Evaluating explain plans in Precise for Oracle

This section includes the following topics:

About the SQL tab

The process of explaining statements is a prerequisite for tuning. The explain process is designed to clarify the access path chosen for a statement and translate it into a visual medium. The SQL tab helps you identify bottlenecks and guides you through the steps required to tune SQL statements.

After a statement is explained, the explain results are stored in the PMDB. This information includes the objects referenced by the statement and the operations performed on these objects. The top statements are automatically explained every day. You can control how many statements to explain using a setting for the Explain Statements PMDB process in AdminPoint. For more information, see the Precise Administration Guide.

Statements are also parsed by a proprietary Precise for Oracle parser. Parsing allows Precise for Oracle for example to highlight statement objects from the execution plan and to format execution plans. The number of statements that are parsed is again governed by a setting in the PMDB. For more information, see the Precise Administration Guide.

Understanding the execution plan chosen by the Oracle Optimizer is extremely important when tuning your application. You can ensure optimal system performance by ensuring that the best plans are used for your queries.

Precise for Oracle provides you with a special tab only for this purpose, the SQL tab. This tab provides several different views of the same execution plan that makes analyzing long and complicated queries much easier.

If you are running Oracle 10g and later, you can also access the execution plan associated with a CREATE TABLE as SELECT statements. In addition to examining the plan, this lets you gain additional insight into a problem by analyzing the findings displayed for these statements.

> The statement will only explained. It will not be parsed.

> To analyze an execution plan, Precise for Oracle provides you with a full picture of the objects (table, indexes, and so on) participating in the plan. You can change a plan by modifying your query or changing the schema, for example by adding an index. To assist in this, Precise for Oracle can provide you with index recommendations for your statement.

One technique to identify the source of the problem is to view the historical information of statements, showing performance degradation. A change in the schema, volume, or execution plan may explain the impact on performance.

An important concept within the SQL tab is that of alternative or related SQL. You can take a statement and create different alternatives for it as part of the tuning process. There are a number of different ways of doing this. For example, you may re-write a statement yourself so that it gives the same result in a more efficient manner; or you may click **Actions>Generate Alternatives** to get Precise for Oracle to generate alternatives for you; or you may change a statement's execution plan by simulating the addition of an index in the What-If tab.

The related SQL is stored along with the original in the same cabinet and folder. However, it does not have an independent life of its own; it is always tied to the original. It does not have a resource consumption profile because it has not been captured by the Collector; it only exists in the SQL tab. The related SQL is intended to be transient until one of the alternatives is implemented as the replacing statement.

The following table lists the tabs and entities from which you can launch the SQL tab.

Table 1 Launching the SQL tab in context

Tab	Entities
Dashboard	Statement
Current	Statement, Active or Current Session that is currently executing a statement.
Activity	Statement, PL/SQL
Objects	Statement, PL/SQL
What-If	Statement that is affected by one or more index evaluations.

In the Current, Activity, and Objects tabs, you can either launch the SQL tab by clicking the SQL tab button when the selected entity in the Main area is a statement, or by selecting a statement in the Association area and clicking the Tune icon that appears before the SQL text.

> For a PL/SQL, the SQL tab offers only limited functionality.

If the SQL tab is opened with no statement in context, a message prompts you to open a statement.

If Precise has already collected the steps of a real plan, it appears highlighted in blue; otherwise, it appears highlighted in gray. If you click on a gray statement, Precise for Oracle will access the monitored instance and try to retrieve the selected plan's steps.

There can be statements with many execution plans. Only the latest 7 plans appear on the tree. Click `More...`, if available, to view additional execution plans. See About the Dashboard tab, About the Current tab, About the Activity tab, and About the Objects tab.

How the SQL tab is structured

The SQL tab lets you analyze execution plans and explain results so that you can tune statements and achieve optimal results.

The following controls determine the information displayed in the SQL tab:

- Related SQL selector
- View tabs
- Tuning Action buttons

About the Related SQL selector

The Related SQL selector appears only in the Plan view if a statement has a related SQL. The first time you visit the SQL tab, it will not be visible. It lets you choose one of the alternative statements that are associated with an original statement. An original statement is one captured by the Collector, imported, or entered manually. Full details on the related SQL are held in the Related SQL view.

Do any of the following to create a related SQL:

- When you edit a statement and save an alternative. See Editing an existing statement.
- When you click New Alternatives in the Related SQL view, and Precise for Oracle creates an alternative SQL for you. See Generating new alternatives.
- When an index recommendation is made in the What-If tab and you click Compare to switch to the SQL tab, in which case a copy of the statement is saved along with its new virtual index and execution plan. See About the What-If tab.

About the View tabs

The SQL tab divides the information on the selected statement into different views. Each view has a different layout and presents different information.

The following views are available:

- Plan
- Run Alternatives
- More ...

About viewing the execution plan of a statement

The Plan view displays the execution plan of a statement and related information, such as statistics, referenced objects, and operations performed in the execution plan. It lets you assess and tune statements based on real information.

Two types of execution plans are available when launching to the SQL tab:

- **Real execution plan**. The plan which was collected from Oracles V$ tables. Additional real-time statistics are available for these plans. For example, you can view a breakdown of In Oracle time for a plan as compared to other plans, or a breakdown of its I/O when accessing Oracle objects.
  When launching to the SQL tab with a specific SQL statement in context, Precise for Oracle presents actual execution plan information, including actual plan steps, and information and statistics for multiple plans. For example, if a statement has several different execution plans, all are displayed.

  > This feature is only available for Oracle 10.1.0.4 monitored instances and later.

- **Estimated execution plan**. Depending upon the version of Oracle running on your system (such as Oracle 8i), you may be only able to generate and run an estimated execution plan. At times, real execution plan information is lost (as when they are removed from the system before Precise for Oracle could access them). If this happens, all real-time statistics (such as, I/O, and In Oracle) are considered to belong to other plans.

The Plan view is divided into two panes. The Execution Plan tree is displayed in the left pane. The Details area is displayed in the right pane. The information displayed in the Details area is controlled by the information tabs (Highlights, Expanded Text, Objects, and More...) that are located above this area.

The following table describes the information displayed in the Plan view.

Table 2 Information tabs

Information Tabs	Description
Highlights	Displays the execution plan's formatted text and findings that enable you to identify the most probable cause of the problem your statement is experiencing.
Expanded Text	Shows all views and synonyms expanded inline allowing you to match the access steps to the version of SQL that Oracle is actually executing.
Objects	Displays information on all referenced objects in the execution plan, including their indexes and columns. Statistical details and general details are displayed for each object.
More ...	Shows the following additional information for a plan: • **Statistics view**. Displays statistical information on all steps in the execution plan tree • **Workshop view**. Displays details on the execution plan, where the statement is stored, and the SQL text of the statement

If you use the Oracle rule-based Optimizer, cost information is not available. If tables and indexes are not analyzed, statistics changes are not available.

