EMBARCADERO
TECHNOLOGIES®

Product Documentation

DB Optimizer™

Evaluation Guide

Version 2.0.1

Published November 27, 2009

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

Contents

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: 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. Profiling Details of the SQL Profiler have been expanded to show Session Details for Sybase, and SQL Server and the SQL that ran in the selected session for Sybase, SQL Server, and DB2.

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. The Analysis tab of the tuner provides a graphical diagram of an SQL query to show how the tables in the query would be joined to satisfy the query and also provides suggestions for the creation of indexes that may increase query performance.

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.

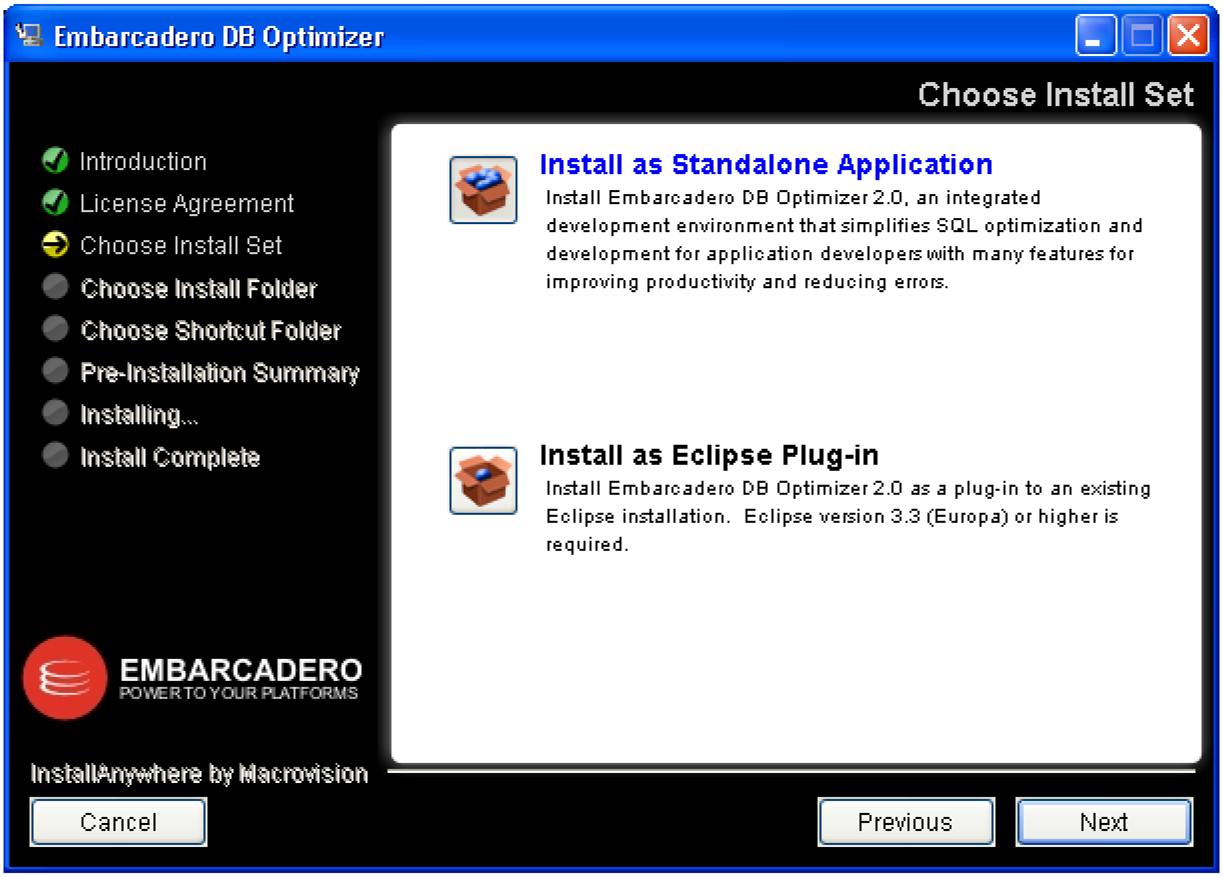
Session 1: Getting Started with DB Optimizer

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.



Install as Standalone Application

Install Embarcadero DB Optimizer 2.0, an integrated development environment that simplifies SQL optimization and development for application developers with many features for improving productivity and reducing errors.

Install as Eclipse Plug-in

Install Embarcadero DB Optimizer 2.0 as a plug-in to an existing Eclipse installation. Eclipse version 3.3 (Europa) or higher is required.

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.

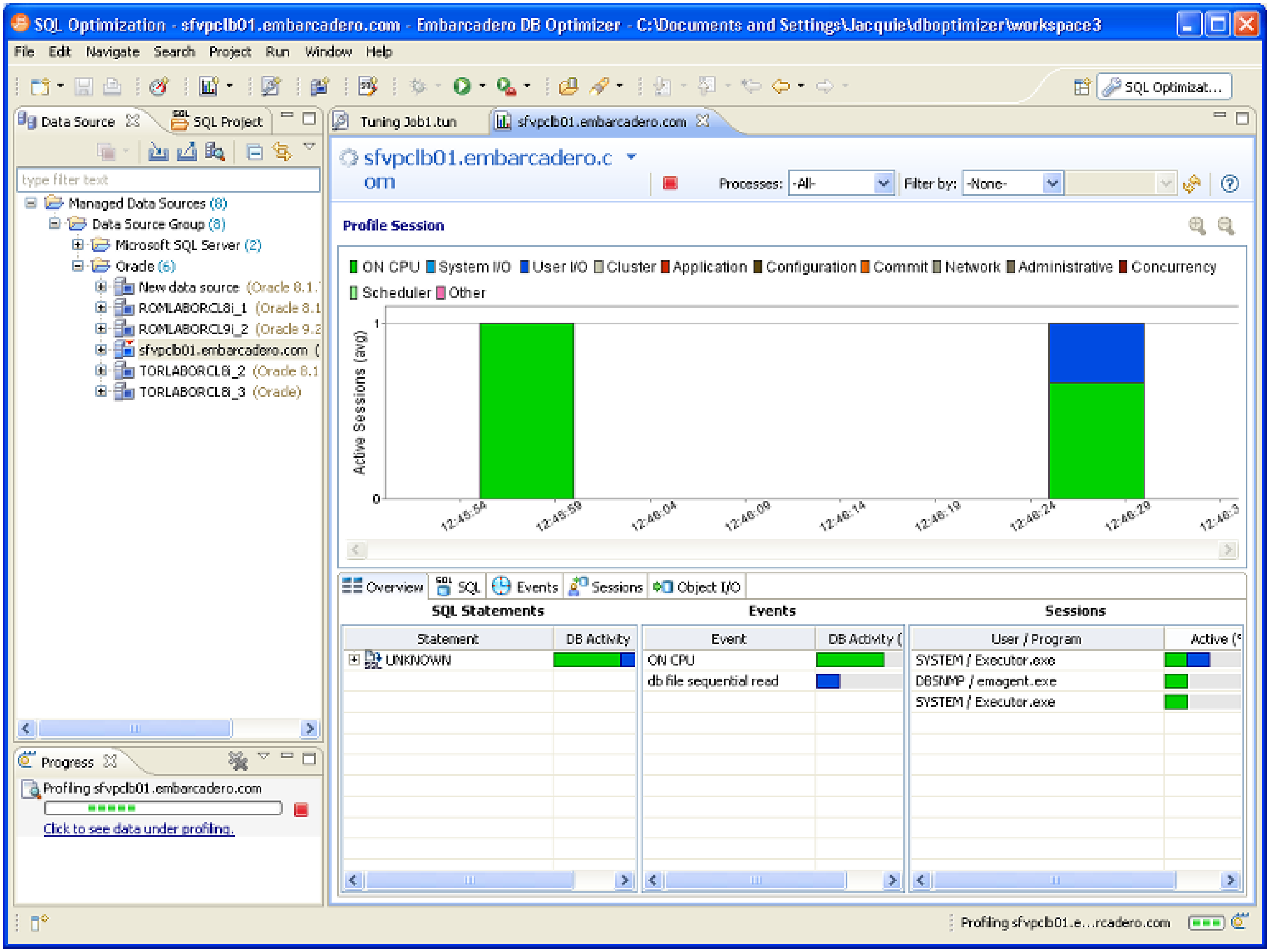
To start DB Optimizer:

- 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.


SQL Optimization - sfvpclb01.embarcadero.com - Embarcadero DB Optimizer - C:\Documents and Settings\Jacquie\dboptimizer\workspace3
File
Edit
Navigate
Search
Project
Run
Window
Help
SQL Optimizat...
Data Source
SQL Project
type filter text
Managed Data Sources (8)
Data Source Group (8)
Microsoft SQL Server (2)
Oracle (6)
New data source (Oracle 8.1.
ROMLABORCL8i_1 (Oracle 8.1
ROMLABORCL9i_2 (Oracle 9.2
sfvpclb01.embarcadero.com (
TORLABORCL8i_2 (Oracle 8.1
TORLABORCL8i_3 (Oracle)
Progress
Profiling sfvpclb01.embarcadero.com
Click to see data under profiling.
Tuning Job1.tun
sfvpclb01.embarcadero.com
sfvpclb01.embarcadero.c om
Processes: -All-
Filter by: -None-
Profile Session
🟩 ON CPU 🟦 System I/O 🟦 User I/O □ Cluster 🟥 Application 🟫 Configuration 🟧 Commit ▨ Network ▨ Administrative 🟥 Concurrency
🟩 Scheduler 🟪 Other
Active Sessions (avg)
1
0
12:45:54
12:45:59
12:46:04
12:46:09
12:46:14
12:46:19
12:46:24
12:46:29
12:46:3
Overview
SQL
Events
Sessions
Object I/O
Profiling sfvpclb01.e...rcadero.com

SQL Statements

Statement	DB Activity
UNKNOWN	🟩🟦

Events

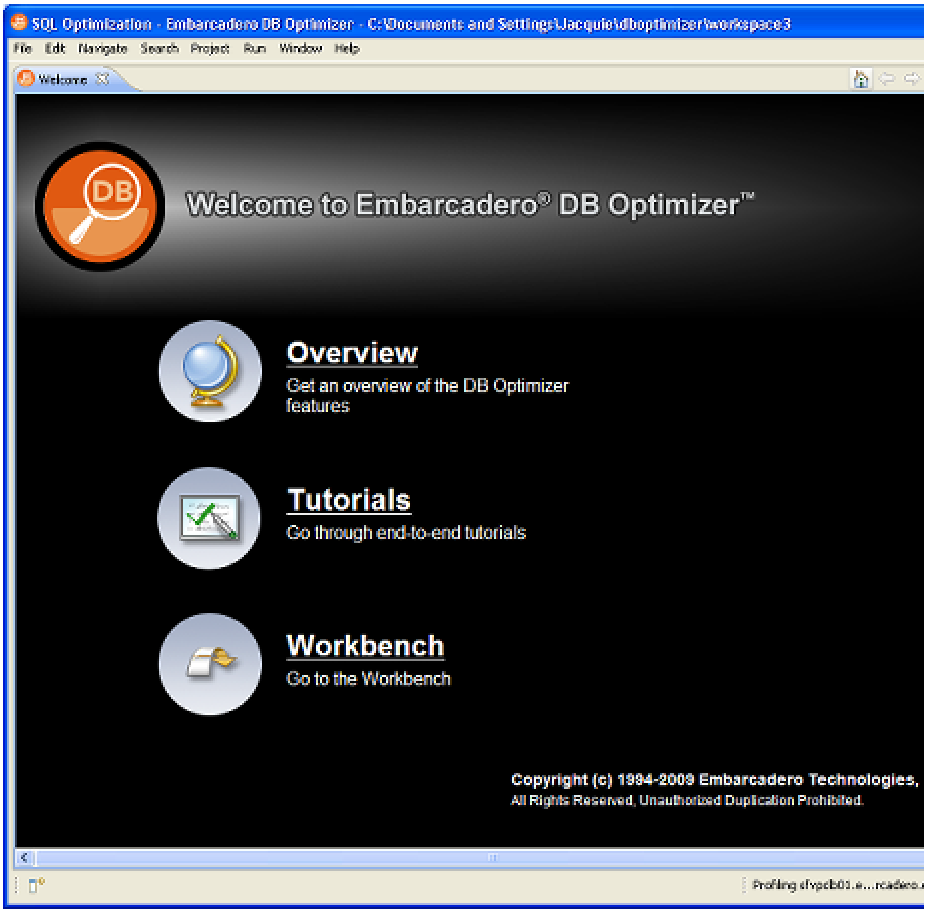
Event	DB Activity (
ON CPU	🟩
db file sequential read	🟦

Sessions

User / Program	Active (°
SYSTEM / Executor.exe	🟩🟦
DBSNMP / emagent.exe	🟩
SYSTEM / Executor.exe	🟩

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

- The **Welcome Page** is the first item in the Help menu. Click **Help > Welcome** to gain access to documentation and help files/tutorials to familiarize yourself with the product. It provides easy access to information that may be of value to new users.


SQL Optimization - Embarcadero DB Optimizer - C:\Documents and Settings\Jacquie\dboptimizer\workspace3
File Edit Navigate Search Project Run Window Help
Welcome
DB

Welcome to Embarcadero® DB Optimizer™

Overview
Get an overview of the DB Optimizer features

Tutorials
Go through end-to-end tutorials

Workbench
Go to the Workbench

Copyright (c) 1994-2009 Embarcadero Technologies,
All Rights Reserved, Unauthorized Duplication Prohibited.

Profiling dlvpcb01.e...rcadero.

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.

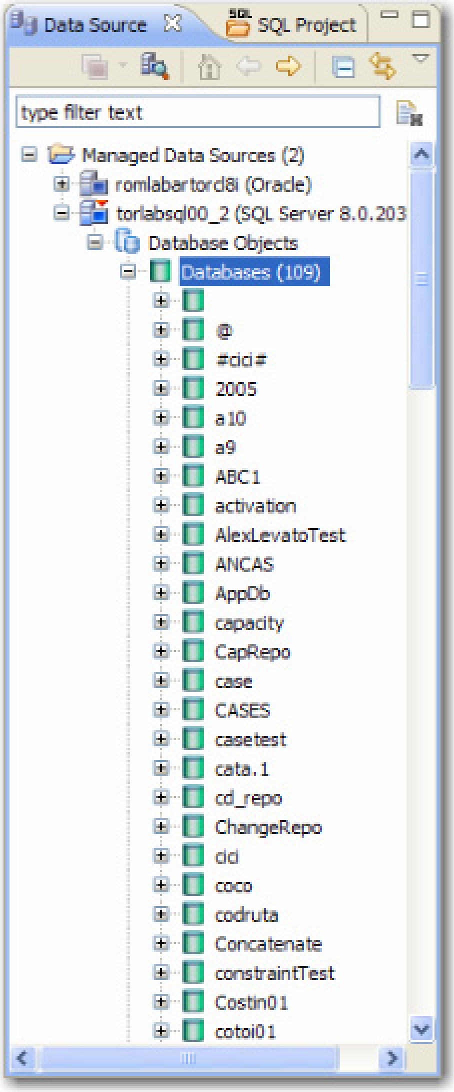
- The **Menu Bar** and **Command Toolbar** contain commands that execute various functional aspects of the application such as launching Views and Editors, navigation commands, and environment preference settings. The Toolbar contains icons that represent specific menu commands. For example, the **Search** 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.



type filter text

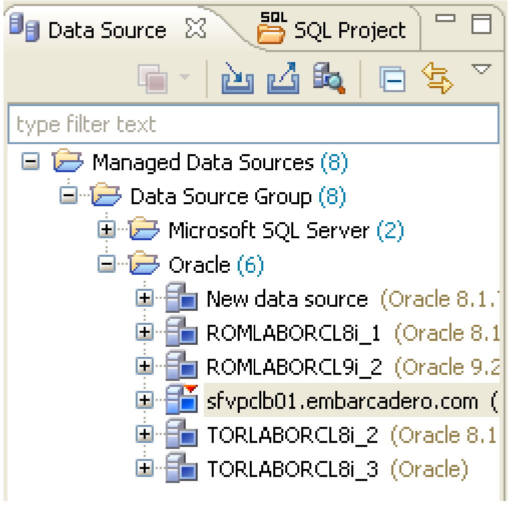
```
Managed Data Sources (2)
  romlabartord8i (Oracle)
  torlabsql00_2 (SQL Server 8.0.203
    Database Objects
      Databases (109)
        
        @
        #cici#
        2005
        a10
        a9
        ABC1
        activation
        AlexLevatoTest
        ANCAS
        AppDb
        capacity
        CapRepo
        case
        CASES
        casetest
        cata.1
        cd_repo
        ChangeRepo
        cici
        coco
        codruta
        Concatenate
        constraintTest
        Costin01
        cotoi01
```

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.



Managed Data Sources (8)
Data Source Group (8)
Microsoft SQL Server (2)
Oracle (6)
New data source (Oracle 8.1.
ROMLABORCL8i_1 (Oracle 8.1
ROMLABORCL9i_2 (Oracle 9.2
sfvpclb01.embarcadero.com (
TORLABORCL8i_2 (Oracle 8.1
TORLABORCL8i_3 (Oracle)

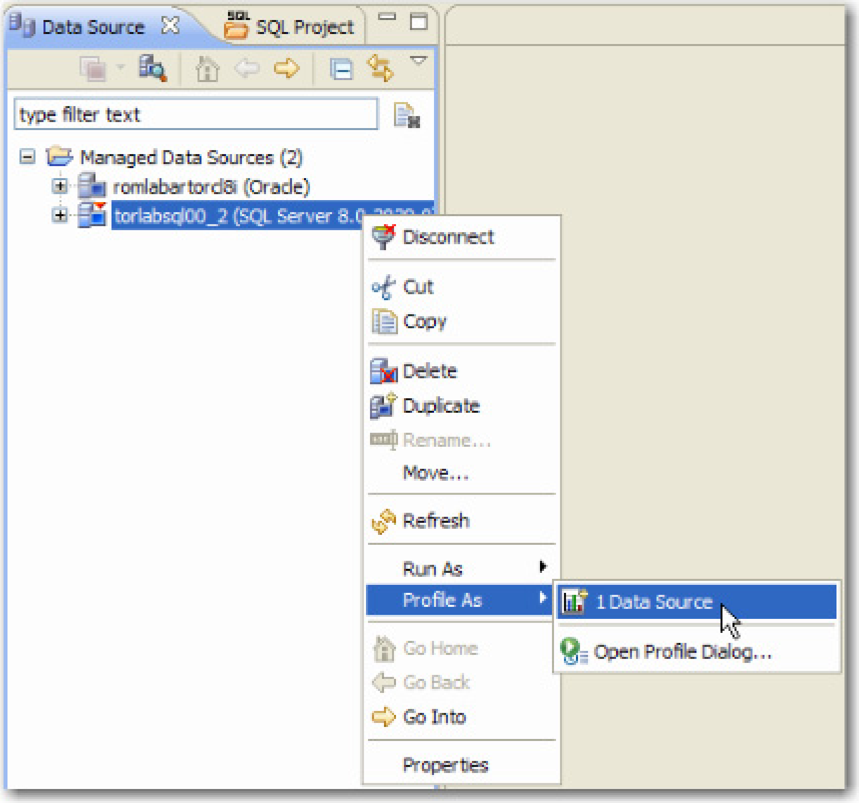
The Data Source Explorer view provides an organizational tree of registered data sources and the parameters associated with them.

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.

