|  |
| --- |
| EPAM Systems, RD Dep.  MTN.\*NIX.07 Oracle DB. Introduction to DWH |
| MTN.\*NIX.07 Labs - Access and Join Methods Part 2 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| REVISION HISTORY | | | | | |
| Ver. | Description of Change | Author | Date | Approved | |
| Name | Effective Date |
| 1.0 | Initial status of document | [**Kiryl Bucha**](mailto:Kiryl_bucha@epam.com) | 16-JAN-2012 |  |  |
|  |  |  |  |  |  |

*Contents*

[1. Auto Trace & Explain Plan 3](#_Toc315038644)

[1.1. Task 1: Auto Trace configuration training 3](#_Toc315038645)

[2. Join Methods 3](#_Toc315038646)

[2.1. Task 2: Nested Loops Joins 4](#_Toc315038647)

[2.2. Task 3: Sort-Merge Joins 4](#_Toc315038648)

[2.3. Task 4: Hash Joins 4](#_Toc315038649)

[2.4. Task 5: Cartesian Joins 4](#_Toc315038650)

[2.5. Task 6: Left/Right Outer Joins 4](#_Toc315038651)

[2.6. Task 7: Full Outer Join 4](#_Toc315038652)

[2.7. Task 8: Semi Joins 4](#_Toc315038653)

[2.8. Task 9: Anti Joins 5](#_Toc315038654)

[2.9. Task 10: Prepare summary table 5](#_Toc315038655)

# Auto Trace & Explain Plan

## Task 1: Auto Trace configuration training

Below all possible variants of SQL plus utilities autotrace:

set autotrace off

set autotrace on

set autotrace traceonly

set autotrace on explain

set autotrace on statistics

set autotrace on explain statistics

set autotrace traceonly explain

set autotrace traceonly statistics

set autotrace traceonly explain statistics

set autotrace off explain

set autotrace off statistics

set autotrace off explain statistics

**Task Results:**

Expected:

Summary table with all result and text description of analyses this results.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Auto Trace Configuration Options | Expected Results | Description |
|  | off | does not display a trace report | not display report statistic and explain plan only SQL statements |
|  | on | displays a trace report | Displays a report on the execution of successful SQL DML statements (SELECT, INSERT, UPDATE or DELETE) |
|  | traceonly | displays a trace report | but does not print query data, if any |
|  | on explain | shows the query execution path by performing an EXPLAIN PLAN | Execute statistics, but not display her |
|  | on statistics | displays SQL statement statistics | Execute explain plan, but not display |
|  | on explain statistics | displays a trace report | Show execute DML statements |
|  | traceonly explain | displays a trace report and explain plan | Not display statistics, but does not print query data |
|  | traceonly statistics | displays a trace report and statistic | Not display explain plan, but does not print query data |
|  | traceonly explain statistics | displays a trace report (explain, statistic) | but does not print query data, if any |
|  | off explain | Not does not display a explain plan | Execute explain plan, but not display |
|  | off statistics | does not display a statistics | Does’t execute statistics |
|  | off explain statistics | Not displays a trace report | Execute explain plan, but not display |

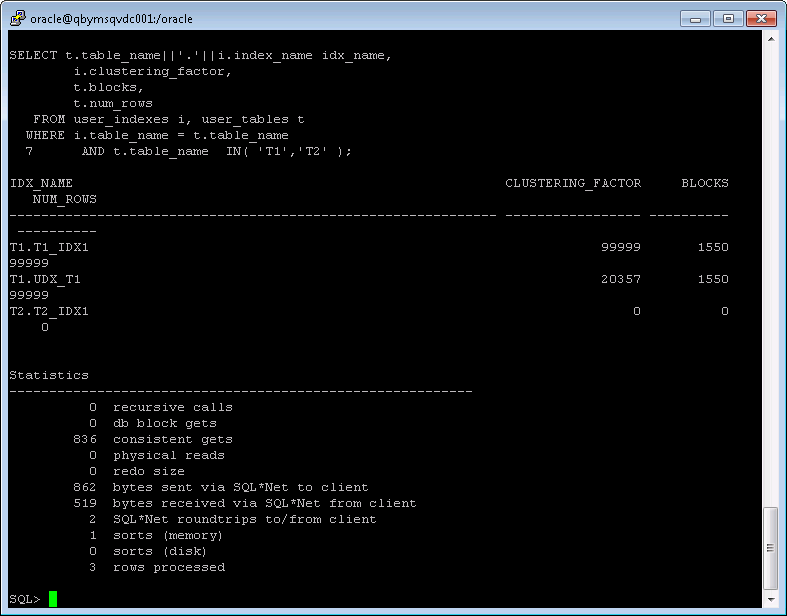
# Join Methods

**The Main Task** is to create SQL and prepare execution plan of statements with join methods on Task Topics (Task 2 - 9)

