- Viewing All Fields in a Table
- Viewing Object SQL
- Viewing Relationships
- Expanding Views in the VST Diagram

Displaying or Hiding the Diagram Legend

Click the **Diagram Legend** toggle [ ] to view the legend, and then click it again to hide it. All of the icons used in the VST diagram are identified in the **Diagram Legend**.

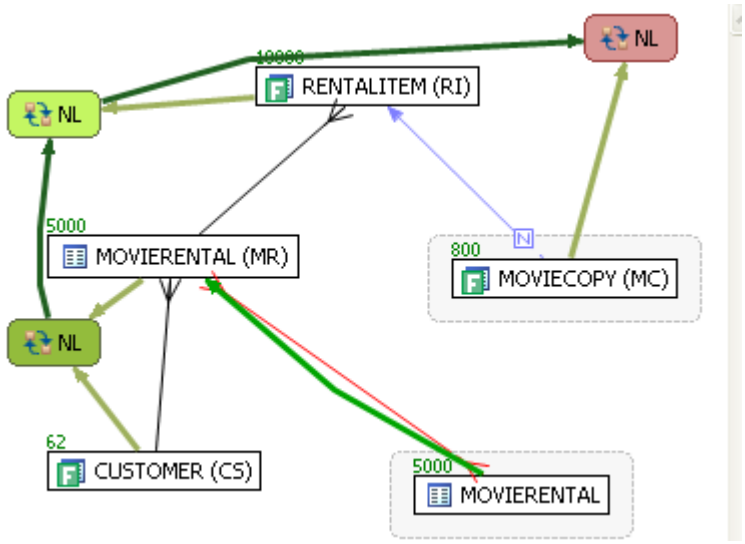


Displaying or Hiding the Explain Plan

> ⚠ The Explain Plan is available only for the Oracle 10g and 11g platforms.

Hover the mouse over the VST diagram to see the **Explain Plan** icon [ ], and then click it to view the **Explain Plan Overlay**. Click the **Explain Plan** icon again to hide the overlay.

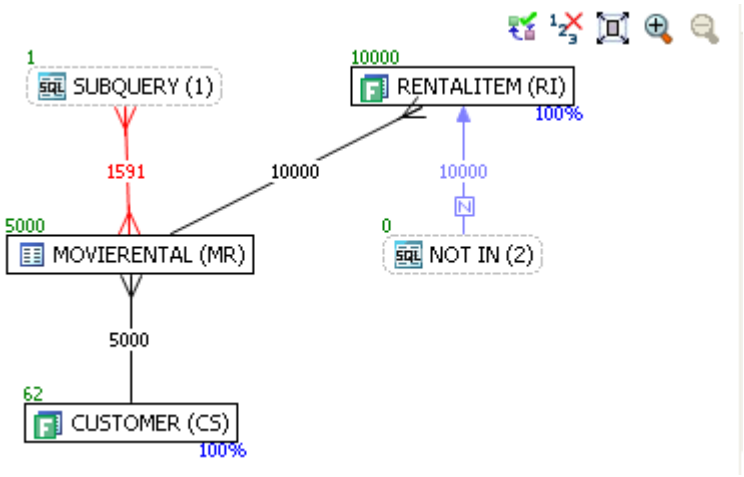
The additional nodes shown in the Explain Plan overlay provide details on the flow of the query plan, with operations (such as nested loops, sorts, and joins) showing connecting tables and other operations.



Hover the mouse over the objects or relationships in the overlay to view additional details.

Displaying or Hiding Table Counts and Ratios

To view or hide table counts, two table join sizes and filtered result set ratios, click the **Ratios and Counts** icon [ ].

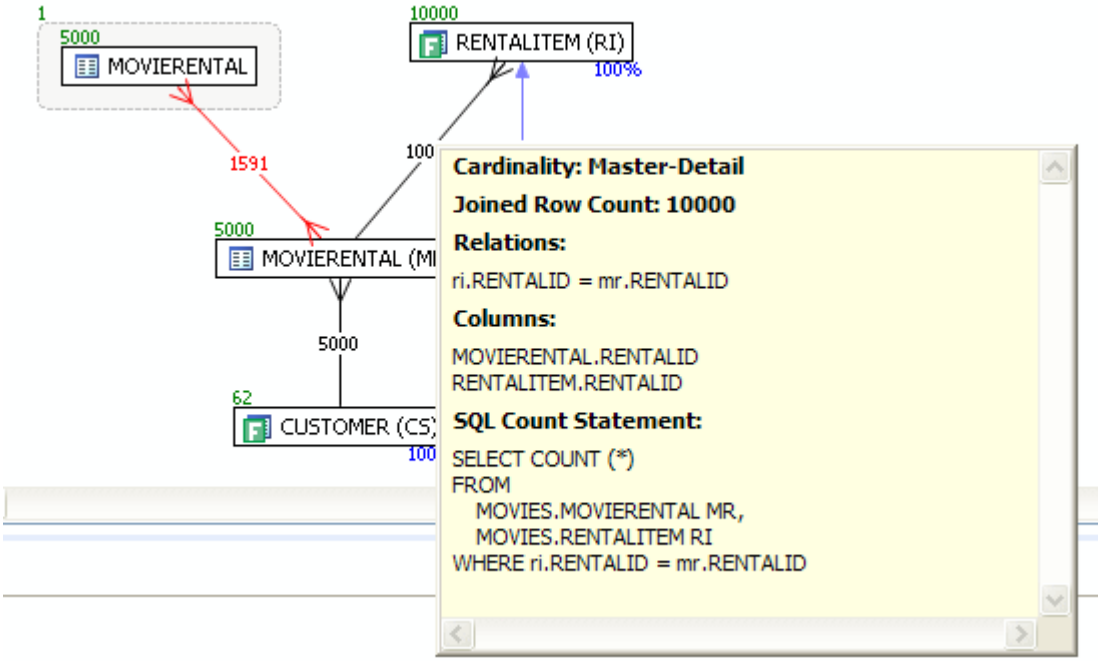


Green numbers at the top left of table represent the total number of rows in that table. In the above example, the MOVIERENTAL (MR) table has 5000 rows.