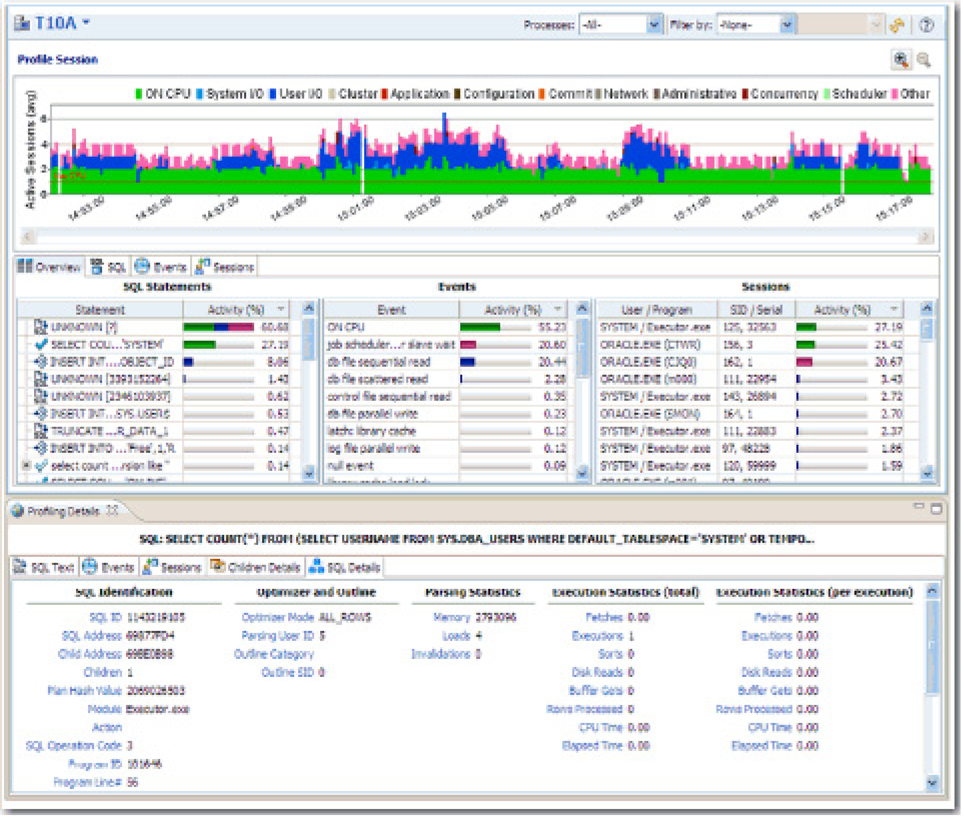
To browse a data source:

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

NOTE: The SQL Profiler is not available in the “Basic” version of DB Optimizer.

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 Statements

Statement	Activity (%)
UNKNOWN [?]	60.68
SELECT COU...'SYSTEM'	27.19
INSERT INT....OBJECT_ID	8.06
UNKNOWN [3393152264]	1.43
UNKNOWN [2346103937]	0.62
INSERT INT...SYS.USER$	0.53
TRUNCATE ...R_DATA_1	0.47
INSERT INTO ...'Fred',1,'R	0.14
select count ...rsion like ''	0.14

Events

Event	Activity (%)
ON CPU	55.23
job scheduler...r slave wait	20.60
db file sequential read	20.44
db file scattered read	2.28
control file sequential read	0.35
db file parallel write	0.23
latch: library cache	0.12
log file parallel write	0.12
null event	0.09

Sessions

User / Program	SID / Serial	Activity (%)
SYSTEM / Executor.exe	125, 32563	27.19
ORACLE.EXE (CTWR)	156, 3	25.42
ORACLE.EXE (CJQ0)	162, 1	20.67
ORACLE.EXE (m000)	111, 22954	3.43
SYSTEM / Executor.exe	143, 26894	2.72
ORACLE.EXE (SMON)	164, 1	2.70
SYSTEM / Executor.exe	111, 22883	2.37
SYSTEM / Executor.exe	97, 48228	1.86
SYSTEM / Executor.exe	120, 59999	1.59

Profiling Details

SQL: SELECT COUNT(*) FROM (SELECT USERNAME FROM SYS.DBA_USERS WHERE DEFAULT_TABLESPACE='SYSTEM' OR TEMPO...

SQL Text | Events | Sessions | Children Details | SQL Details

SQL Identification
SQL ID: 1143219105
SQL Address: 69877FD4
Child Address: 698E0B98
Children: 1
Plan Hash Value: 2069026503
Module: Executor.exe
Action:
SQL Operation Code: 3
Program ID: 101646
Program Line#: 56

Optimizer and Outline
Optimizer Mode: ALL_ROWS
Parsing User ID: 5
Outline Category:
Outline SID: 0

Parsing Statistics
Memory: 2793096
Loads: 4
Invalidations: 0

Execution Statistics (total)
Fetches: 0.00
Executions: 1
Sorts: 0
Disk Reads: 0
Buffer Gets: 0
Rows Processed: 0
CPU Time: 0.00
Elapsed Time: 0.00

Execution Statistics (per execution)
Fetches: 0.00
Executions: 0.00
Sorts: 0.00
Disk Reads: 0.00
Buffer Gets: 0.00
Rows Processed: 0.00
CPU Time: 0.00
Elapsed Time: 0.00

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:

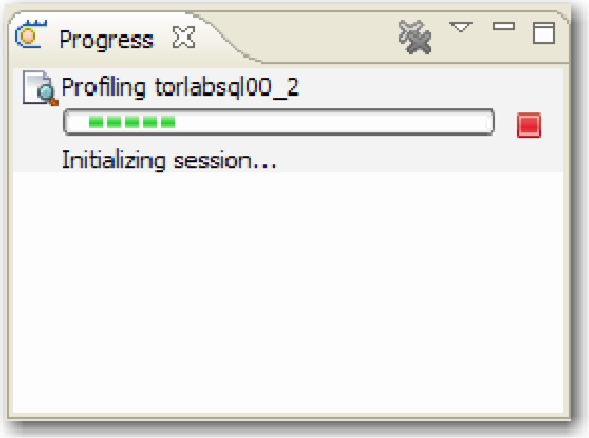
- The **Load Graph** provides a display of the overall load on the system. The bars represent individual aspects of the data source.
- The **Top Activity** section displays where load originates. Specifically, **Top Activity** displays the top SQL statements, the top events the database spends time in, and the top activity sessions that may be causing issues in terms of query time and response.
- The **Profiling Details View** displays detailed information 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. 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.



Profiling torlabsql00_2

Initializing session...

The Progress view indicates that the 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

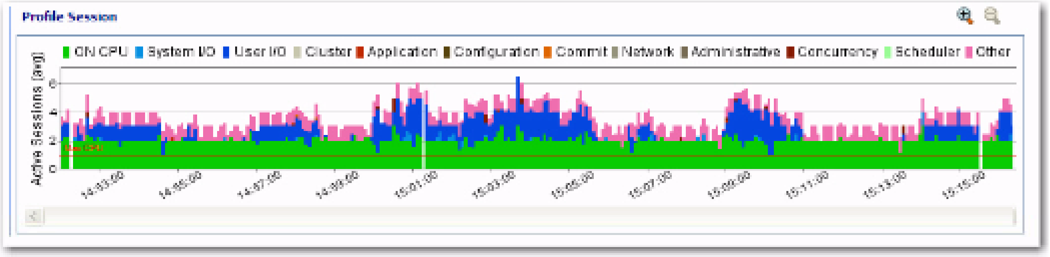
Profiler is composed of three essential diagnostic views that provide information about load on a particular database in the system. These views enable you to identify bottlenecks and view details about specific queries that execute and wait over the course of a profiling session.

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

- 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.

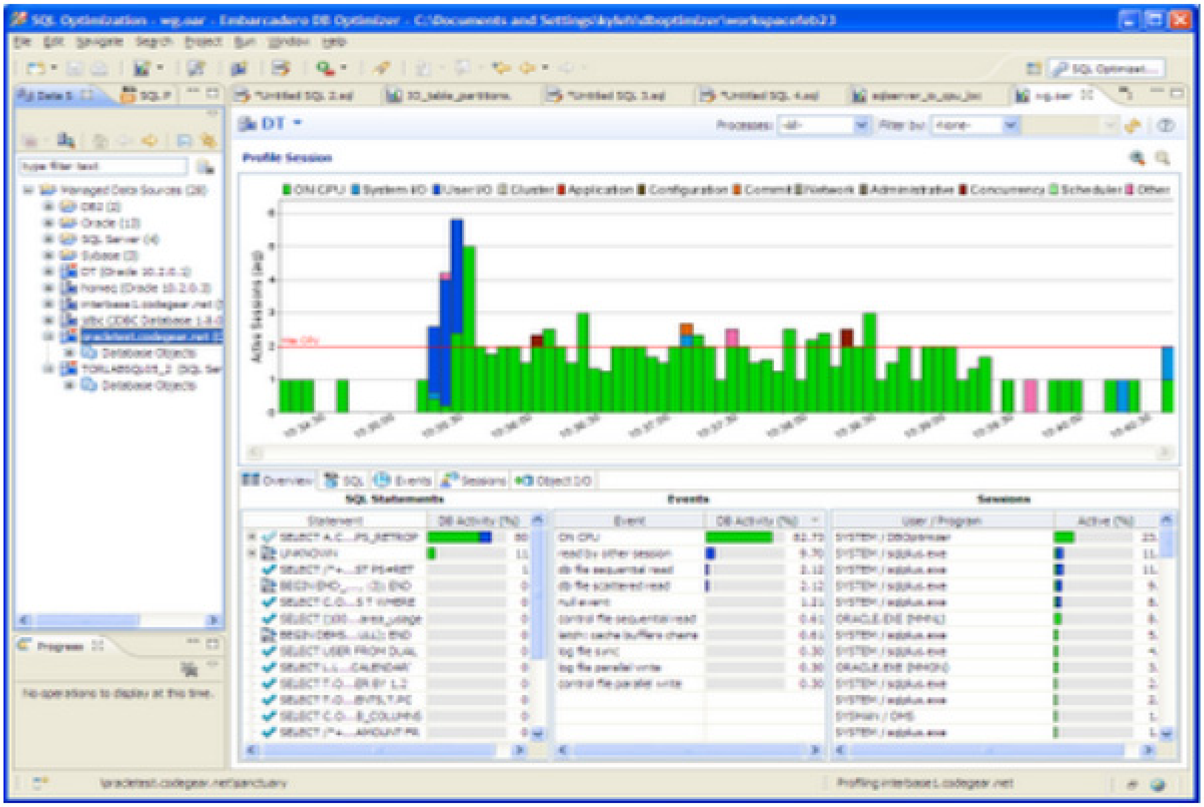


The Load Chart displays the overall load of the database that you analyzed with Profiler.

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.



SQL Statements

Statement	DB Activity (%)
SELECT A.C....PS_RETROP	80
UNKNOWN	11
SELECT /*+....ST PS#RET	1
BEGIN DHD_.... (3); END	0
SELECT C.O....S T WHERE	0
SELECT DIX....area_usage	0
BEGIN DBMS....ULL); END	0
SELECT USER FROM DUAL	0
SELECT L.L....CALENDAR	0
SELECT T.O....ER BY L.2	0
SELECT T.O....BVPS,T.PC	0
SELECT C.O....B_COLUMNS	0
SELECT /*+....AMOUNT FR	0

Events

Event	DB Activity (%)
ON CPU	82.75
read by other session	9.70
db file sequential read	2.12
db file scattered read	2.12
null event	1.21
control file sequential read	0.61
latch: cache buffers chains	0.61
log file sync	0.30
log file parallel write	0.30
control file parallel write	0.30

Sessions

User / Program	Active (%)
SYSTEM / DBOptimizer	23.
SYSTEM / sqlplus.exe	11.
SYSTEM / sqlplus.exe	11.
SYSTEM / sqlplus.exe	9.
SYSTEM / sqlplus.exe	8.
ORACLE.EXE (MMNL)	8.
SYSTEM / sqlplus.exe	5.
SYSTEM / sqlplus.exe	4.
ORACLE.EXE (MMON)	3.
SYSTEM / sqlplus.exe	3.
SYSTEM / sqlplus.exe	2.
SYSMAN / OMS	1.
SYSTEM / sqlplus.exe	1.

\\oracletest.codegear.net\sanctuary

Profiling interbase1.codegear.net

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.

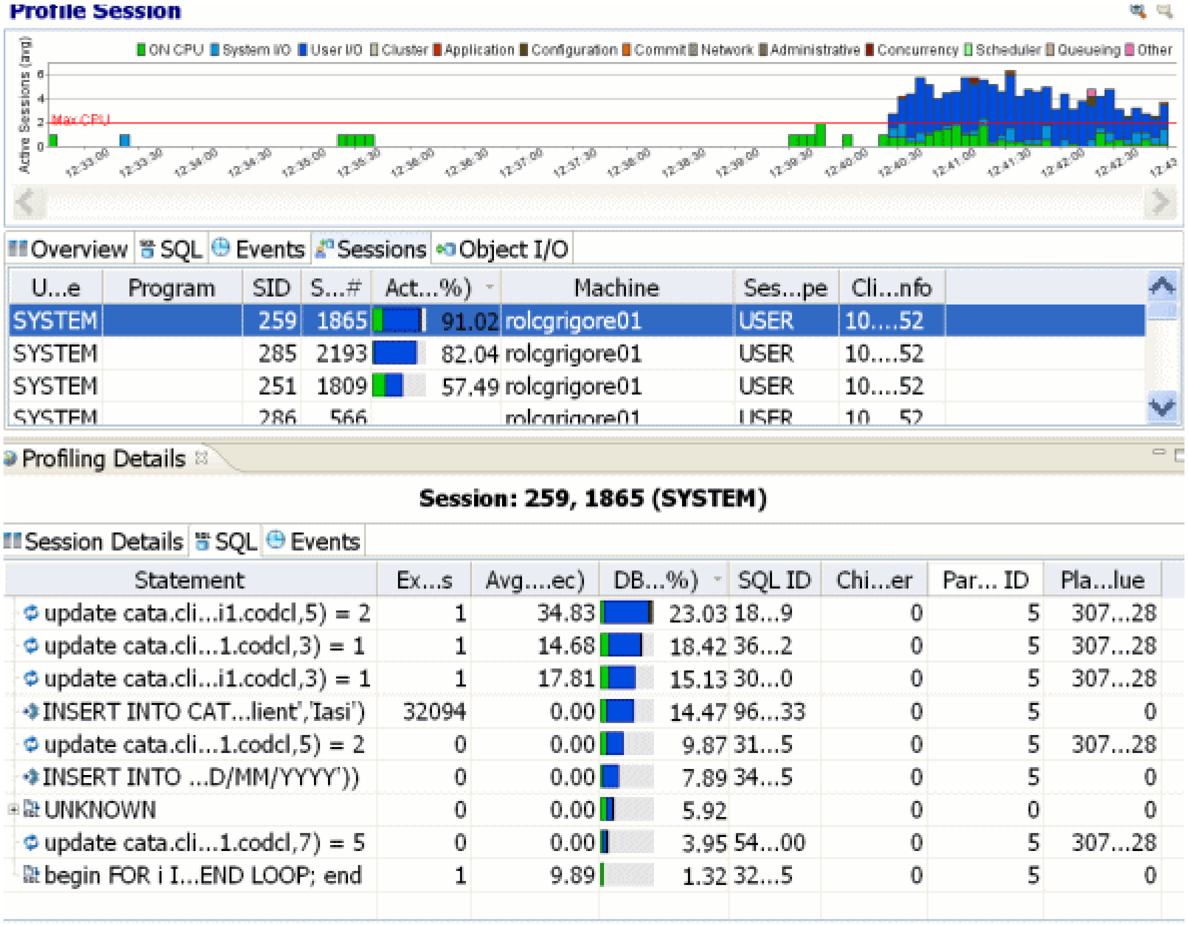
- The **Overview** tab provides summary information about SQL statements and events and their activity levels, and sessions, their system process ids and their activity level. You can reorder the rows in any of the three sections of this tab. For example, clicking the Event column in the Events section changes the alphabetical order to ascending or decending.
- 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 **Object I/O** tab provides information about I/O. This tab will not be displayed if you are profile a database on a platform other than Oracle.

Profiling Details

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

> **TIP:** 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.



U...e	Program	SID	S...#	Act...%)	Machine	Ses...pe	Cli...nfo
SYSTEM		259	1865	91.02	rolcgrigore01	USER	10....52
SYSTEM		285	2193	82.04	rolcgrigore01	USER	10....52
SYSTEM		251	1809	57.49	rolcgrigore01	USER	10....52
SYSTEM		286	566		rolcgrigore01	USER	10....52

Session: 259, 1865 (SYSTEM)

Statement	Ex...s	Avg....ec)	DB...%)	SQL ID	Chi...er	Par... ID	Pla...lue
update cata.cli...i1.codcl,5) = 2	1	34.83	23.03	18...9	0	5	307...28
update cata.cli...1.codcl,3) = 1	1	14.68	18.42	36...2	0	5	307...28
update cata.cli...i1.codcl,3) = 1	1	17.81	15.13	30...0	0	5	307...28
INSERT INTO CAT...lient','Iasi')	32094	0.00	14.47	96...33	0	5	0
update cata.cli...1.codcl,5) = 2	0	0.00	9.87	31...5	0	5	307...28
INSERT INTO ...D/MM/YYYY'))	0	0.00	7.89	34...5	0	5	0
UNKNOWN	0	0.00	5.92		0	0	0
update cata.cli...1.codcl,7) = 5	0	0.00	3.95	54...00	0	5	307...28
begin FOR i I...END LOOP; end	1	9.89	1.32	32...5	0	5	0

Profiling Details displays detailed parameter information on a statement, event, or session that was selected in Top Activity. The data displayed also varies by database platform.

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.

Statement Selected

When a **Statement** is selected the following Profile Detail tabs are available.

Tab Name	Description	Supported Platform			
		Oracle	Sybase	DB2	SQL Server
SQL Text	Displays the full code of the selected SQL statement.	yes	yes	yes	yes

Tab Name	Description	Supported Platform			
		Oracle	Sybase	DB2	SQL Server
SQL Details	Provides details on statement, like execution statistics.	yes		yes	
Events	Provides database activity details about events the statement is associated with.	yes	yes	yes	yes
Sessions	Shows which sessions executed this statement.	yes	yes	yes	yes
Children Details	Lists all copies of the cursor or sql query, if Oracle has cached multiple copies of the same statement.	yes			
Object I/O	If the SQL query has done physical I/O, then these are the objects, such as tables, and indexes that were read to satisfy the query. Temporary objects with not have values in Object and Type columns.	yes			

Event Selected

When an **Event** is selected the following Profile Detail tabs are available.

Tab Name	Description	Supported Platform			
		Oracle	Sybase	DB2	SQL Server
SQL	Shows which SQL statements waited on this event.	yes	yes	yes	yes
Sessions	Provides information about the sessions associated with the event.	yes	yes	yes	yes
Raw Data	Raw data that was sampled from the database, specifically the following: • Sample time • SID • Serial # • User name • Program • Sql ID • P1 • P2 • P3	yes			
Analysis	Displays for “buffer busy waits” and “cache buffer chains latch” waits. The analysis shows data and documentation to assist in solving these bottlenecks.	yes			

Session Selected

When a **Session** is selected the following Profile Detail tabs are available.

