Report

Laboratory Work 4

Dmitry Ladutsko

July 19, 2022

## Task 1: Auto Trace configuration training

|  |  |  |  |
| --- | --- | --- | --- |
| № | Auto Trace Configuration Options | Expected Results | Description |
|  | set autotrace off | No AUTOTRACE report is generated. | This is the default. |
|  | set autotrace on | The AUTOTRACE report | Includes both the optimizer execution path and the SQL statement execution statistics. |
|  | set autotrace traceonly | Like SET AUTOTRACE ON | Suppresses the printing of the user's query output, if any. This option is useful when you are tuning a large query, but do not want to see the query report. |
|  | set autotrace on explain | The AUTOTRACE report | Shows only the optimizer execution path. |
|  | set autotrace on statistics | The AUTOTRACE report | Shows only the SQL statement execution statistics. |
|  | set autotrace on explain statistics | Shows only the SQL statement execution statistics. | = set autotrace on |
|  | set autotrace traceonly explain | Execution plan w/o query result | These options should be used when a large result set is expected. |
|  | set autotrace traceonly statistics | Statistics w/o query result |
|  | set autotrace traceonly explain statistics | Execution plan + statistics w/o query result |
|  | set autotrace off explain | Disables autotrace utility | |
|  | set autotrace off statistics |
|  | set autotrace off explain statistics |

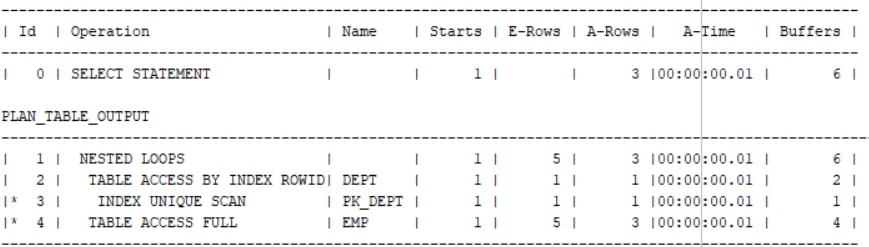
## Task 2: Nested Loop Joins

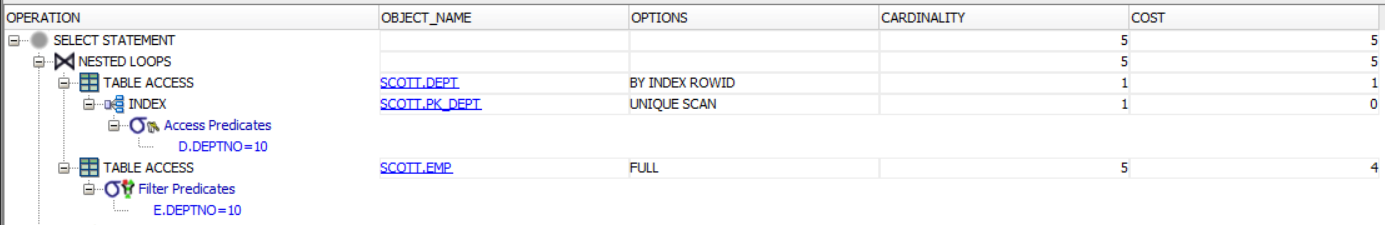
Example:

SELECT /\*+ gather\_plan\_statistics \*/ \*

FROM scott.emp e, scott.dept d

WHERE e.deptno = d.deptno

AND d.deptno = 10 



Picture 2.1 - Query Result

*Note.* I used /\*+ gather\_plan\_statistics \*/ hint because for some reason explain plan statement did not give me full needed stats (only id, operation and name columns). Nested loops join compare every row of table B with current row in table A. If it is a row, which satisfies the specified condition, it is selected, otherwise goes next and compare second raw from table A with all rows from table B. This method is quite useful with small table, but not with big.