Blue percentage at the bottom right of the table represent the percentage of rows in that table that meet the selection criteria. In the above example, 100 percent of the rows in the RENTALITEM (RI) table have met the selection criteria.

The numbers on the table joins indicate the total number of rows that meet the selection criteria for both tables.

You can also view the SQL Query that created a relationship by hovering over the relationship. If the tooltip content is larger than the size of the tooltip rectangle, you can hover the mouse on top of the tooltip for a second, and then it will turn into a dialog you can re-size, scroll in, and select text from to copy into the Clipboard.



Displaying or Hiding Table Columns and Indexes

Click the **Detail/Summary** switch to display or hide details of the table display, including table columns and indexes.

Only fields that are used in the WHERE clause are displayed in detail mode. For information on interpreting the icons used in Detail mode, activate the Diagram Legend.

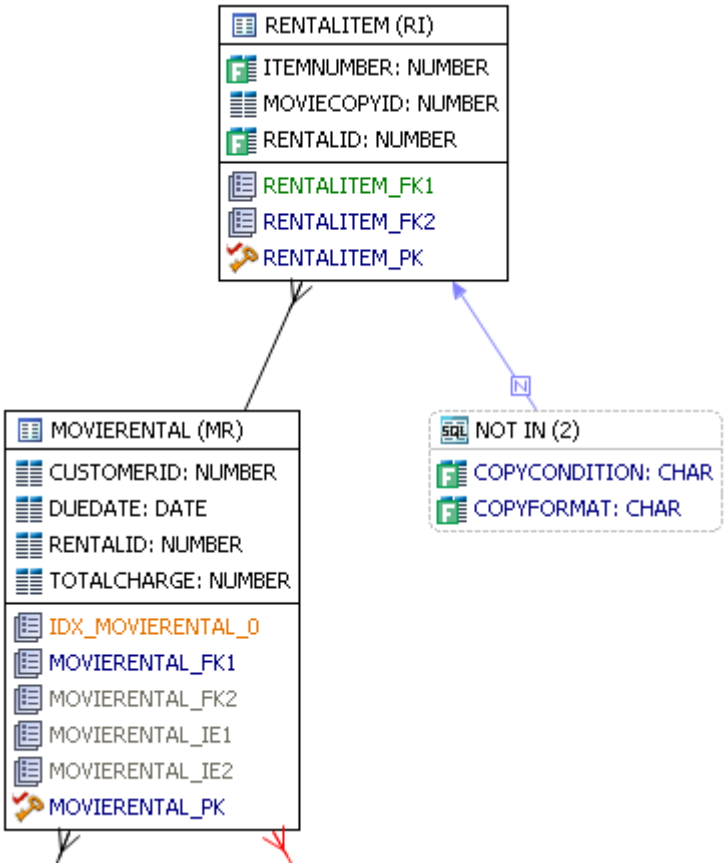
You can also switch between Summary Mode and Detail Mode for a specific table or view by double-clicking the object name.

You can move tables in the diagram by clicking and dragging them to the desired location. The position of the connecting lines is automatically adjusted.

Viewing All Fields in a Table

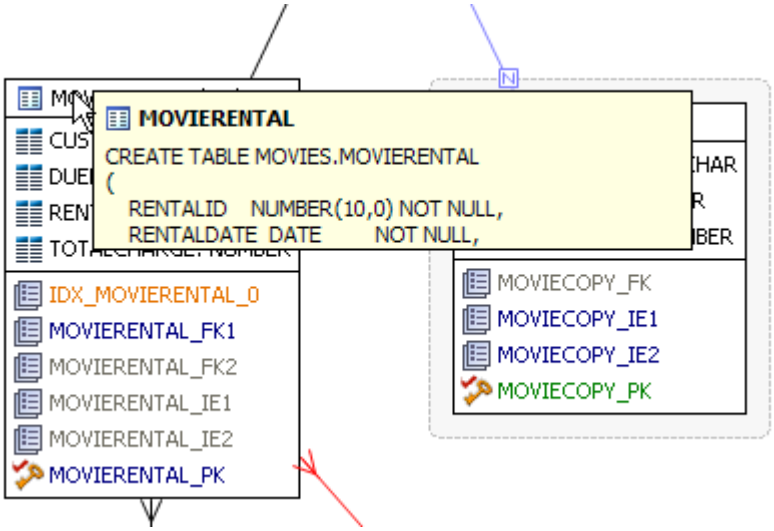
Hover the mouse over the table name to view all fields in the table in a pop-up window.

The following illustration shows the pop-up window that appears when hovering over the table.