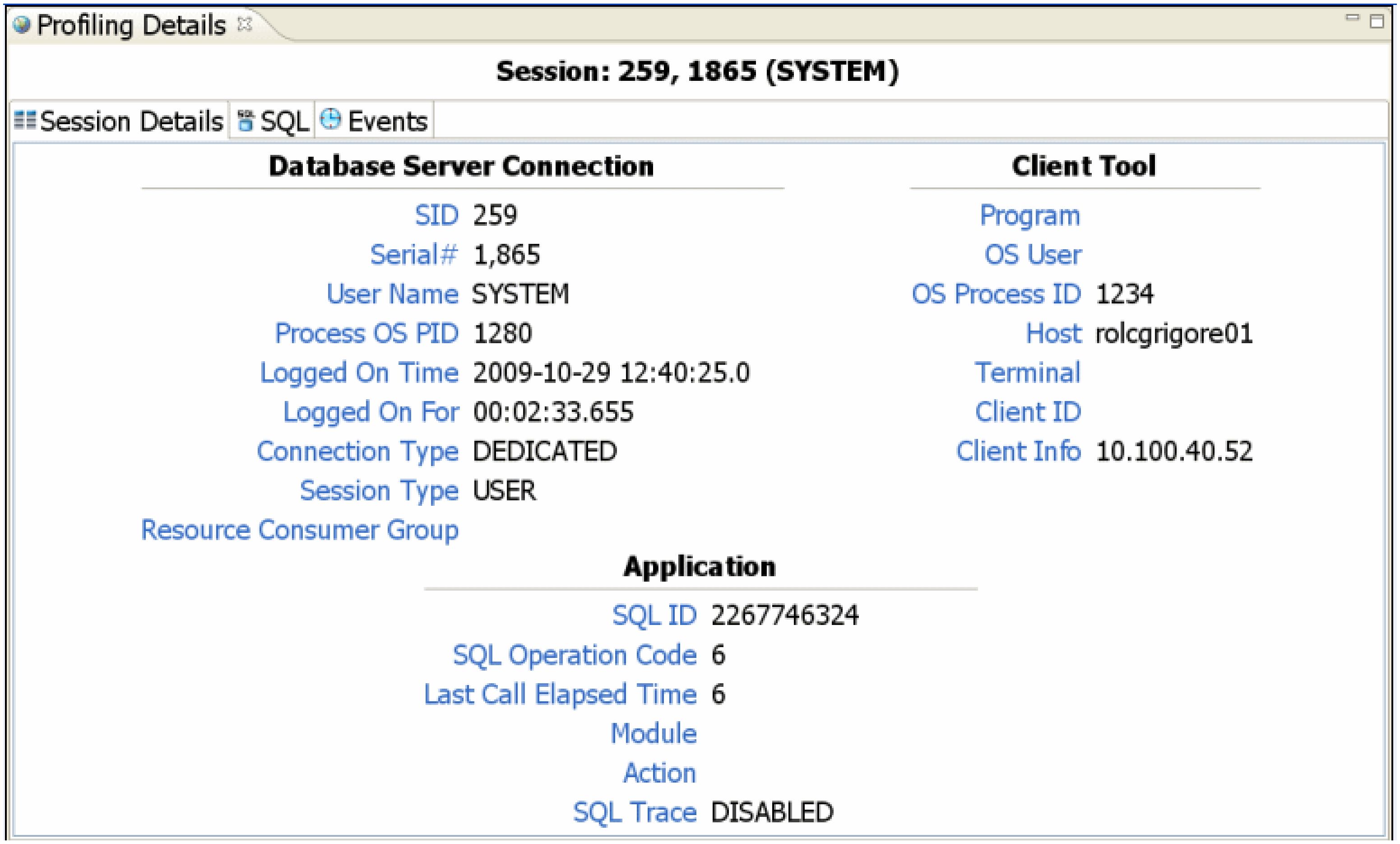
Tab Name	Description	Supported Platform			
		Oracle	**Sybase**	**DB2**	**SQL Server**
Session Details	Provides parameters regarding the session. For example, database server connection information, and data regarding the client tool and application.	yes	yes		yes
SQL	Shows which SQL statements this session ran.	yes	yes	yes	yes
Events	Shows which events this session waited on.	yes	yes	yes	yes

NOTE: When right-clicking on a SQL statement in the **Top Activity Section** in Profiling, if the SQL statement is run by a different user than the user who is running DBO, than 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 tuning under the original user.

To view session details:

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 **Sessions Details** tab.

   Details of the session are displayed.



Session: 259, 1865 (SYSTEM)

Database Server Connection

SID	259
Serial#	1,865
User Name	SYSTEM
Process OS PID	1280
Logged On Time	2009-10-29 12:40:25.0
Logged On For	00:02:33.655
Connection Type	DEDICATED
Session Type	USER
Resource Consumer Group	

Client Tool

Program	
OS User	
OS Process ID	1234
Host	rolcgrigore01
Terminal	
Client ID	
Client Info	10.100.40.52

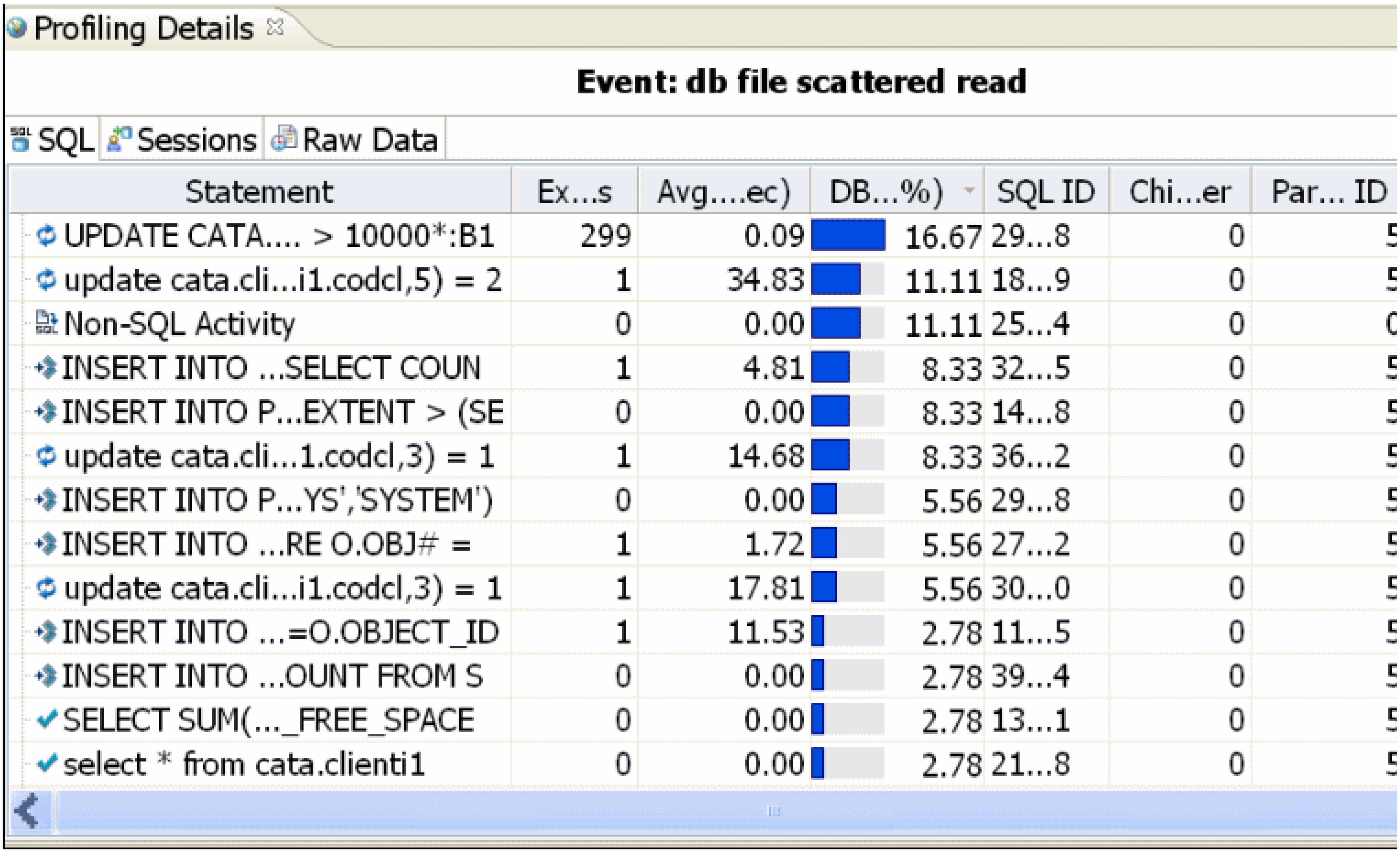
Application

SQL ID	2267746324
SQL Operation Code	6
Last Call Elapsed Time	6
Module	
Action	
SQL Trace	DISABLED

To view session 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** tab.

   The SQL for the application that selected displays.



Profiling Details

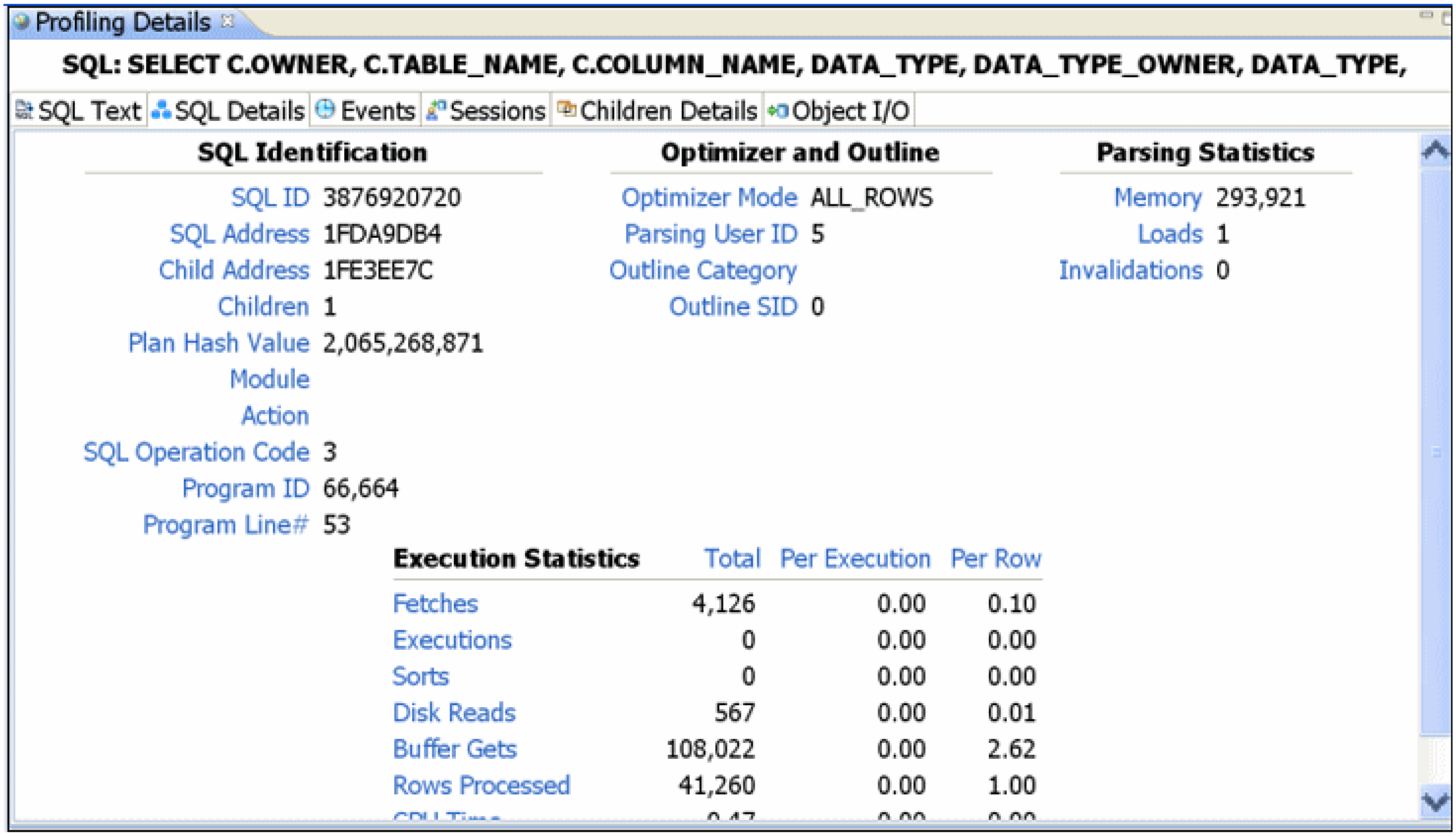
Event: db file scattered read

SQL | Sessions | Raw Data

Statement	Ex...s	Avg....ec)	DB...%)	SQL ID	Chi...er	Par... ID
UPDATE CATA.... > 10000*:B1	299	0.09	16.67	29...8	0	5
update cata.cli...i1.codcl,5) = 2	1	34.83	11.11	18...9	0	5
Non-SQL Activity	0	0.00	11.11	25...4	0	0
INSERT INTO ...SELECT COUN	1	4.81	8.33	32...5	0	5
INSERT INTO P...EXTENT > (SE	0	0.00	8.33	14...8	0	5
update cata.cli...1.codcl,3) = 1	1	14.68	8.33	36...2	0	5
INSERT INTO P...YS','SYSTEM')	0	0.00	5.56	29...8	0	5
INSERT INTO ...RE O.OBJ# =	1	1.72	5.56	27...2	0	5
update cata.cli...i1.codcl,3) = 1	1	17.81	5.56	30...0	0	5
INSERT INTO ...=O.OBJECT_ID	1	11.53	2.78	11...5	0	5
INSERT INTO ...OUNT FROM S	0	0.00	2.78	39...4	0	5
SELECT SUM(..._FREE_SPACE	0	0.00	2.78	13...1	0	5
select * from cata.clienti1	0	0.00	2.78	21...8	0	5

TIP: 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.

3 Click the **SQL Details** tab for summary details of the SQL.



SQL: SELECT C.OWNER, C.TABLE_NAME, C.COLUMN_NAME, DATA_TYPE, DATA_TYPE_OWNER, DATA_TYPE,

SQL Identification

SQL ID	3876920720
SQL Address	1FDA9DB4
Child Address	1FE3EE7C
Children	1
Plan Hash Value	2,065,268,871
Module	
Action	
SQL Operation Code	3
Program ID	66,664
Program Line#	53

Optimizer and Outline

Optimizer Mode	ALL_ROWS
Parsing User ID	5
Outline Category	
Outline SID	0

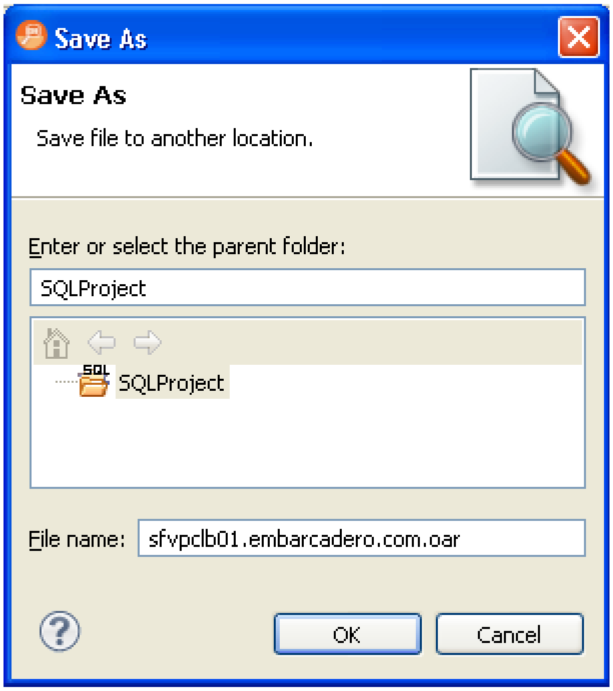
Parsing Statistics

Memory	293,921
Loads	1
Invalidations	0

Execution Statistics	Total	Per Execution	Per Row
Fetches	4,126	0.00	0.10
Executions	0	0.00	0.00
Sorts	0	0.00	0.00
Disk Reads	567	0.00	0.01
Buffer Gets	108,022	0.00	2.62
Rows Processed	41,260	0.00	1.00
CPU Time	0.47	0.00	0.00

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 enables you to open the file at a later time for analysis.



Save As

Save file to another location.

Enter or select the parent folder: SQLProject

File name: sfvpclb01.embarcadero.com.oar

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

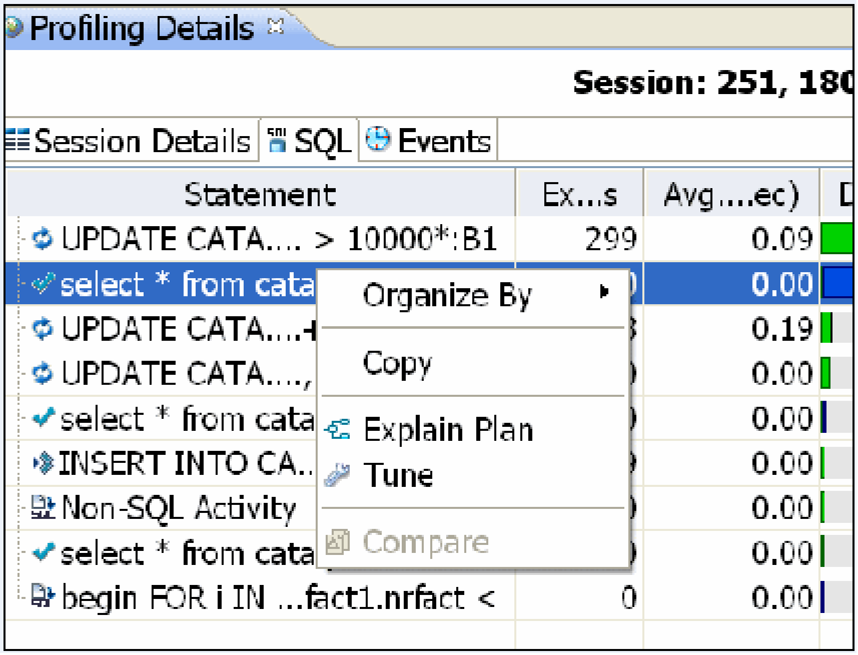
Once you have saved a profiling session, it appears in the **SQL Project Explorer** under the name you saved it as. It can be opened again by double-clicking the project name.

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 **OK**. 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.



Statement	Ex...s	Avg....ec)	D
UPDATE CATA.... > 10000*:B1	299	0.09	
select * from cata		0.00	
UPDATE CATA....+		0.19	
UPDATE CATA....,		0.00	
select * from cata		0.00	
INSERT INTO CA..		0.00	
Non-SQL Activity		0.00	
select * from cata		0.00	
begin FOR i IN ...fact1.nrfact <	0	0.00	

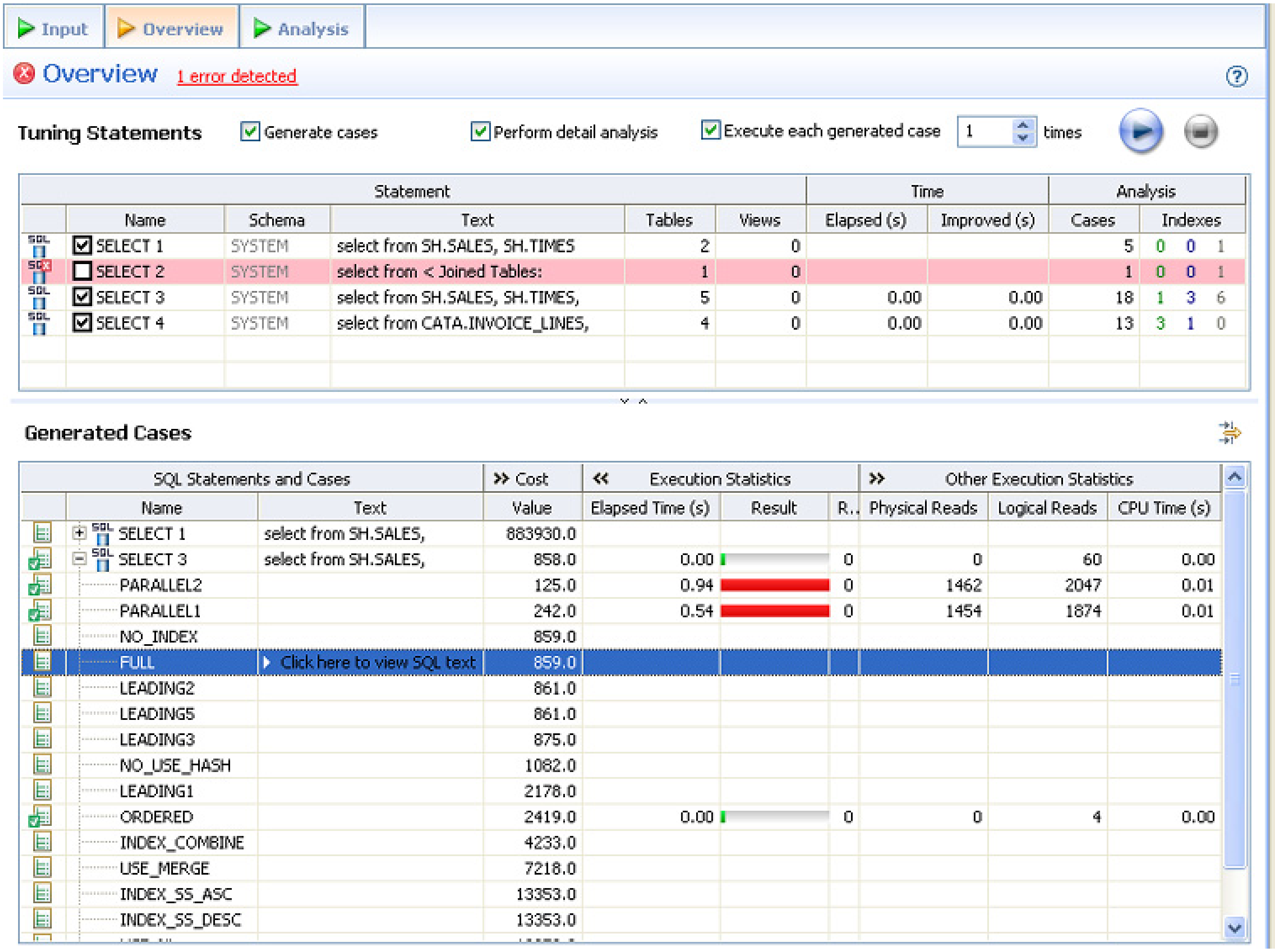
Context menu commands in SQL Profiler enable you to import statements to a tuning job directly from the Profiler interface.