About the Execution Plan tree

The Execution Plan tree displays both the real execution plan (Oracle 10.1.0.4 and later) and the estimated execution plan, of the specified statement. When you analyze the access plan of a selected statement, you can examine the access path that was chosen by the Oracle Optimizer.

The execution plan's text or expanded text (with all referenced views expanded) appears both at the bottom of the execution plan tree (by clicking the plus sign (+) to view text) or at the top of the Details area (right pane). The text that relates to the selected step in the Plan tree is highlighted. This lets you view the text of the statement, the execution plan, and additional information, such as the objects referenced by the statement, all at once. See About expanded text of a statement in the Plan view.

Actions that can be performed on the tree

You can perform the following actions on the execution plan tree:

- The Playback controls, located at the top of the tree, enable you to freely move within the execution plan of an explained statement.
- By moving the pointer over the execution plan steps, you can view more statistical information, such as Estimated Cost, Estimated Rows, and Estimated I/O Cost, and description of the specified step.
- Selecting a specific step highlights nested steps and affects the information displayed in the information tabs: the relevant SQL text is highlighted on the Text tab; the referenced tables, indexes, and columns are shown on the Objects tab; and the relevant step is highlighted on the Statistics tab.
- Clicking the plus sign (+), located at the bottom of the tree, displays and highlights the statement's formatted text or expanded text.

About the text of a statement in the Plan view

The Highlights tab, in the SQL tab, displays the execution plan's formatted text and findings that enable you to identify the most probable cause of the problem your statement is experiencing. The Findings table and Findings Details area provide additional information on what steps you can take to continue your analysis and lets you launch to the appropriate tab, with the statement in context.

The text that relates to the selected step in the execution plan tree (on the left) is highlighted. The highlight color is based on the selected step type. The structured text of the execution plan is displayed in the right pane.

When a table is highlighted, all table columns appearing in the text are highlighted in the same color as the table. The following is an explanation of the major step types.

About steps that access a table or index

The table and its columns are highlighted and color-coded. The columns that are accessed by an index are underlined.

About sort steps

All tables whose columns were used by sort steps are highlighted and color-coded. Each table and its corresponding columns have their own color. The columns that participate in the sort are underlined.

About join steps

The methodology for highlighting nested loop steps differs slightly from that of the merge join and hash join steps. Nested loop steps are highlighted as follows: all tables and their corresponding columns participating in a sub-tree of the outer table (the first step that is a direct descendant of the nested loop step) are highlighted in blue; all tables and their corresponding columns participating in a sub-tree of the inner table (the second step that is a direct descendant of the nested loop step) are highlighted in red.

The highlight methodology for the other join steps (hash join and merge join) differs slightly. The first table and all its corresponding columns are highlighted in blue. The columns that are used for the join are underlined. The second table and all its corresponding columns are highlighted in red. The columns that are used for the join are underlined.

About expanded text of a statement in the Plan view

Oracle provides views and synonyms as a way of simplifying the complexity of an application. However, the Oracle Optimizer must generate an execution plan against the correct underlying tables even if they do not appear in the SQL text. The Expanded Text tab shows all views and synonyms expanded inline enabling you to match access steps to the version of SQL that Oracle is actually executing.

In the SQL tab, a statement with the original text appears in the bottom left corner and the expanded text on the right at the top. The expanded view text is shown in bold. The statement highlighting picks out the columns within the view referenced in the explain plan.

About the objects that are referenced by the execution plan

The Objects tab displays three tables (Tables in use, Indexes defined on table, Columns in table or index) that list all referenced objects in the execution plan, including their indexes and columns. Statistical details and general details are displayed for each object.

Each table has a title denoting the entities highlighted. The lower table could have a title denoting either the columns of a table or the columns of an index.

> If tables and indexes are not analyzed, statistics information is not available.

About tables in use

The Tables in use lists all the referenced tables in the tree.

The following table describes the information displayed for the tables that are accessed in the execution plan.

Table 3 Information on tables used in the plan

Column	Description
	Launches the Objects tab so that you can focus on the specified table.
	Locates and highlights all the steps in the execution plan that access the specified table.
Used	Indicates whether the specified table is used in the selected step in the execution plan tree.
Table	Displays the table name. A ToolTip displays the full name in the following format: *Owner.Table_Name.*
Rows	Number of rows in the table based on data dictionary statistics.
Blocks	Total number of blocks in the table.
Non-Empty Blocks	Number of used blocks in the table.
Chained Rows	Number of chained rows in the table.
Partitioned	Indicates whether the table is a partitioned table.
Index Organized	Indicates whether the table is an index organized table.
Temporary	Indicates whether the table is a temporary table.
Object ID	ID of the object.
Owner	Owner of the table.
I/O Wait	Time of IO wait in the specific plan.

About the indexes defined for the selected table

The Indexes defined for the selected table lists all the indexes that are used to access the selected table, whether they were used in the explain plan or not.

The following table describes the information displayed for the indexes that are used to access the selected table.

Table 4 Information on the indexes used to access the selected table

Column	Description
	Launches the Objects tab so that you can focus on the specified index.
	Locates and highlights all the steps in the execution plan that access the specified index.
Used	Indicates whether the specified index is used in the selected step in the execution plan tree.
Index	Index name. A ToolTip gives the full name, such as Owner.Index_name.
Unique	Indicates whether the index is unique.
Type	Index type, such as Normal (B*Tree), Bitmap, and so on.
Partitioned	Indicates whether the index is a partitioned index.
blocks	Total number of blocks in the index.
Leaf Blocks	Number of leaf blocks in the index.
Distinct Keys	Number of distinct keys or values in the index.
B-Level	Depth of a B*Tree index.
Clustering Factor	Clustering factor of the index. The clustering factor is an important factor in determining how efficiently an index range scan will retrieve data from the table. It measures the degree to which the data in the index and its table are in the same order or, put another way, the probability that the next row to be fetched from the table is in the same block as the current row. It can vary between the number of blocks in the table (the best case because they are in the same order) and the number of rows in the table (the worst case because they are completely out of sync). The clustering factor tends to become worse over time as data is inserted and deleted. Note that the clustering factor makes no difference for a unique index lookup.
I/O Wait	Time of IO wait for index in the specific plan.
Last Analyzed	Time when the table was last analyzed.
Object ID	ID of the Object.
Owner	Owner of the table.
Locality	Indicates if the index is local or global.

About the columns in table or index

The Columns in table or index lists all columns in the selected table displayed in Tables Used in Plan. When an index is selected in Indexes of Table, the first column constitutes the index column sorted by the position of the column in the index, and marked with an ascending or descending arrow.

The following table describes the information displayed for the columns of the table.

