Contents

[**Oracle SQL Processing Overview** 2](#_Toc482174013)

[**Explain Plan** 2](#_Toc482174014)

[**Generating Explain Plan using SQL plus** 3](#_Toc482174015)

[**Create a Plan Table** 3](#_Toc482174016)

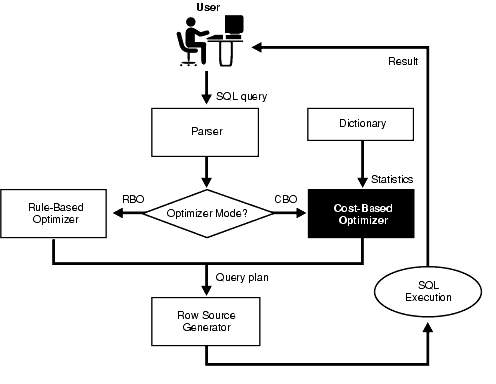
[**Run Explain Plan** 4](#_Toc482174017)

[**Reading Explain Plan Output** 4](#_Toc482174018)

[**Execution Plan Can Change** 5](#_Toc482174019)

# 

# **Oracle SQL Processing Overview**



Oracle optimizer determines the most efficient way to execute a SQL statement after considering many factors related to the objects referenced and the conditions specified in the query.

# 

# **Explain Plan**

A statement's execution plan is the sequence of operations Oracle performs to run the SQL statement.

SQL statements can be executed in many different ways, including the following:

* Full table scans
* Index scans
* Nested loops
* Hash joins

The row source tree is the core of the execution plan. It shows the following information:

* An ordering of the tables referenced by the statement
* An access method for each table mentioned in the statement
* A join method for tables affected by join operations in the statement
* Data operations like filter, sort, or aggregation

In addition to the row source tree, the explain plan contains information about the following:

* Optimization, such as the cost and cardinality of each operation
* Partitioning, such as the set of accessed partitions
* Parallel execution, such as the distribution method of join inputs

# **Generating Explain Plan using SQL plus**

## **Create a Plan Table**

Oracle writes explain plan details to a Table. The first thing you will need to do is make sure you have a table called PLAN\_TABLE available in your schema

(Oracle provided a script to create this table, script should be available in /rdbms/admin/utlxplan.sql)

## **Run Explain Plan**

SQL> EXPLAIN PLAN For your\_sql\_stmt;

SQL> EXPLAIN PLAN FOR select \* from dept where deptno = 40;

Explained.

# **Reading Explain Plan Output**

After running EXPLAIN PLAN, Oracle populates the PLAN\_TABLE table with data that needs to be formatted to present to the user in a more readable format. Several scripts exist for this, however, one of the easiest methods available is to cast *dbms\_xplan.display* to a table and select from it.

SQL> SELECT \* FROM TABLE(***dbms\_xplan.display***);

PLAN\_TABLE\_OUTPUT

---------------------------------------------------------------------------------------

Plan hash value: 2852011669

---------------------------------------------------------------------------------------

| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |

---------------------------------------------------------------------------------------

| 0 | SELECT STATEMENT | | 1 | 20 | 1 (0)| 00:00:01 |

| 1 | TABLE ACCESS BY INDEX ROWID| DEPT | 1 | 20 | 1 (0)| 00:00:01 |

|\* 2 |  **INDEX UNIQUE SCAN** | PK\_DEPT | 1 | | 0 (0)| 00:00:01 |

We can also use database tools like Toad, SQL Developer, OEM etc. to view the explain plan in a readable format.

By examining this plan, we can find out if Oracle is picking the right [indexes](http://www.orafaq.com/wiki/Index) and [joining](http://www.orafaq.com/wiki/Join) your [tables](http://www.orafaq.com/wiki/Table) in the most efficient manner. This will often reveal that the query is not using the relevant indexes, or indexes to support the query are missing.

Eg., Below explain plan shows query is causing index full scan, which would cause bad performance if table size is very big.

|  |  |  |  |
| --- | --- | --- | --- |
| **Plan** | | | |
| **SELECT STATEMENT** ALL\_ROWS Cost: 20,056 Bytes: 2,565 Cardinality: 5 | | | |
|  | **3 TABLE ACCESS BY INDEX ROWID TABLE** SA.TABLE\_OPP Cost: 20,056 Bytes: 2,565 Cardinality: 5 | | |
|  |  | **2 BITMAP CONVERSION TO ROWIDS** | |
|  |  |  | **1 BITMAP INDEX FULL SCAN INDEX (BITMAP)** SA. TYPE\_BMX INDX |

Sometime execution plan operation alone cannot differentiate between well-tuned statements and those that perform poorly. For example, an EXPLAIN PLAN output that shows that a statement uses an index does not necessarily mean that the statement runs efficiently. Sometimes indexes can be extremely inefficient. In this case, you should examine the following:

* The columns of the index being used
* Their selectivity (fraction of table being accessed)

## **Execution Plan Can Change**

EXPLAIN PLAN output shows how Oracle runs the SQL statement when the statement was explained. This can differ from the plan during actual execution for a SQL statement, because of differences in the execution environment and explain plan environment.

*References: Oracle Documentation*