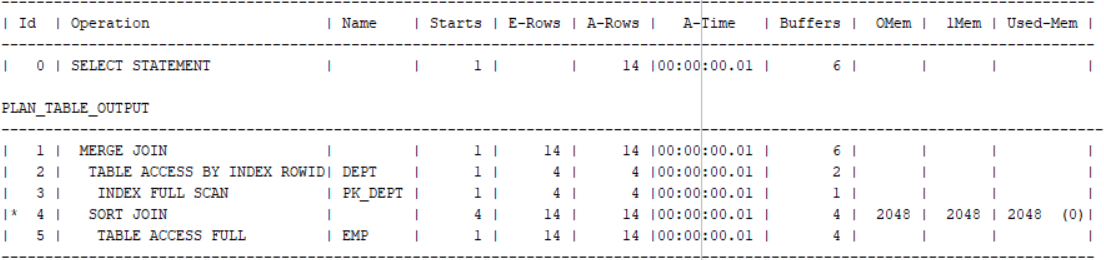
## Task 3: Sort-Merge Joins

set autotrace on explain;

SELECT /\*+ use\_merge(e d) gather\_plan\_statistics \*/ \*

FROM scott.emp e, scott.dept d

WHERE e.deptno = d.deptno



Picture 3.1 - Query Result

*Note*. Using this type of join we need to be sure that all used tables are sorted by columns, mentioned in query condition. Oracle scans each row once. This gives us nice benefits, but we should keep in mind that sorting loads the system a lot if we use big tables.

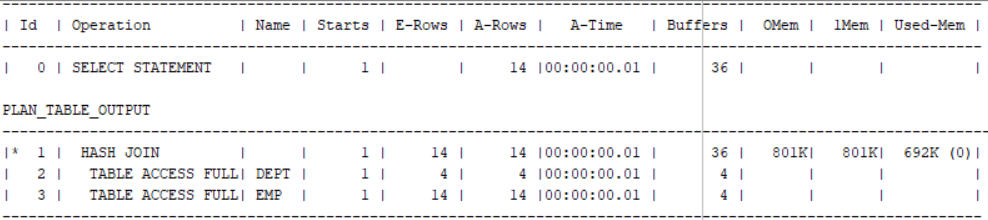
## Task 4: Hash Joins

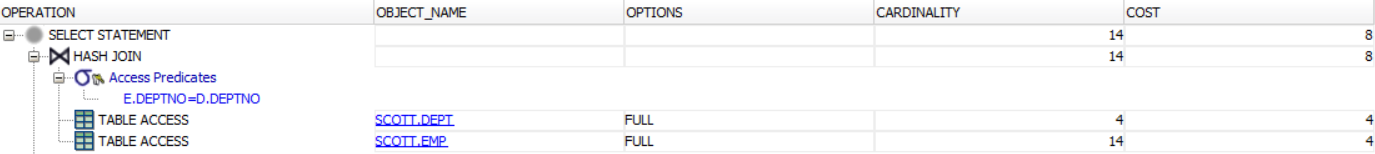
set autotrace on explain;

SELECT /\*+ gather\_plan\_statistics USE\_HASH(e d) \*/ \*

FROM scott.emp e, scott.dept d

WHERE e.deptno = d.deptno



****

Picture 4.1 - Query Result

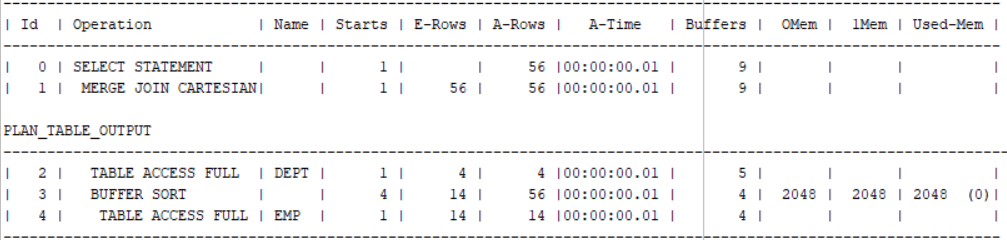
*Note*. Hash join is used when projections of the joined tables are not sorted on the join columns. In this case, the optimizer builds an in-memory hash table on the inner table's join column. The optimizer then scans the outer table for matches to the hash table, and joins data from the two tables accordingly.

