



## **Embarcadero® DB Optimizer™ 1.5 Evaluation Guide**

Published: March 16, 2009

# Contents

<b>INTRODUCTION TO DB OPTIMIZER .....</b>	<b>4</b>
Product Benefits .....	4
<b>ABOUT THIS EVALUATION GUIDE.....</b>	<b>6</b>
<b>SESSION 1: GETTING STARTED WITH DB OPTIMIZER .....</b>	<b>7</b>
Install DB Optimizer.....	7
User Interface Overview .....	8
<b>SESSION 2: WORKING WITH DATA SOURCE EXPLORER.....</b>	<b>10</b>
Adding Data Sources .....	11
Browsing Data Sources.....	12
<b>SESSION 3: PROFILING A DATA SOURCE.....</b>	<b>13</b>
Starting a Profiling Session .....	14
Analyzing Session Data .....	15
Load Chart.....	15
Top Activity.....	15
Profiling Detail.....	16
Saving a Profiling Session.....	18
Importing Statements to SQL Tuner.....	18
<b>SESSION 4: TUNING SQL STATEMENTS.....</b>	<b>19</b>
Creating a New Tuning Job.....	20
Adding SQL Statements.....	21
Running a Tuning Job .....	22
Analyzing Tuner Results .....	23
Applying Tuner Results on the Data Source .....	24
<b>SESSION 5: SQL CODE ASSIST AND EXECUTION .....</b>	<b>25</b>
Code Extraction.....	27
Code Highlighting .....	27
Automatic Error Detection .....	29
Code Complete .....	30
Hyperlinks .....	30
Code Formatting .....	30
Code Folding .....	31
Code Quality Checks.....	32

SQL Execution .....	33
Configuring SQL Execution Parameters.....	34
<b>ADDITIONAL EVALUATION RESOURCES .....</b>	<b>35</b>
Embarcadero Technologies Product Support.....	35
Licensing Your Embarcadero Technologies Product .....	35
Embarcadero Technologies Technical Support.....	35
Embarcadero Technologies on the Web .....	35

## Introduction to DB Optimizer

DB Optimizer is an enterprise SQL development and optimization tool that provides an environment for building code, analyzing database performance, and optimizing query paths on specific data sources. This enables users to ensure the efficiency of the overall enterprise, and provides a single application from which to develop, diagnose, and optimize. This product provides support for IBM DB2 for LUW, Microsoft SQL Server, Sybase, Oracle, and generic JDBC data source platforms. It is available as a stand-alone application, or as an Eclipse plug-in.

The application is structured and composed into three main interface parts. This design provides you with a comprehensive workflow that enables development, query analysis, and tuning capabilities. This workflow, in turn, leads to more efficient task management in terms of time and efficiency, overall.

The three major interface components of DB Optimizer are as follows:

### **SQL Profiler**

SQL Profiler provides continuous data source monitoring that builds a statistical model, or profile, of the specified data source, and highlights top SQL, event, and session activity. This component is used to locate and diagnose problematic SQL code and event-based bottlenecks via its graphical interface, which is used to identify problem areas and drill down to individual, problematic statements. Additionally, Profiler enables the investigation of execution and wait time event details for individual stored routines.

### **SQL Tuner**

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 in data retrieval through hint injections. Users are supplied with a list of possible cases generated by Tuner, and can select and update a statement with the most efficient path to reduce load and improve efficiency, overall.

### **SQL Editor**

SQL Editor simplifies SQL development by utilizing features that improve productivity and reduce errors. It provides a rich interface that offers code completion, real-time error caching, code formatting, and sophisticated object validation tools. In context with SQL Profiler and SQL Tuner, it provides an interface for viewing and editing SQL files and database packages, as accessed through Profiler and Tuner functionality.

## **Product Benefits**

- Interface provides a comprehensive and task-based workflow to enable a simple process, allowing for statement analysis and subsequent database tuning in a short number of convenient and intuitive steps.
- Real-time and continuous database monitoring enables the application to run in the background, and presents you with a live view of statement execution processes and problem areas.
- Drill-down functionality enables granular views of statements, events, and sessions being monitored by the application, and access to the underlying code for modification, duplication, or examination purposes.
- Migration and enterprise management functionality organizes all data sources within your enterprise, as well as objects and other application information under one roof.

- Code execution occurs directly on data sources registered in the application, but is always initiated on the interface. This provides a single point of entry to all of your data sources, and enables a faster and more efficient way to manage your enterprise.

## About This Evaluation Guide

This evaluation guide enables you to get started with DB Optimizer.

Its purpose is to provide you with the foundation needed to fully utilize the features and benefits of the products, and apply them to your own enterprise. This guide contains information on how to get the application up and running, 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 well as identify and prevent system bottlenecks and other wait-related issues.

This guide is divided into five sections:

**Session 1: Getting Started with DB Optimizer**

**Session 2: Working with Data Source Explorer**

**Session 3: Profiling a Data Source**

**Session 4: Tuning SQL Statements**

**Session 5: SQL Code Assist and Execution**

Once you have started the application, you can select **Help** from the Menu Bar to find additional resources that compliment and build upon the features and tasks presented in this guide.

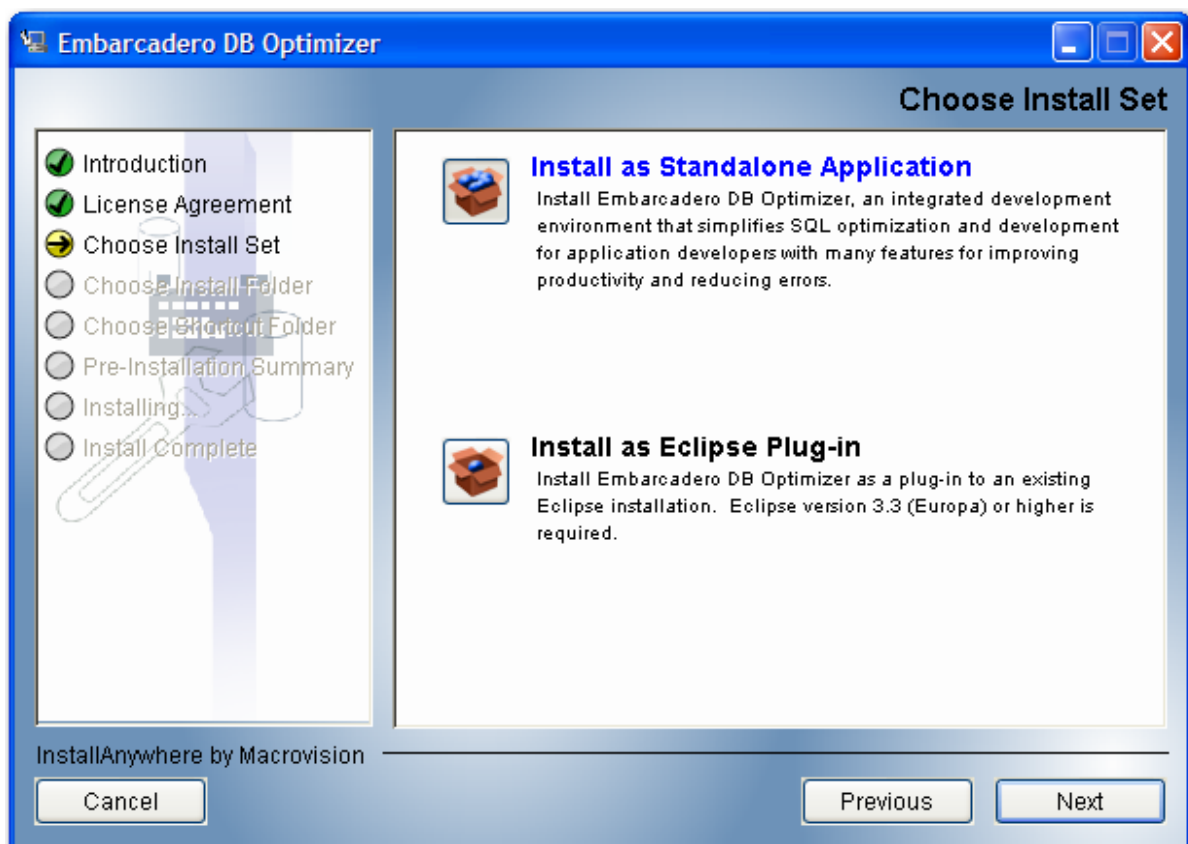
## Session 1: Getting Started with DB Optimizer

### Install DB Optimizer

Launch the installation program via the installer executable included in your software package and follow the instructions in the wizard to complete it.

DB Optimizer can be installed as a standalone application (RCP installation) or as an Eclipse plug-in (Plug-In installation).

- The RCP installation provides an enclosed application that runs DB Optimizer in a contained framework.
- The Eclipse plug-in installation leverages plug-in capabilities to run within the Eclipse framework. You must have a previously installed version of Eclipse (version 3.3 or higher) on your machine prior to installing DB Optimizer as a plug-in.



*DB Optimizer can be installed as a standalone application or as a plug-in for an existing Eclipse installation.*

Both the RCP and plug-in versions of DB Optimizer provide full product functionality.

You may be required to restart your machine to complete the installation of DB Optimizer. It is also recommended that you copy the installer to the desktop of your machine prior to running the installer.