Table 5 Information on the columns of a table

Column	Description
	Launch the Objects tab so that you can focus on the specified column.
Column	Column name.
Type	Physical storage type of the column.
Distinct Values	Number of distinct values.
Key Number	Position of the column in the index selected in Indexes of Table; otherwise blank.
Asc/Desc	Indicates whether the column is part of the selected index and whether it is sorted in ascending or descending order.
In Clause	Location of the column in the statement. The column may appear in the Select clause, Where clause, Group by clause, Having clause, and Sort by clause.

Indexable	Indicates whether the column could be used as part of an index. A column may be indexable even though it is not currently part of any index. Alternatively, a column may not be indexable, even though it is currently part of an index, if the column does not appear within the where clause (or group by, or order by clauses) or there is a function on the column (and there is no function-based index). The Optimizer will not be able to use an index if the leading columns in the index are non-indexable (unless it can employ a skip-scan search).
Indexes	List of indexes in which the column is used.

About statistical information on all operators in the execution plan

The Statistics tab displays statistical information on all steps in the execution plan tree. You can sort steps by their estimated cost or by any other column in the table.

The following table shows the information displayed in statistics table for the execution plan.

Table 6 Execution plan statistics

Column	Description
	Locates and highlights the step in the Execution Plan tree that matches the selected step in the grid.
Step ID	Displays the number of the explain plan step.
Step	Provides a short description of the explain plan step.
Estimated Cost	Estimated cost of the current operation. A high cost value may indicate a problem in the current implementation of the operation. Check the Estimated I/O cost and Estimated CPU cost values to determine whether the operation is an I/O consuming operation or a CPU consuming operation (or both).
Estimated Rows	Estimated number of rows returned by this step.
Estimated Bytes	Estimated number of bytes returned by this step.
Estimated CPU Cost	The estimated CPU cost of the current operation. A high cost value may indicate a problem in the current implementation of the operation.
Estimated I/O Cost	Estimated I/O cost of the current operation. A high cost value may indicate a problem in the current implementation of the operation.
Partitioned ID	Name of the table or index partition, if the step involves a partitioned table or index.
Partition Start	Low value of the partition key.
Partitioned Stop	High value of the partition key.
Access Predicates	Predicates used to locate rows in an access structure; for example start or stop predicates for an index range scan (as defined in Oracle documentation).
Filter Predicates	Predicates used to filter rows before producing them (as defined in Oracle documentation).
#	Step #.
In Oracle	In Oracle time for the specific step.
Cost	Cost for the specific step.
Other Tag	Data as seen in Oracle for specific step.
Plan Hash	Plan hash value.

About information on statement plan, location, and text

The Workshop tab displays details on the execution plan, where the statement is stored, and the SQL text of the statement.

The following table describes how the information on the Workshop tab is structured.

Table 7 Structure of information on the Workshop tab

Item	Description

Explained on	Date the latest explain plan was first generated.
Cost	Estimated cost calculated by the Oracle cost-based Optimizer.
Parsing User	In the case of an imported or manually saved statement, this is the Oracle user that was specified at the time; in the case of an automatically captured statement, it is the first Oracle user that the Precise for Oracle Collector found running the statement.
Cabinet	Cabinet where the statement is stored.
Folder	Folder within the cabinet where the statement is stored.
Origin	Source of the statement. The Precise for Oracle Collector will automatically capture most statements when they are executing in Oracle, but some may be imported from source files or saved manually. Possible values are: • Automatically collected • Generated as related SQL • Predicted plan (What-If) • Entered by user • Imported from source file • Generated as related SQL by user
Saved on	Date that the statement was saved.
Parsing Information	Indicates whether Precise for Oracle has performed an extra level of parsing above that performed by Oracle to support more detailed analysis and syntax color highlighting.
Comment	User-defined comment that can be entered against a statement when it is saved. See Editing the properties of a statement.

About the Recommend button

The Recommend button above the right pane is used to activate the Oracle Advisor (default) or Precise Advisor. The result of the recommendation is shown in the What-If Tier.

We recommend you to use the Oracle Advisor, as this provides an extended Oracle functionality. The recommendation van be evaluated by the What-If analysis.

> The Precise Advisor on the SQL tab will be used instead of the Oracle Advisor when: the Oracle Advisor is not installed, the Oracle version does not support it, or the usage of the Oracle Advisor was manually disabled in our application (for more information on how to manually disable the default Oracle Advisor, see Disabling the Oracle Advisor.

Disabling the Oracle Advisor

When you decide to disable the Oracle Advisor, you have to make changes in the registry file per instance. The result will be that the Precise Advisor will be used with the following limitation:

- When clicking the Recommend link in the Finding on the Activity tab the following message will appear:
  `Unable to perform the Recommended function. Verify that the Oracle Advisor is enabled in the Precise registry.`

To disable the Oracle Advisor

1. For each instance, open the `<SID_name>.xml` file in the following location:
   `\products\i3fp\registry\products\indepth-oracle\hosts\<host_name>\oracle`
2. Under the <recommend> tag, add/edit the following tag:
   `<oem-enabled>N</oem-enabled>`
   Where `N` disables the Oracle Advisor.
3. Save the `<SID_name>.xml` file.
4. Restart the Precise for Oracle FocalPoint.

About the Run Alternatives view

The Run Alternatives view lets you run different statement alternatives under different conditions (bind variables and session parameters) and view details on how the statement was run.

Each row in the Alternatives table (displayed in the left pane) can represent details of one of the following:

- The original statement.
- Any of the statements alternatives.
- If you decide to run the original statement or any of its alternatives, a new row, describing the details of the run, is added to the table.

The following table describes the information displayed for the alternative statements.

Table 8 Alternative Statements

Column	Description
	Launches to the Plan view with the statement in context and lets you view the statement's execution plan, in the Highlights tab.
	Lets you compare the alternative statement with the original statement.
Status	Displays whether or not the statement was successfully run, or whether it is still running or has timed out.
Name	The information displays for the following statement names: • For an alternative statement, it displays the alternative statement's name. • For the original statement, it displays the original statement's name. • For an alternative run of a statement, it displays the name of the alternative or statement used as a base for running the alternative statement.
Timestamp	Displays the time the statement was run.
Plan Hash Value	Displays the hash value for the plan associated with the selected statement.
Statement ID	Displays the statement ID.
Duration (Avg.)	Displays how long the statement ran.
Cost	Displays the average cost of the plan.
Logical I /O (Avg.)	Displays the average number of logical I/O operations per executed statement.
Physical Reads	Number of physical reads from disk.
Hit Ratio	Number of physical reads or logical reads, expressed as a percentage.
Table Scans	Number of full table scans performed.
Write Requests	Number of requests to write data to disk, usually of temporary data during joins and sorts.
Sorts	Number of sorts performed.
Sorted Rows	Total number of sorted rows.
Table Scans by Row ID	Number of accesses to a table by Row ID. A Row ID contains the address of a row in a table and is the fastest way to gain access to an individual row, although not necessarily to multiple rows. Access via Row ID usually follows an index scan, because Row IDs are stored in the leaf blocks of an index.
Table Scan Rows Gotten	Number of rows retrieved from tables.
Table Scan Blocks	Number of blocks fetched from tables.
Recursive Calls	Number of recursive SQL calls that Oracle made to the data dictionary while executing the statement.
Processe d Rows	Number of rows processed during the execution. This includes rows retrieved from tables, indexes, and temporary segments. It may be many more than are returned.
Host Name	Displays the host name of the server that the statement is running on.
Description	If you chose to run an alternative of the statement, this field displays the description assigned to the run in the Run Statement dialog box.