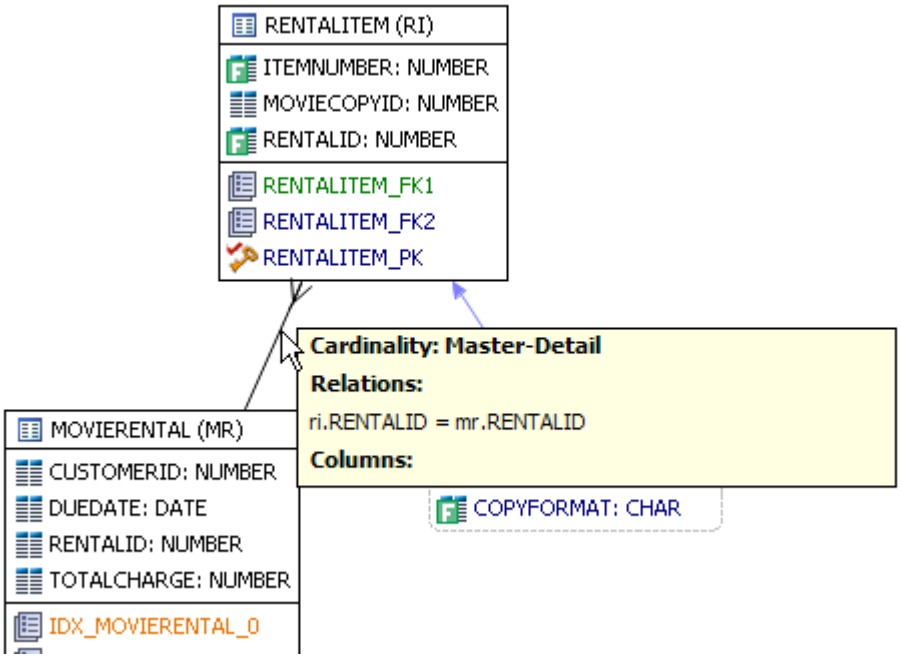
Viewing Object SQL

Hovering the mouse over the table name, field, or index displays the SQL required to create that object.



Viewing Relationships

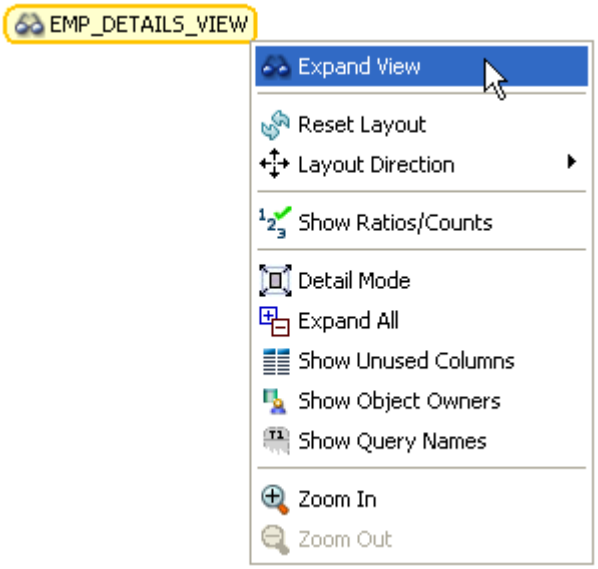
Hovering over the join between two tables displays the relationship between the two tables.



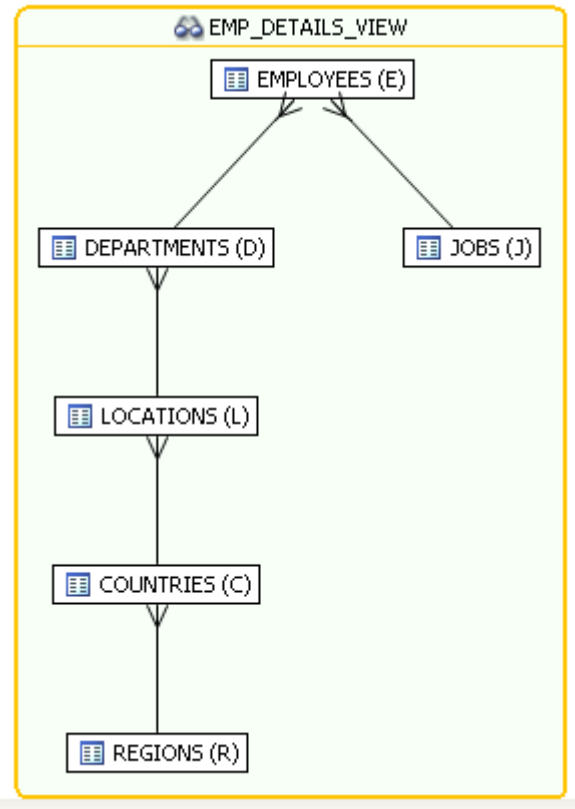
Expanding Views in the VST Diagram

For example, the following is the default layout from query join table CLIENT (c) to view TRANSACTIONS (t).

Right-click on the view, and then choose **Expand View**.

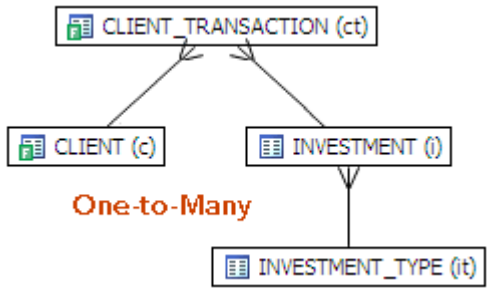


Now you can see the objects in the view.

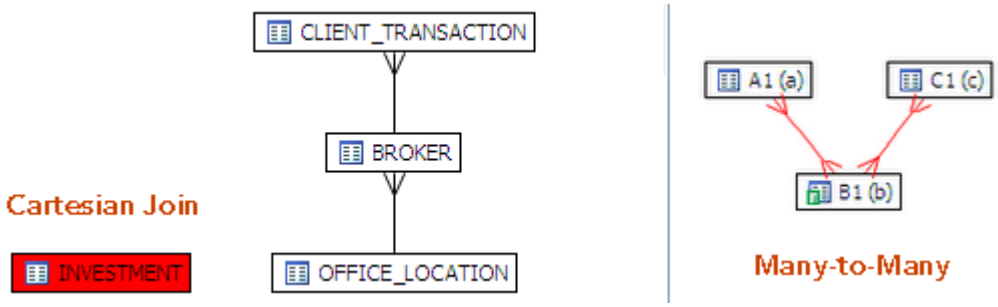

EMP_DETAILS_VIEW
EMPLOYEES (E)
DEPARTMENTS (D)
JOBS (J)
LOCATIONS (L)
COUNTRIES (C)
REGIONS (R)

Finding Problematic SQL or Schema

In the Visual SQL Tuning (VST) diagram, a well-formed query would resemble the following:



Problems such as Cartesian joins, implied Cartesian joins, and many-to-many relationships are clearly represented in the VST.