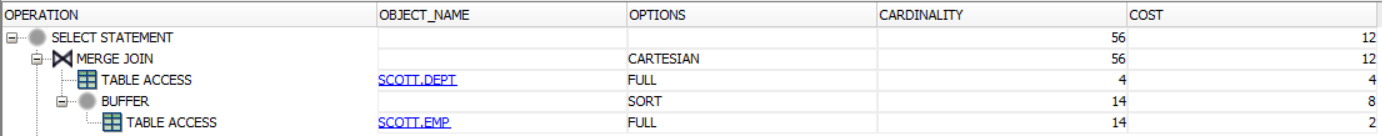
## Task 5: Cartesian Joins

set autotrace on explain;

SELECT /\*+ gather\_plan\_statistics \*/ \*

FROM scott.emp, scott.dept





Picture 5.1 - Query Result

*Note*. Cartesian joins are used when all rows in all tables listed in a query: each row in the first table is paired with all the rows in the second table. This happens when there is no relationship defined between the two tables.

## Task 6: Left/Right Outer Joins

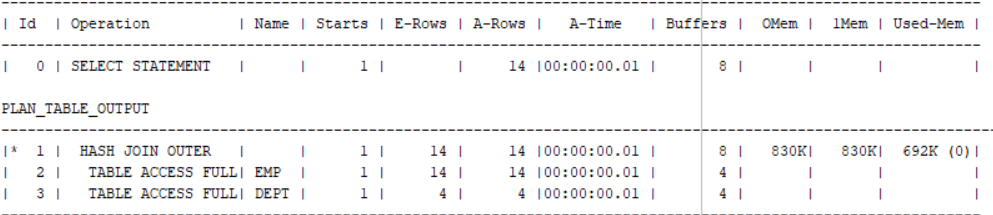
1. Left outer JOIN

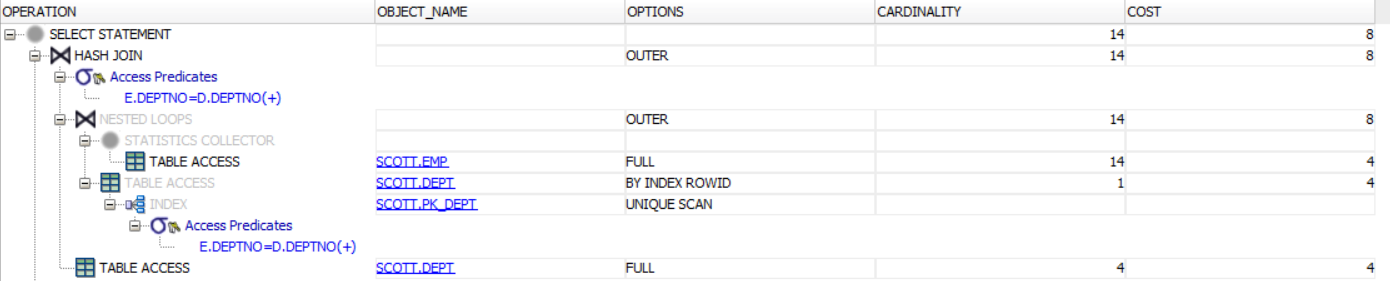
set autotrace on explain;

select /\*+ gather\_plan\_statistics \*/ e.ename , e.deptno, e.job, d.dname

from scott.emp e, scott.dept d

where e.deptno = d.deptno(+);





Picture 6.1 - Query Result

*Note*. The specific of this type of join is that result includes unmatched rows from only the table that is specified before the LEFT OUTER JOIN clause. If we are joining two tables and want the result set to include unmatched rows from only one table, we use a LEFT OUTER JOIN clause (or a RIGHT OUTER JOIN) clause.