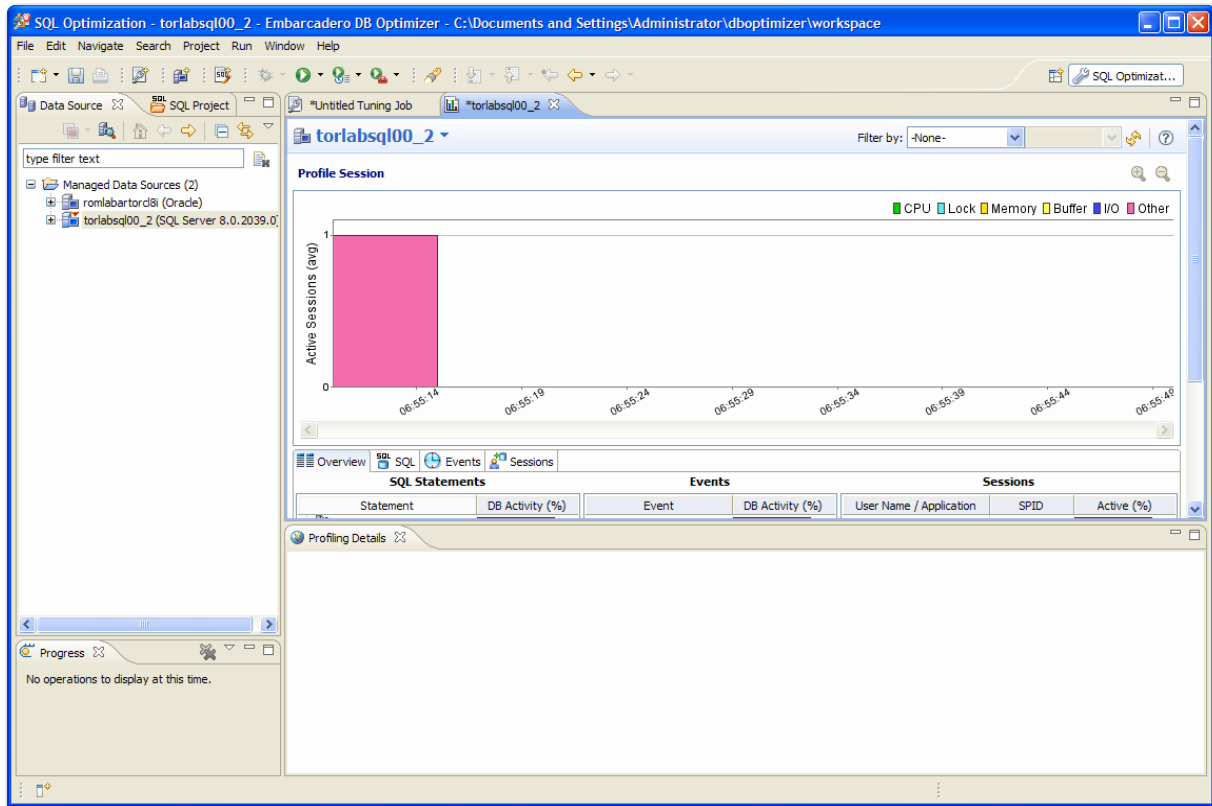
### To start DB Optimizer:

- From the Desktop, choose the Windows Start menu and then select **Programs > Embarcadero > Embarcadero DB Optimizer 1.5 > Embarcadero DB Optimizer 1.5**.

## User Interface Overview

The DB Optimizer application environment is known as the **Workbench**. The Workbench provides an interface through which you manage data sources and analyze and tune statements.

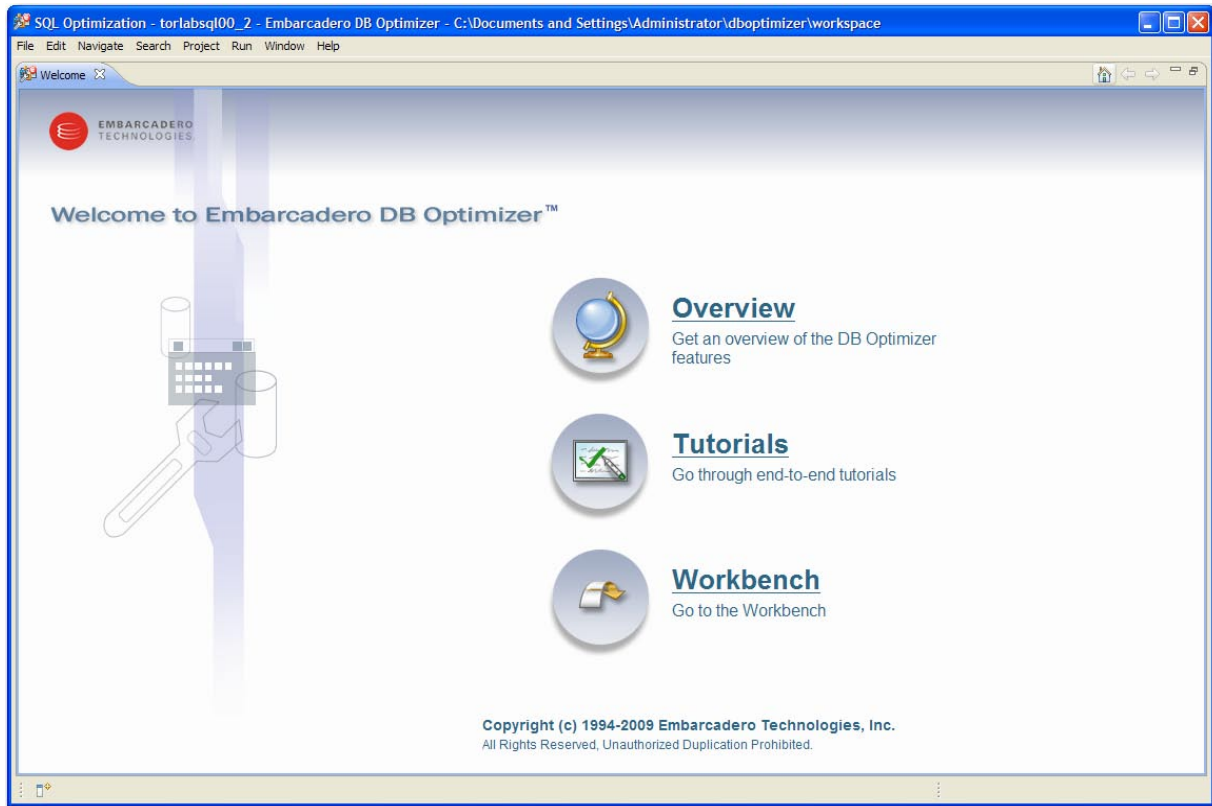
The Workbench is composed of common interface elements that provide tools to accomplish these tasks. These objects provide a uniform system for working with tuning jobs, profile sessions, and data sources.



*The Workbench provides an environment to build profiling sessions and tune query statements.*



- The **Welcome Page** is the first screen you will see when you initially launch the application. This screen provides links to documentation and other help files/tutorials to further familiarize you with the product. It provides easy access to information that may be of value to new users.

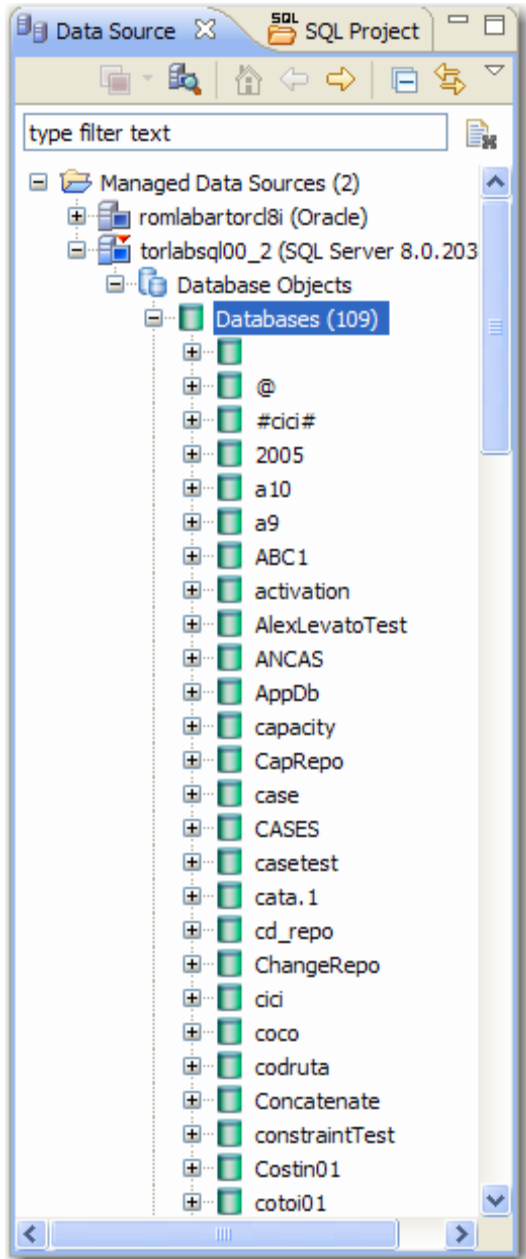


*The Welcome page provides links to help tutorials and provides access to the application environment.*

- The **Workbench Space** is composed of **Views**, **Editors**, and the **Menu Bar** and **Command Toolbar**. Views and Editors are interface components that enable you to perform DB Optimizer functions and work tasks, as well as manage resources.
- 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 Toolbar contains icons that represent specific menu commands. For example, the **Search** command can be launched via the **Search > Search** menu in the Menu Bar, and also via the **Search** icon in the Command Toolbar.
- **Views** are Workbench interface components typically 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 Workbench interface components typically used to access DB Optimizer functionality. For example, the **SQL Editor** provides a means to view, edit, and save SQL code. Editors are largely differentiated from Views in that they operate on an individual level rather than a supportive one. The SQL Profiler and SQL Tuner components of DB Optimizer can be considered Editors for the purposes of understanding the Workbench interface.