To Import a statement from Profiler into Tuner

Select one or more statements in Profiler, 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, and index and statistics analysis.



Overview 1 error detected

Tuning Statements ☑ Generate cases ☑ Perform detail analysis ☑ Execute each generated case 1 times

Statement						Time		Analysis	
	Name	Schema	Text	Tables	Views	Elapsed (s)	Improved (s)	Cases	Indexes
SQL	☑ SELECT 1	SYSTEM	select from SH.SALES, SH.TIMES	2	0			5	0 0 1
SQL	□ SELECT 2	SYSTEM	select from < Joined Tables:	1	0			1	0 0 1
SQL	☑ SELECT 3	SYSTEM	select from SH.SALES, SH.TIMES,	5	0	0.00	0.00	18	1 3 6
SQL	☑ SELECT 4	SYSTEM	select from CATA.INVOICE_LINES,	4	0	0.00	0.00	13	3 1 0

Generated Cases

SQL Statements and Cases			>> Cost	<< Execution Statistics			>> Other Execution Statistics		
	Name	Text	Value	Elapsed Time (s)	Result	R..	Physical Reads	Logical Reads	CPU Time (s)
	⊞ SELECT 1	select from SH.SALES,	883930.0						
	⊟ SELECT 3	select from SH.SALES,	858.0	0.00		0	0	60	0.00
	PARALLEL2		125.0	0.94		0	1462	2047	0.01
	PARALLEL1		242.0	0.54		0	1454	1874	0.01
	NO_INDEX		859.0						
	FULL	▸ Click here to view SQL text	859.0						
	LEADING2		861.0						
	LEADING5		861.0						
	LEADING3		875.0						
	NO_USE_HASH		1082.0						
	LEADING1		2178.0						
	ORDERED		2419.0	0.00		0	0	4	0.00
	INDEX_COMBINE		4233.0						
	USE_MERGE		7218.0						
	INDEX_SS_ASC		13353.0						
	INDEX_SS_DESC		13353.0						

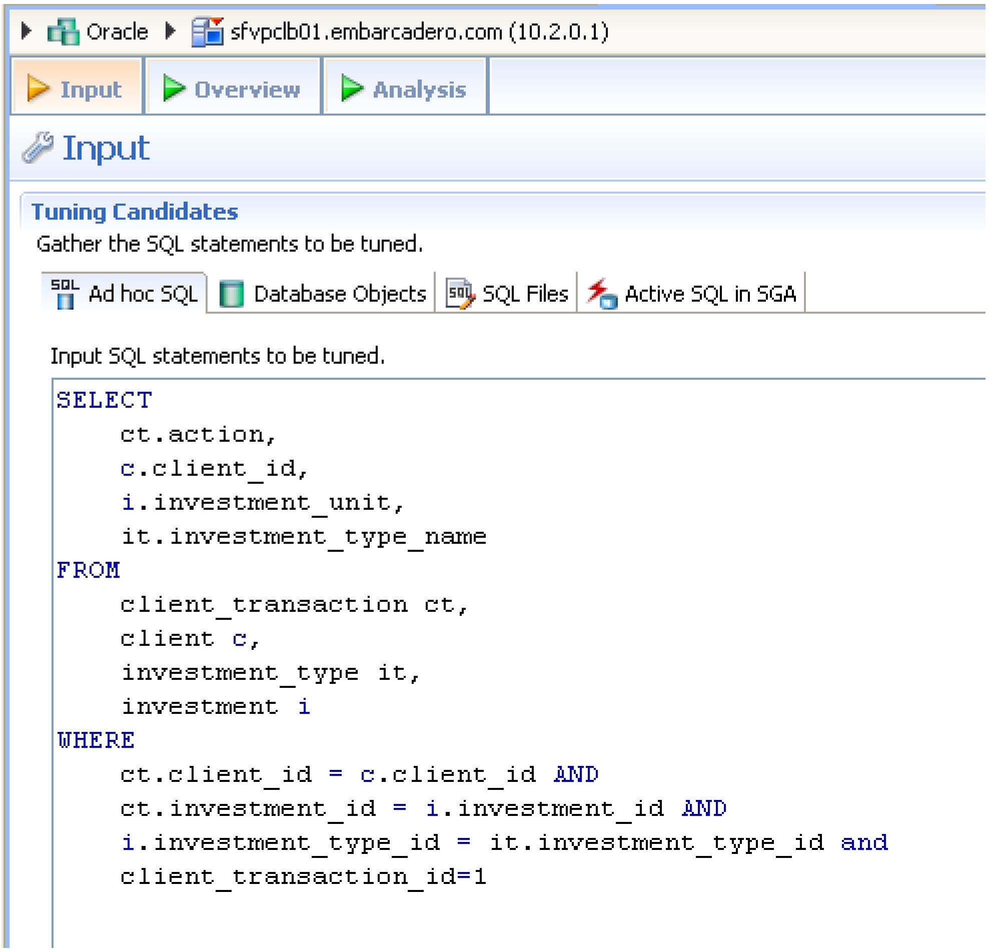
Tuner analyzes specified SQL statements and then supplies execution path directives. This enables you to select alternate paths for queries, thus optimizing system performance based on analysis.

For example, if tuning 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, 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 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.



Gather the SQL statements to be tuned.

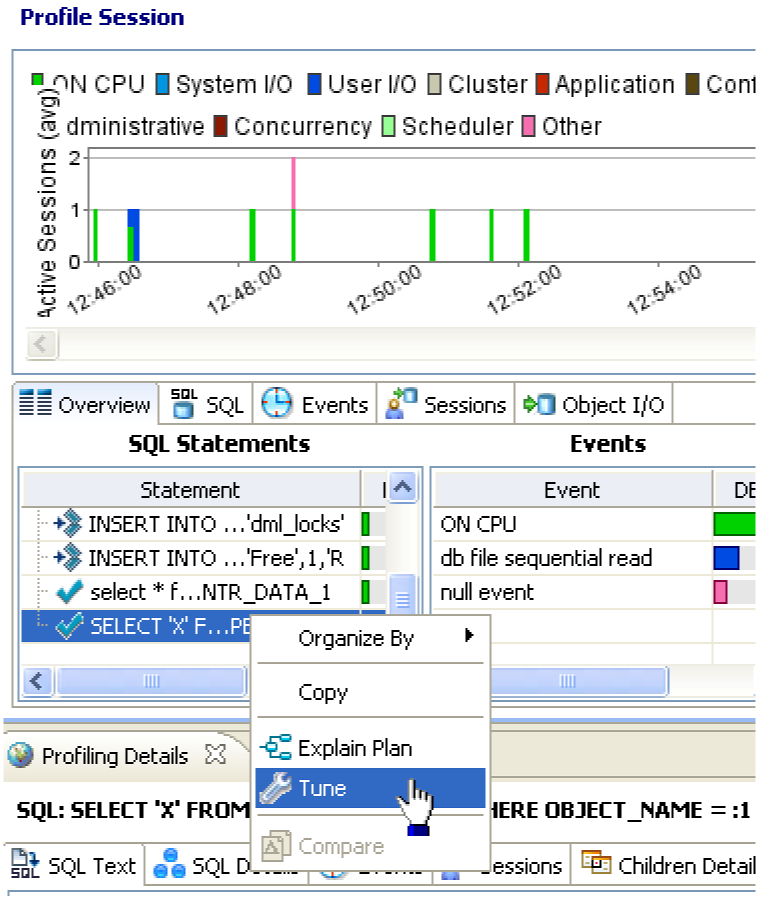
Input SQL statements to be tuned.

```
SELECT
    ct.action,
    c.client_id,
    i.investment_unit,
    it.investment_type_name
FROM
    client_transaction ct,
    client c,
    investment_type it,
    investment i
WHERE
    ct.client_id = c.client_id AND
    ct.investment_id = i.investment_id AND
    i.investment_type_id = it.investment_type_id and
    client_transaction_id=1
```

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.

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.



Statement
INSERT INTO ...'dml_locks'
INSERT INTO ...'Free',1,'R
select * f...NTR_DATA_1
SELECT 'X' F...PE

Event
ON CPU
db file sequential read
null event

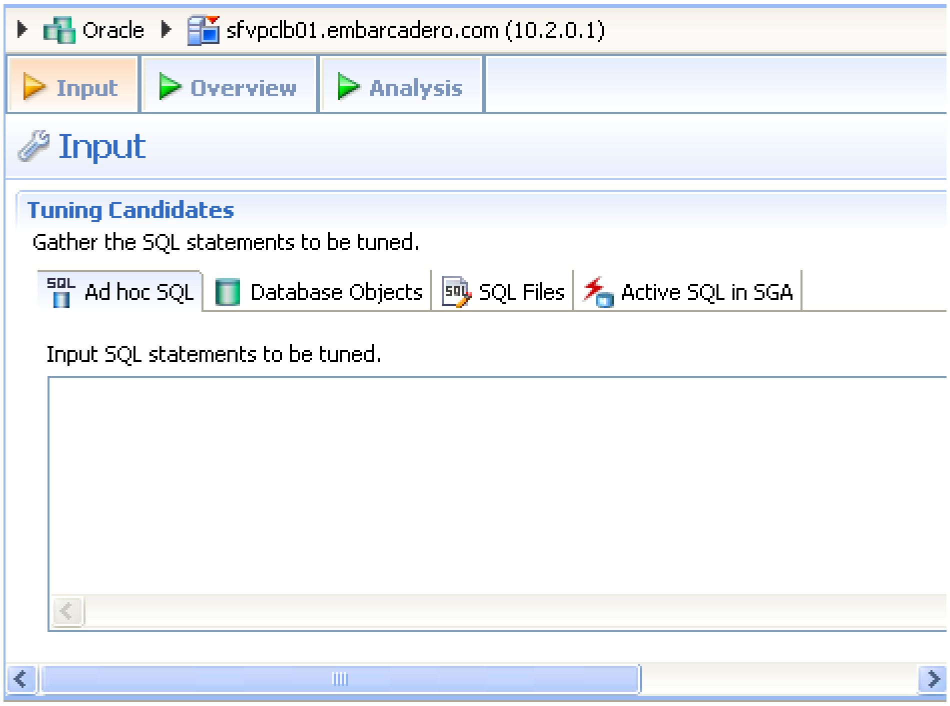
OR

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 have defined 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

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 **Ad hoc SQL** box. All standard DML statements (SELECT, INSERT, DELETE, UPDATE) are viable for the tuning procedure.



Gather the SQL statements to be tuned.

Input SQL statements to be tuned.

Statements are added to a tuning job via the Input tab.

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

Use the **Input tab** to specify which SQL statements to tune.

- **Ad hoc SQL**: Copy/paste SQL statements to the Ad hoc SQL tab or write queries by hand.
- **Database Objects**: Drag and drop data base objects from the Data Source Explorer to the Database Objects tab.
- **SQL Files**: Browse the workspace or file system and select SQL files.
- **Active SQL in SGA**: For the Oracle platform only, you can also scan the System Global Area (SGA) for statements to tune.

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.

To add Active SQL in SGA:

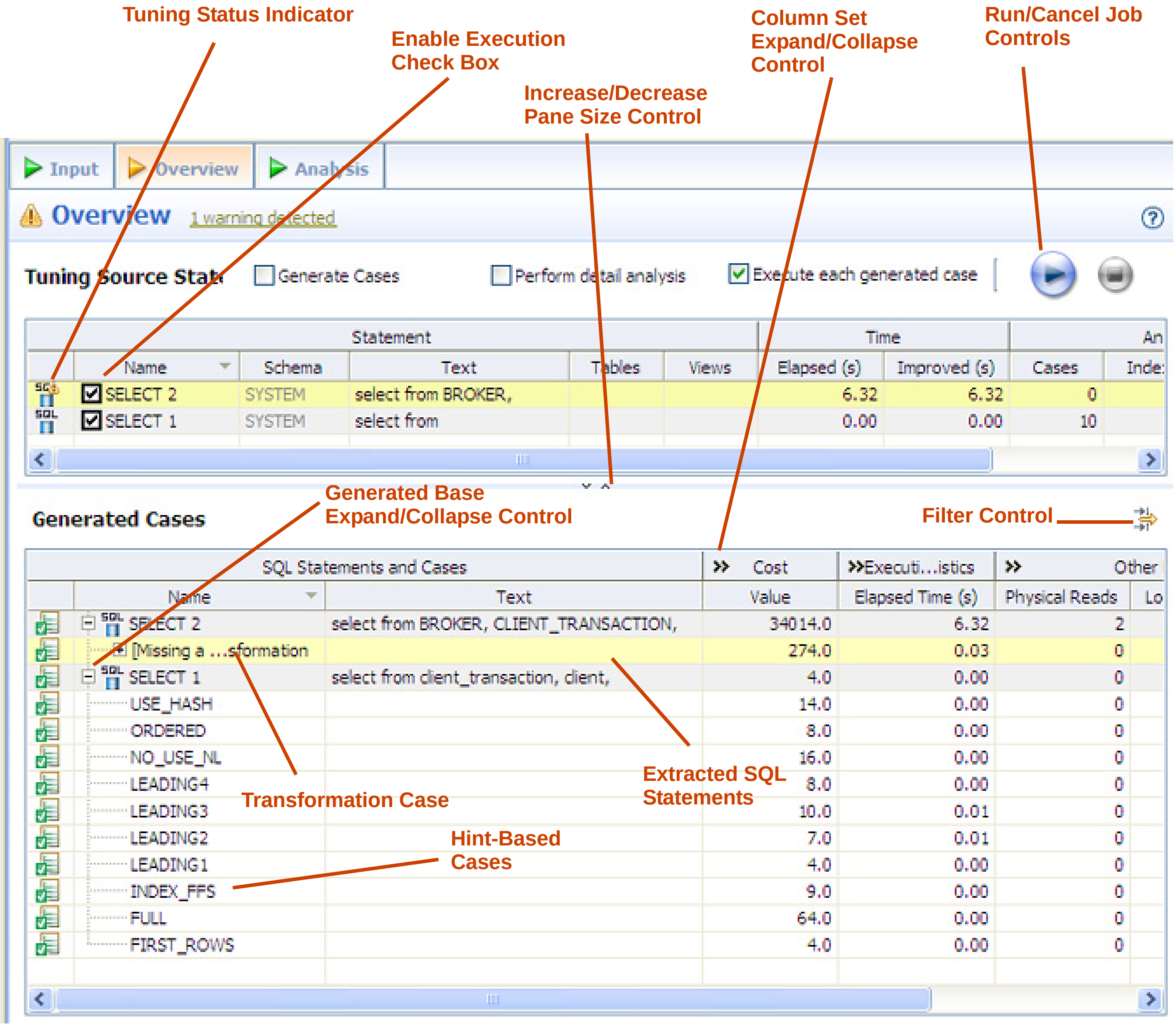
Select the **Active SQL in SGA** tab and click **Scan**. Specify any filters as required and then click **Next**. From the list of SQL statements retrieved from the SGA, select those you want to optimize, and then click **Finish**. The selected statements are copied to the text area of the **Active SQL in SGA** tab.

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.

TIP: You can click the **Text** field of a generated case to view the SQL source of the statement.



Tuning Status Indicator

Enable Execution Check Box

Increase/Decrease Pane Size Control

Column Set Expand/Collapse Control

Run/Cancel Job Controls

Generated Base Expand/Collapse Control

Filter Control

Transformation Case

Extracted SQL Statements

Hint-Based Cases

Tuning Source Stat

□ Generate Cases □ Perform detail analysis ☑ Execute each generated case

	Statement					Time		An
	Name	Schema	Text	Tables	Views	Elapsed (s)	Improved (s)	Cases	Inde:
	☑ SELECT 2	SYSTEM	select from BROKER,			6.32	6.32	0	
	☑ SELECT 1	SYSTEM	select from			0.00	0.00	10	

Generated Cases

	SQL Statements and Cases		>> Cost	>>Executi...istics	>> Other	
	Name	Text	Value	Elapsed Time (s)	Physical Reads	Lo
	⊟ SELECT 2	select from BROKER, CLIENT_TRANSACTION,	34014.0	6.32	2	
	⊞ [Missing a ...sformation		274.0	0.03	0	
	⊟ SELECT 1	select from client_transaction, client,	4.0	0.00	0	
	USE_HASH		14.0	0.00	0	
	ORDERED		8.0	0.00	0	
	NO_USE_NL		16.0	0.00	0	
	LEADING4		8.0	0.00	0	
	LEADING3		10.0	0.01	0	
	LEADING2		7.0	0.01	0	
	LEADING1		4.0	0.00	0	
	INDEX_FFS		9.0	0.00	0	
	FULL		64.0	0.00	0	
	FIRST_ROWS		4.0	0.00	0	

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 **Execute each generated 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:

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
3. *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 tab is performed when you click the Analysis tab.

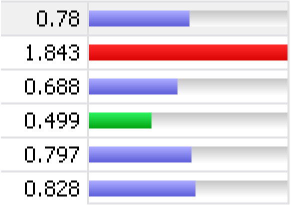
   To have the system generate execution statistics, click **Execute each generate case** and then select the number of time the system should execute each generated case. Multiple executions can verify that the case results are not skewed by caching. For example, the first time a query is run, data might 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.
4. Then 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 be 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.



0.78
1.843
0.688
0.499
0.797
0.828

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.