Attempted Executions	Number of executions started.
Actual Executions	Number of executions completed.

About the text of an alternative statement

This view displays the text associated with the alternative statement.

About the plan tree of an alternative statement

This view displays the plan associated with the alternative statement, or the original statement.

About the Run Info view

The Run Info view displays bind variables and session parameters used during the run of the selected alternative statement.

About the Extended Statistics view

The Extended Statistics view displays additional statistics on the alternative runs.

The following table describes the additional statistics information displayed for each alternative.

Table 9 Extended Statistics table

Column	Description
#	Displays the sequential number of the step.
Step ID	Displays the number of the explain plan step.
Step	Provides a short description of the explain plan step.
Rows	Displays the number of rows returned by this step.
Time	Displays when the alternative statement was last run.
Blocks	Displays the total number of blocks in the alternative statement.
Executions	Displays the number of executions of the alternative statement.

About the History view of resource consumption

The History view displays the resource consumption over time vs. cost, and changes over time. This lets you determine which changes led to performance problems.

The History view is divided into two areas:

- Main area
- Change History area

> If you use the Oracle rule-based Optimizer, cost information is not available. If tables and indexes are not analyzed, statistics changes are not available.

About the Main area of the History view

The Main area displays resource consumption over time, estimated cost over time, changes over time, and executions over time.

The following views are available in the Main area:

- Overview
- Text

About the overview of history information

The Overview, in the SQL tab, displays the following overtime graphs:

Table 10 Graphs

Graph	Description
In Oracle (Avg.)	Illustrates the average resource consumption of the statement over time; does not detail the states or substates.
Cost	Displays the estimated cost calculated by the Oracle cost-based optimizer over time. This information is generated whenever you explain the statements using Precise for Oracle.
Changes	Marks significant events that affect the statement and the objects that it accesses over time, namely analyze statistics changes, schema changes, and execution plan changes. The execution plan changes information is generated whenever you explain the statements using Precise for Oracle, either using the PMDB process or the SQL tab. The statistics changes information is generated whenever you run the PMDB process Collect Schema Changes. It scans all the objects referenced by the statement. The schema changes information is generated whenever you run the PMDB process Collect Schema Changes. It scans all the objects referenced by the statement. See About the Change History area.
Executions	Displays the number of executions of the selected statement over time.

About the text of a statement in the History view

The Text view, in the SQL tab, displays the full SQL text of the statement and information on the statement and execution plan, as follows:

Table 11 Text view display

Information	Description
Cabinet	The cabinet where the statement is stored.
Folder	The folder within the cabinet where the statement is stored.
Origin	The source of the statement. The Precise for Oracle Collector automatically captures most statements when they are executing in Oracle, but you can also import statements from source files or save them manually. A statement can be automatically collected, generated as related SQL, part of a predicted plan in the What-If tab, entered by users, imported from a source file, or generated as a related SQL by a user.
Saved on	The date that the statement was saved.
Parsing Information	Indicates whether Precise for Oracle has performed an extra level of parsing above that performed by Oracle to support more detailed analysis and syntax color highlighting. For statements that are explained in the background, parsing must be enabled in the Explain Statements PMDB process in AdminPoint (see the *Precise Administration Guide*) for statements that are explaining in the background. For statements that you explain in the SQL tab, parsing must be enabled from **Settings>General Settings>SQL**. By default, parsing is enabled. Some statements cannot be parsed. See the Precise Administration Guide and About SQL settings.
Explained on	The date the latest explain plan was generated.
Cost	The estimated cost calculated by the Oracle cost-based optimizer.
Parsing User	In the case of an imported or manually saved statement, the Oracle user that was specified at the time; in the case of an automatically captured statement, the first Oracle user that the Precise for Oracle Collector found running the statement.

About the Change History area

The Change History area displays the changes that may have affected the performance of the statement over time. Information on the following type of changes is displayed:

- All changes
- Schema changes
- Statistics changes
- Execution plan changes
- Run statistics history

About viewing all changes

The All Changes view displays a list of all changes made. The changes can be statistics changes, schema changes, and execution plan changes. If there are many changes, you may want to use one of the other change types or filter your search using the More... option. See Associating entities with data that meets specific criteria.

The following table describes the information displayed in the All Changes table.

Table 12 All Changes table

Column	Description
	Switches to the Plan view and displays the first access plan that was created by the Explain Statements process after the specified change was detected.
	If the object that was changed is one of the entities in the Objects tab (table, view, index, or column), this icon will launch the Objects tab with the selected object in context. Also, if the change type is Access Plan Changed, the icon will launch the Compare view with the specified access plan compared to the last access path, as detected by the Explain Statements process.
Timestamp	Indicates the date that the change was detected.
Change Type	Displays the type of change detected. Can be one of the following values: • Table Created • Table Dropped • Table Altered • Index Created • Index Dropped • Index Altered • Table Statistics Changed • Initial Execution Plan • Different Execution Plan
Object	Indicates the object that was changed, such as table or index.
Change Details	Displays information about the change that was made. For example, if the Change Type is Different Execution Plan, the Change Details column gives the estimated cost.

About schema changes

The Schema Changes view displays a list of all schema type changes made.

The following table describes the information displayed in the Schema Changes table.

Table 13 Schema Changes table

Column	Description
	Switches to the Plan view and displays the first access plan that was created by the Explain Statements process after the specified change was detected.
	If the object that was changed is one of the entities in the Objects tab (table, view, index, or column), this icon will launch the Objects tab with the selected object in context.
Timestamp	Indicates the date the change was detected.
Change Type	Displays the type of change detected. It can be one of the following values: • Table Created • Table Dropped • Table Altered • Index Created • Index Dropped • Index Altered
Object	Indicates the object that was changed, such as table or index.
Columns	Indicates the columns in the object that were affected by the change.

About statistics changes

The Statistics Changes view displays a list of changes made to Oracle statistics on tables, indexes, and columns. The following table describes the information displayed in the Statistics Changes table.