## Session 2: Working with Data Source Explorer

Data Source Explorer provides a tree view of all data sources registered in DB Optimizer. In addition, 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.*

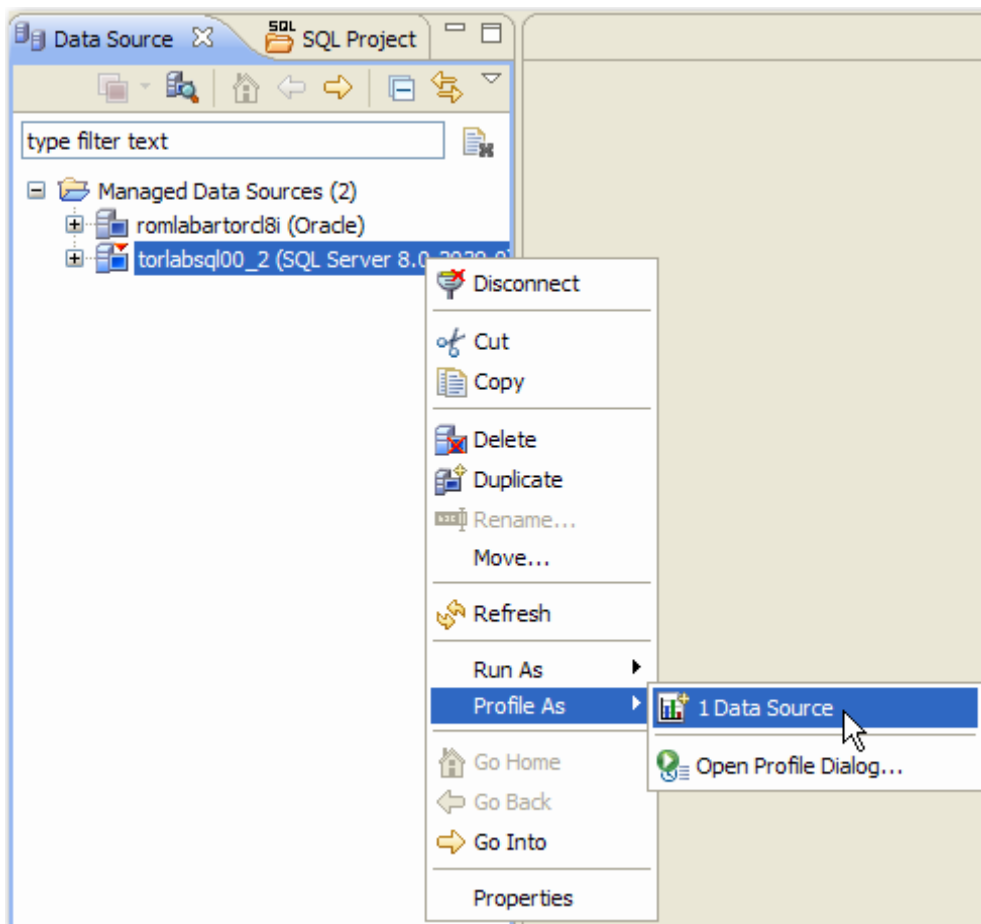
### To add a data source:

Select Data Source Explorer and choose **File > New > Data Source** from the Menu Bar. The Data Source Wizard appears. Follow the steps provided to register the data source. When you are finished, the data source is added to Data Source Explorer.

## Browsing Data Sources

In order to analyze and tune statements on data sources within your enterprise, you need to access the data source objects within the application environment. It is important to be able 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.*

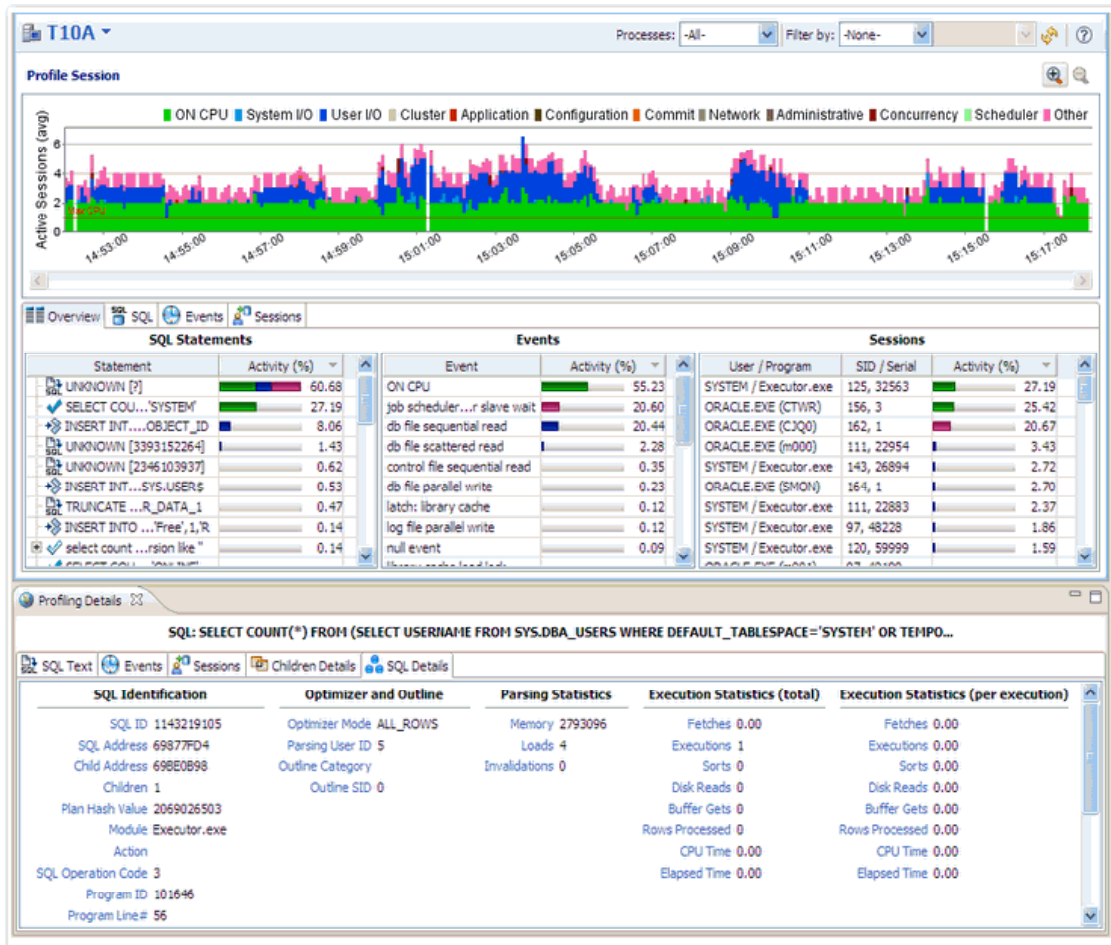
All objects listed in Data Source Explorer are organized, in descending order, by data source. Additionally, if you double-click on an object, SQL Editor will open and display the underlying SQL code.

### To browse a data source:

Expand the node of the data source that you want to examine. Double-click an object to view the code in SQL Editor.

## Session 3: Profiling a Data Source

SQL Profiler is an interface component that constantly 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. Additionally, Profiler enables you to investigate execution and wait time event details for individual stored routines. Results are presented in the profiling editor, which enables you to identify problem areas and view individual SQL statements, as needed.



SQL Profiler analyzes and provides a statistical model of database load that identifies problematic code and event-based bottlenecks.

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 load originates. Specifically, **Top Activity** displays 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 on 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.

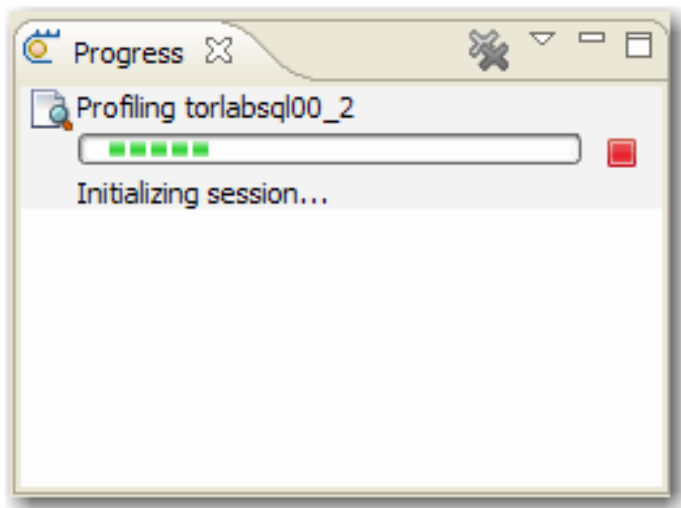
## Starting a Profiling Session

In order to access SQL Profiler, you need to run a profiling session on a data source registered in Data Source Explorer. You can do this by right-clicking on a data source and selecting **Profile As > Data Source** from the context menu. You can also click the **Profile** icon in the Toolbar, which initiates a profile session on the last selected data source in Data Source Explorer.