**Task Results:**

There are several tasks below with the same main expected result points:

* Create SQL using next tables: scott.emp, scott.dept
* Create additional needed Tables and Indexes
* Prepare screenshots of execution plan



## Task 2: Nested Loops Joins

**Example:**

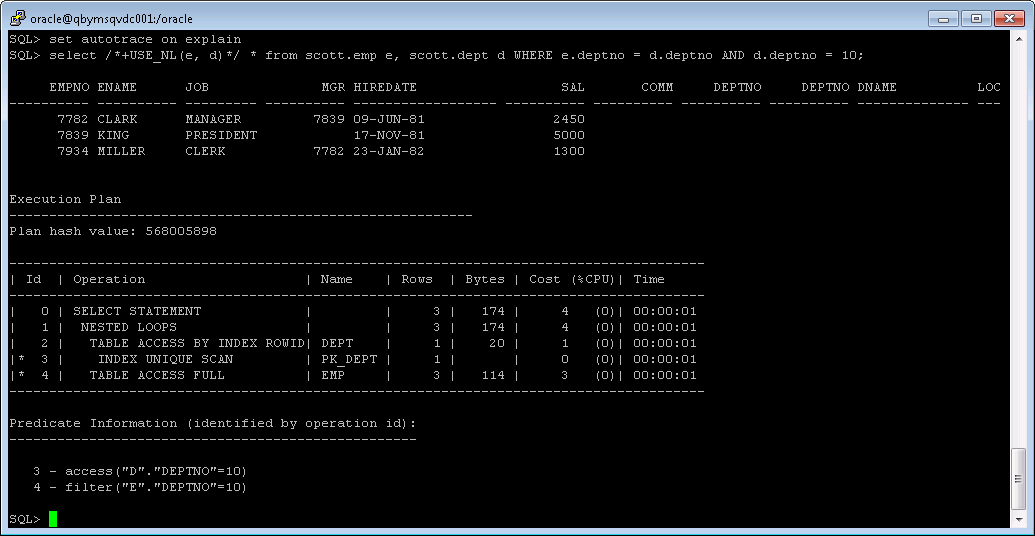
# SELECT \*

FROM scott.emp e, scott.dept d

WHERE e.deptno = d.deptno

AND d.deptno = 10;

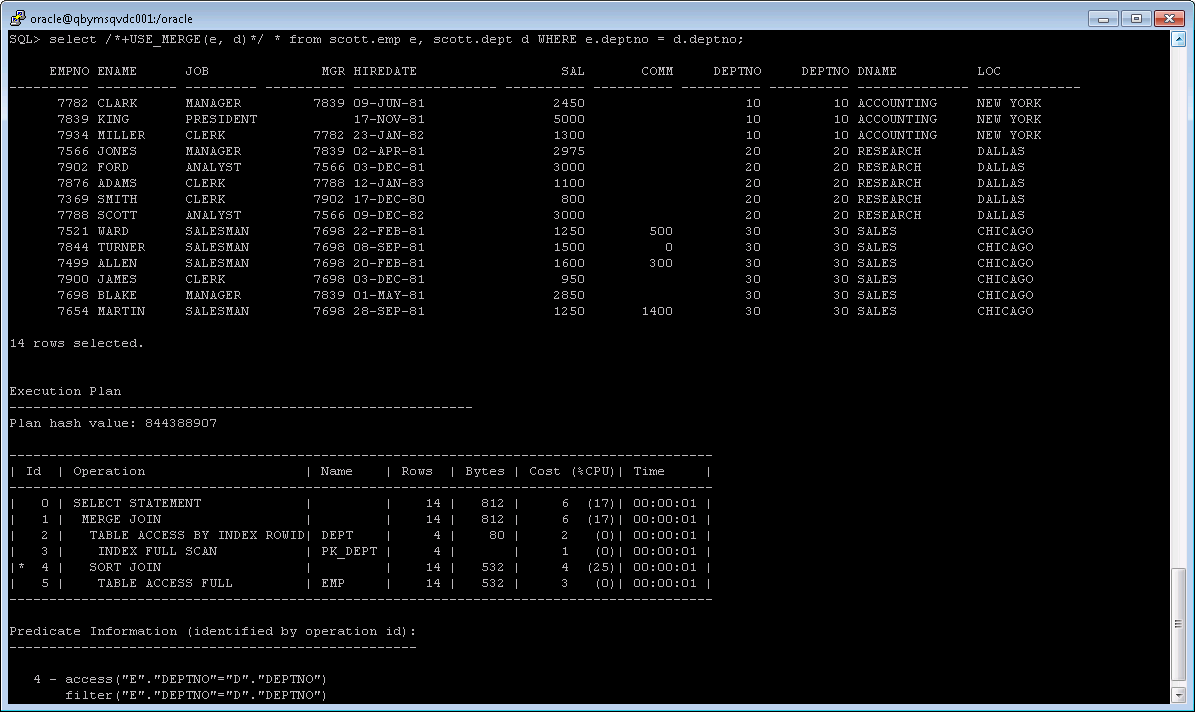
**Task:** Prepare SQL **explain plan** using: SQL plus Auto Trace Utility.