Table 14 Statistics Changes table

Column	Description

	Switches to the Plan view and displays the first access plan that was created by the Explain Statements process after the specified change was detected.
	Launches the Objects tab with the selected instance.
Timestamp	Indicates the date the change was detected.
Object	Indicates the object that was changed, such as table or index.
Type	Displays the type of object, such as table or index.
Rows	A change in the number of rows.
Non-Empty Blocks	A change in the number of non-empty blocks.
Free Space (Avg.)	A change in the average free space in a block.
Chained Rows	A change in the number of chained rows.
Distinct Keys	A change in the number of distinct keys in an index.
Leaf Blocks	A change in the number of leaf blocks in an index.
Clustering Factor	A change in the clustering factor of an index. The clustering factor is an important factor in determining how efficiently an index range scan will retrieve data from the table. It measures the degree to which the data in the index and its table are in the same order or, put another way, the probability that the next row to be fetched from the table is in the same block as the current row. It can vary between the number of blocks in the table (the best case because they are in the same order) and the number of rows in the table (the worst case because they are completely out of sync). The clustering factor tends to become worse over time as data is inserted and deleted. Note that the clustering factor makes no difference for a unique index lookup.

About execution plan changes

The Execution Plan Changes view displays a list of all the access plans of the statement as detected by the Explain Statement process.

The following table describes the information displayed in the Execution Plan Changes table.

Table 15 Execution Plan Changes table

Column	Description
	Switches to the Plan view and displays the first access plan that was created by the Explain Statements process after the specified change was detected.
	Opens the access plan in the Compare view and lets you compare the specified access plan with the last access plan, as detected by the Explain Statements process.
Timestamp	Indicates the date the specified access plan was detected by the Explain Statements process.
Cost	Estimated cost calculated by the Oracle cost-based Optimizer.
Nested Loops	Number of nested loop operations performed.
Hash Joins	Number of hash joins performed.
Merge Joins	Number of merge joins performed.
Sorts	Number of sorts performed.
Table Scans	Number of full table scans performed.
Index Scans	Number of index scans performed.

Steps	Bar graph showing the number of steps in the execution plan and a visual breakdown of the different types of operations performed.

About the run statistics history

The Run Statistics History view displays Oracle run statistics from when the statement has been run manually from within the tool. Any changes that it makes are rolled back. The only overhead is the execution time. A time-out may be specified in the Run dialog box to prevent the statement from taking too long. See Running a statement.

The Run Statistics History is a useful way of testing the performance of different SQL alternatives to see which is the fastest before it is put into production. Until it is run, the Collector will not have captured its execution performance. Another use is to benchmark the performance of certain key statements over time.

The information in the Run Statistics History table is displayed on the following tabs: General and Statistics. The statistics information is the same as you would get from the V$SESSTAT table or if you set the autotrace function in SQL*Plus.

The following table explains the information shown on the General tab.

Table 16 Information shown on the General tab

Column	Description
Timestamp	Displays when the statement was run.
Duration	How long the statement took to run.
Processed Rows	Number of rows processed during the execution. This includes rows retrieved from tables, indexes, and temporary segments. It may be many more than are returned.
Physical Reads	Number of physical reads from disk.
Attempted Executions	Number of executions started.
Actual Executions	Number of executions completed.
Timed Out	Denotes whether the timeout specified in the Run dialog was reached. The timeout is for all executions.
Timeout (Sec.)	Timeout in seconds that was specified in the Run dialog box.
Status	One of the following states: Started, Completed, or an error message if an error has occurred.

The following table explains the information shown on the Statistics tab.

Table 17 Information shown on the Statistics tab

Column	Description
Timestamp	When the statement was run.
Logical Reads	Number of logical reads.
Hit Ratio	Number of physical reads or logical reads, expressed as a percentage.
Write Requests	Number of requests to write data to disk, usually of temporary data during joins and sorts.
Sorts	Number of sorts performed.
Sorted Rows	Total number of sorted rows.
Table Scans	Number of full table scans performed.
Table Scans by Row ID	Number of accesses to a table by Row ID. A Row ID contains the address of a row in a table and is the fastest way to gain access to an individual row, although not necessarily to multiple rows. Access using Row ID usually follows an index scan, because Row IDs are stored in the leaf blocks of an index.
Table Scan Rows Gotten	Number of rows retrieved from tables.

Table Scan Blocks	Number of blocks fetched from tables.
Recursive Calls	Number of recursive SQL calls that Oracle made to the data dictionary while executing the statement.

Comparing execution plans

The Compare view lets you compare a statement's text and its execution plan with one of its alternatives or with one of its historical execution plans. By default, the last execution plan is displayed in the left pane. The execution plan you want to compare it with is displayed in the right pane. The statement's text is displayed at the bottom of each execution plan pane, and the selected step is highlighted.

The Compare view is usually opened in context from other views within the SQL tab (such as History or Related SQL) or other tabs (such as What-If). To open the Compare view in the context of the selected execution plan and compare it with the latest execution plan, click the Compare icon.

 Click the Compare icon to display the list of execution plans available for a statement.

To compare the execution plans of a specified statement

1. Click the Compare icon.
2. From the list of access plans that belong to the statement and related SQL, choose the access plan or related SQL you want to use for a comparison from the list.
3. Click **OK**.

The statement you selected is displayed in the right pane. The statement's text is displayed in the bottom pane.

About the All Plans view

The All Plans view displays information on all the available plans. Use this view to analyze and compare the plans to determine which is the best plan to run. Every row in the All Plans table (displayed in the left pane), describes a different plan belonging to the current statement.

The following table describes the information displayed in the All Plans table.

Table 18 All Plans table

Column	Description
	Launches to the SQL tab with the selected plan in context.
	Launches to the Compare view so that you can compare the current plan and the selected plan.
Plan Hash Value	Displays the plan's hash value, as computed by Oracle.
Plan Type icon	Indicates whether a plan is a real or estimated execution plan.
First Detected	For actual plans, displays when the plan was first encountered. For estimated plans, displays when the statement was first explained and the plan was produced.
Last Detected	For actual plans, displays when the plan was last encountered. For estimated plans, displays when the statement was last explained and the plan was produced.
In Oracle	Displays the total time spent in Oracle by statements which were using this plan during the selected time frame.
Duration (Avg.)	Displays the average amount of time it took the plan to run.
Cost	Displays the last cost retrieved for the plan.
Executions	Displays the number of times the plan was executed, during the selected time frame.

About the all plans overview

The All Plans overview lets you analyze how plans changed over time. This information is displayed in the following graphs:

- **In Oracle (Avg.) vs. Executions**. It displays the average time spent in Oracle vs. the number of executions, over time. This graph lets you analyze the scalability of the different plans displayed in the All Plans table.
  It notices that at a certain point in time one of the plans was replaced by another and that the AVG in-oracle time of the second plan is considerably higher than the first one.