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 upper left-hand side of the Profiler.

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

The **Progress** view indicates that the profiling session has been initiated.



*The Progress view indicates that a profiling session has been initiated.*

## 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.

## Analyzing Session Data

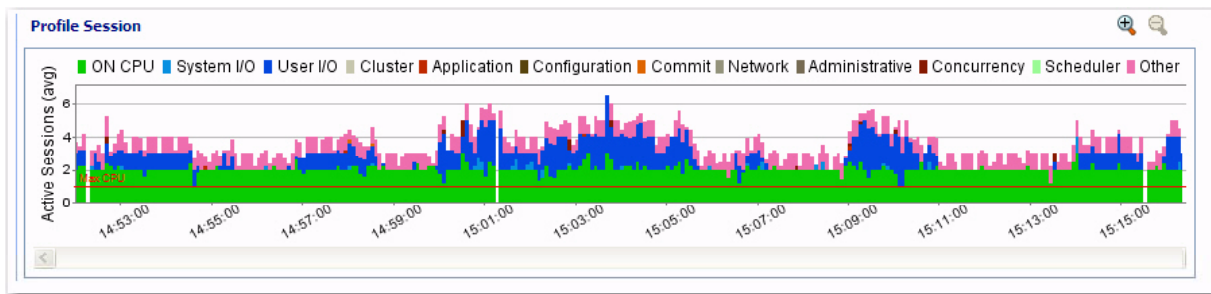
Profiler is composed of three essential diagnostic views that provide information about load on a particular database in the system. These views enable you to identify bottlenecks and view details about specific queries that execute and wait over the course of a profiling session.

SQL Profiler is composed of the following three components, listed in descending order of granularity:

1. **Load Chart**
2. **Top Activity**
3. **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.



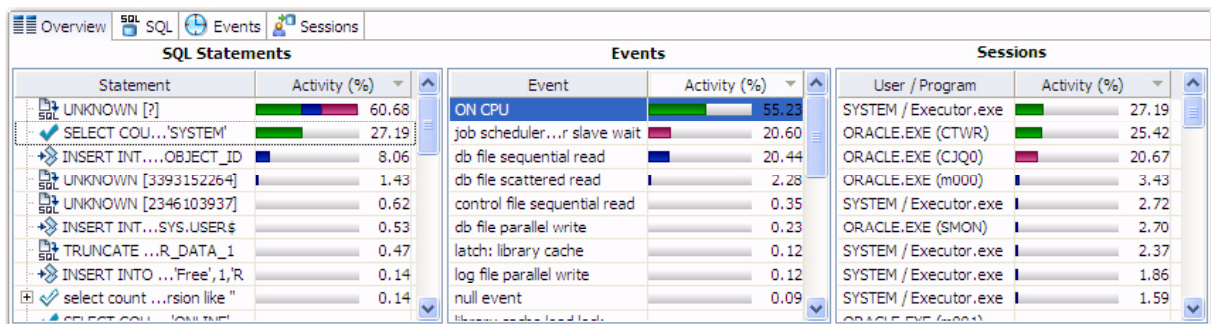
The Load Chart displays the overall load of the database that you analyzed with Profiler.

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.

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 the top SQL statements, top events, and the top activity sessions on the database. It is composed of a series of tabs that provide detailed statistics on individual SQL statements and sessions that are waiting or executing over the length of the profiling session.



The Top Activity section displays a more detailed view of the Load Graph. It identifies top statements, events, and sessions that are waiting or executing over the length of the profiling session



- 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.
- If you are profiling an Oracle data source, the **I/O** tab provides information about I/O. This tab will not be displayed if the platform being profiled is not specific to Oracle.

### Profiling Details

When you select any item from **Top Activity**, details are displayed on the **Profiling Details** view. The tabs that compose this display are dependent on the nature of the object selected in **Top Activity**, in order to reflect that item's specific information.

The screenshot shows the 'Profiling Details' window with the following tabs: SQL Text, SQL Details, Events, Sessions, and Children Details. The 'SQL Details' tab is active, displaying the following information:

SQL Identification	Optimizer and Outline	Execution Statistics (total)	per execution	per row
SQL ID 1135785965	Optimizer Mode ALL_ROWS	Fetches 0.00	0.00	0.00
SQL Address 6998CE60	Parsing User ID 5	Executions 1	1.00	1.00
Child Address 698AA520	Outline Category	Sorts 0	0.00	0.00
Children 1	Outline SID 0	Disk Reads 1004	1,004.00	1,004.00
Plan Hash Value 3592252508		Buffer Gets 13261	13,261.00	13,261.00
Module Executor.exe	<b>Parsing Statistics</b>	Rows Processed 1	1.00	1.00
Action	Memory 162433	CPU Time 93,750.00	93,750.00	93,750.00
SQL Operation Code 2	Loads 134	Elapsed Time 70,279,820.00	70,279,820.00	70,279,820.00
Program ID 101644	Invalidations 132			
Program Line# 195				

*Profiling Details displays detailed parameter information on a statement, event, or session that was selected in Top Activity.*

The parameters that appear in **Profiling Details** will be different depending on whether you selected a statement, session, or an event. This is to accommodate the parameter specifics of the item you selected.

- When you select a statement, the **SQL Text**, and **Events** tabs appear. Additionally, if you are profiling an Oracle data source, the **SQL Details** and **Sessions** tab appears as well.

Tab Name	Description
<b>SQL Text</b>	Displays the full code of the selected SQL statement.
<b>Events</b>	Provides parameter details about events the statement is associated with.
<b>SQL Details (Oracle only)</b>	Provides parameter details related to how the selected SQL statement is executing. For example, parsing statistics, and SQL identification details.
<b>Sessions (Oracle only)</b>	Provides parameter details on different parameters about any sessions the statement is associated with. For example, SID number and machine directory location.



- When you select an event, the **Session Details**, **SQL**, and **Sessions** tabs appear.

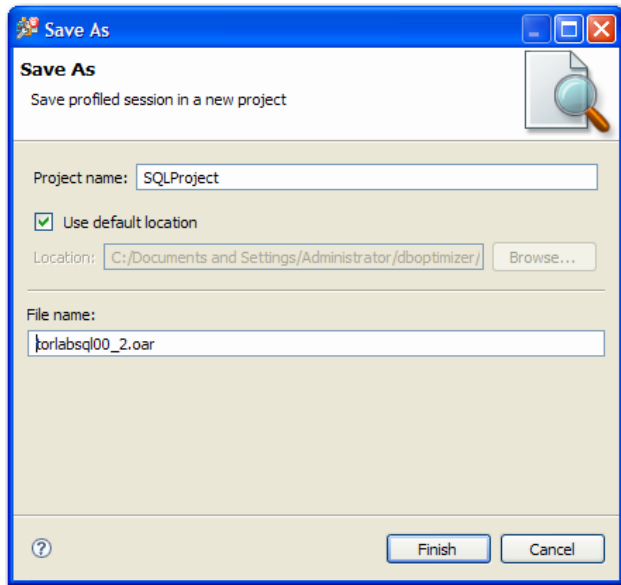
Tab Name	Description
<b>SQL</b>	Displays information about the statements associated with the event.
<b>Sessions</b>	Provides information about the sessions associated with the event.

- When you select a session, the **Session Details**, **SQL**, and **Events** tabs appear.

Tab Name	Description
<b>Session Details</b>	Provides parameters regarding the session. For example, database server connection information, and data regarding the client tool and application.
<b>SQL</b>	Displays information about the statements associated with the session.
<b>Events</b>	Provides information about the events associated with the session.

## 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. This provides you with the ability to open the file at a later time for subsequent analysis.



*Profiling sessions can be saved as **.oar** files for use at a later time.*

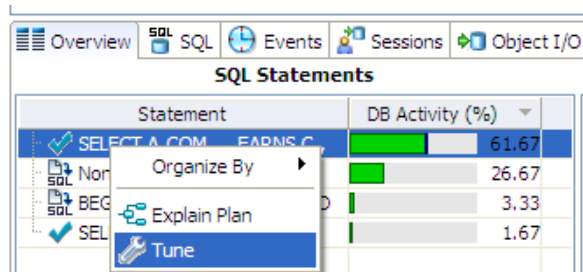
Once you have saved a profiling session, it appears in **SQL Project Explorer** under the name you saved it as. It can be opened again by double-clicking the project name.