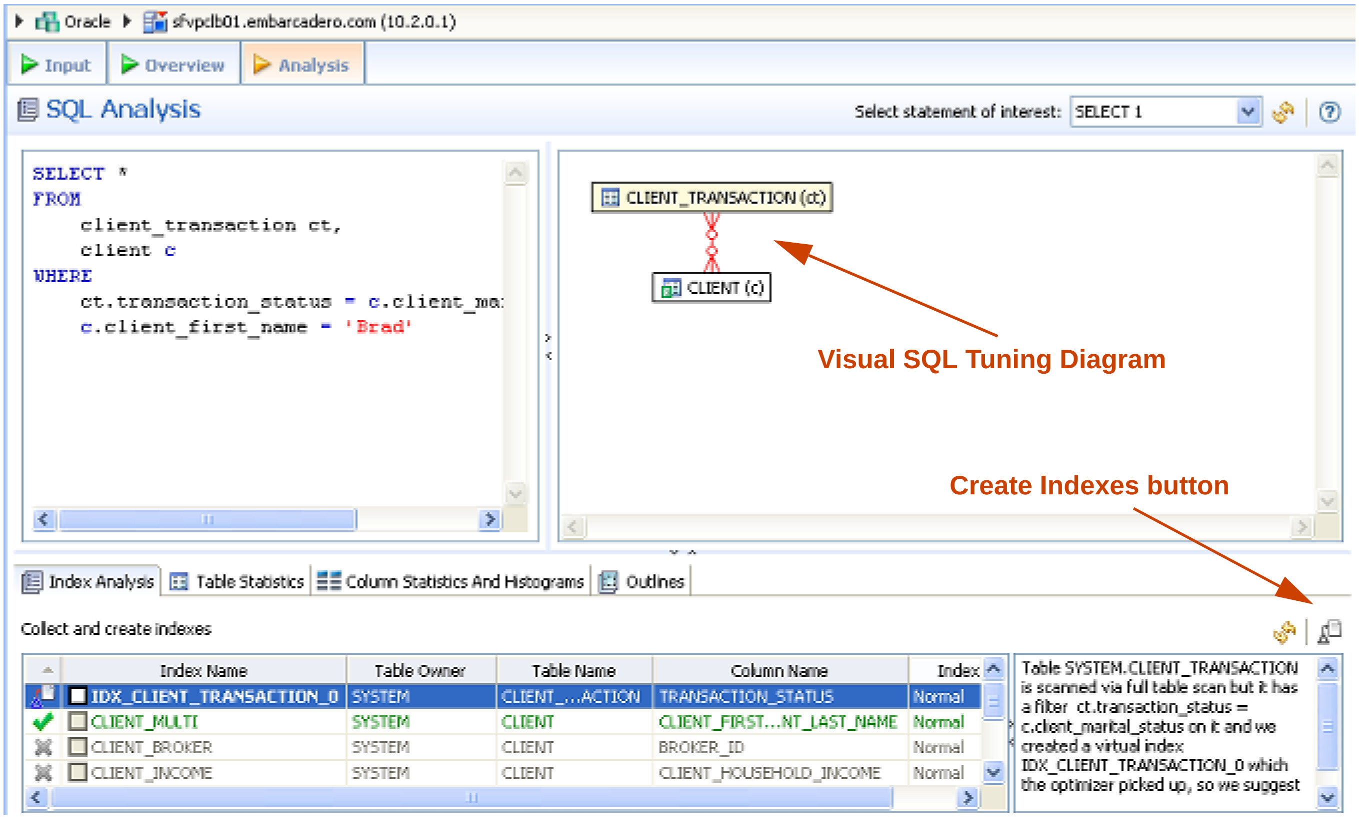
To determine the best cases for statement execution path time:

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.


Oracle
sfvpclb01.embarcadero.com (10.2.0.1)
Input
Overview
Analysis
SQL Analysis
Select statement of interest:
SELECT 1
CLIENT_TRANSACTION (ct)
CLIENT (c)
Index Analysis
Table Statistics
Column Statistics And Histograms
Outlines
Collect and create indexes

```
SELECT *
FROM
    client_transaction ct,
    client c
WHERE
    ct.transaction_status = c.client_ma
    c.client_first_name = 'Brad'
```

Visual SQL Tuning Diagram

Create Indexes button

	Index Name	Table Owner	Table Name	Column Name	Index
	□ IDX_CLIENT_TRANSACTION_0	SYSTEM	CLIENT_...ACTION	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

Table SYSTEM.CLIENT_TRANSACTION is scanned via full table scan but it has a filter ct.transaction_status = c.client_marital_status on it and we created a virtual index IDX_CLIENT_TRANSACTION_0 which the optimizer picked up, so we suggest

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, which can be displayed in either Summary Mode or Detail Mode, 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.

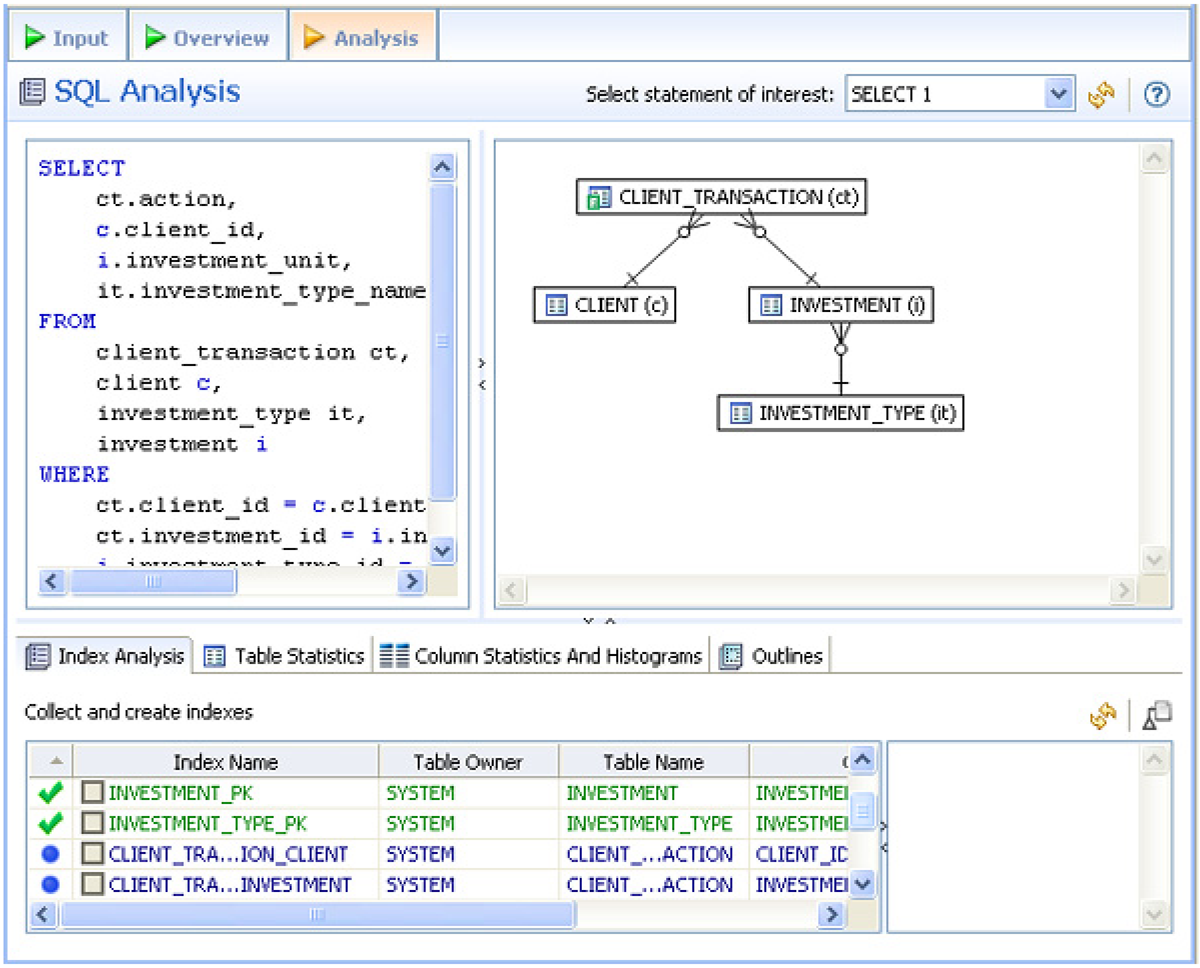
This section is comprised of the following topics:

- Finding Missing Indexes
- Finding Missing Indexes
- Interpreting the VST Diagram Graphics Conventions
- Interpreting the VST Diagram Graphics Conventions
- Implementing Recommendations on the Overview Tab

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.

> **TIP:** Indexes that are used are color coded **green**. Indexes that are present but not used in this execution path are color coded **grey**.



```
SELECT
     ct.action,
     c.client_id,
     i.investment_unit,
     it.investment_type_name
FROM
     client_transaction ct,
     client c,
     investment_type it,
     investment i
WHERE
     ct.client_id = c.client
     ct.investment_id = i.in
```

Collect and create indexes

	Index Name	Table Owner	Table Name	
✓	□ INVESTMENT_PK	SYSTEM	INVESTMENT	INVESTMEI
✓	□ INVESTMENT_TYPE_PK	SYSTEM	INVESTMENT_TYPE	INVESTMEI
●	□ CLIENT_TRA...ION_CLIENT	SYSTEM	CLIENT_...ACTION	CLIENT_ID
●	□ CLIENT_TRA...INVESTMENT	SYSTEM	CLIENT_...ACTION	INVESTMEI

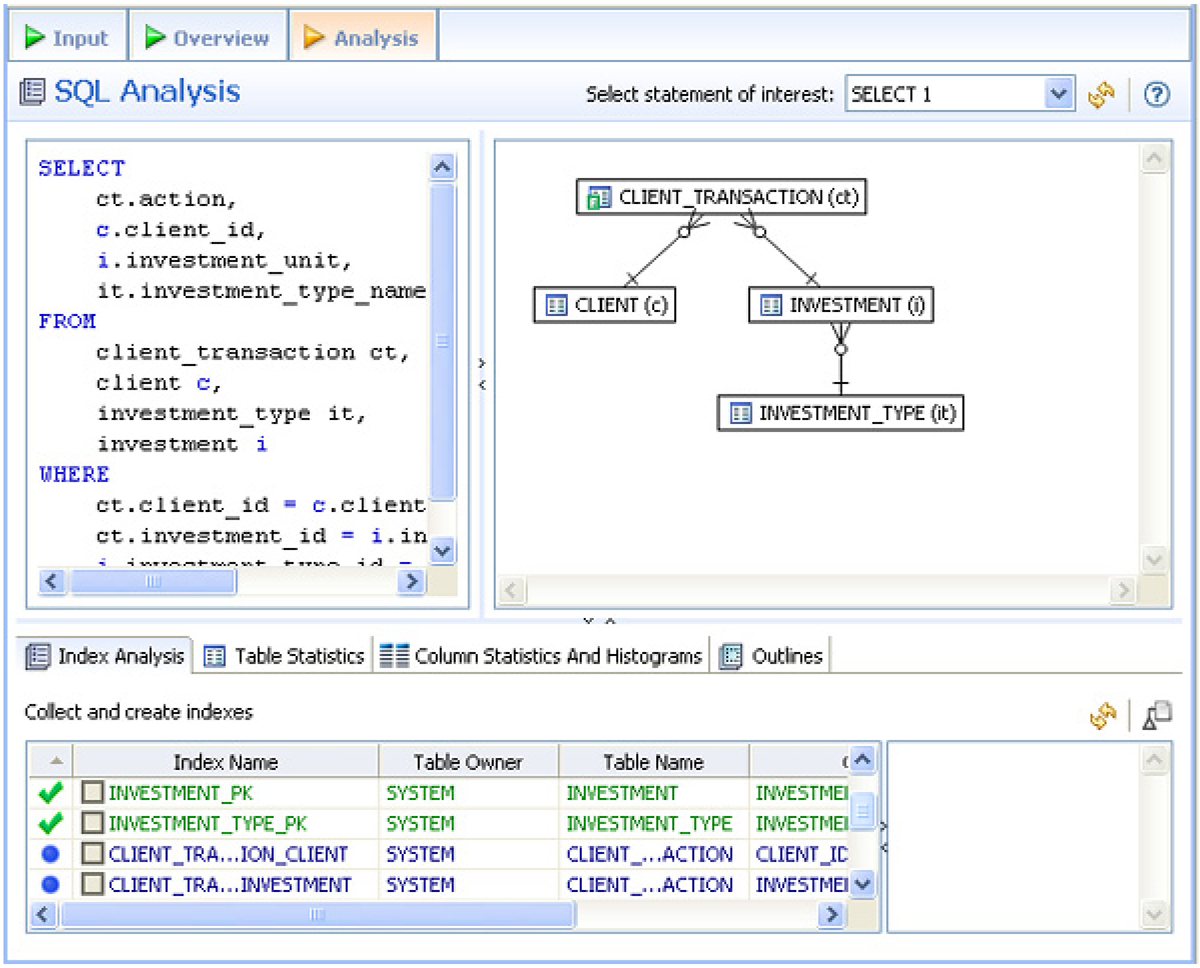
Changing Diagram Detail Display

This section is comprised of the following topics:

- Viewing the VST Diagram in Summary Mode
- Viewing the VST Diagram in Detail Mode
- Changing Detail Level for a Specific Table
- Viewing All Table Fields
- Viewing Diagram Object SQL
- Expanding Views in the VST Diagram

Viewing the VST Diagram in Summary Mode

By default the diagram displays Summary Mode, showing only table names and joins, as seen in the following illustration.

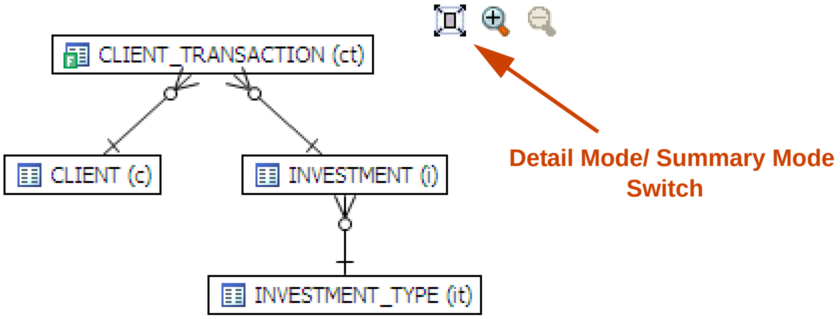


```
SELECT
    ct.action,
    c.client_id,
    i.investment_unit,
    it.investment_type_name
FROM
    client_transaction ct,
    client c,
    investment_type it,
    investment i
WHERE
    ct.client_id = c.client
    ct.investment_id = i.in
    i.investment_type_id =
```

	Index Name	Table Owner	Table Name	(
✓	□ INVESTMENT_PK	SYSTEM	INVESTMENT	INVESTMEI
✓	□ INVESTMENT_TYPE_PK	SYSTEM	INVESTMENT_TYPE	INVESTMEI
●	□ CLIENT_TRA...ION_CLIENT	SYSTEM	CLIENT_...ACTION	CLIENT_ID
●	□ CLIENT_TRA...INVESTMENT	SYSTEM	CLIENT_...ACTION	INVESTMEI

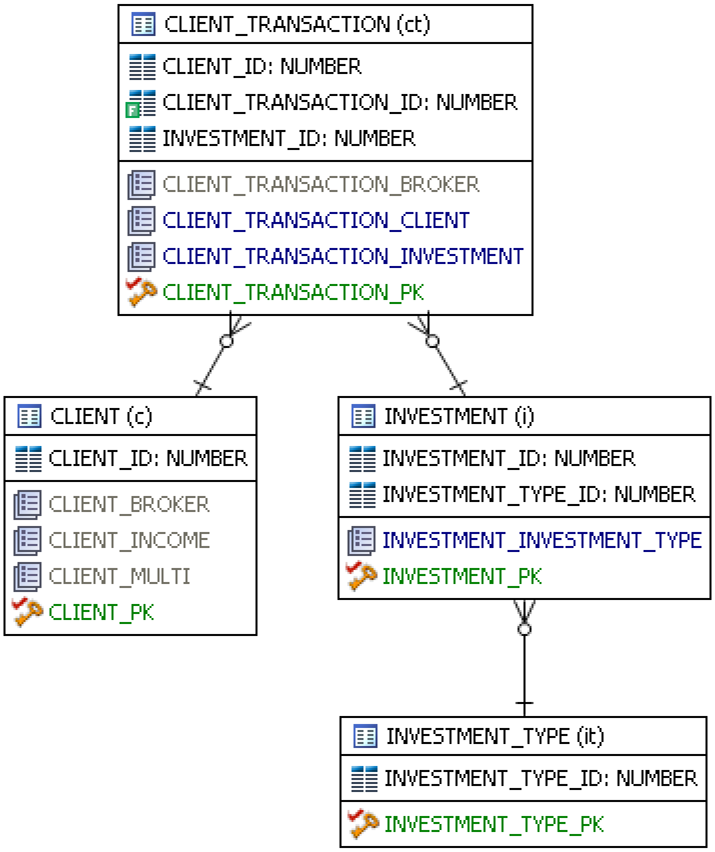
Viewing the VST Diagram in Detail Mode

By default, the VST diagram displays in Summary Mode, but by clicking the **Detail Mode/Summary Mode** switch



Detail Mode/ Summary Mode Switch

additional details of the tables display, including table columns and indexes:

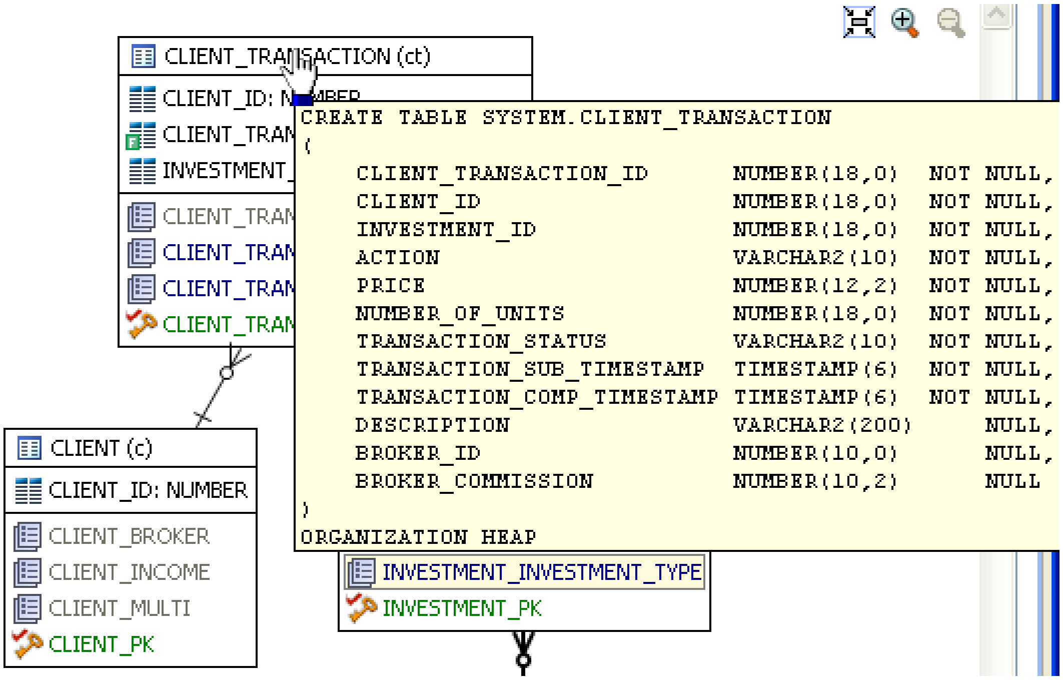


Changing Detail Level for a Specific Table

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

Viewing All Table Fields

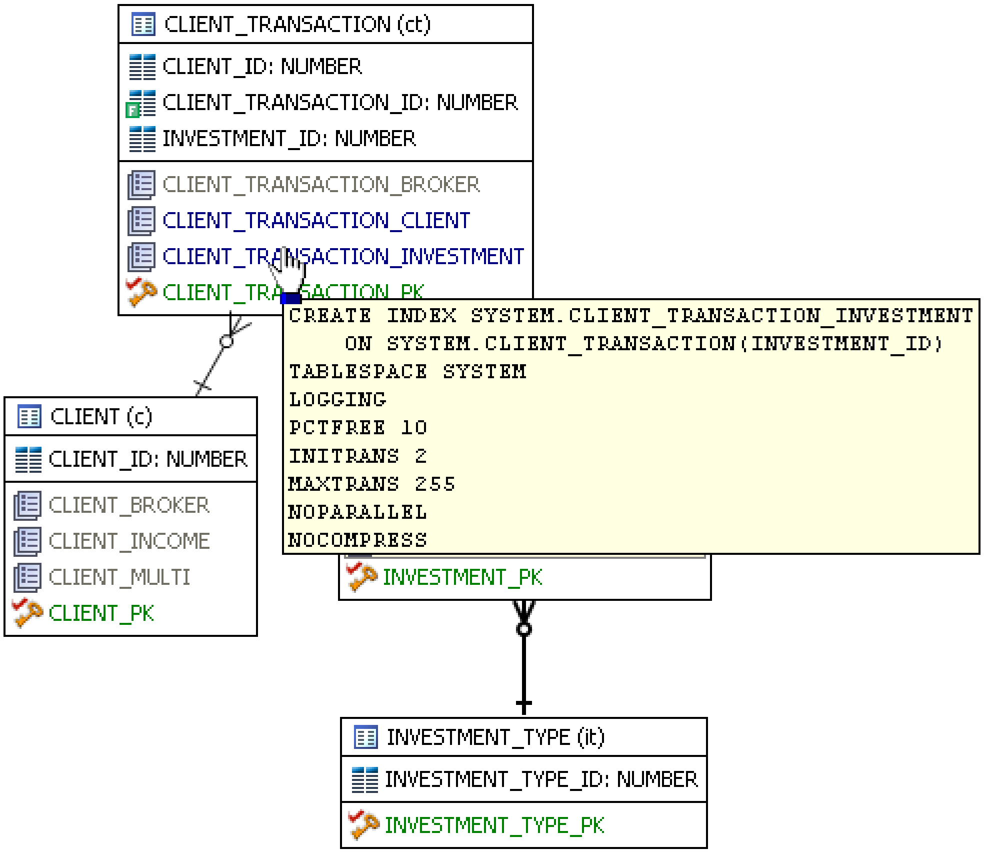
Only fields that are used in the WHERE clause are displayed in detail mode; however, all fields in the table can be seen in a pop-up window when you hover the mouse over the table name. The illustration below shows the pop-up window that appears when hovering over the CLIENT_TRANSACTION (ct) table .



```
CREATE TABLE SYSTEM.CLIENT_TRANSACTION
(
    CLIENT_TRANSACTION_ID       NUMBER(18,0)  NOT NULL,
    CLIENT_ID                   NUMBER(18,0)  NOT NULL,
    INVESTMENT_ID               NUMBER(18,0)  NOT NULL,
    ACTION                      VARCHAR2(10)  NOT NULL,
    PRICE                       NUMBER(12,2)  NOT NULL,
    NUMBER_OF_UNITS             NUMBER(18,0)  NOT NULL,
    TRANSACTION_STATUS          VARCHAR2(10)  NOT NULL,
    TRANSACTION_SUB_TIMESTAMP   TIMESTAMP(6)  NOT NULL,
    TRANSACTION_COMP_TIMESTAMP  TIMESTAMP(6)  NOT NULL,
    DESCRIPTION                 VARCHAR2(200)     NULL,
    BROKER_ID                   NUMBER(10,0)      NULL,
    BROKER_COMMISSION           NUMBER(10,2)      NULL
)
ORGANIZATION HEAP
```