**Note:**  If you would like change in execution plan the type of join method use oracle performance hints. (USE\_NL)

## Task 3: Sort-Merge Joins

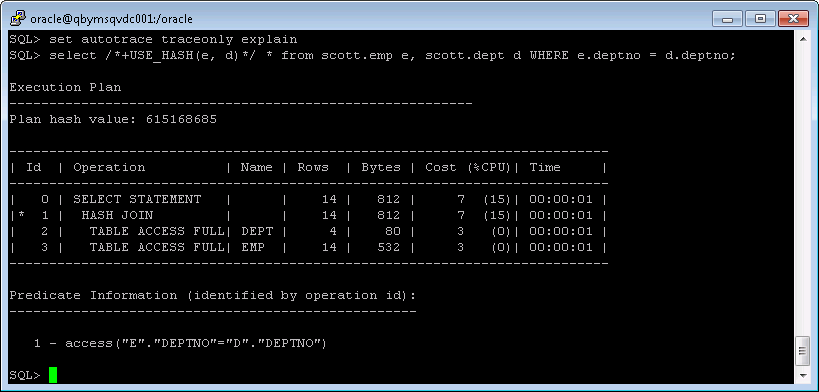
**Task:** Prepare SQL **explain plan** using: SQL plus Auto Trace Utility.



**Note:**  If you would like change in execution plan the type of join method use oracle performance hints. (USE\_MERGE)

## Task 4: Hash Joins

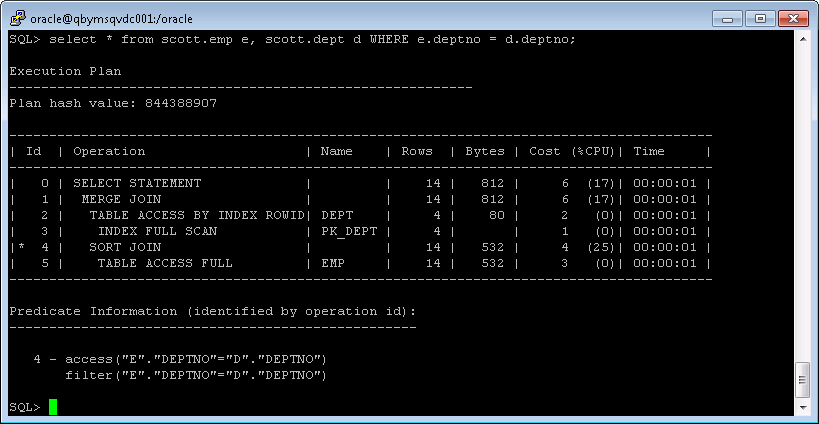
**Task:** Prepare SQL **explain plan** using software: Oracle SQL Developer.