### To save a profiling session:

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

## 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.*

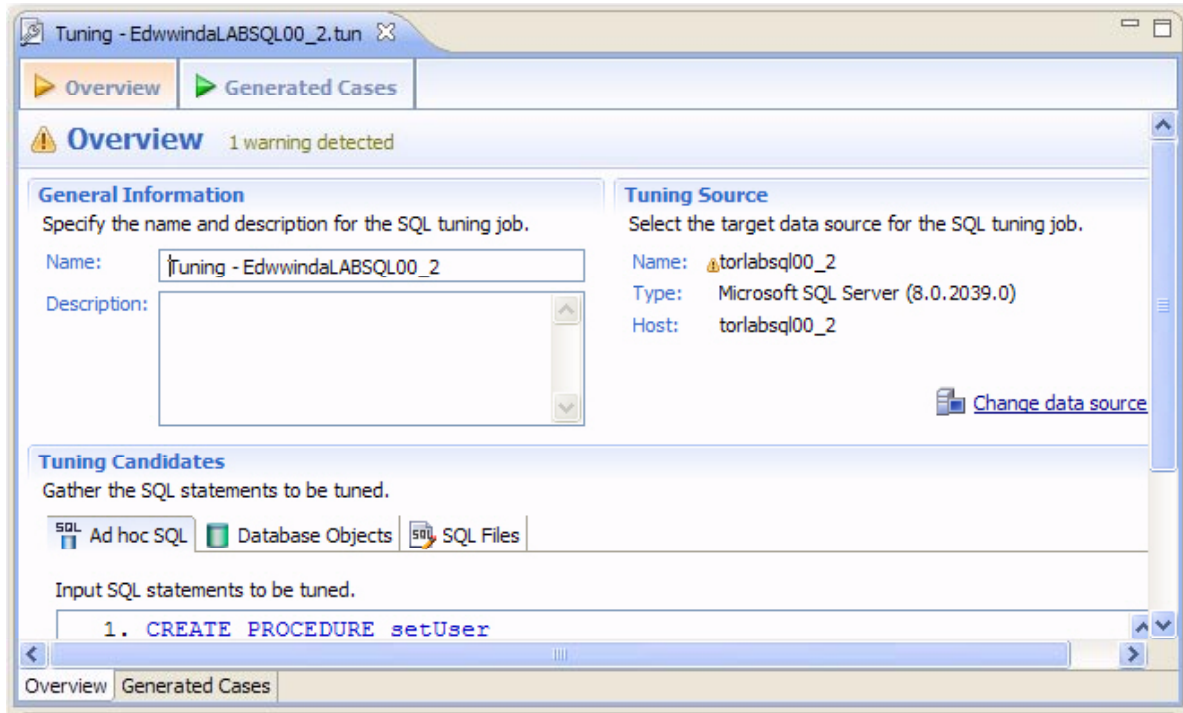
### To Import a statement from Profiler into Tuner

Select one or more statements in Profiler and right-click and 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.

## Session 4: Tuning 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, but on Oracle platforms, Tuner also supports index and statistic analysis.



*Tuner analyzes specified SQL statements and then supplies execution path directives. This enables you to select alternate paths for queries, thus optimizing system performance based on analysis.*

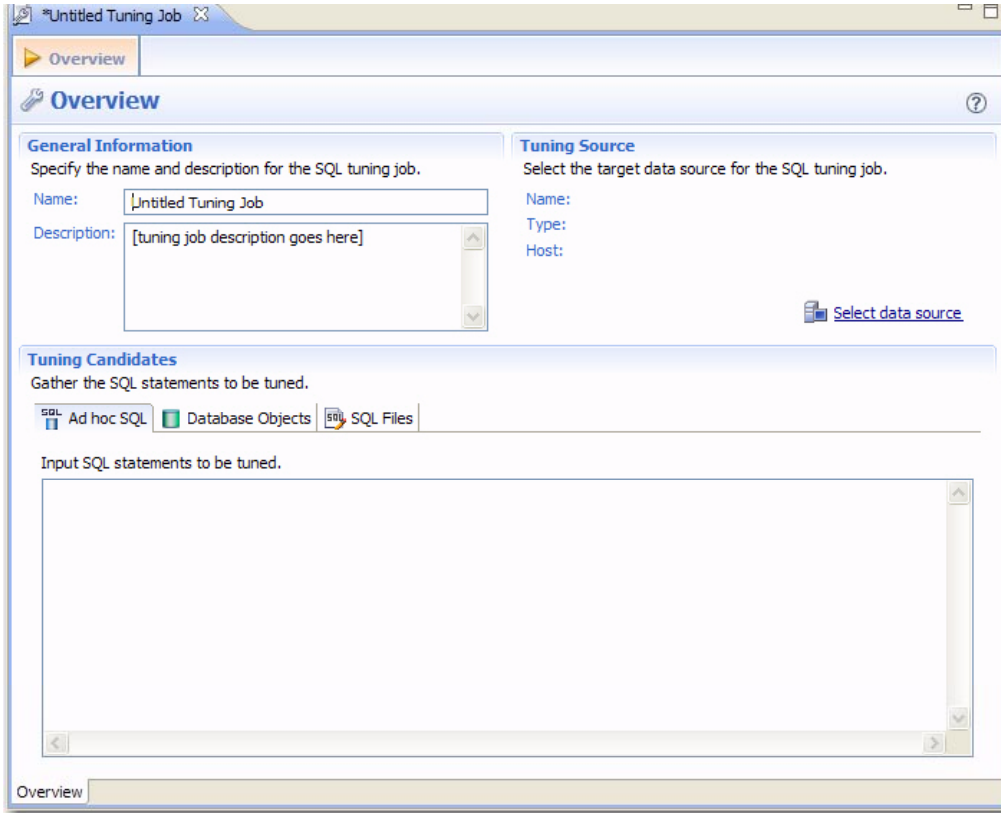
For example, if Tuner is selecting from two tables (A and B), it will enable the joining of A to B, or B to A, as well as the join form.

Additionally, different joining methods such as nested loops or hash joins can be used and will be tested. Tuner selects alternate paths, and enables you to change the original path to one of these alternates. Execution paths slower than the original are eliminated altogether, which enables you to select and initiate the quickest of the returned selection and improve query times, overall.

This is a better method than relying on the native platform optimizer, as it can be wrong through incorrect or missing object statistics, skewed data, correlated predicates, or a bug in the optimizer.

## 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.



*Tuning jobs are defined in SQL Tuner by specifying the data source and corresponding statements to be tuned and then executing the tuning job process.*

When SQL Tuner first opens, you need to define the parameters of the tuning job. This includes specifying a job name and description, as well as selecting the data source and corresponding statements to be tuned.

- A **Job Name** identifies the tuning job in the application, and should be specified with this in mind. Specify a meaningful name that clearly identifies the job in the views and dialogs of the working environment.
- The **Description** is an optional field, but should still reflect the nature of the tuning job.
- Click **Select Data Source** and choose a data source from which you want to tune statements. The **Tuning Source** box identifies the data source as displayed by **Name**, **Type**, **Host**, and **Schema** (if applicable).

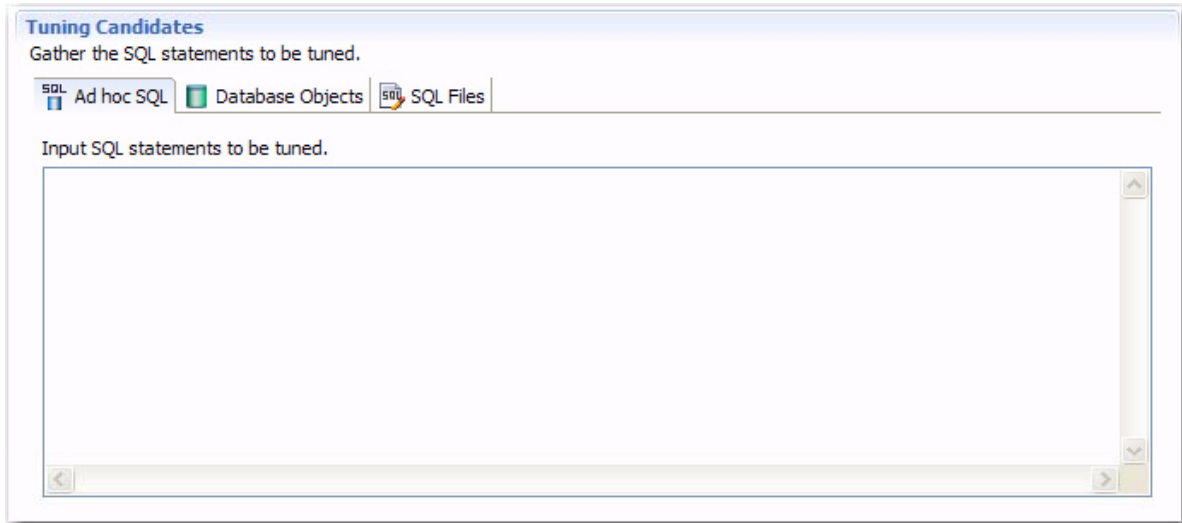
### To create a new tuning job:

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

## Adding SQL Statements

Once you have created a name for the tuning job and indicated its source, you need to add the SQL statements that you want the job to tune. Statements are added to a job via the **Tuning Candidates** box. All standard DML statements (SELECT, INSERT, DELETE, UPDATE) are viable for the tuning procedure.



Statements are added to a tuning job via the **Tuning Candidates** box.

There are three different ways to add SQL statements to a job, as reflected by the three tabs in the **Tuning Candidates** box:

- **Ad Hoc SQL** enables you to tune statements by manually typing, or cutting and pasting a statement into the input field.
- **Database Objects** enables you to select stored packages from a list on the data source you specified in the **Tuning Source** box. This includes functions, materialized views, packages, package bodies, procedures, stored outlines, triggers, and views.
- The **SQL Files** tab enables you to choose an SQL file that has been saved to the system directory. This includes project files residing in the application, or files that reside on your machine or a network directory.