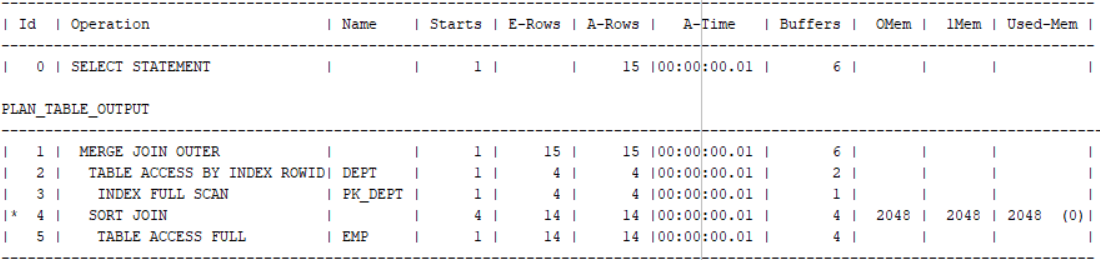
1. Right outer JOIN

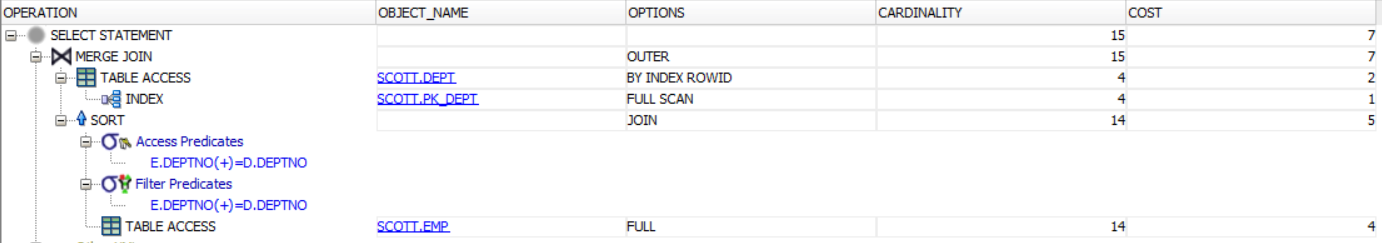
set autotrace on explain;

select /\*+ gather\_plan\_statistics \*/ e.ename , e.deptno, e.job, d.dname

from scott.emp e, scott.dept d

where e.deptno(+) = d.deptno;





Picture 6.2 - Result

## Task 7: Full Outer Join

set autotrace on explain;

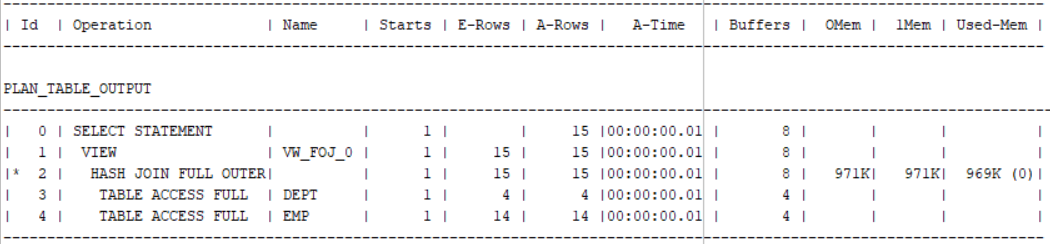
select /\*+ gather\_plan\_statistics \*/ e.ename, e.deptno, e.job, d.dname

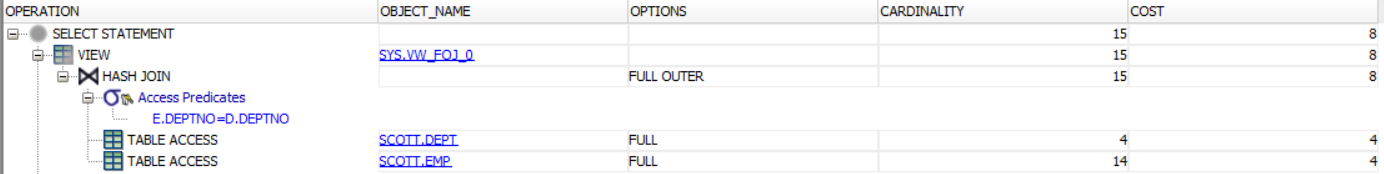
from scott.emp e

full outer join

scott.dept d

on(e.deptno = d.deptno);





Picture 7.1 - Result

*Note*. We use Full outer join if we want the result set to include unmatched rows from both tables.