**Note:**  If you would like change in execution plan the type of join method use oracle performance hints. (USE\_HASH)

## Task 5: Cartesian Joins

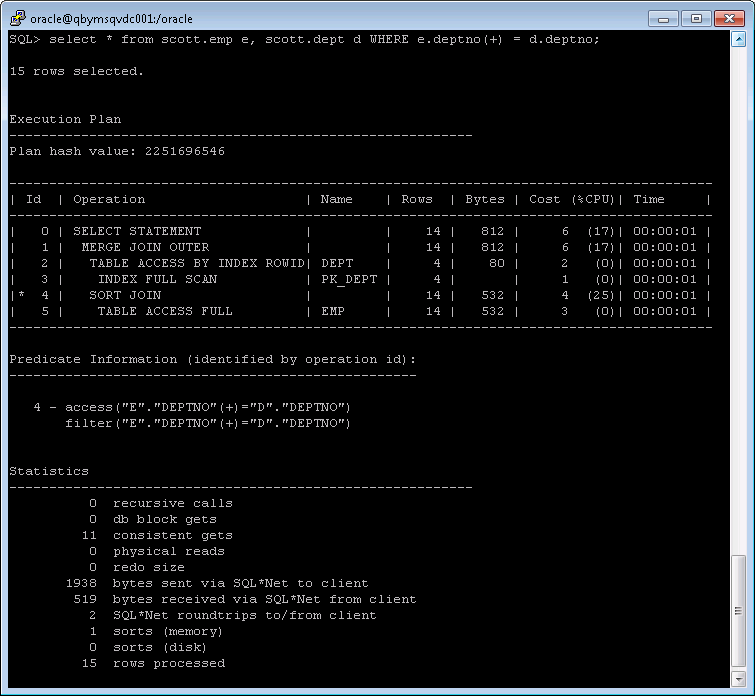
**Task:** Prepare SQL **explain plan** using software: Oracle SQL Developer.



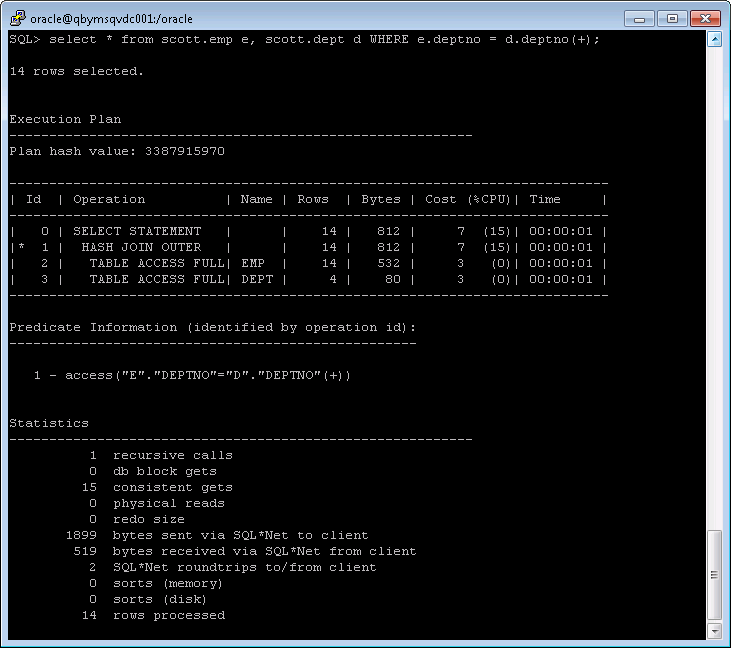
## Task 6: Left/Right Outer Joins

**Tasks:**

* Prepare SQL **trace protocol** using software: Oracle SQL Developer.
* Create SQL left outer join