### To add an ad hoc statement:

Select the **Ad Hoc SQL** tab and manually type an SQL statement in the window. Alternatively, copy/paste the statement from another source.

### To add a database object:

Select the **Database Objects** tab and click **Add**. The **Data Source Object Selection** dialog appears. Type an object name prefix or pattern in the field provided and choose a statement from the window as it populates to match what you typed.

### To add a saved SQL file:

Select the **SQL Files** tab and click **Workspace** or **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 job.

## Running a Tuning Job

As you add SQL statements to the job, DML is parsed from the statements and added to the **Generated Cases** tab.

Each extracted statement is listed by **Name**, **Text**, and **Source**. 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.

**Generated Cases**

SQL Statements and Cases					Cost	Elapsed Time			
	Name	Text	Source	Index Analysis	Value	Value (s)	Result	Phy	
<input checked="" type="checkbox"/>	SQL Statement 1	select from	Ad Hoc	Click to optimize	3.0	7.825			
<input checked="" type="checkbox"/>	SQL Statement 2	select from	Ad Hoc	OK					
<input checked="" type="checkbox"/>	USE_CONCAT								
<input checked="" type="checkbox"/>	NO_EXPAND								
<input checked="" type="checkbox"/>	SQL Statement 3	select from	Ad Hoc	Unable to					
<input checked="" type="checkbox"/>	SQL Statement 4	select from	Ad Hoc	OK	1.0				
<input checked="" type="checkbox"/>	SQL Statement 5	select from	Ad Hoc	Unable to					
<input checked="" type="checkbox"/>	SQL Statement 10	select from	Ad Hoc	Click to optimize	3.0				
<input checked="" type="checkbox"/>	[Null c...rmation				3.0				
<input checked="" type="checkbox"/>	ALL_ROWS				3.0				
<input checked="" type="checkbox"/>	FIRST_ROWS				3.0				
<input checked="" type="checkbox"/>	FULL				3.0				
<input checked="" type="checkbox"/>	INDEX_FFS				3.0				

Number of times to execute each case: 1

Run/Cancel Job controls

The **Generated Cases** tab enables you to measure the various load costs of the original tuning statements and generated cases for each, which suggest alternative query paths to optimizing your data source.

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.

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

Once you have executed a tuning job, the **Generated Cases** tab will reflect SQL Tuner's analysis of the specified statements. Once these have been analyzed, you can proceed to modifying the Tuner results and applying specified cases on the data source to optimize its performance.

### To execute a tuning job:

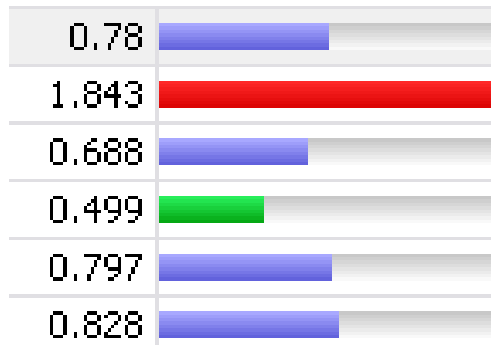
Navigate to the **Generated Cases** tab and modify the number of times to execute in the **Number of Times to Execute Each Case** field. Then click the execution icon in the lower right-side of the screen. The tuning job runs, analyzing each statement and case, and providing values in the appropriate columns.

## Analyzing Tuner Results

When you have executed a tuning job, the **Generated Cases** tab reflects Tuner's analysis of the specified statements and cases.

Once a tuning job has executed, use the **Cost** and **Elapsed Time** value columns to determine the fastest execution path for each statement.

In the **Cost** and **Elapsed Time** columns, the values of the original statement are considered the baseline values. A **Cost** field can be expanded to provide a graphical representation of the values for statements and cases. Similarly, the **Elapsed Time** field can 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 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.













In terms of color-coding, values within the degradation and improvement threshold are represented with a neutral color of light blue. Values less than the improvement threshold are represented with a color of green, and values greater than the degradation threshold are represented by the color red.

### To determine the best cases for statement execution path time:

Once the tuning job has executed, view the **Generated Cases** tab and determine the best possible case in terms of the **Cost** and **Elapsed Time** 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.

## Applying Tuner Results on the Data Source

Once Tuner has generated cases and statement results, you can apply those suggested changes to the data source from the **Generated Cases** tab.

SQL Statements and Cases >>		Cost >>	Elapsed Time >>	Other Execution Statistics		
	Name	Value ▾	Value (s)	Physical Reads	Logical Reads	Consistent Gets
	<input checked="" type="checkbox"/> Statement 1					
	<input checked="" type="checkbox"/> [Null column comparison]					
	<input checked="" type="checkbox"/> Statement 2					
	<input checked="" type="checkbox"/> Statement 3					
	<input checked="" type="checkbox"/> Statement 4					
	<input checked="" type="checkbox"/> Statement 5					
	<input checked="" type="checkbox"/> Statement 11					
	<input checked="" type="checkbox"/> Statement 13					
	<input checked="" type="checkbox"/> Statement 14	2.0	0	0	3	3
	<input checked="" type="checkbox"/> ALL_ROWS	2.0	0.829	1	3	3
	<input checked="" type="checkbox"/> FIRST_ROWS	2.0				
	<input checked="" type="checkbox"/> NO PARALLEL	2.0				

Cases can be selected and applied on the data source where the statement originated. This process improves the former statement's execution path and therefore lessens overall data source load.

### To apply a change:

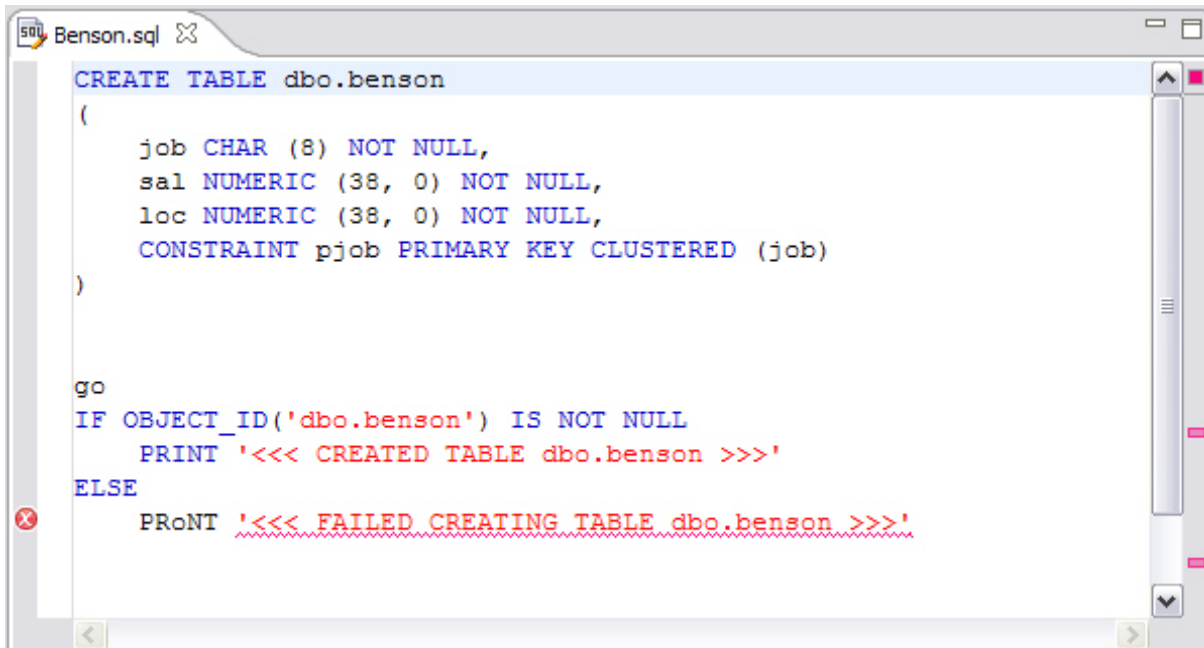
Right-click on the **Name** field of the case you want to modify the original statement with and choose **Apply Change**. The **Apply Change** dialog appears. Choose **Execute** to apply the change to the statement automatically. Alternatively, you can select **Open in New SQL Editor** to make manually changes or save it to file.