Joins are represented with connecting lines between nodes. The following describes when a particular type of connecting line is used and the default positioning of the line.

Connecting Lines	When Used
	One-to-One Join relationships are graphed horizontally using **blue** lines. One-to-one joins exist when two tables are joined on their primary key.
	One-to-Many Join relationships are graphed with the many table above the one table.
	Cartesian Join shows the table highlighted in red with no connectors to indicate that it is joined in via a Cartesian join. The query is missing join criteria. A Cartesian join problem could be resolved by implementing a rewrite suggestion shown in the Generated Cases area of the Overview tab.

Connecting Lines	When Used
	Many-to-Many Join relationships are connected by a **red** line and the relative location is not restricted. If master detail information is missing then the VST diagram will have many-to- many connectors. The optimizer can more consistently optimize a well-formed query, so the query will run faster.

Implementing Recommendations on the Index Analysis Tab

Once you have added tuning candidates to a tuning job, DB Optimizer can analyze the effectiveness of the indexes in the database and recommend the creation of new indexes where the new indexes can increase performance.

In the **Collect and create indexes** table, an index that DB Optimizer recommends that you create is marked in orange and has a create index icon next to its checkbox.

Index Name	Table Owner	Table Name	Column Name	Index
IDX_CLIENT_TRANSACTION_0	SYSTEM	CLIENT_TRANSACTION	TRANSACTION_STATUS	Normal
CLIENT_MULTI	SYSTEM	CLIENT	CLIENT_FIRST...NT_LAST_NAME	Normal
CLIENT_BROKER	SYSTEM	CLIENT	BROKER_ID	Normal
CLIENT_INCOME	SYSTEM	CLIENT	CLIENT_HOUSEHOLD_INCOME	Normal
CLIENT_PK	SYSTEM	CLIENT	CLIENT_ID	Unique
CLIENT_TRANSACTION_BROKER	SYSTEM	CLIENT_TRANSACTION	BROKER_ID	Normal

Accept the Suggestion and Automatically Generate an Index

To accept the sugestion and automatically generate an index

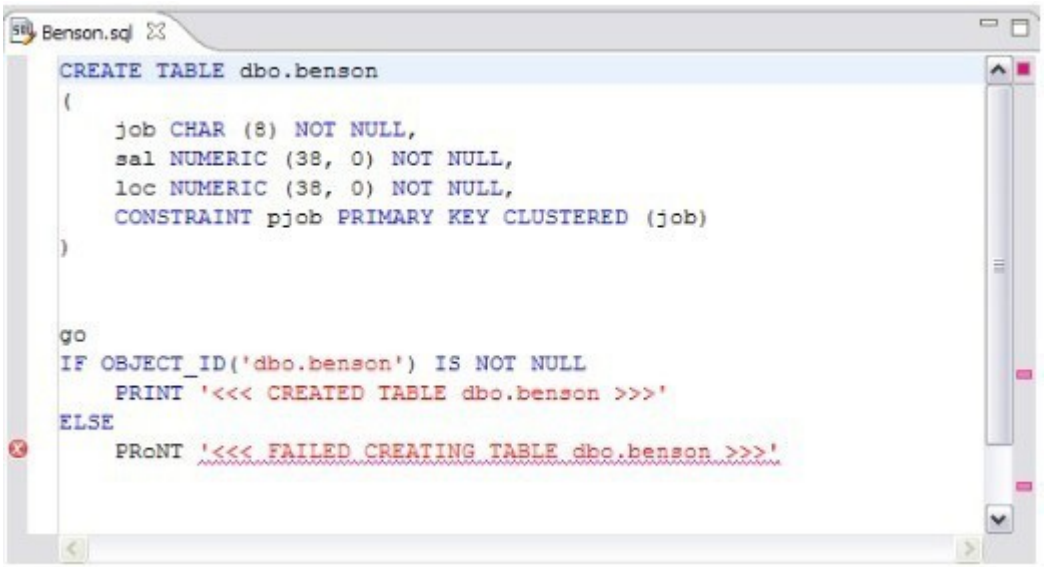
1. For any recommended index, click the check box to the left of the index you want to create.
   For a selected index, you can modify the Index type by clicking in the **Index Type** column, and then selecting a type from the list: **Normal**, **Bitmap**, **Reverse Key**, **Reverse Key Unique**, or **Unique**.

2. Click the **Create Index** button. The Index Analysis dialog appears.
3. To view the index SQL in an editor for later implementation, click the statement, and then click **Open** in a SQL editor.
4. To run the index SQL and create the index on the selected database, click **Execute**.

SQL Code Assist and Execution

SQL Code Assist is an interface component that enables the development and formatting of SQL code for the purposes of creating and modifying database objects. It provides a front-end application for the delivery of code through its object code extraction capabilities.

Code Assist provides a number of key features that assist in the coding process, ensuring greater accuracy in coding, faster development cycles, and a general increase in efficiency overall.