- **In Oracle**. It displays the total time spent in Oracle by statements which were using this plan over time.

- **Cost**. It displays the cost of the retrieved for the plan over time.

About the Plan tree

The Plan tree displays execution plan of the selected statement.

About the Bind Variables view

A bind variable is a placeholder in a SQL statement that must be replaced with a valid value (or address of a value) before the statement can be successfully executed. The execution program passes the value of the bind variable to Oracle when the statement is processed. Understanding which values were used when a statement was run, can be important to the tuning process.

Precise for Oracle can capture statements with bind variables if the following conditions exist:

- Oracle 10.1.0.4 or later is running
- The relevant PMDB process (Collect Bind Variables) is up and running.
- The statement is a heavy statement that is run frequently.

Different values for the same bind variable can dramatically change the performance of a statement. By analyzing a statement's bind values, you can:

- Run a statement with a real set of bind values, and observe how different binds sets influence the statement's execution).
- Note the existence of different sets of bind values for a statement.
- Determine if a correlation can be drawn between a statements performance to the bind values used during the period in question.

About the Bind Variables table

The following information is displayed for each bind set in the Bind Variables table in the left pane:

Table 19 Bind set

Bind Set	Description
Plan Match icon	Icon that indicates whether the plan's hash value matches the hash value of the real execution plan.
Estimated Cost	Displays the estimated cost of the plan.
Best Plan Hash Value	Displays the best hash value of the selected plan. This information is displayed after you run Get Best Plan. If different hash values are displayed, this may indicate that there is a matching problem.
Date Last Captured	Displays the date and time the bind set was last captured.
Bind Variables (B1B50)	Displays the values of bind variables B1 through B50. Precise for Oracle can display up to 50 bind variables.
Duplicated Sets	The number of times this specific bind set was collected.

Viewing information displayed for a particular bind set

You can view the text of a particular bind set, and copy an estimation of the text to another tool, in the Details area (right pane) of the Bind Variables view. You can view the following text for a bind set:

Table 20 Bind set

Bind Set	Description
Text, with bind variables replaced	Displays source text with bind variables replaced by the actual values run by the statement.
Text for Estimation	Displays source text and hints that instruct Oracle how to sample the table's data and create an optimal plan for the same bind variables.
Bind variables metadata	Displays additional information for the bind variables of the selected bind set (such as, bind name and type, and column name and type). This information can be useful in cases where the same bind name is used for different columns, with different data types.

To view the text of a bind set, with variables replaced

1. In the Bind Variables table, click on the plan whose text you want to view.
2. In the Details area, click on the `Text, with bind variables replaced` tab.

> You can select and copy this text to another tool for further analysis.

To obtain an estimate of the bind set's text

1. In the Bind Variables table, click on the plan whose text you want to view.
2. In the Details area, click on the Text for Estimation tab.
3. You can select and copy this text to another tool for further analysis.

About Findings in the Details area

Whenever you select a step on the tree, the children appear as formatted text under Highlights in the Details area. At the top of the Details area, you find the date and time of the sampled execution plan, and its formatted text. Findings appear below the formatted text; it includes a table that lists the severity of each type of sort or other operation, and the name of each object that is referenced by the sampled execution plan.

About the findings table

The Findings table is the result of a proprietary tuning algorithm that recommends normal B*Tree indexes, bitmap indexes, index-only access, or statistics gathering on tables and/or indexes for the selected statement. Based on these recommendations, the Oracle Optimizer can choose a better access plan and improve the performance of the statement.

The following table describes the Findings table.

Table 21 Findings table

Column	Description
Severity	Indicates the severity alert that occurred during the sampled execution plan. Severity is displayed using the following colors: • Red. Top findings. • Orange. Medium findings. • Yellow. Other findings. An orange or yellow finding in an execution plan with many findings, may become a red finding in an execution plan with minimum findings because it becomes one of the top findings.
	Launches to a tab in context with the chosen object.
Type	Indicates the type of operation. Notice that the given type may be underlined. This indicates a live link ToolTip. Select the live link type to view ToolTip recommendations to better access the execution plan and improve the performance of the statement.
Object	Lists the object in the type of operation that is referenced by the execution plan.
Impact (%)	Indicates the maximum theoretical saving, expressed as a percentage of the total In Oracle time consumed. Note that the actual saving that can be made depends on the activity.
In Oracle	Indicates the time used to perform this operation In Oracle for this execution plan.

About the Expanded view

The Expanded view displays recommendations that the Oracle Optimizer can use to create a better access plan and improve the performance of the statement.

The Expanded view is divided into two areas:

- Highlights
- What To Do Next

About the Highlights area

The Highlights area displays a brief description of the findings for this type of operation. It also contains a link which provides more help.

About the What To Do Next area

The What To Do Next area displays one or more recommended steps to identify the cause of the problem. Carefully review all data for the finding before continuing.

How to investigate Findings

When you start investigating the findings, it is good practice to start with the finding that has the highest severity rank in the Findings table.

To investigate a finding

1. Identify the finding with the highest severity rank in the Findings table.
2. Select the finding type to view the expanded information for this type of operation.
3. Read the Highlights and What To Do Next areas for the finding and perform the advice that best suits your needs.
4. Follow up on performance to verify that the problem was resolved.

About a statement's different versions

The Compare view lists the different versions of a statement that have been saved together. It lets you compare Oracle Optimizer cost and execution statistics for each alternative so that you can choose the most efficient one. You can create related statements by doing any of the following:

- Clicking Edit in the Action menu and saving a new version of the statement.
- Clicking New Alternatives in the Action menu to generate alternatives. Statements that are generated this way are named Alternative-nnn.
- Running a What-If simulation. Statements that are generated this way are named Predicted-nnn.

> The alternative versions of a statement only appear in the Activity or What-If tabs or in the Open Statement dialog box after they have been run and captured by the Collector.

The information displayed in the Related SQL view is controlled by the information tabs (Details, Text, Run Statistics) that are located above this area.

About Optimizer cost and more for an alternative

When you click the Run Alternative tab (on the right pane), the cost and more is shown in the Details tab (left pane) for each alternative.

The following table describes the information displayed on the Details tab.

Table 22 Details tab information

Column	Description
	Switches to Plan view, in context with the selected related SQL statement.
	Available for related SQL statements but not for the original statement.
Name	Name of the statement.
Statement ID	ID of the statement.
Timestamp	Last run date and time of the statement.
Cost	Bar graph showing the Oracle Optimizer cost for the statement.
Plan Hash Value	Display plan hash value.
Duration (Avg.)	Average time of how long the statement took to run.
Logical I /O (Avg.)	Number of average logical I/Os from the disk.
Physical Reads	Number of physical reads from disk.
Hit Ratio	Number of physical reads or of logical reads, expressed as a percentage.
Table Scans	Number of full table scans performed.
Write Requests	Number of requests to write data to disk, usually of temporary data during joins and sorts.
Sorts	Number of sorts performed.
Sorted Rows	Number of sorted rows.