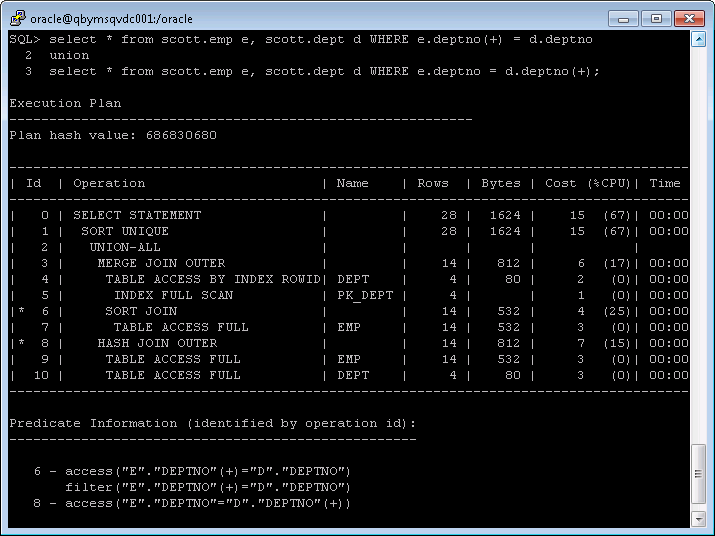


* Create SQL right outer join



## Task 7: Full Outer Join

**Task:** Prepare SQL **explain plan** using: SQL plus Auto Trace Utility.

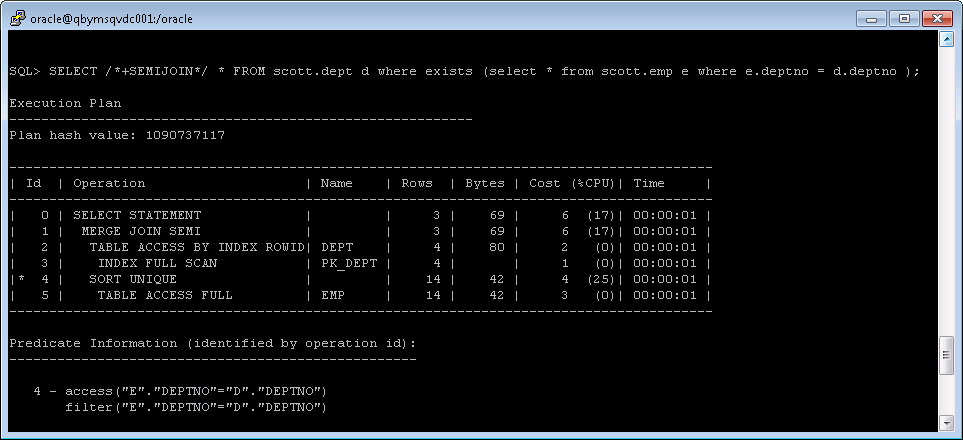


## Task 8: Semi Joins

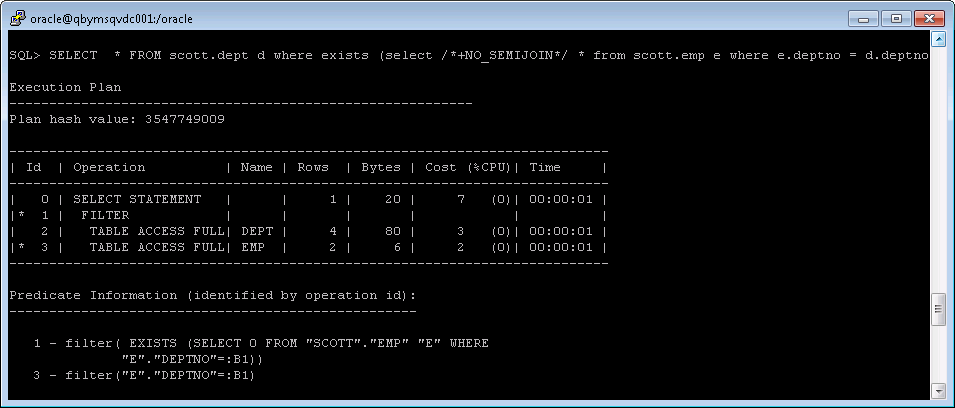
**Task:** Prepare All possible variants of SEMI JOIN SQL **explain plan** using: SQL plus Auto Trace Utility.

**Note:**  If you would like change in execution plan the type of join method use oracle performance hints.

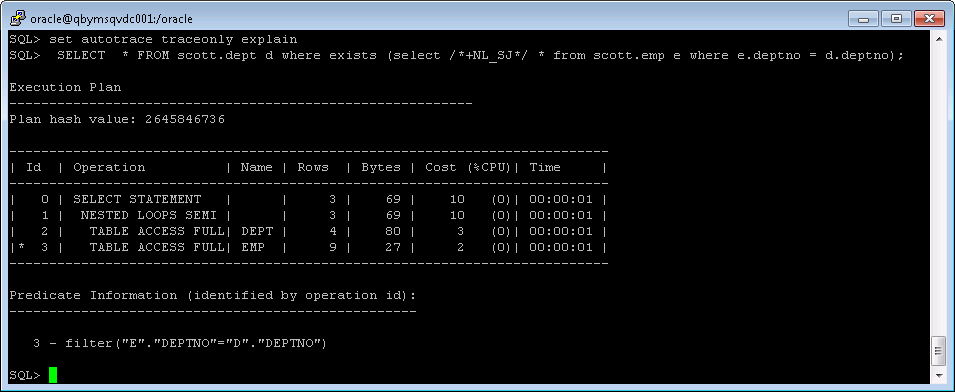
1. SEMIJOIN – perform a semi-join (the optimizer gets to pick which kind)