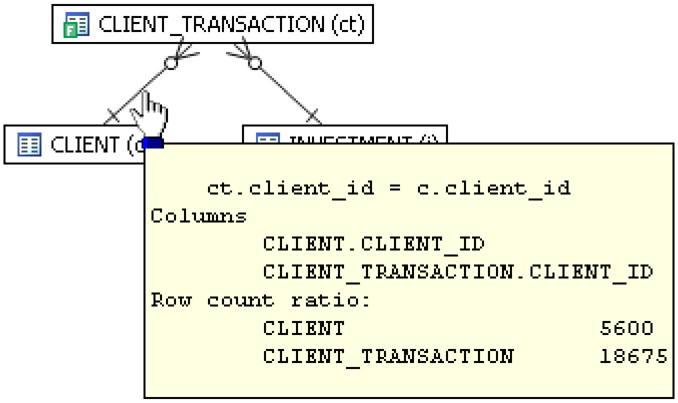
Viewing Diagram Object SQL

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



```
CREATE INDEX SYSTEM.CLIENT_TRANSACTION_INVESTMENT
    ON SYSTEM.CLIENT_TRANSACTION(INVESTMENT_ID)
TABLESPACE SYSTEM
LOGGING
PCTFREE 10
INITRANS 2
MAXTRANS 255
NOPARALLEL
NOCOMPRESS
```

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

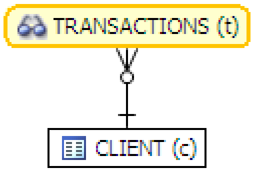


```
    ct.client_id = c.client_id
Columns
        CLIENT.CLIENT_ID
        CLIENT_TRANSACTION.CLIENT_ID
Row count ratio:
        CLIENT                    5600
        CLIENT_TRANSACTION        18675
```

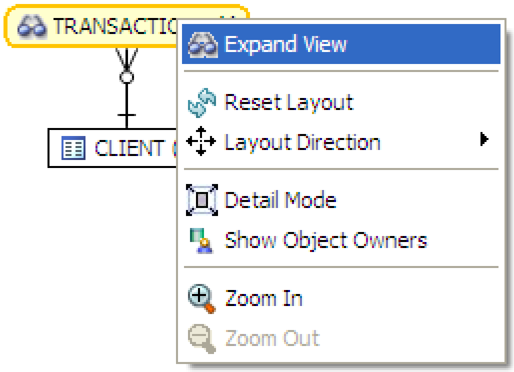
Expanding Views in the VST Diagram

If there are views in the Visual SQL Tuning diagram, they can be expanded by right clicking the view name and choosing **Expand View**:

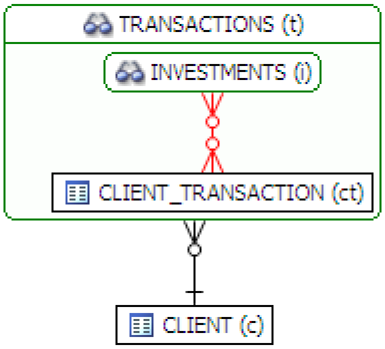
For example, the following is the default layout from query join table CLIENT (c) to view TRANSACTIONS (t):



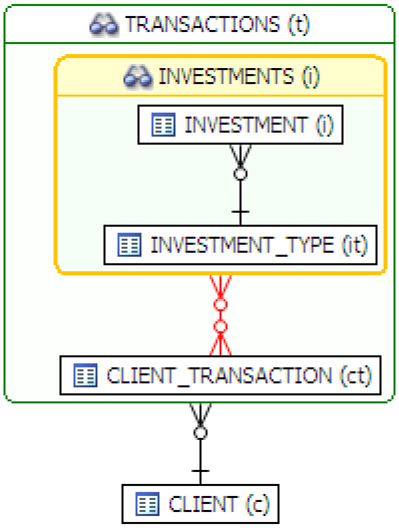
Right click on the view, TRANSACTION (t) and choose **Expand View**



Now we can see the objects in the view:

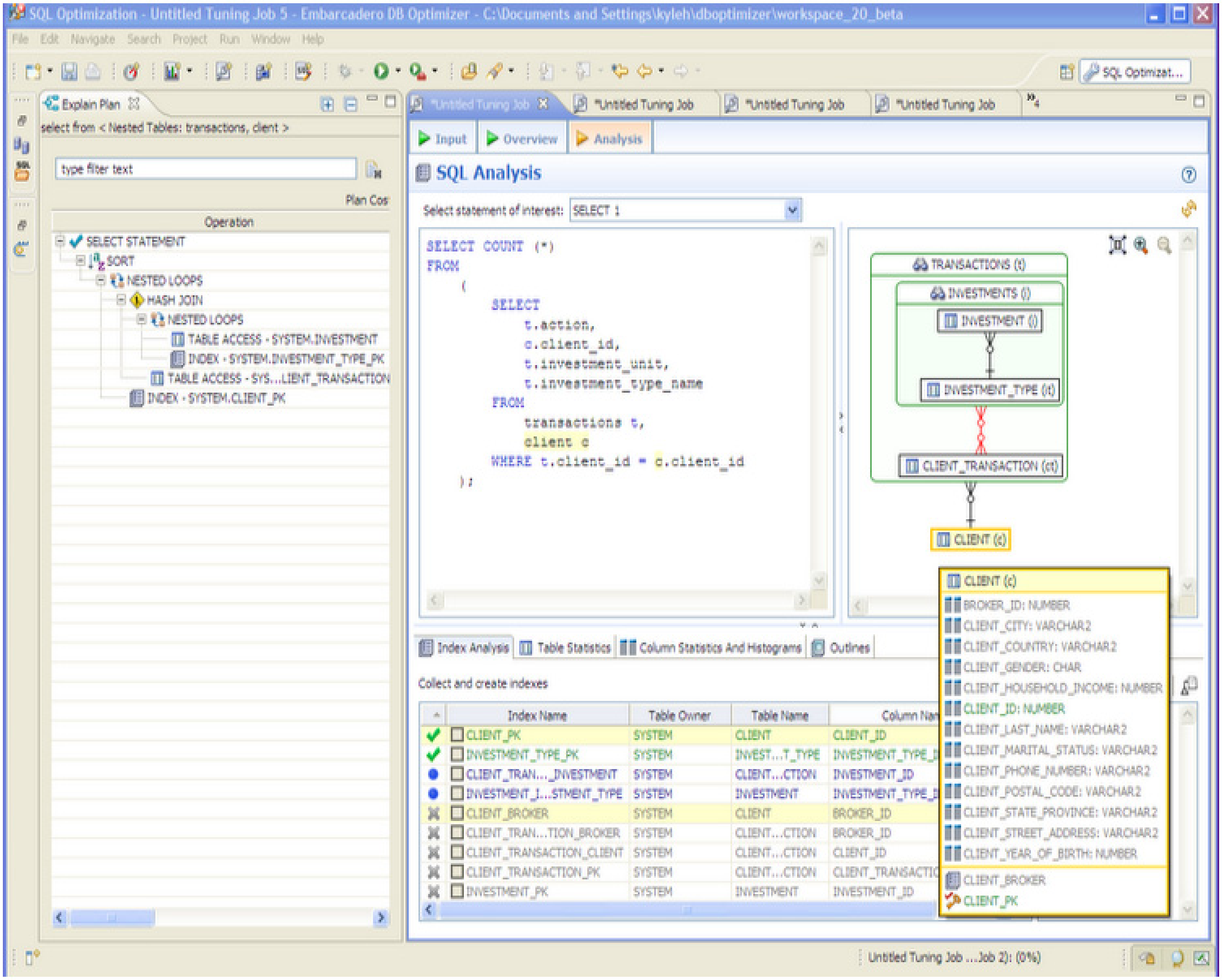


We can further expand the sub-view within the original view:



The following is an example of view expansion along with the Explain Plan to the left.

Notice in the view expansion a list of all the indexes on all the underlying tables in the views and sub views and which of those indexes is used in the default execution plan.


SQL Optimization - Untitled Tuning Job 5 - Embarcadero DB Optimizer - C:\Documents and Settings\kyleh\dboptimizer\workspace_20_beta
File Edit Navigate Search Project Run Window Help
SQL Optimizat...
Explain Plan
select from < Nested Tables: transactions, client >
type filter text
Plan Cos
Operation
SELECT STATEMENT
SORT
NESTED LOOPS
HASH JOIN
NESTED LOOPS
TABLE ACCESS - SYSTEM.INVESTMENT
INDEX - SYSTEM.INVESTMENT_TYPE_PK
TABLE ACCESS - SYS...LIENT_TRANSACTION
INDEX - SYSTEM.CLIENT_PK
*Untitled Tuning Job
*Untitled Tuning Job
*Untitled Tuning Job
*Untitled Tuning Job
Input
Overview
Analysis
SQL Analysis
Select statement of interest:
SELECT 1
TRANSACTIONS (t)
INVESTMENTS ()
INVESTMENT ()
INVESTMENT_TYPE (it)
CLIENT_TRANSACTION (ct)
CLIENT (c)
Index Analysis
Table Statistics
Column Statistics And Histograms
Outlines
Collect and create indexes
Untitled Tuning Job ...Job 2): (0%)

```
SELECT COUNT (*)
FROM
    (
        SELECT
            t.action,
            c.client_id,
            t.investment_unit,
            t.investment_type_name
        FROM
            transactions t,
            client c
        WHERE t.client_id = c.client_id
    );
```

	Index Name	Table Owner	Table Name	Column Nam
✓	□ CLIENT_PK	SYSTEM	CLIENT	CLIENT_ID
✓	□ INVESTMENT_TYPE_PK	SYSTEM	INVEST...T_TYPE	INVESTMENT_TYPE_I
●	□ CLIENT_TRAN..._INVESTMENT	SYSTEM	CLIENT...CTION	INVESTMENT_ID
●	□ INVESTMENT_I...STMENT_TYPE	SYSTEM	INVESTMENT	INVESTMENT_TYPE_I
✗	□ CLIENT_BROKER	SYSTEM	CLIENT	BROKER_ID
✗	□ CLIENT_TRAN...TION_BROKER	SYSTEM	CLIENT...CTION	BROKER_ID
✗	□ CLIENT_TRANSACTION_CLIENT	SYSTEM	CLIENT...CTION	CLIENT_ID
✗	□ CLIENT_TRANSACTION_PK	SYSTEM	CLIENT...CTION	CLIENT_TRANSACTIO
✗	□ INVESTMENT_PK	SYSTEM	INVESTMENT	INVESTMENT_ID

CLIENT (c)
BROKER_ID: NUMBER
CLIENT_CITY: VARCHAR2
CLIENT_COUNTRY: VARCHAR2
CLIENT_GENDER: CHAR
CLIENT_HOUSEHOLD_INCOME: NUMBER
CLIENT_ID: NUMBER
CLIENT_LAST_NAME: VARCHAR2
CLIENT_MARITAL_STATUS: VARCHAR2
CLIENT_PHONE_NUMBER: VARCHAR2
CLIENT_POSTAL_CODE: VARCHAR2
CLIENT_STATE_PROVINCE: VARCHAR2
CLIENT_STREET_ADDRESS: VARCHAR2
CLIENT_YEAR_OF_BIRTH: NUMBER
CLIENT_BROKER
CLIENT_PK

Interpreting the VST Diagram Graphics Conventions

This section will help you understand the following graphic usages:

- Icons
- Colors
- Connecting Lines/Joins

Icons

The following describes the icons used in tables displayed in Detail Mode.

Table Icon	Description
▦	Table Name
☰	Field
F☰	Field with a filter, used in the WHERE clause
▤	Index
🔑	Primary Key

Colors

The color of the index entries in the **Collect and Create Indexes** table is interpreted as follows:

Text Color	Interpretation
🟩	Index is used in the query.
🟦	Index is usable but not used by the current execution path.
🟧	This index is missing. DB Optimizer recommends that you create this index.
⬛	This index exists on the table but not usable in this query as it is written.

Connecting Lines/Joins

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

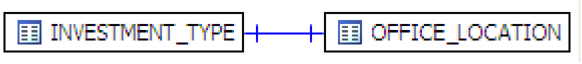
Connecting Lines	When used
┼──┼	One-to-One Join relationships are graphed horizontally using blue lines.
>o──┼	One-to-Many Join relationships are graphed with the many table above the one table.
▦ INVESTMENT	Cartesian Join shows the table highlighted in red with no connectors to indicate that it is joined in via a Cartesian join.
>o──o<	Many-to-Many Join relationships are connected by a red line and the relative location is not restricted.

One-to-One Join

If two tables are joined on their primary key, such as:

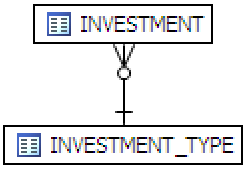
```
SELECT COUNT (*)
        FROM
            investment_type it,
            office_location ol
        WHERE investment_type_id = office_location_id;
```

Then graphically, these would be laid out side-by-side, with a one-to-one connector:



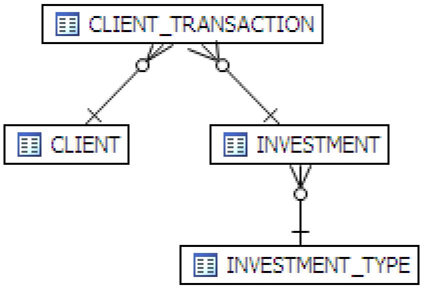
One-to-Many Join

This is the default positioning of a one-to-many relationship, where INVESTMENT_TYPE is the master table and INVESTMENT is the details table.



The following is an example of a query that consists of only many-to-one joins, which is more typical:

```
SELECT
        ct.action,
        c.client_id,
        i.investment_unit,
        it.investment_type_name
    FROM
        client_transaction ct,
        client c,
        investment_type it,
        investment i
    WHERE
        ct.client_id = c.client_id AND
        ct.investment_id = i.investment_id AND
        i.investment_type_id = it.investment_type_id and
        client_transaction_id=1
```

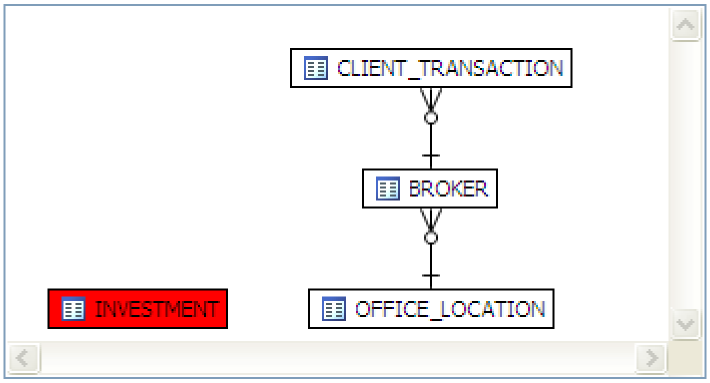


Cartesian Join

A Cartesian join is described in the following example where the query is missing join criteria on the table INVESTMENT:

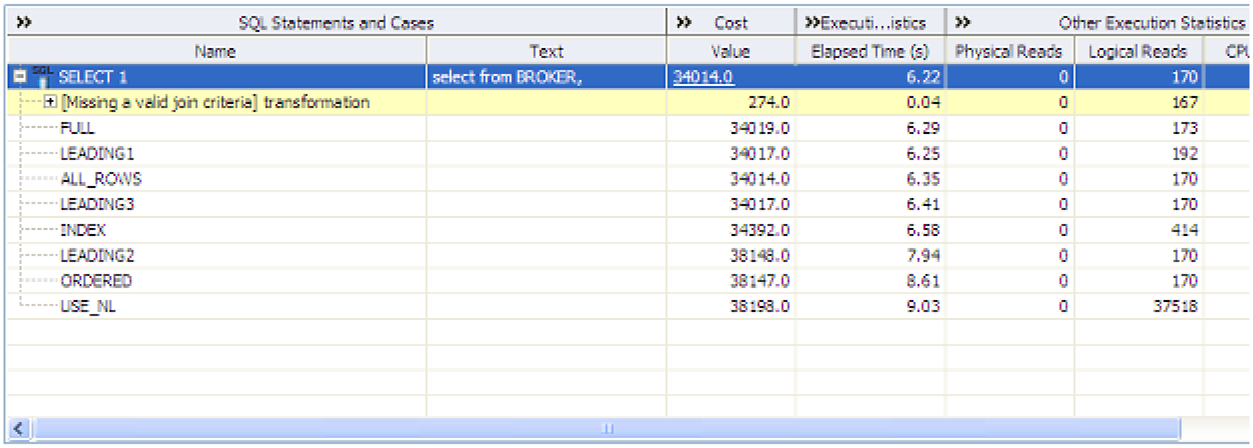
```
SELECT
        A.BROKER_ID BROKER_ID,
        A.BROKER_LAST_NAME BROKER_LAST_NAME,
        A.BROKER_FIRST_NAME BROKER_FIRST_NAME,
        A.YEARS_WITH_FIRM YEARS_WITH_FIRM,
        C.OFFICE_NAME OFFICE_NAME,
        SUM (B.BROKER_COMMISSION) TOTAL_COMMISSIONS
    FROM
        BROKER A,
        CLIENT_TRANSACTION B,
        OFFICE_LOCATION C,
        INVESTMENT I
    WHERE
        A.BROKER_ID = B.BROKER_ID AND
        A.OFFICE_LOCATION_ID = C.OFFICE_LOCATION_ID
    GROUP BY
        A.BROKER_ID,
        A.BROKER_LAST_NAME,
        A.BROKER_FIRST_NAME,
        A.YEARS_WITH_FIRM,
        C.OFFICE_NAME;
```

Graphically, this looks like:



INVESTMENT is highlighted in red with no connectors to indicate that it is joined in via a Cartesian join.