## 

## Task 8: Semi Joins

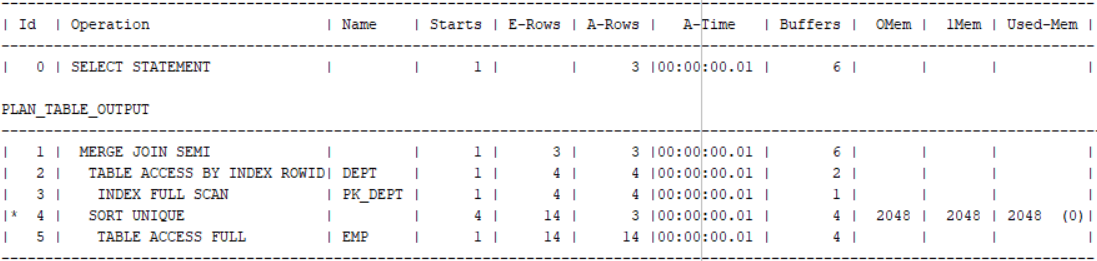
1. Using /\*semijoin\*/ :

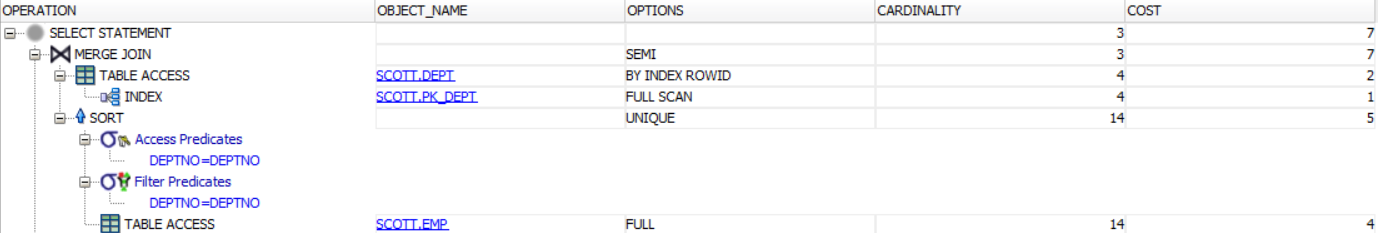
set autotrace on explain;

select /\*+ gather\_plan\_statistics semijoin \*/ dname

from scott.dept d

where deptno in (select deptno from scott.emp e);





Picture 8.1 -Query Result

*Note*. A semi-join returns one copy of each row in first table for which at least one match is found. Semi-joins are also written using the EXISTS construct.

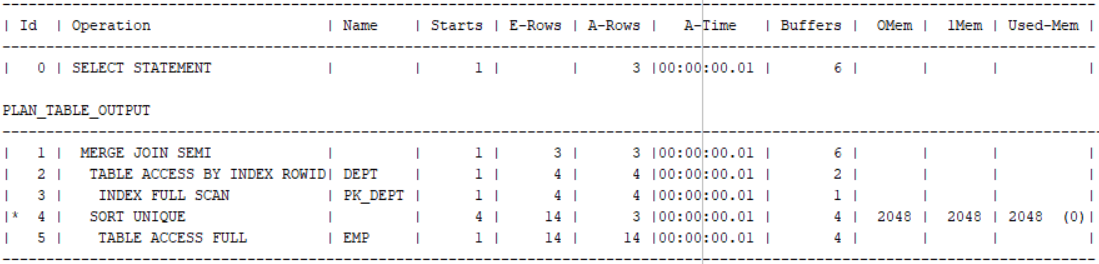
1. Using /\*no\_semijoin\*/ :

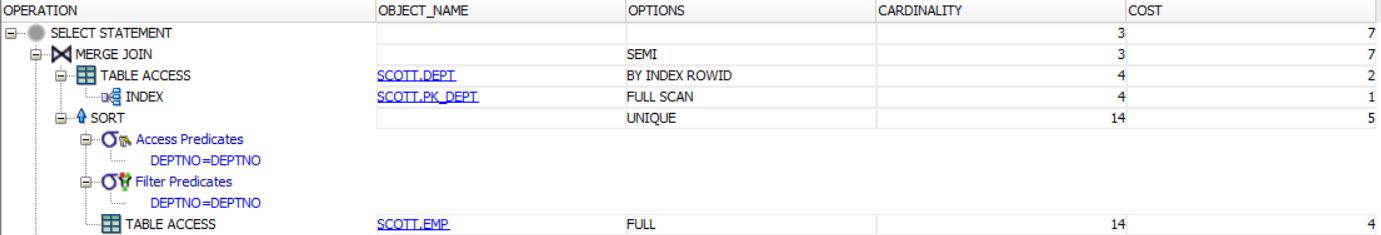
set autotrace on explain;

select /\*+ gather\_plan\_statistics no\_semijoin \*/ dname

from scott.dept d

where deptno in (select deptno from scott.emp e);