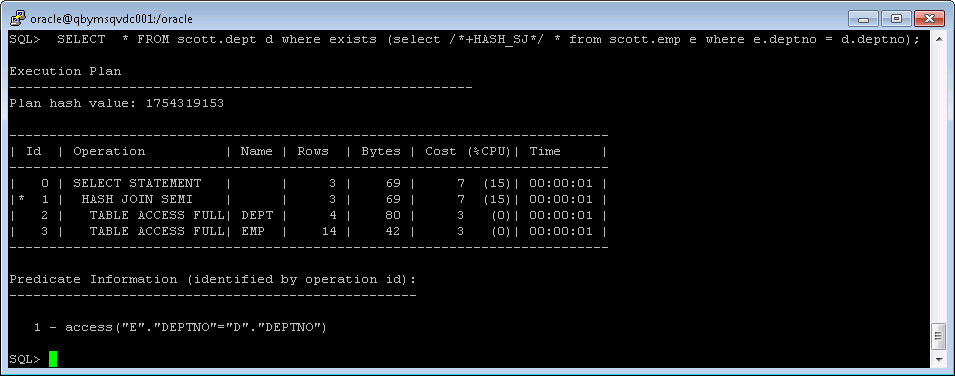
1. NO\_SEMIJOIN – obviously means don’t perform a semi-join



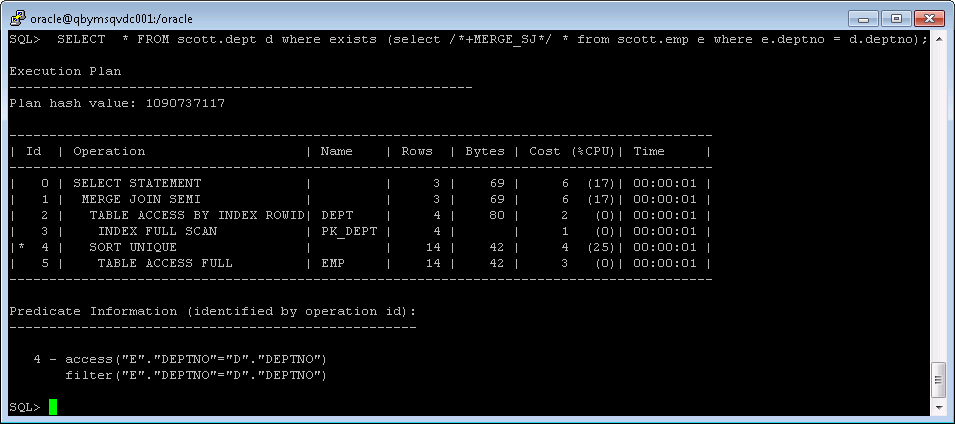
1. NL\_SJ – perform a nested loops semi-join (deprecated as of 10g)