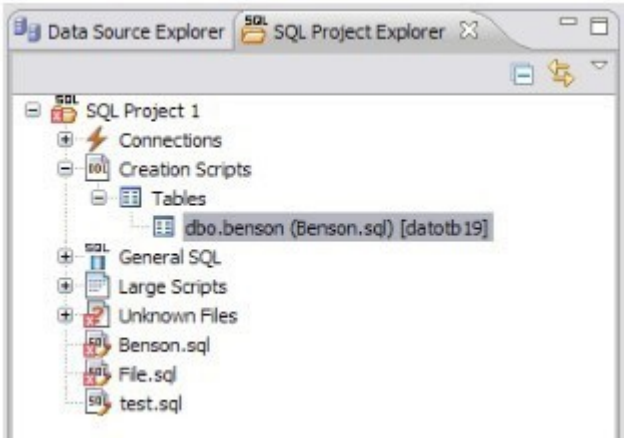


The following key features are provided with SQL Code Assist:

- Code Extraction
- Code Highlighting
- Automatic Error Detection
- Code Complete
- Hyperlinks
- Code Formatting
- Code Folding
- Code Quality Checks
- SQL Execution
- Configuring SQL Execution Parameters

In order to access SQL Editor, you need to create a new file or edit existing code from Data Source Explorer.

Additionally, SQL Project Explorer provides a tree structure for all files created in DB Optimizer. You can access files from here by double-clicking the file name you want to open as well.



To create a new file

- Choose **File > New > SQL File**. A blank instance of SQL Editor appears in the Workbench. If you save this file, it is automatically added to the SQL Project Explorer.

To edit an existing file

- Use Data Source Explorer or Project Explorer to navigate to the code you want to modify, and then double-click it. An instance of SQL Editor appears in the Workbench, populated with the extracted code of the specified object or SQL project file.

Code Extraction

SQL Editor provides the ability to extract the underlying SQL code of database objects registered in DB Optimizer to provide a front-end application for the development and modification of data sources in your enterprise.

To extract underlying SQL code

- Navigate to a database object in Data Source Explorer, and then select **Extract** via the right-click menu. The object's underlying SQL code appears in SQL Editor and is ready for editing and further modification.

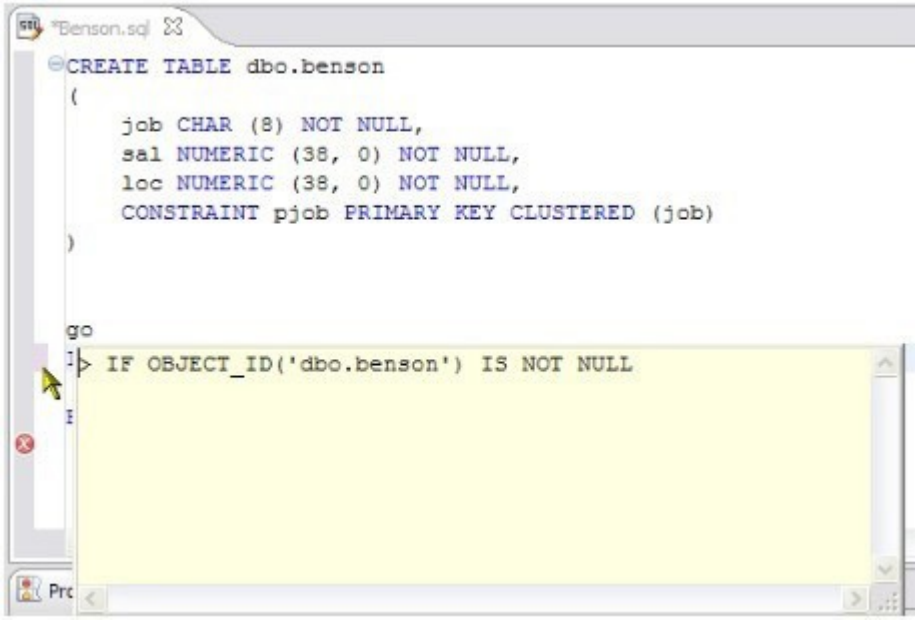
Code Highlighting

SQL Editor identifies commands and provides syntax highlighting changes that are automatically added to the code as you add lines, which enables you to clearly and quickly understand code when you read it in the interface.

The following syntax highlighting is automatically added to lines of code in SQL Editor:

Code	Formatting
Comments	*green italics*
SQL Comments	dark blue
Syntax Errors	red underline
Coding Errors	red

Additionally, SQL Editor provides a purple change bar in the left-hand column that indicates if a line of code has been modified from the original text. You can hover over this change bar to view the original code line. A red square icon in the right-hand column indicates that there are errors in the code line. You can hover over the icon to view the error count.