Picture 8.2 -Query Result

*Note*. As a consequence, NO vital difference

1. Using /\*using exists\*/

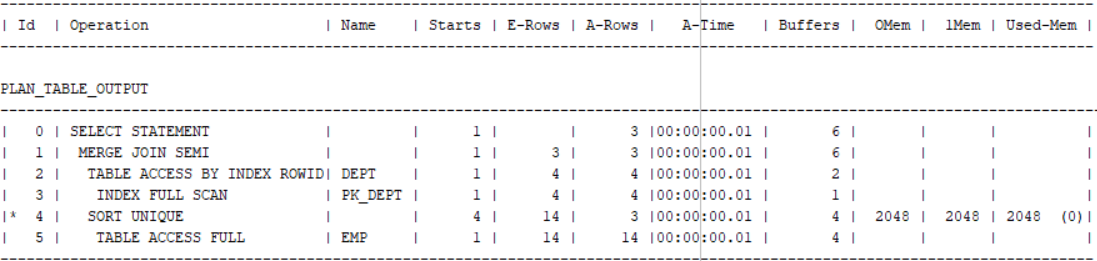
set autotrace on explain;

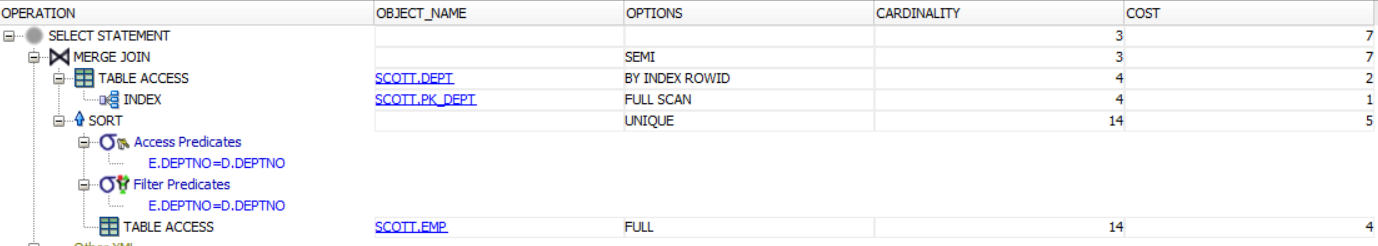
select /\*+ gather\_plan\_statistics \*/ dname

from scott.dept d

where exists (select null from scott.emp e

where e.deptno = d.deptno);





Picture 8.3 -Query Result

*Note*. The EXISTS function checks to find a single matching row to return the result in a subquery. Because the IN function retrieves and checks all rows, so it is slower.

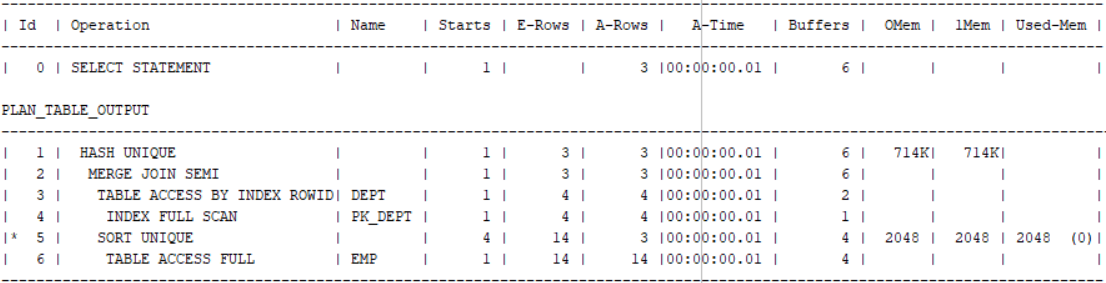
1. Select distinct using /\* inner join with distinct \*/ hint