1. HASH\_SJ – perform a hash semi-join (deprecated as of 10g)



1. MERGE\_SJ – perform a merge semi-join (deprecated as of 10g)

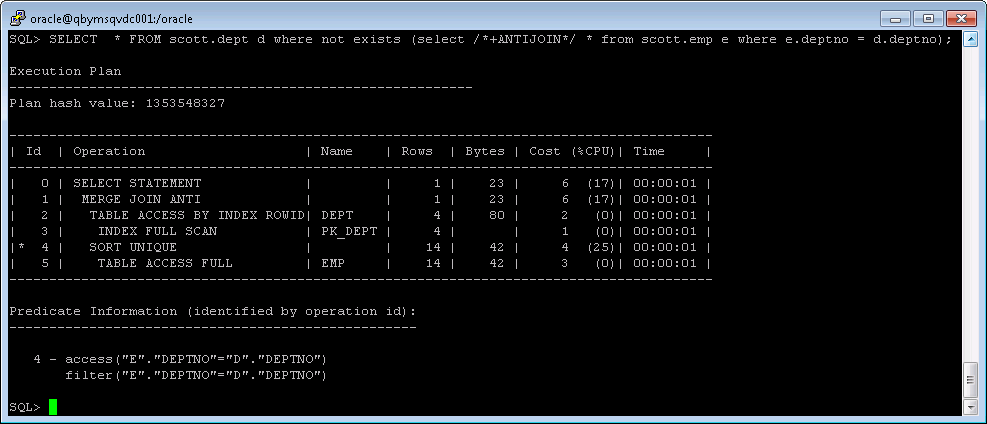


## Task 9: Anti Joins

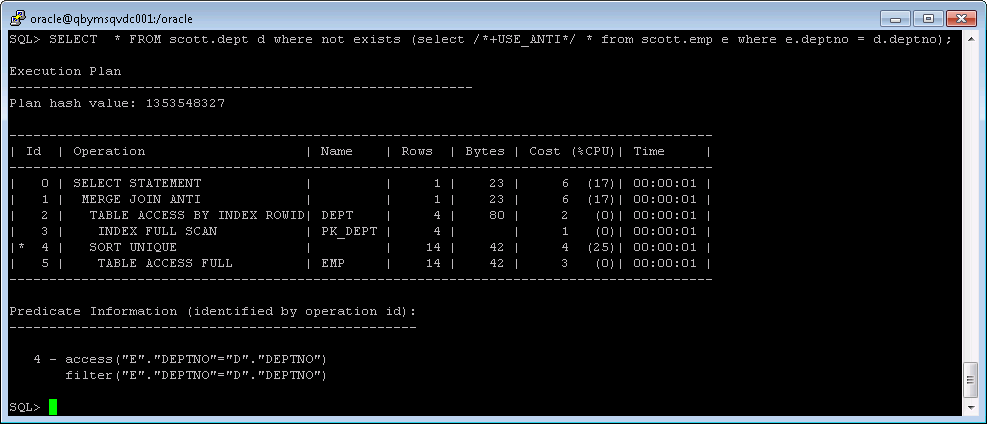
**Task:** Prepare All possible variants of ANTI JOIN SQL **explain plan** using: SQL plus Auto Trace Utility.

**Note:**  If you would like change in execution plan the type of join method use oracle performance hints.

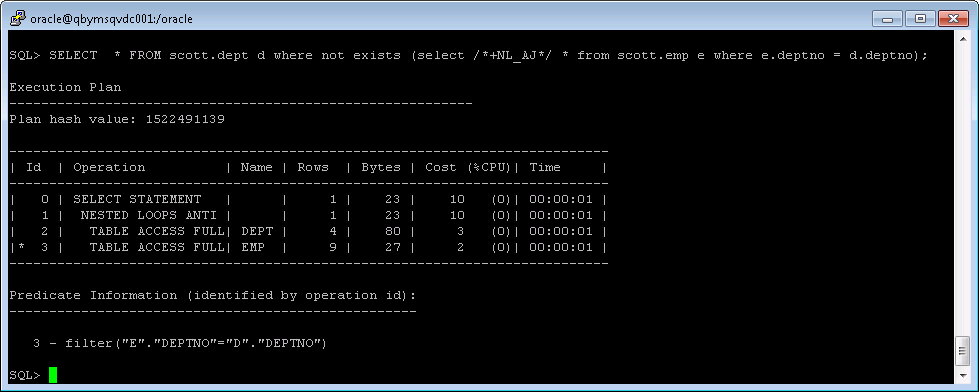
1. ANTIJOIN – perform an anti-join (the optimizer gets to pick which kind)



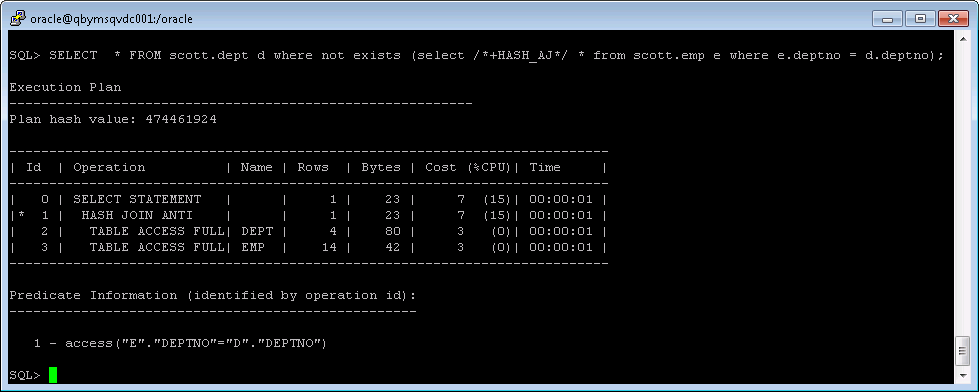
1. USE\_ANTI – older version of ANTIJOIN hint