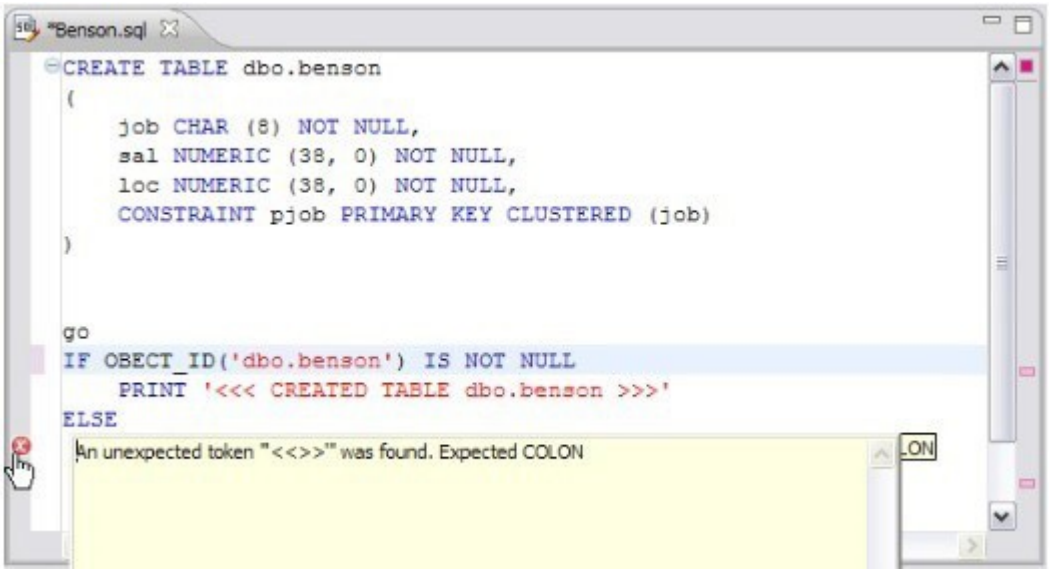


The purple change bar indicates if a line of code has been changed from its original text. Hover your mouse over the change bar to view the original text.

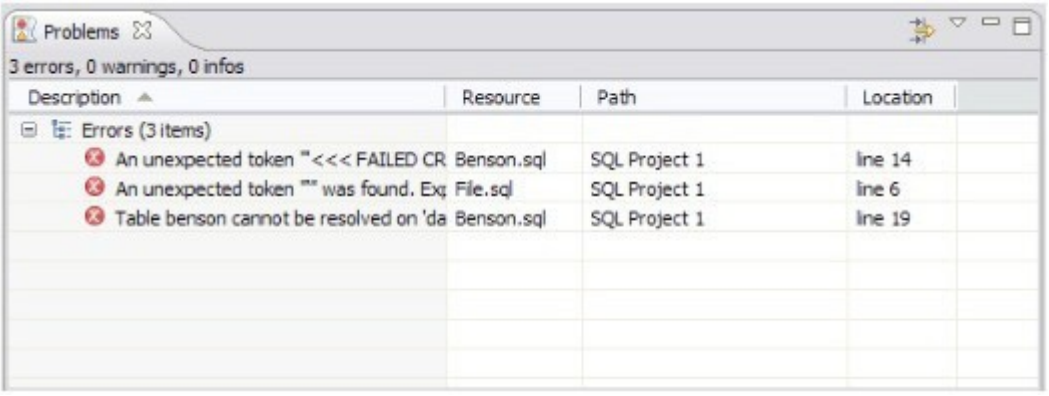
Automatic Error Detection

The automatic error detection functionality of the editor highlights errors and typos in the code as you work in "real time."

Automatic error detection automatically identifies and analyzes SELECT, FROM, WHERE, GROUP BY, HAVING, and ORDER BY statements. If it detects any syntax errors while you type these statements, the line is automatically flagged by the error icon in the left-hand column of SQL Editor. You can hover your mouse over the icon to view any errors.



Additionally, all semantic errors are recorded in the **Problems** view, an interface component that automatically logs errors and warnings as you work with files.



The Problems view logs errors and warnings as you work with files in SQL Editor.

You can double-click on a line in the Problems view and DB Optimizer will automatically navigate you to that issue in SQL Editor.

Code Complete

SQL Editor provides suggestion capabilities for both DML statements and objects. It has the ability to look up object names in order to avoid errors when defining table names, columns, etc. in the development process. This feature provides lists of object and code suggestions that will make the development process more efficient as you will not be forced to manually look up object names and other statement values. SQL Editor provides code assist for SELECT, UPDATE, INSERT, and DELETE statements (as well as for MERGE statements on SQL Server 2008 and higher) and object suggestion support for tables, alias tables, columns, alias columns, schemas, and catalogs.

To activate code assist

1. Click the line on which you want to activate the code assist feature.
2. Press **CTRL + Spacebar** on your keyboard. Code assist analyzes the line and presents a list of suggestions as appropriate based on the elements of the statement.