## Session 5: 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.



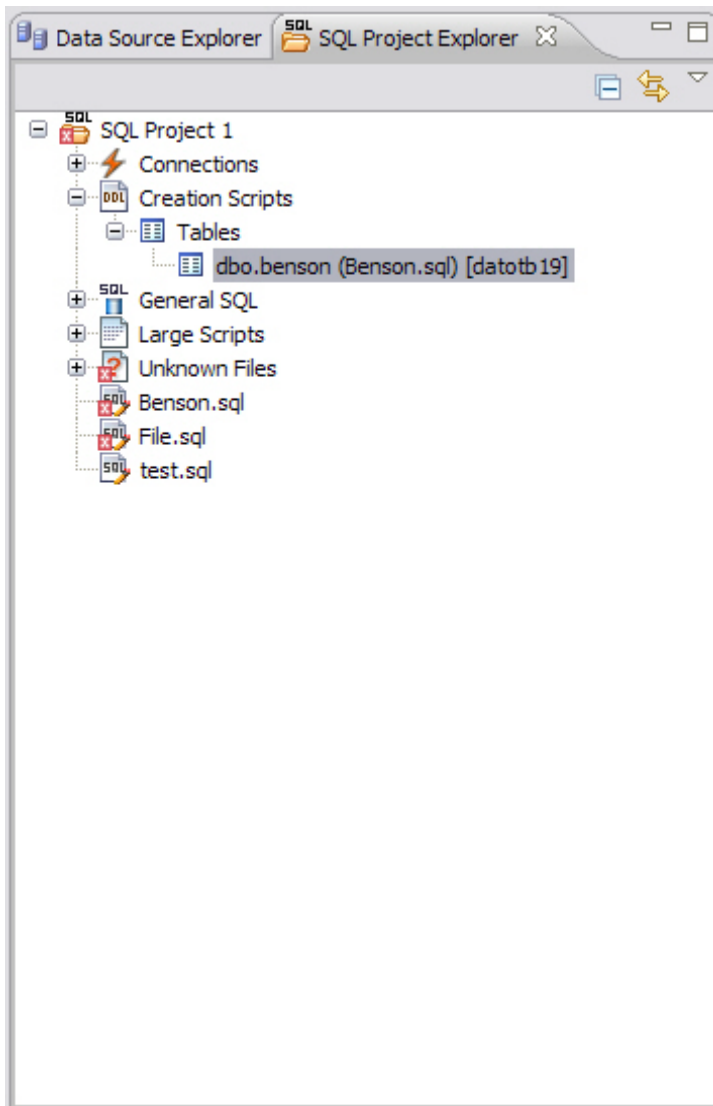
SQL Editor provides key features to ensure an increase in code accuracy and overall development efficiency.

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

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.



*SQL Project Explorer provides a tree of all files developed and saved in DB Optimizer.*

#### **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 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 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

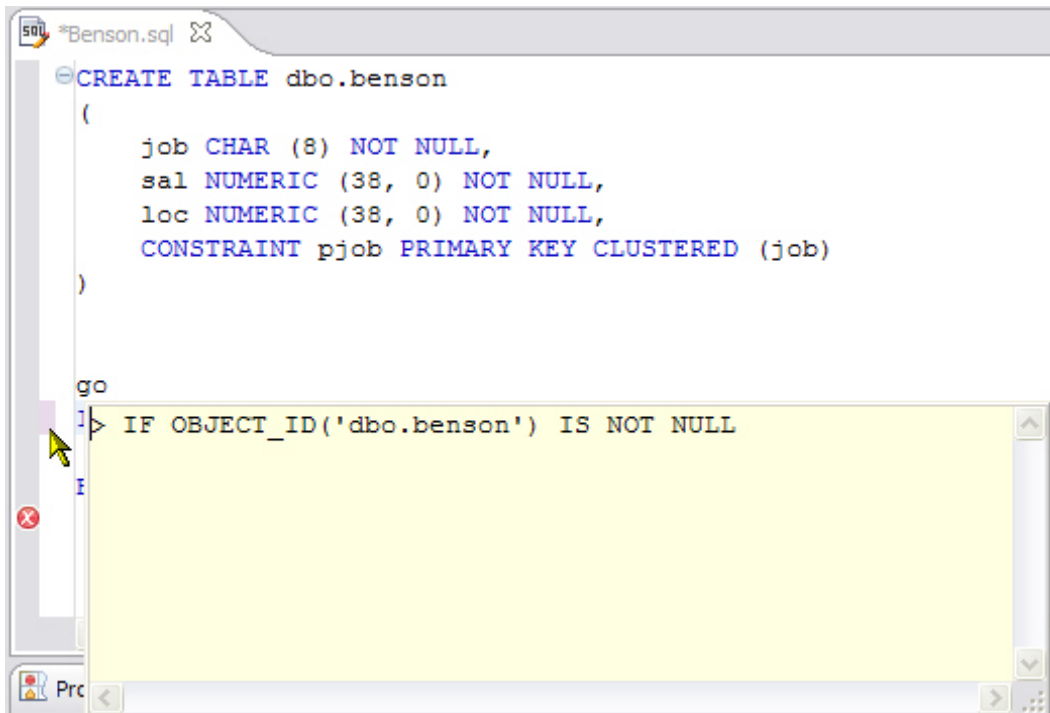
SQL Editor identifies commands and provides syntax highlighting changes that are automatically added to the code as you add lines, which enables you to clearly and quickly understand code when you read it in the interface.

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

Code	Formatting
Comments	Green italic font
SQL Commands	Dark blue font
Syntax Errors	Red underlined font
Coding Errors	Red font

- **Comments:** green italics
- **SQL Commands:** 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.

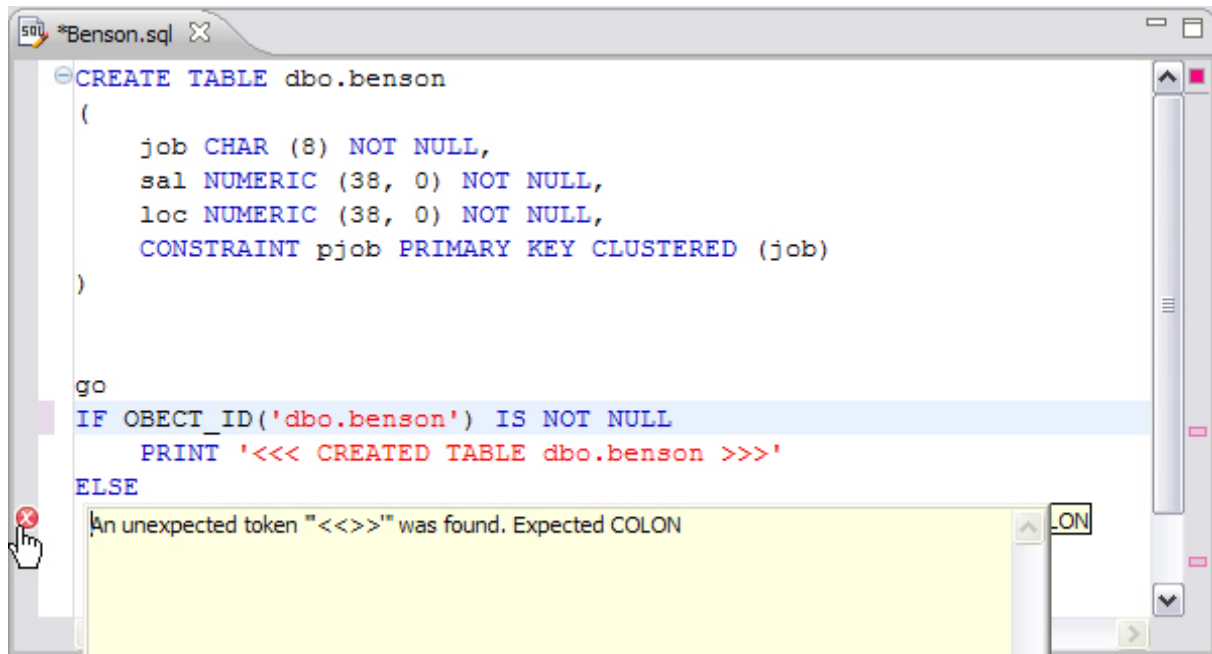


The purple change bar indicates if a line of code has been changed from its original text. Hover your mouse over the change bar to view the original text.

## Automatic Error Detection

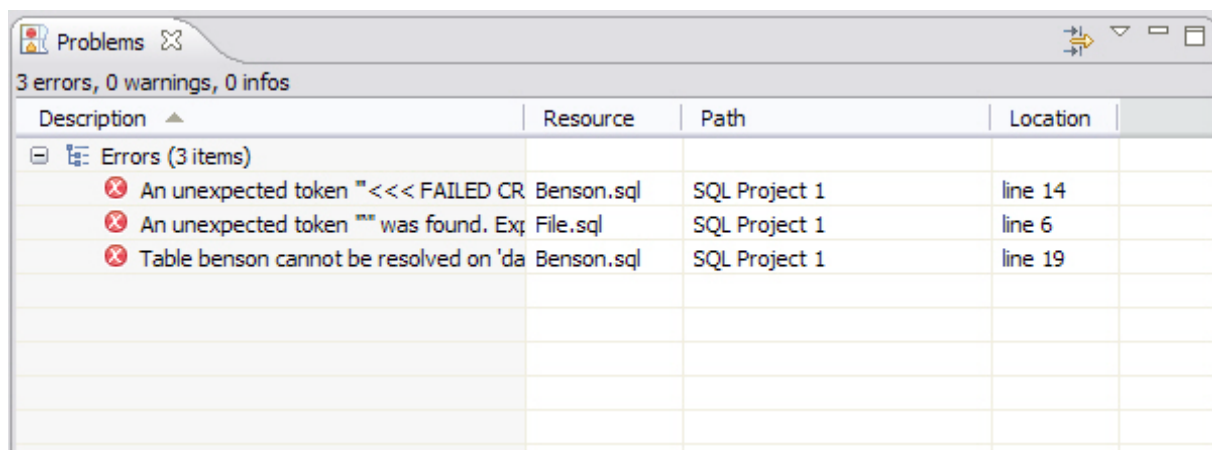
The automatic error detection functionality of the editor highlights errors and typos in the code as you work in “real time”.