Possible missing join conditions are displayed in the **Overview** tab under **Generated Cases** in the transformations area. DB Optimize recommends that you create these joins.



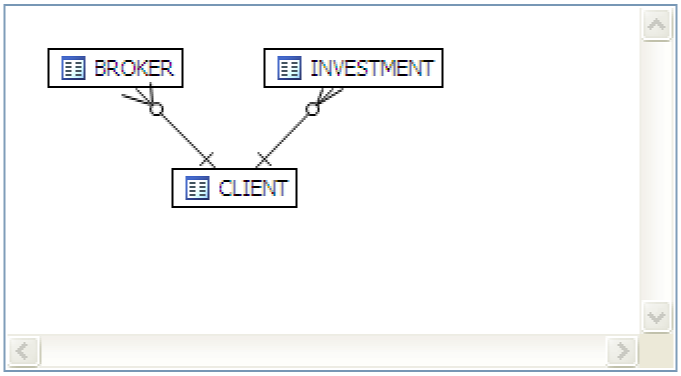
>> SQL Statements and Cases		>> Cost	>>Executi...istics	>> Other Execution Statistics		
Name	Text	Value	Elapsed Time (s)	Physical Reads	Logical Reads	CP
SELECT 1	select from BROKER,	34014.0	6.22	0	170	
[Missing a valid join criteria] transformation		274.0	0.04	0	167	
FULL		34019.0	6.29	0	173	
LEADING1		34017.0	6.25	0	192	
ALL_ROWS		34014.0	6.35	0	170	
LEADING3		34017.0	6.41	0	170	
INDEX		34392.0	6.58	0	414	
LEADING2		38148.0	7.94	0	170	
ORDERED		38147.0	8.61	0	170	
USE_NL		38198.0	9.03	0	37518	

NOTE: Transformations are highlighted in yellow.

Implied Cartesian Join

If there are different details for a master without other criteria then a Cartesian-type join is created:

```
SELECT *
    FROM
        investment i,
        broker b,
        client c
    WHERE
        b.manager_id=c.client_id and
        i.investment_type_id=c.client_id;
```



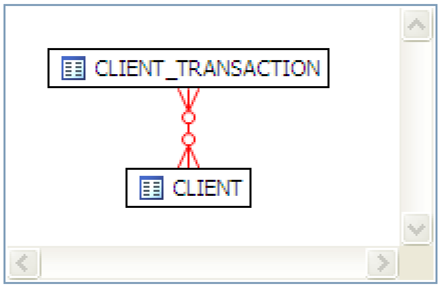
The result set of BROKER to CLIENT will be multiplied by the result set of INVESTMENT to CLIENT.

Many-to-Many Join

If there is no unique index at either end of a join then it can be assumed that in some or all cases the join is many-to-many; there are no constraints preventing a many-to-many join. For example, examine the following query:

```
SELECT *
    FROM
        client_transaction ct,
        client c
    WHERE
        ct.transaction_status=c.client_marital_status;
```

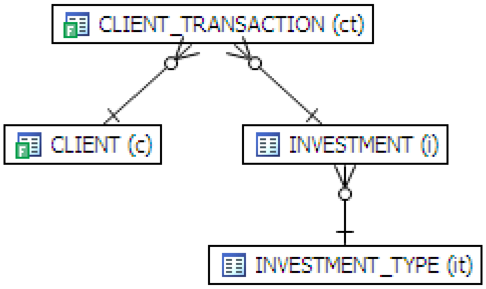
There is no unique index on either of the fields being joined so the optimizer assumes this is a many-to-many join and the relationship is displayed graphically as:



If one of the fields is unique, then the index should be declared as such to help the optimizer.

Finding Problematic SQL or Schema

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

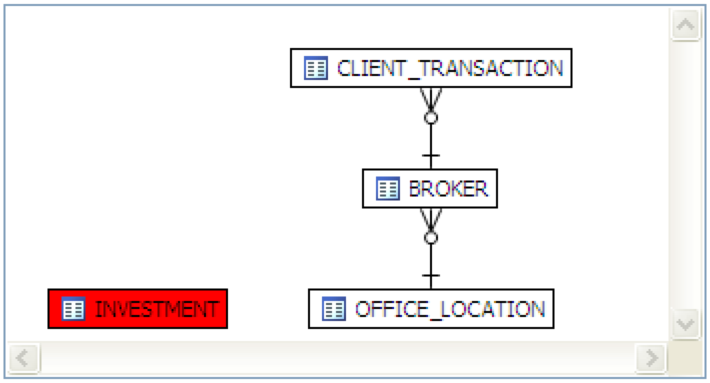


Problems such as Cartesian joins, implied cartesian joins, and many-to-many relationship are clearly represented in the VST. The following describes these potential query problems.

- Cartesian Join
- Implied Cartesian Join
- Many-to-Many Relationships

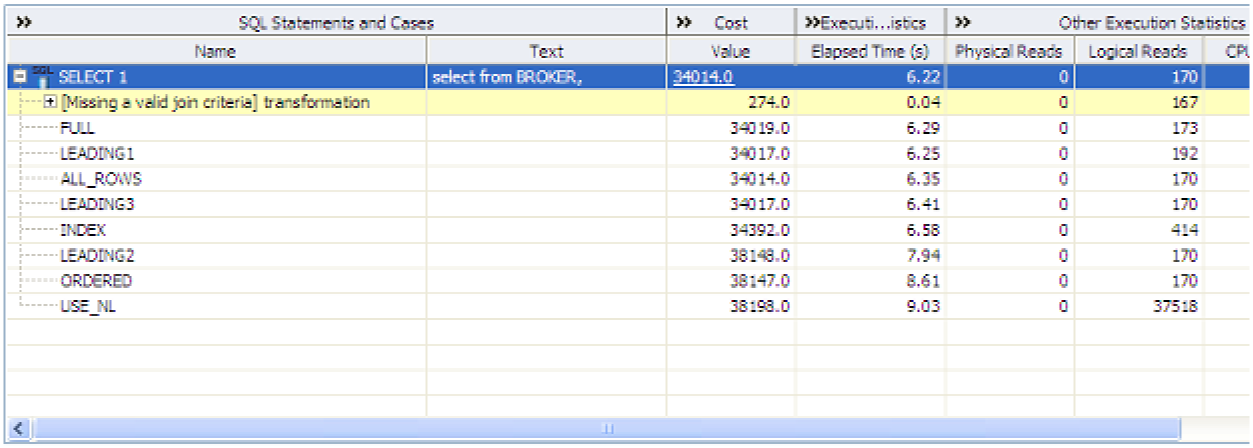
Cartesian Join

The diagram on the Analysis tab can help identify SQL problems such as missing or cartesian joins. For example, you might see a table or view marked in red as in the following diagram.



A red table or view indicates that here is Cartesian join that could be resolved by implementing a rewrite suggestion shown in the **Generated Cases** area of the **Overview** tab.

Possible missing join conditions are displayed in the **Overview** tab under **Generated Cases** in the transformations area. DB Optimize recommends that you create these joins.



SQL Statements and Cases		Cost	Executi...istics	Other Execution Statistics		
Name	Text	Value	Elapsed Time (s)	Physical Reads	Logical Reads	CPU
SELECT 1	select from BROKER,	34014.0	6.22	0	170	
[Missing a valid join criteria] transformation		274.0	0.04	0	167	
FULL		34019.0	6.29	0	173	
LEADING1		34017.0	6.25	0	192	
ALL_ROWS		34014.0	6.35	0	170	
LEADING3		34017.0	6.41	0	170	
INDEX		34392.0	6.58	0	414	
LEADING2		38148.0	7.94	0	170	
ORDERED		38147.0	8.61	0	170	
USE_NL		38198.0	9.03	0	37518	

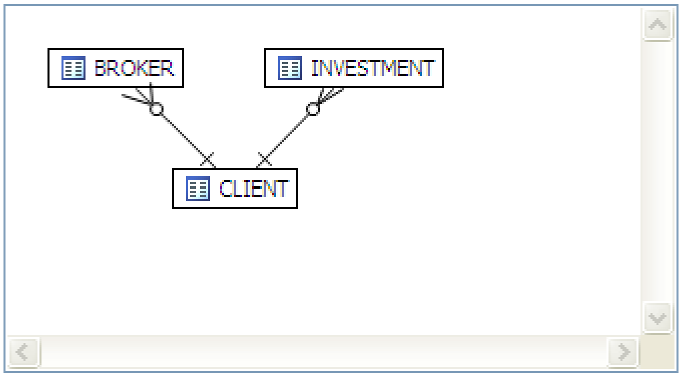
NOTE: Transformations are highlighted in yellow.

Implied Cartesian Join

In a well-formed query, there should only be at most one detail table above any master table.The following diagram shows an implied Cartesian join, here we have two details tables above the BROKER table. When there can be more than one row that satisfies the join between the first join CLIENT and BROKER and the second join BROKER and CLIENT_TRANSACTION, DB Optimizer presents an implied Cartesian Join. This could signify a flaw in the query or a flaw in the schema design.

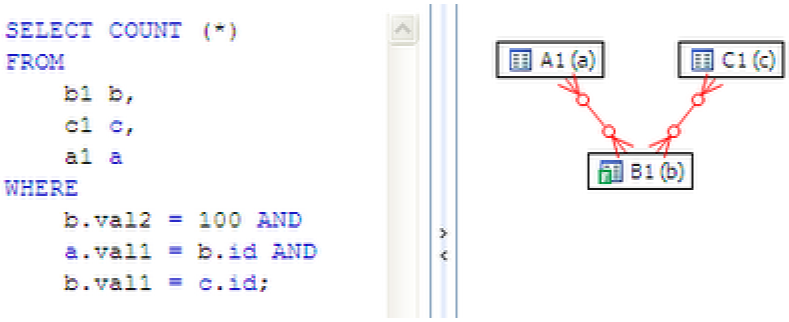
In the following case, the result set of BROKER to CLIENT will be multiplied by the result set of INVESTMENT to CLIENT.

```
SELECT *
FROM
    investment i,
    broker b,
    client c
WHERE
    b.manager_id=c.client_id AND
       i.investment_type_id=c.client_id;
```



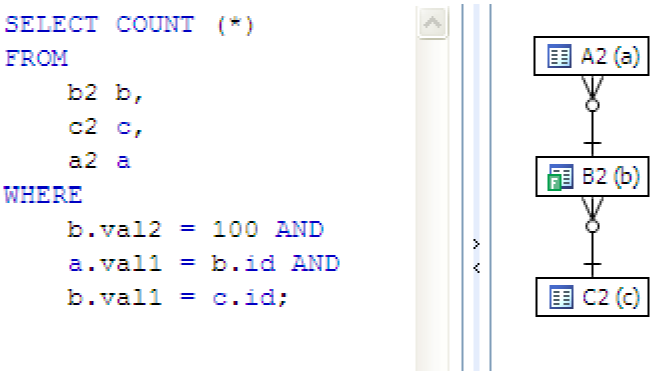
Many-to-Many Relationships

If master detail information is missing then the VST diagram will have many-to-many connectors:



```
SELECT COUNT (*)
FROM
    b1 b,
    c1 c,
    a1 a
WHERE
    b.val2 = 100 AND
    a.val1 = b.id AND
    b.val1 = c.id;
```

The same set of data with properly defined unique indexes or primary key and foreign key definitions would look like:



```
SELECT COUNT (*)
FROM
    b2 b,
    c2 c,
    a2 a
WHERE
    b.val2 = 100 AND
    a.val1 = b.id AND
    b.val1 = c.id;
```

The optimizer can more consistently optimize a well-formed query, so the query will run faster.

Applying Tuner Results to the Data Source

Once Tuner has generated cases and statement results and analyzed the indexes, you can apply the suggested changes to the data source from the **Generated Cases** area of the **Overview** tab. You can create any recommended indexes from the **Index Analysis** area of the **Analysis** tab.

Implementing Recommendations on the Overview Tab

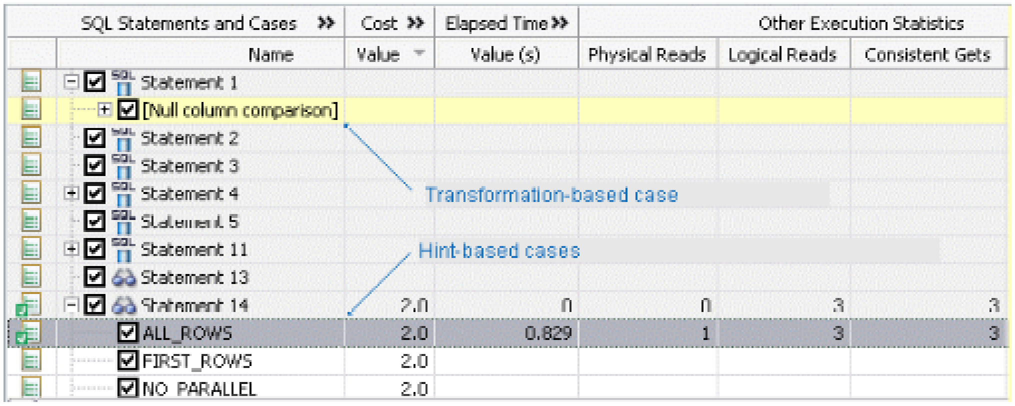
To change an SQL statement based on a transformation or hint-based case:

1. In the **Generated Cases** area, right-click the **Name** field of the case you want to modify the original statement with and choose **Apply Change**.

   The **Apply Change** dialog appears.

2. Choose **Execute** to apply the change to the statement automatically.

   TIP: Alternatively, you can select **Open in New SQL Editor** to make manual changes or save it to a file.



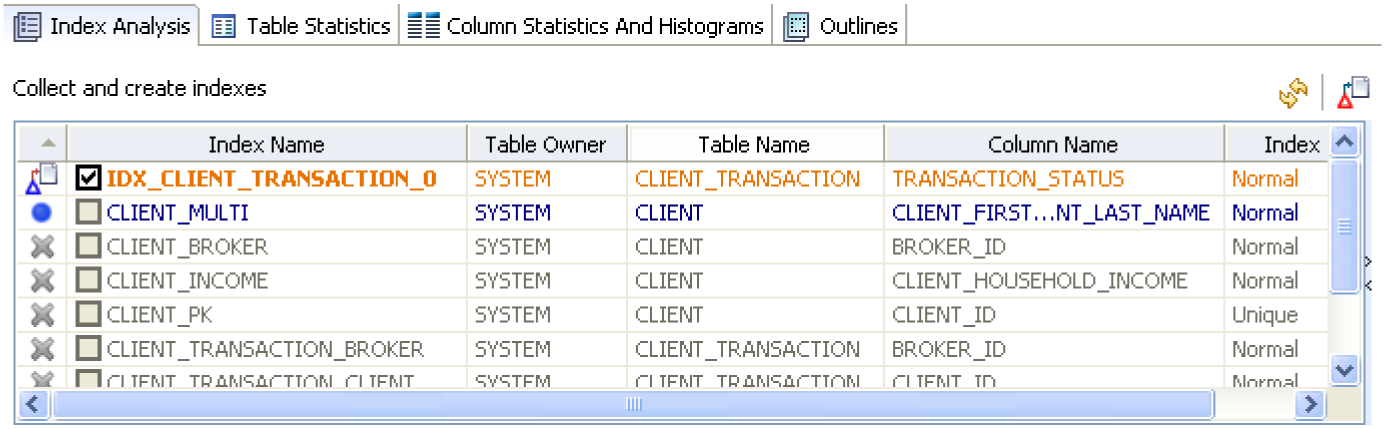
SQL Statements and Cases	Cost	Elapsed Time	Other Execution Statistics		
Name	Value	Value (s)	Physical Reads	Logical Reads	Consistent Gets
☑ Statement 1					
☑ [Null column comparison]					
☑ Statement 2					
☑ Statement 3					
☑ Statement 4					
☑ Statement 5					
☑ Statement 11					
☑ Statement 13					
☑ Statement 14	2.0	0	0	3	3
☑ ALL_ROWS	2.0	0.829	1	3	3
☑ FIRST_ROWS	2.0				
☑ 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.

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, any indexes DB Optimizer recommends you create are marked in orange and have the little create index.

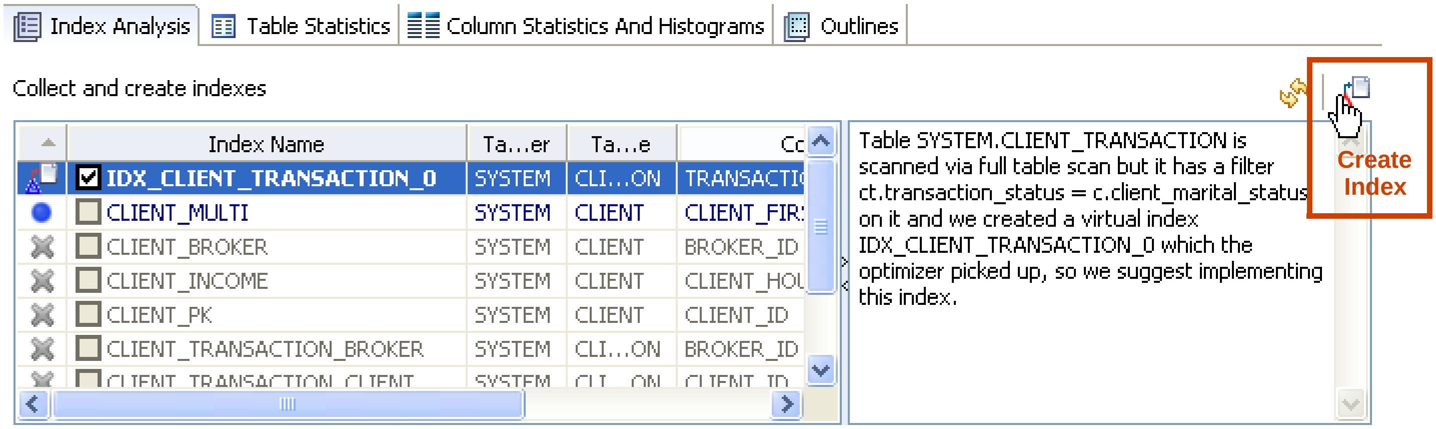


	Index Name	Table Owner	Table Name	Column Name	Index
	☑ IDX_CLIENT_TRANSACTION_0	SYSTEM	CLIENT_TRANSACTION	TRANSACTION_STATUS	Normal
●	☐ CLIENT_MULTI	SYSTEM	CLIENT	CLIENT_FIRST…NT_LAST_NAME	Normal
✗	☐ CLIENT_BROKER	SYSTEM	CLIENT	BROKER_ID	Normal
✗	☐ CLIENT_INCOME	SYSTEM	CLIENT	CLIENT_HOUSEHOLD_INCOME	Normal
✗	☐ CLIENT_PK	SYSTEM	CLIENT	CLIENT_ID	Unique
✗	☐ CLIENT_TRANSACTION_BROKER	SYSTEM	CLIENT_TRANSACTION	BROKER_ID	Normal
✗	☐ CLIENT_TRANSACTION_CLIENT	SYSTEM	CLIENT_TRANSACTION	CLIENT_ID	Normal

To accept the suggestion and have tuning automatically generate an index:

1 For any recommended index, click the checkbox 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.



	Index Name	Ta…er	Ta…e	Cc
	☑ IDX_CLIENT_TRANSACTION_0	SYSTEM	CLI…ON	TRANSACTIO
●	☐ CLIENT_MULTI	SYSTEM	CLIENT	CLIENT_FIR:
✗	☐ CLIENT_BROKER	SYSTEM	CLIENT	BROKER_ID
✗	☐ CLIENT_INCOME	SYSTEM	CLIENT	CLIENT_HOU
✗	☐ CLIENT_PK	SYSTEM	CLIENT	CLIENT_ID
✗	☐ CLIENT_TRANSACTION_BROKER	SYSTEM	CLI…ON	BROKER_ID
✗	☐ CLIENT_TRANSACTION_CLIENT	SYSTEM	CLI…ON	CLIENT_ID

Table SYSTEM.CLIENT_TRANSACTION is scanned via full table scan but it has a filter ct.transaction_status = c.client_marital_status on it and we created a virtual index IDX_CLIENT_TRANSACTION_0 which the optimizer picked up, so we suggest implementing this index.