Table Scans by Row ID	Number of accesses to a table by Row ID. A Row ID contains the address of a row in a table and is the fastest way to gain access to an individual row, although not necessarily to multiple rows. Access using Row ID usually follows an index scan, because Row IDs are stored in the leaf blocks of an index.
Table Scan Rows Gotten	Number of rows retrieved from tables.
Table Scan Blocks	Number of blocks fetched from tables.
Recursive Calls	Number of recursive SQL calls that Oracle made to the data dictionary while executing the statement.
Processe d Rows	Number of rows processed during the execution. This includes rows retrieved from tables, indexes, and temporary segments. It may be many more than are returned.
Host Name	Name of the host.
Description	Description
Attempted Executions	Number of executions started.
Actual Executions	Number of actual executions performed.

About the SQL text for each alternative

The Text tab displays the SQL text for each alternative.

About Tuning Actions

The Tuning Actions icon lets you perform additional tuning functions. These action options are accessed by clicking on their related button or by accessing them through the Actions menu.

It is possible to perform the following tuning actions:

Table 23 Tuning actions

Action	Description
Open	Open an existing statement.
New	Create a new statement.
Edit Properties	Edit the properties on an existing statement. See Editing the properties of a statement.
Edit Text	Edit an existing statement.
Run	Run a statement.
Re-Explain	Re-explain a statement.
Recommend	Generate new recommendations.
Generate Alternatives	Generate new alternatives.
Get Best Plan	Get the best plan according to collected bind sets.

Opening an existing statement

The Open option lets you open an existing statement in the PMDB that was previously created by using the New button, collected by the Precise for Oracle Collector, or imported from the Activity tab. The SQL tab is populated with the explain plan of your chosen statement. Alternatively, you can open a statement by selecting it in another tab and clicking the SQL tab button or the Tune icon.

You can open existing statements in the following views:

- Plan
- Recommend
- Run Alternatives
- History

- Compare
- All Plans
- Bind Variables

> You cannot open statements that were generated as related SQL. You must first open the original statement and then select the related SQL by using the Related SQL tab. See Editing the properties of a statement.

To open an existing statement

1. Click **Actions>Open**.
2. In the Open Statement dialog box, choose the relevant properties for the statement that you want to view from the drop-down lists, as follows:
   - **Instance**. Indicates the name of the instance that the statement belongs to.
   - **Cabinet**. Indicates the name of the cabinet that the statement is saved in.
   - **Folder**. Indicates the name of the folder that the statement is saved in.
   - **Statement**. Indicates the ID or user-defined name of the statement.
   - **Comment**. Provides an optional comment that was previously associated with the statement.
   - **Text**. Provides a preview of the SQL text for the currently selected statement.
     The maximum number of items displayed in the Cabinet, Folder, and Statement lists is limited.
3. Click **OK**.

Creating a new statement

The Create option lets you create a new statement and save it in the PMDB in a logical cabinet and folder hierarchy. However, you cannot create a new statement with the same combination of cabinet, folder, and name as an existing statement. You cannot create a statement with exactly the same text as an existing statement.

You can create a new statement in the following views:

- Plan
- Recommend
- Run Alternatives
- History
- Compare
- All Plans
- Bind Variables

Statements collected by the Collector are automatically saved in the default cabinet and folder, so it is recommended to choose another cabinet and folder. See Editing the properties of a statement.

To create a new statement

1. Click **Actions>New**.
2. In the New Statement dialog box, choose the relevant properties for the new statement from the drop-down lists and fill in the fields, as follows:
   - **Instance**. Indicates the name of the instance that the statement belongs to.
   - **User**. Indicates the Oracle user name that will parse the statement.
   - **Cabinet**. Indicates the name of the cabinet that the statement is saved in.
   - **Folder**. Indicates the name of the folder that the statement is saved in.
   - **Statement**. Indicates the user-defined name of the statement.
   - **Comment**. Provides an optional tuning comment that is to be associated with the statement.
   - **Text**. Indicates the SQL text for the statement.
     The maximum number of items displayed in the Cabinet, Folder, and Statement lists is limited. You may also type the names.
3. In the Text field, enter the text for the statement.
4. In the Comment field, type a tuning comment that is associated with the statement, if required.
5. Click **OK**. The new statement is saved in the PMDB in the cabinet and folder hierarchy.

Editing an existing statement

The Edit Text option lets you edit a statement that was saved in the PMDB through the New option. Collected and imported statements cannot be edited. However, you can use the Edit dialog box to save a related SQL statement and then edit the related SQL statement.

> You cannot save a statement with the same combination of cabinet, folder, and name as an existing statement.

You can edit existing statements in the following views:

- Plan
- Run Alternatives
- All Plans

See Editing the properties of a statement.

To edit an existing statement

1. Click **Actions>Edit Text**. The Edit Statement dialog box opens.
2. In the Edit Statement dialog box, do one of the following:

- Enter a new name for the statement in the text field to rename the existing statement.
- Enter a new name for the statement, check the **Save as alternative** option to save your changes under a different name.

Either choose or save it as a related statement by entering a new name. If the statement was collected by the Collector, you can only save it under a new name. In this case, the Save as alternative option is pre-selected and cannot be unchecked. See About a statement's different versions.

3. In the Text box, edit the SQL text for the statement.
4. Click **Save**. The edited statement is saved in the same cabinet and folder as the original.

Running a statement

The Run option lets you run a statement in the monitored database so that you can gather its execution statistics. Any changes that it makes are rolled back. The only overhead is the execution time. A timeout may be specified in the Run dialog box to prevent the statement execution from running too long.

You can run a statement in the Run Alternatives view. See Editing the properties of a statement.