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.



Syntax errors are automatically flagged by line as you work with code in SQL Editor. Hover the mouse over the error icon to view the specific error message.

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 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.

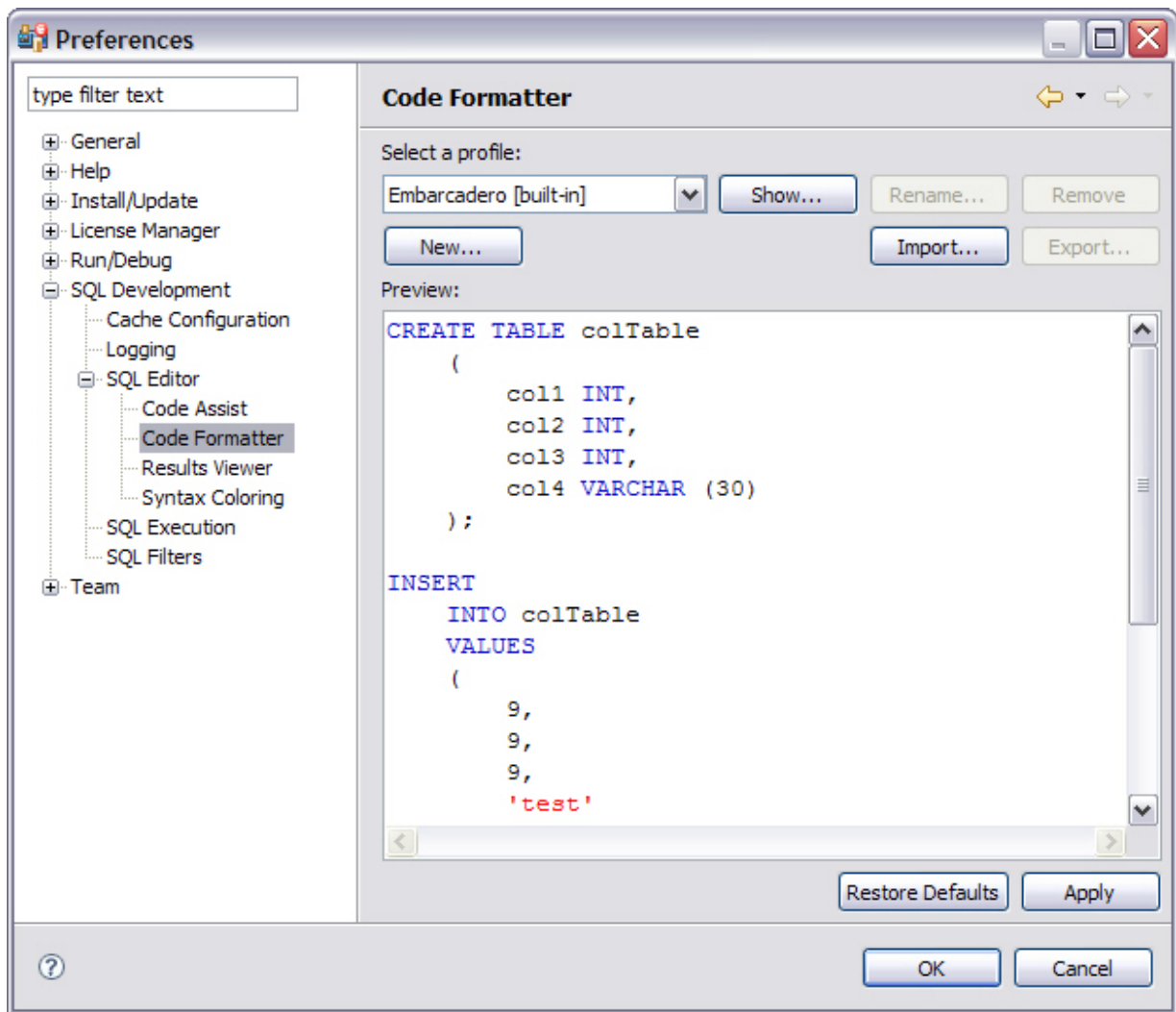
Additionally, hyperlinks can be used to link procedures or the function of a call statement, as well as function calls in DML statements.

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 it in SQL code. This enables you to set global formatting preferences one time, and then apply it 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.)



*Code formatting parameters can be globally set and then applied to all development work in SQL Editor.*

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 that you want to apply formatting to and execute the **Format** command from the right-click menu. The files will be 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 provides 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 + I** 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 + I** on your keyboard.

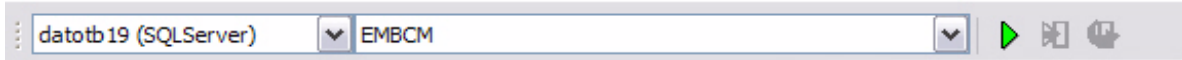
The editor suggestions appear in a window beneath the selected statement. When you click a suggested amendment, the affected code is automatically updated.



## 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 indicate 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 beside the **Execute** icon.

### To execute a file:

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

### To execute a transaction:

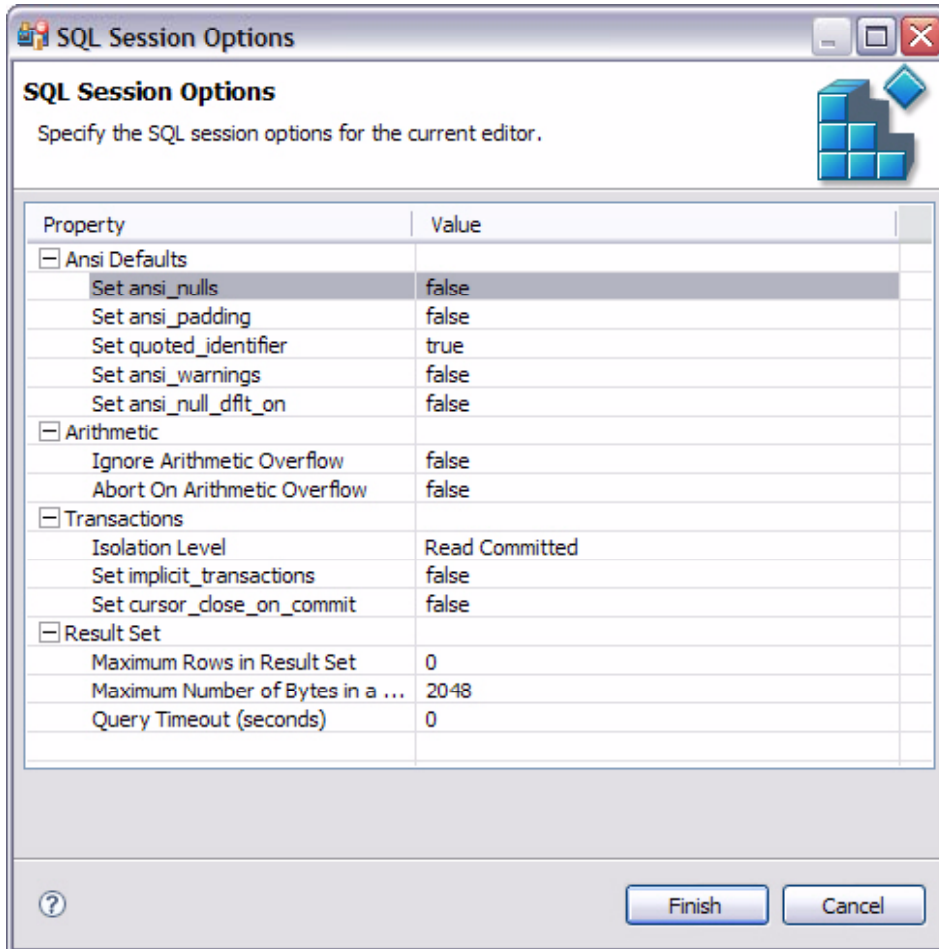
Open the transaction file you want to run and ensure it is associated with the correct database, then click the **Start Transaction** icon. DB Optimizer executes the transaction on the database you specified.

### To commit a transaction:

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

## Configuring SQL Execution Parameters

If you do not want to use the default execution options provided by DB Optimizer, 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 a execution-per-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.*

### To modify SQL session options:

1. Click the SQL Session Options icon in 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.

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

## Additional Evaluation Resources

### Embarcadero Technologies Product Support

The Embarcadero Web site is an excellent source of 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](#)

### Licensing Your Embarcadero Technologies Product

All Embarcadero Technologies products include a 30-day trial period. To continue using the product without interruption, we recommend that you license it as soon as possible. To license your product, use the License Request Wizard found in the Help menu of your respective product. If you have not yet purchased your Embarcadero Technologies product, contact [sales@embarcadero.com](mailto:sales@embarcadero.com), or [uk.sales@embarcadero.com](mailto:uk.sales@embarcadero.com) for sales in the EMEA region.

### 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 the application. 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.

For additional information regarding Embarcadero Technologies Technical Support, go to the Support page on our Web site.

### Embarcadero Technologies on the Web

To download evaluations of other Embarcadero Technologies products or to learn more about our company and our products visit us at [www.embarcadero.com](http://www.embarcadero.com).