2 Click the **Create Indexes** button.

The **Index Analysis** dialog appears.

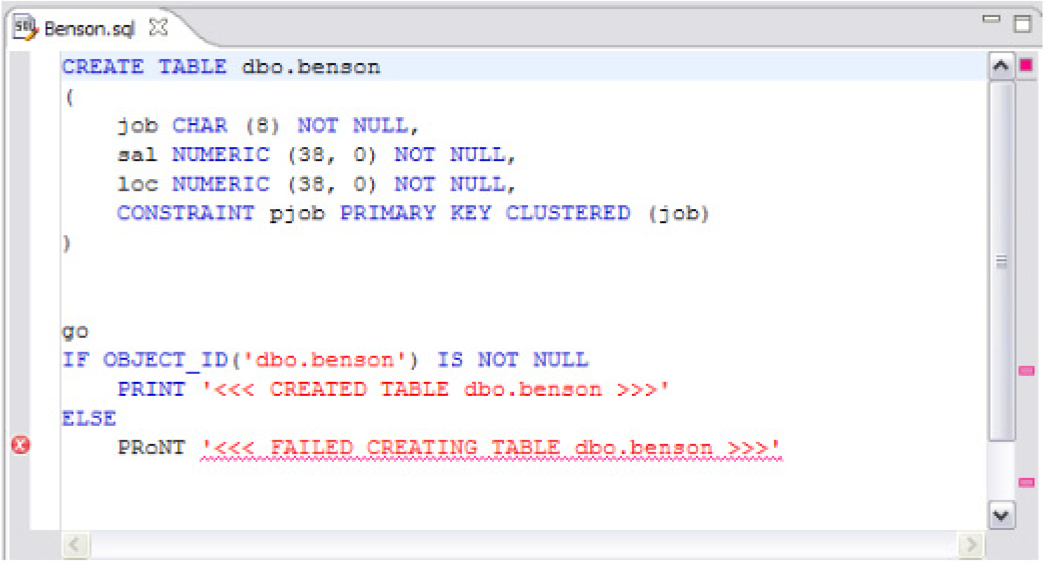
3 *To view the index SQL in an editor for later implementation*, click the statement and then click **Open in a SQL editor**.

4 *To run the index SQL and create the index on the selected database*, click **Execute**.

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.



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


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

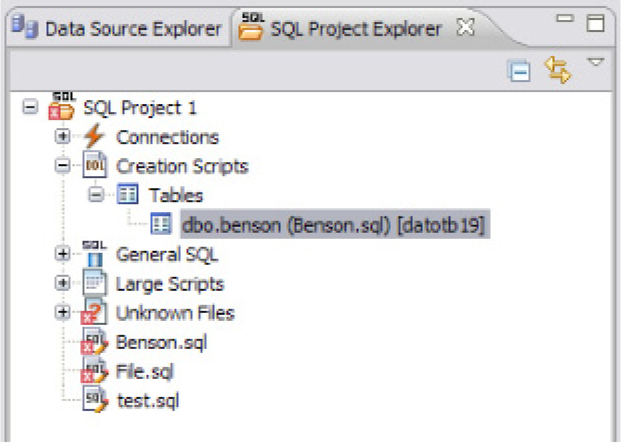
SQL Editor provides key features to ensure an increase in code accuracy and overall development efficiency.

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

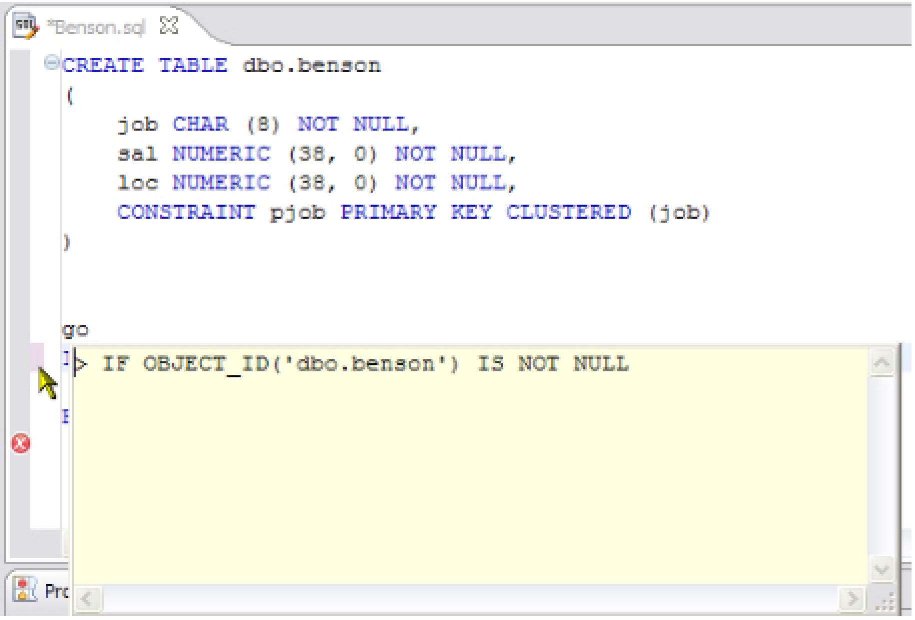
SQL Editor identifies commands and provides syntax highlighting changes that are automatically added to the code as you add lines, which enables you to clearly and quickly understand code when you read it in the interface.

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.



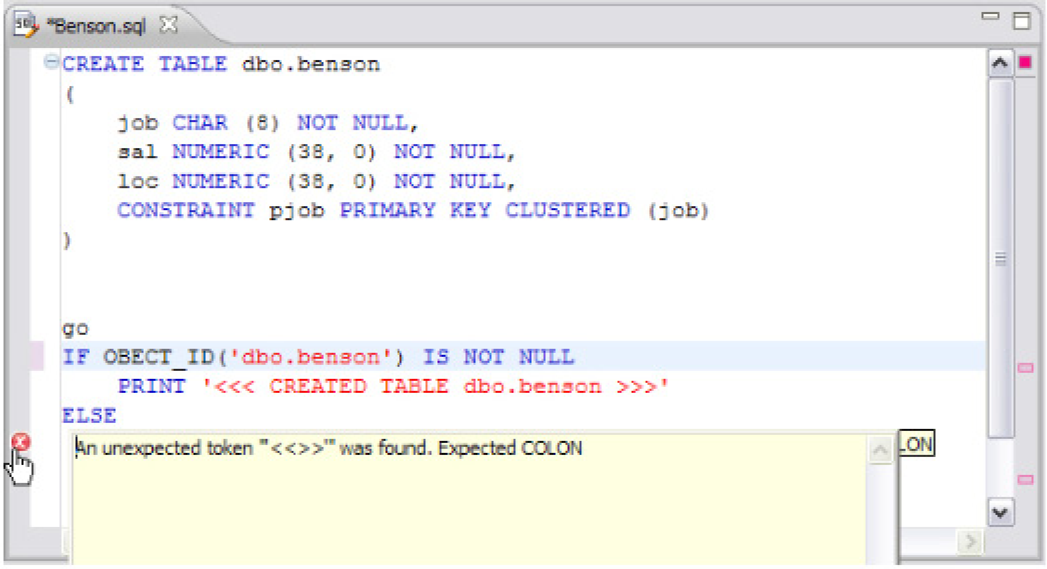
```
CREATE TABLE dbo.benson
(
    job CHAR (8) NOT NULL,
    sal NUMERIC (38, 0) NOT NULL,
    loc NUMERIC (38, 0) NOT NULL,
    CONSTRAINT pjob PRIMARY KEY CLUSTERED (job)
)


go
> IF OBJECT_ID('dbo.benson') IS NOT NULL
```

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.

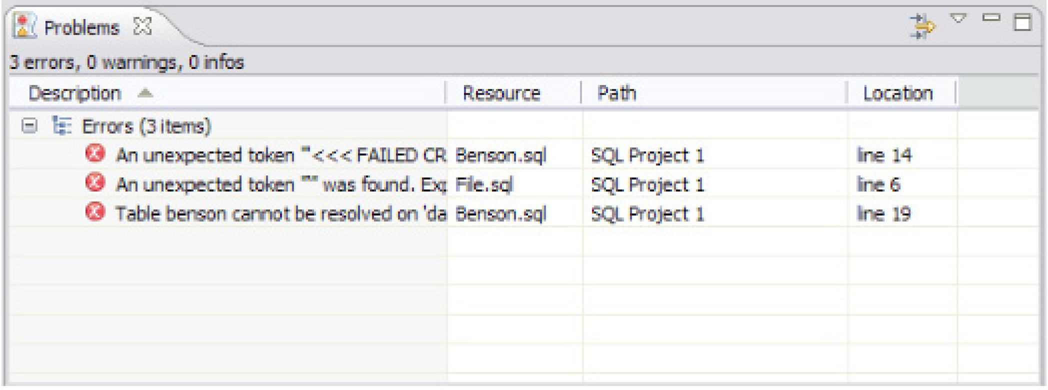


```
CREATE TABLE dbo.benson
(
    job CHAR (8) NOT NULL,
    sal NUMERIC (38, 0) NOT NULL,
    loc NUMERIC (38, 0) NOT NULL,
    CONSTRAINT pjob PRIMARY KEY CLUSTERED (job)
)


go
IF OBECT_ID('dbo.benson') IS NOT NULL
    PRINT '<<< CREATED TABLE dbo.benson >>>'
ELSE
```

An unexpected token "<<>>'" was found. Expected COLON

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.



3 errors, 0 warnings, 0 infos

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

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:

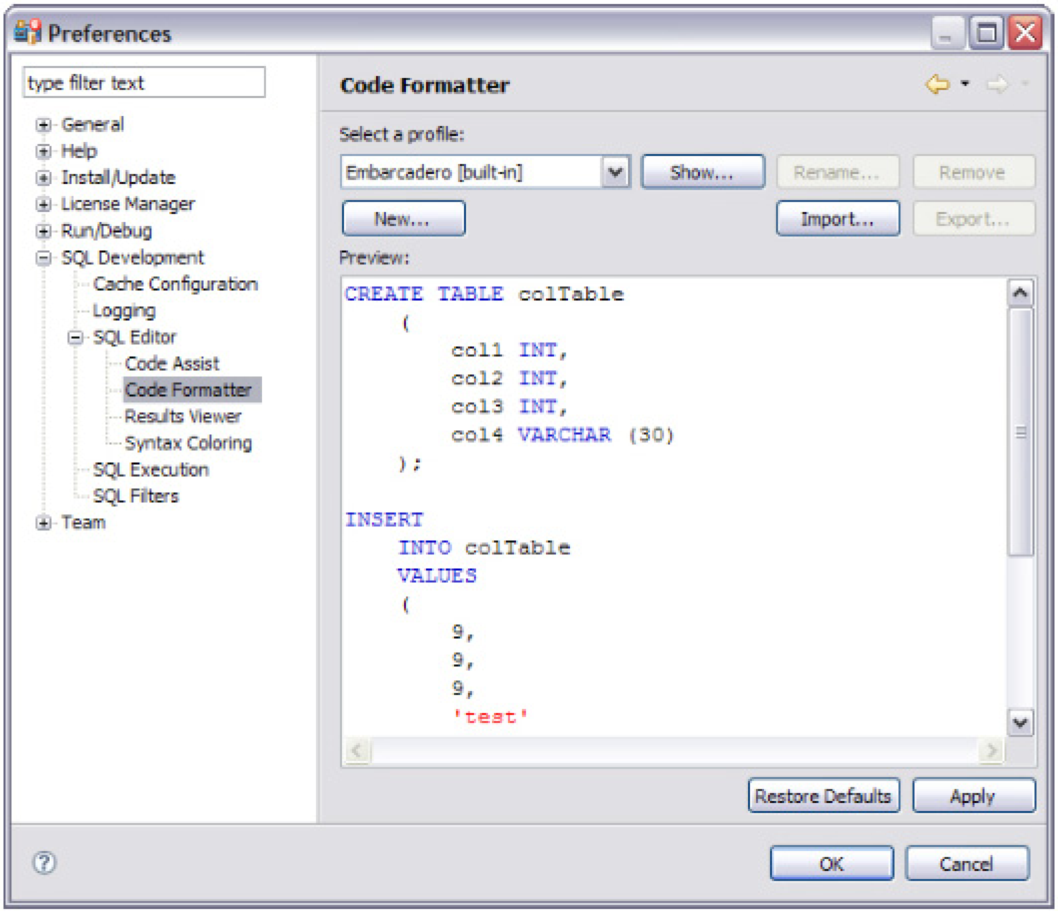
1. Click the line on which you want to activate the code assist feature.
2. Press **CTRL + Spacebar** on your keyboard. Code assist analyzes the line and presents a list of suggestions as appropriate based on the elements of the statement.

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.)



Select a profile: Embarcadero [built-in]

Preview:

```
CREATE TABLE colTable
    (
        col1 INT,
        col2 INT,
        col3 INT,
        col4 VARCHAR (30)
    );

INSERT
    INTO colTable
    VALUES
    (
        9,
        9,
        9,
        'test'
```

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 + 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:

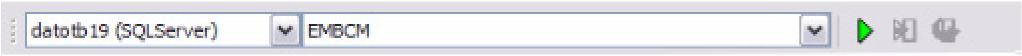
- Click the light bulb icon in the margin of the editor or select **Ctrl + l** on your keyboard.

  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 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:

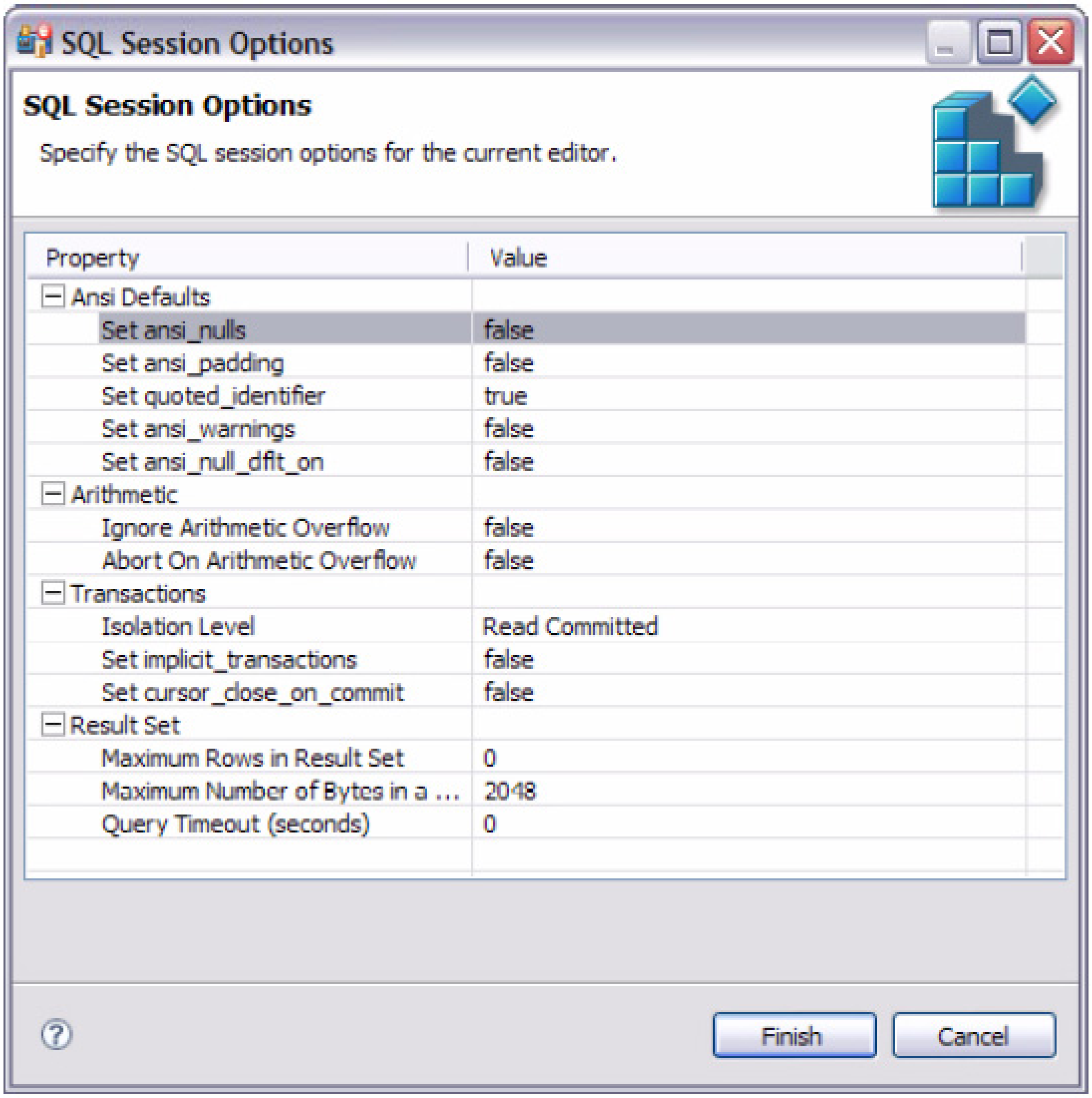
Open the transaction file you want to run and ensure it is associated with the correct database, then click the **Start Transaction** icon. DB Optimizer executes the transaction on the database you specified.

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 Session Options

Specify the SQL session options for the current editor.

Property	Value
⊟ Ansi Defaults	
Set ansi_nulls	false
Set ansi_padding	false
Set quoted_identifier	true
Set ansi_warnings	false
Set ansi_null_dflt_on	false
⊟ Arithmetic	
Ignore Arithmetic Overflow	false
Abort On Arithmetic Overflow	false
⊟ Transactions	
Isolation Level	Read Committed
Set implicit_transactions	false
Set cursor_close_on_commit	false
⊟ Result Set	
Maximum Rows in Result Set	0
Maximum Number of Bytes in a ...	2048
Query Timeout (seconds)	0

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: 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 provides a variety of resources to help support your evaluation and selection of a Data Modeling product for your organization.

Web site

Visit our Web site for current product and company information, educational materials and supporting information. Visit www.embarcadero.com.

To download an evaluation copy of DB Optimizer, please visit: www.embarcadero.com/downloads

Electronic Documentation

Detailed reference documentation is available on the DB Optimizer Evaluation CD or online at docs.embarcadero.com

Online FAQ

The DB Optimizer online FAQ provides answers to commonly asked questions regarding licensing, installation and other helpful topics.

To review the FAQs for DB Optimizer visit: www.embarcadero.com/products/db-optimizer/f

Email Support

You can contact Embarcadero support engineers, consultants and engineers directly by sending inquiries to:

For North America, Latin America and Asia Pacific:

support@embarcadero.com

For Europe, Africa and the Middle East:

uk.support@embarcadero.com

or log a case through embarcadero.com at: www.embarcadero.com/support/open_case.jsp

Telephone Support

We encourage you to call us anytime you would like help or have questions during your evaluation.

For North America, Latin America and Asia Pacific:

Phone: 415.834.3131 x2

Hours: Monday to Friday, 6:00am - 6:00pm Pacific time

For Europe, Africa and the Middle East

Phone: +44 (0) 1628 684499

Hours: Monday to Friday, 9:00am to 5:30pm UK time

Request a Product Key

For North America, Latin America and Asia Pacific:

key@embarcadero.com

For Europe, Africa and the Middle East:

uk.key@embarcadero.com