To run a statement

1. Click **Actions>Run**.
2. On the General tab in the Run Statement dialog box, do the following:
   a. Select the database instance that the statement will run on. The maximum number of items displayed in the Cabinet, Folder, and Statement lists is limited.
   b. In the Description box, type a short description of the run.
   c. Enter the Oracle user name and password with which to log in to Oracle.
   d. Choose one of the following options: Fetch all rows or Fetch first *n* rows
   e. Specify the number of times that you want to run the statement. You may want to run it more than once to reduce the effect of having to load the buffer cache on the first run. Precise for Oracle calculates and displays the average value for each statistic over the series.
   f. Check the **Time out after** *n minutes or hours*, if you want to configure a timeout after which the execution of the statement is aborted if the timeout is exceeded. If this is an alternative version of a statement that you have saved, a reasonable timeout is the duration of the original version because you are unlikely to be interested in slower alternatives.
   g. Check **Run in background**, if you want to run the statement in the background.
3. On the Bind variables tab in the Run Statement dialog box, choose the bind variables with which to run the bind set as follows:
   a. Click **Choose bind set**.
   b. In the Bind Sets dialog box, choose the bind set you want to use to run the statement. see About bind sets.
   c. Click **OK**.
4. On the Advanced tab in the Run Statement dialog box, specify the session parameters to be used when running the statement:
   a. Type in a value for each session variable that you want to run the statement with.
   b. Click Add New Row if you want to add a new session parameter. Choose a session parameter name from the list, or enter a new one. Type in the value you want to run the statement with. See About session parameters.
   c. If you want to delete a session variable from the statement run, select the session variable you want to delete and click **Delete**.
   d. Click **OK**.
   e. Check the **Extended run information (STATIC_LEVEL=ALL)** option if you want additional statistics to be collected. These statistics are displayed in the Extended Statistics tab in the Run Alternatives view.
   f. Check the **Array fetch** option and specify the number of rows you want to be fetched, if you want to use an array fetch operation.
5. Click **Fill Binds**.
6. Click **OK**.

About bind sets

Bind variables are placeholders in a statement. When you use bind variables, you increase the probability that statements will be stored in memory, making them more immediately available to the next operation that needs them. Bind variables obtain their values from the last statement that was run. You can change the values of bind sets and analyze the values used when a statement is run, to help improve your tuning process.

About session parameters

Session variables obtain their values from the last statement run. You can define a new session variable to run the statement with, in the Add Session Parameters dialog box.

Re-explaining a statement

The Re-Explain option instructs Precise for Oracle to re-explain a statement. Precise for Oracle generates a new execution plan if one or more of the following conditions are met:

- The execution plan has changed—one or more of its steps or their order has changed.
- The cost has changed.
- One of the underlying objects has changed.

If the access plan is unchanged, the last explain time is updated.

You can set the parsing user from **Settings>General Settings>SQL** tab. You can re-explain a statement in the following views:

- Plan
- All Plans

See About configuring your settings and Editing the properties of a statement.

To re-explain a statement, in the Plan or All Plans view, click **Actions>Re-Explain**.

Generating new alternatives

The Generate Alternatives option instructs Precise for Oracle to run the alternative SQL generator to see if it can find ways of re-writing the SQL so that the Oracle Optimizer comes up with a more efficient execution plan. It performs various mathematical transformations to the SQL, such as replacing `EXISTS` with `IN`, `OR` with `UNION`, but still so as to give an equivalent result set. The alternatives are saved in the same folder as the original and can be viewed in the Related SQL view.

You can control the settings for generating alternatives from **Settings>General Settings>SQL** tab, such as only generating alternatives that have a different execution plan or indexable columns.

You can generate new statement alternatives in the Run Alternatives view. See About configuring your settings.

To generate a new alternative, in the Run Alternatives view, click **Actions>Generate Alternative**.

How the SQL tab can help you identify performance problems

After identifying a problematic statement that is slowing down the response time of a specific application, the first step in tuning the statement is to understand the access path that Oracle chose for the statement. The explain procedure is designed to clarify the access path chosen for a statement and translate it into a visual medium. Therefore, you can easily see whether the optimizer chose the proper execution plan. For example, you can see whether the optimizer performed an index seek as expected. In addition, you can see schema changes related to the statement's objects and compare these changes with previous access plans and In Oracle over time data for the selected statement to understand how the changes affected a statement's performance. You can also compare previous access plans and locate the steps that have been changed.

You can identify a performance problem by doing one or more of the following:

- Identifying problematic steps in the access plan
- Getting advice on a selected statement
- Creating a related SQL
- Comparing a related SQL

Identifying problematic steps in the access plan

The first step in tuning a statement is to identify problematic steps.

For Oracle 11 and higher, we collect the step activity from Oracle and know the time distributed between the steps. The highest step will be highlighted.

For versions lower than Oracle 11, the problematic steps are defined as steps where the estimated cost is high or where there are a large number of logical I/Os.

To identify problematic steps in the access plan

1. In the Time Frame list, choose the period of time you want to analyze.
2. Do one of the following:
    - Launch to the SQL tab with a statement in context, for tuning.
    - In the SQL tab, open the statement you want to analyze.
3. The SQL tab shows SQL text on the bottom left, and the execution plan on the top left of the SQL tab. For an example, we see that the SQL is accessing the ORDERS table using a full table scan. The top table on the right will show that the ORDERS table has 30000 rows in it. This is the problematic step.

Getting advice on a selected statement

In our example, we can identify in the statement text three conditions in the WHERE clause of the statement that restrict the amount of data required from the ORDERS table. However, the Oracle Optimizer has determined that every record in the ORDERS table must be read. There are several reasons for this occurrence. We can investigate this problem further by clicking the Recommend button.

Creating a related SQL

The Related SQL view lists the original statement along with any related statements that have been created. Initially there will only be the original statement.

For our example, the SQL tab shows the Related SQL view with just the original statement. We want to tune this statement because no alternate statements have been created yet either manually or automatically by Precise for Oracle.

To create a related SQL

1. In the Time Frame list, choose the period of time you want to analyze.
2. Do one of the following:
    - Launch to the SQL tab with a statement in context, for tuning.
    - In the SQL tab, open the statement you want to analyze.
3. Click **Run Alternatives**. It instructs Precise for Oracle to automatically suggest alternate ways of re-writing the SQL. For our example, two alternatives have been suggested for this statement. The first alternative has a lower cost than the original.
4. Click the **Text** tab to see a second alternative. The second alternative has no cost because it is using a `/*+ RULE */` hint to force the use of Oracle's rule-based Optimizer, rather than the cost-based Optimizer.

Comparing a related SQL

You can carry your analysis of the change further by selecting the Compare view. The Compare view compares two access plans for the same statements.

To compare a related SQL

1. In the Time Frame list, choose the period of time you want to analyze.
2. Do one of the following:
    - Launch to the SQL tab with a statement in context, for tuning.
    - In the SQL tab, open the statement you want to analyze.
3. Click the **Compare** icon. The alternative statement opens both the original and the alternative SQL statement in the Compare view.

For our example, we can see that the alternate SQL is now able to use an index to access the CUSTOMER table, whereas before it was doing a Full Table Scan. The re-written SQL will return exactly the same records as the original, but Oracle can use a different access path to the data.

This is because an SQL function, such as SUBST on the C_LAST column, prevents Oracle from using an index on it (unless you have defined a specific function-based index).

The alternate SQL has no function around the C_LAST column. Therefore, Oracle is able to use the index CUSTOMER_I2 index to more efficiently access the records in the CUSTOMER table.