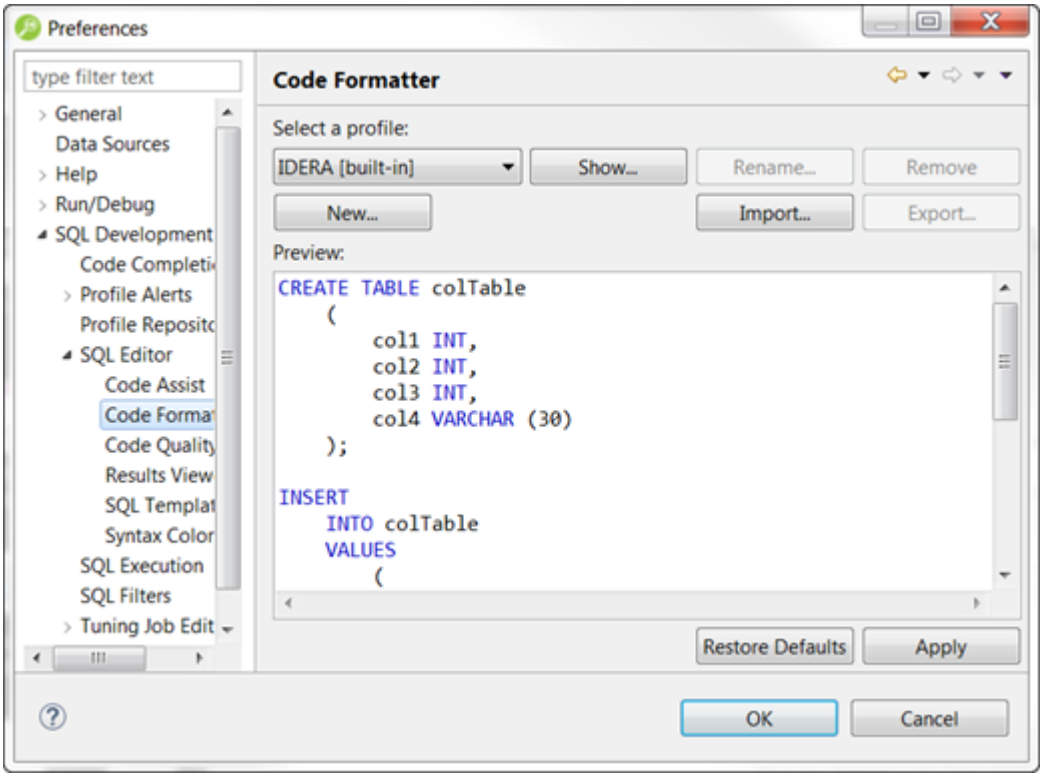
Hyperlinks

Hyperlinks are used in SQL Editor to provide links to tables, columns, packages and other reference objects. When you select a hyperlinked object from a piece of code, a new editor opens and displays the source. Additionally, hyperlinks can be used to link procedures or the function of a call statement, as well as function calls in DML statements. To enable a hyperlink, hover your mouse over the object name and hold the **CTRL** key. It becomes underlined and changes color.

Code Formatting

Code formatting is automatically applied to a file as you develop the file in SQL code. This enables you to set global formatting preferences one time, and then apply them to all code development, saving time and allowing for a more efficient code development process. The code formatter can be accessed by selecting **CTRL + Shift + F** in the editor.

All code is automatically formatted based on parameters specified on the **Code Formatter** node of the **Preferences** panel. (Select **Window > Preferences** in the main menu to access this panel.)



In addition to formatting code per individual file, you can also format an entire group of files from **Project Explorer**. Select the directory of files to which you want to apply formatting, and then execute the **Format** command from the right-click menu. The files are automatically formatted based on the global preferences.

Code Folding

The code folding feature automatically sorts code into a tree-like outline structure in SQL Editor. This increases navigation and clarification capacities during the code development process. It ensures that the file will be easily understood should any future work be required on the code.

As you work in SQL Editor, collapsible nodes are automatically inserted into the appropriate lines of code. Statements can then be expanded or collapsed as needed, and this feature is especially useful when working on parts of particularly large or complicated files.

Code Quality Checks

On Oracle-based code, the code quality checking feature provides suggestions regarding improved code on a statement-by-statement basis. As you work in SQL Editor, markers provide annotations that prevent and fix common mistakes in the code.

Notes regarding code quality suggestions appear in a window on any line of code where the editor detects an error, or otherwise detects that the code may not be as efficient as it might be. Code quality check annotations are activated by clicking the light bulb icon in the margin, or by selecting **Ctrl + l** on your keyboard.

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

- Statement is missing valid JOIN criteria
- Invalid or missing outer join operator
- Transitivity issues
- Nested query in WHERE clause
- Wrong place for conditions in a HAVING clause
- Index suppressed by a function or an arithmetic operator
- Mismatched or incompatible column types
- Null column comparison