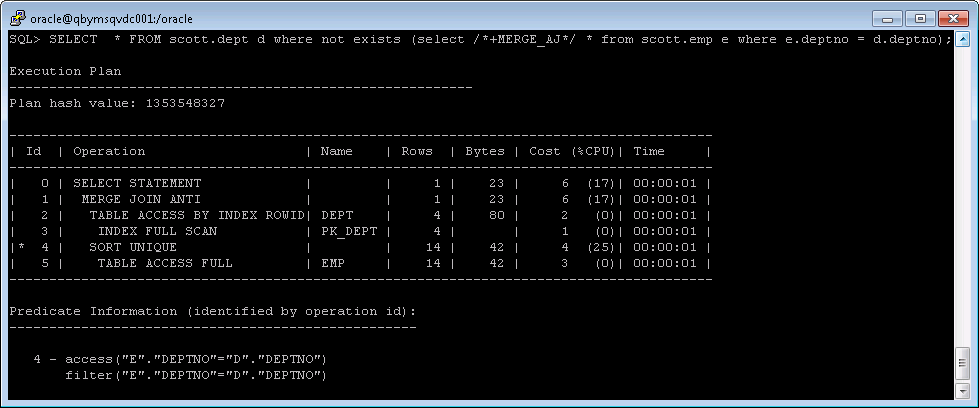
1. NL\_AJ – perform a NESTED LOOPS anti-join (deprecated as of 10g)



1. HASH\_AJ – perform a HASH anti-join (deprecated as of 10g)



1. MERGE\_AJ – perform a MERGE anti-join (deprecated as of 10g)



## Task 10: Prepare summary table

**Task:** Make comparison of all possible variant of join methods and join access methods and fill the table below:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Join Access “A” | Join Access “B” | Nested Loop | Hash Join | Sort-Merge Join | Anti-Join | Semi-Join |
| Small Table | Small Table |  |  |  |  |  |
| Small Table | Indexed Small Table |  |  |  |  |  |
| Big table | Big table |  |  |  |  |  |
| Small table | Index clustered table |  |  |  |  |  |
| Small table | HASH clustered table |  |  |  |  |  |
| Small table | Index unique table |  |  |  |  |  |

**Nested loops** joins use each row of the query result reached through one access operation to drive into

another table. These joins are typically most effective if the result set is limited in size and indexes

are present on the columns used for the join. With nested loops, the cost of the operation is based on

reading each row of the outer row source.

**Sort-merge** joins read the two tables to be joined independently, sorts the rows from each table (but

only those rows that meet the conditions for the table in the WHERE clause) in order by the join key, and

then merges the sorted rowsets. The sort operations are the expensive part for this join method. For

large row sources that won’t fit into memory, the sorts will end up using temporary disk space to

complete. This can be quite memory and time-consuming to complete. But once the rowsets are

sorted, the merge happens quickly. To merge, the database alternates down the two lists, compares the

top rows, discards rows that are earlier in the sort order than the top of the other list, and only returns

matching rows.

**Hash joins**, like sort-merge joins, first reads the two tables to be joined independently and applies the

criteria in the WHERE clause. Based on table and index statistics, the table that is determined to return the

fewest rows will be hashed in its entirety into memory. This hash table includes all the row data for that

table and is loaded into hash buckets based on a randomizing function that converts the join key to a hash

value. As long as there is enough memory available, this hash table will reside in memory. However, if

there is not enough memory available, the hash table may be written to temp disk space.

**Semi-join** is a join between two sets of data (tables) where rows from the first set are returned, based

on the presence or absence of at least one matching row in the other set.

**Anti-joins** are basically the same as semi-joins in that they are an optimization option that can be

applied to nested loop, hash, and merge joins, but are the opposite of semi-joins in terms of

the data they return.