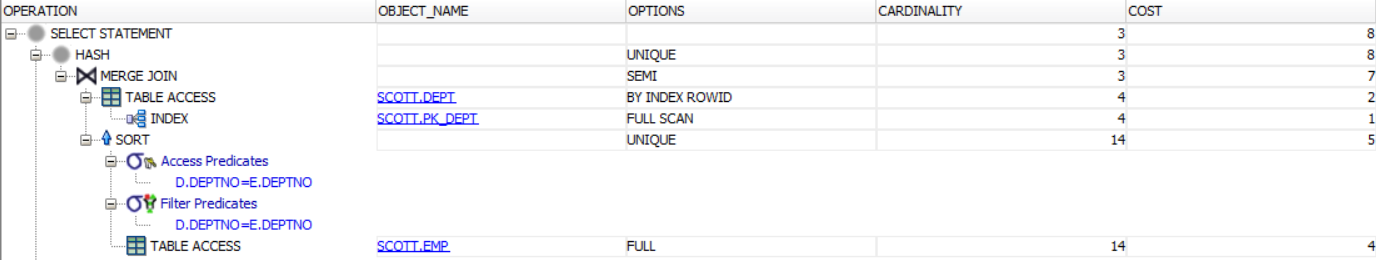
set autotrace on explain;

select /\*+ gather\_plan\_statistics inner join with distinct \*/ distinct dname

from scott.dept d ,scott.emp e

where d.deptno = e.deptno ;





Picture 8.4 -Query Result

*Note*. The DISTINCT clause is used in a SELECT statement to filter duplicate rows in the result set. This way we can ensure that returned rows are unique for the column or columns specified in the SELECT clause.

## Task 9: Anti Joins

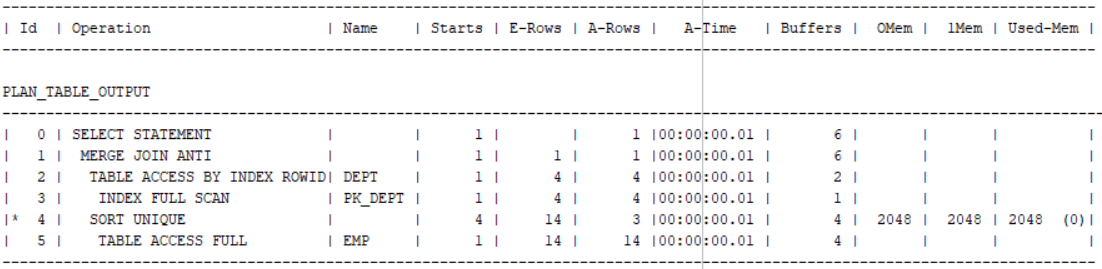
1. Not exists using /\*not exists\*/ hint

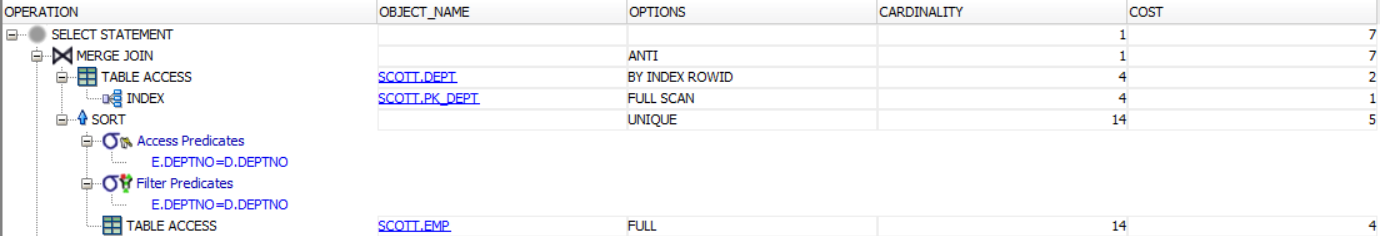
set autotrace on explain;

select /\*+ gather\_plan\_statistics not exists ANTIJOIN \*/ dname

from scott.dept d

where not exists (select null from scott.emp e

where e.deptno = d.deptno); 



*Note*. The NOT EXISTS operator returns true if the subquery returns no row. Otherwise, it returns false. We should keep in mind that NOT EXISTS operator returns false if the subquery returns any rows with a NULL value.