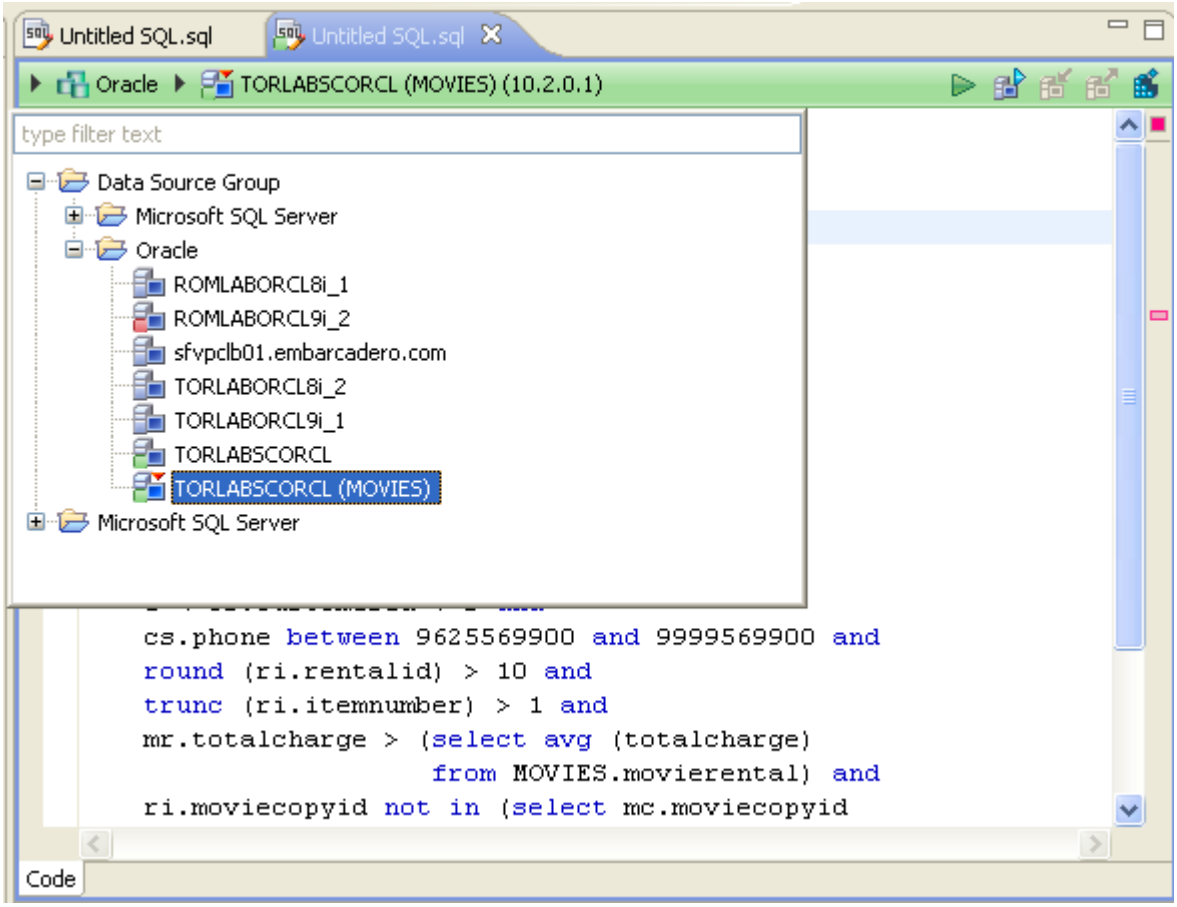
To activate code quality checks

- Click the light bulb icon in the margin of the editor or select **Ctrl + l** on your keyboard. The editor suggestions appear in a window beneath the selected statement. When you click a suggested amendment, the affected code is automatically updated.

SQL Execution

When you have finished developing or modifying code, you can then execute the file from within the DB Optimizer environment, on the database of your choosing. This enables you to immediately execute code upon completion of its development. Alternatively, you can save files for execution at a later point in time.

In order to execute a file, you must first associate it with a target database. This is performed by using the drop-down menus located in the Toolbar. When a SQL file is open in the Workbench, the menus are enabled. Select a data source and a corresponding database to associate the file with and then click the green arrow icon to execute the file.



The pair of drop-down menus indicates that the SQL file is associated with the **dataotb19** data source and EMBCM database. When the green arrow icon on the right-hand side of the menus is selected, the file is executed on the specified data source and database.

Additionally, if you have turned off auto-committal in the Preferences panel (**Window > Preferences**), you can commit and execute transactions via the **Commit Transaction** and **Start Transaction** icons located next to the **Execute** icon.

Execute a File

Open the file you want to run and ensure it is associated with the correct database. Click the **Execute** icon [ ]. DB Optimizer executes the code on the database you specified.

Execute a Transaction

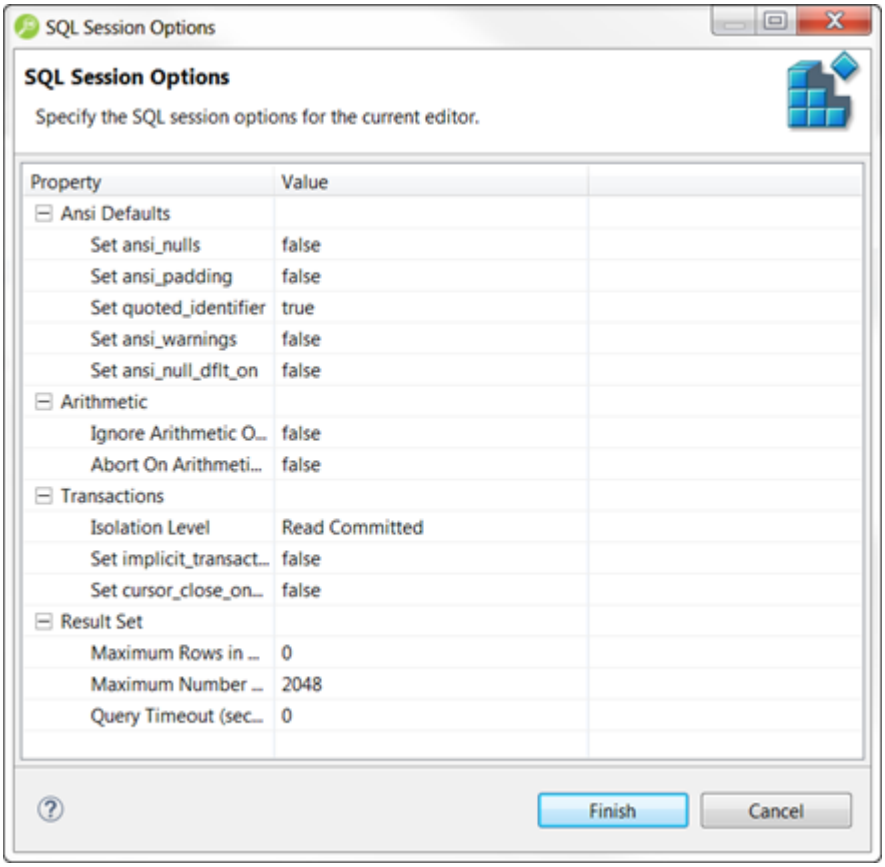
Open the transaction file you want to run and ensure it is associated with the correct database. Click the **Start Transaction** icon [ ]. DB Optimizer executes the transaction on the database you specified.

Commit a Transaction

Open the transaction file you want to commit and ensure it is associated with the correct database. Click the **Commit Transaction** icon [ ]. DB Optimizer commits the transaction on the database you specified.

Configuring SQL Execution Parameters

Use the SQL Session Options dialog to modify the configuration parameters that determine how DB Optimizer executes code. These options ensure that code is executed the way you want on an execution basis, ensuring accuracy and flexibility when running new or modified code.



SQL Sessions Options provide you with the flexibility to adjust execution parameters on a session-by-session basis.

Modify SQL Session Options

1. Click the **SQL Session Options** icon on the toolbar. The SQL Sessions Options dialog appears.
2. Click on the individual parameters in the **Value** column to change the configuration of each property, as specified.
3. Click **Finish**. The session options are changed and DB Optimizer executes the code as specified by your options.

> ⚠ SQL Session Options are only applied to the currently-selected code, and are not retained across different files with regards to execution.