1. Not in using /\*not in\*/ hint

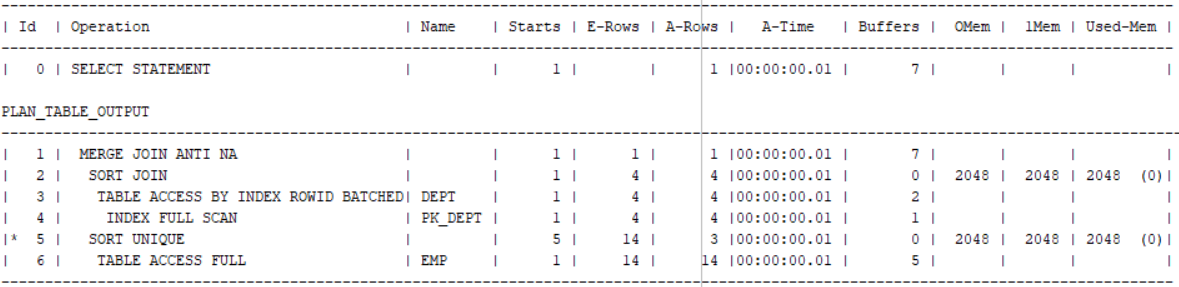
set autotrace on explain;

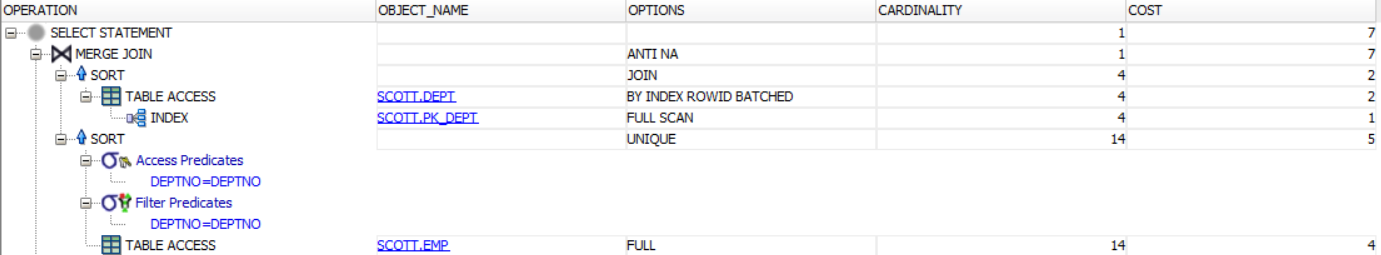
select /\*+ gather\_plan\_statistics not in \*/ dname

from scott.dept d

where deptno not in

(select deptno from scott.emp e)





*Note*. Using NOT IN operator can be risky because it does not return null values

NOT IN can be just as efficient as NOT EXISTS even if an anti-join can be used (if the subquery is known to not return nulls)

1. Minus using /\*minus\*/ hint

set autotrace on explain;

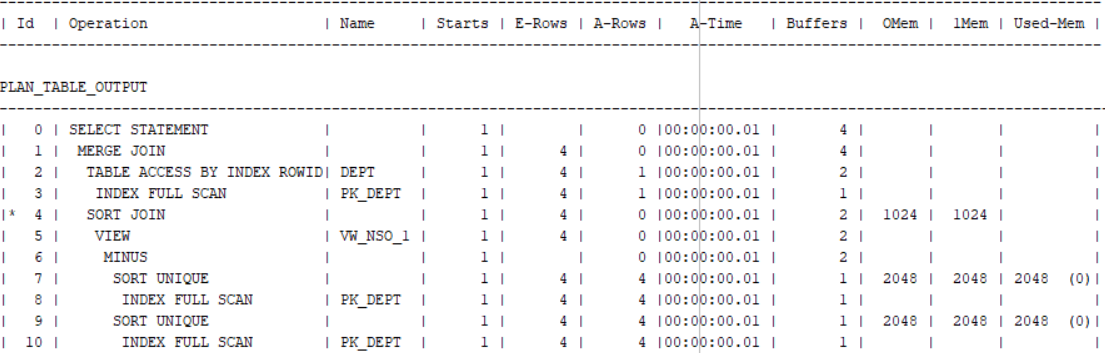
select /\*+ gather\_plan\_statistics minus\*/ dname

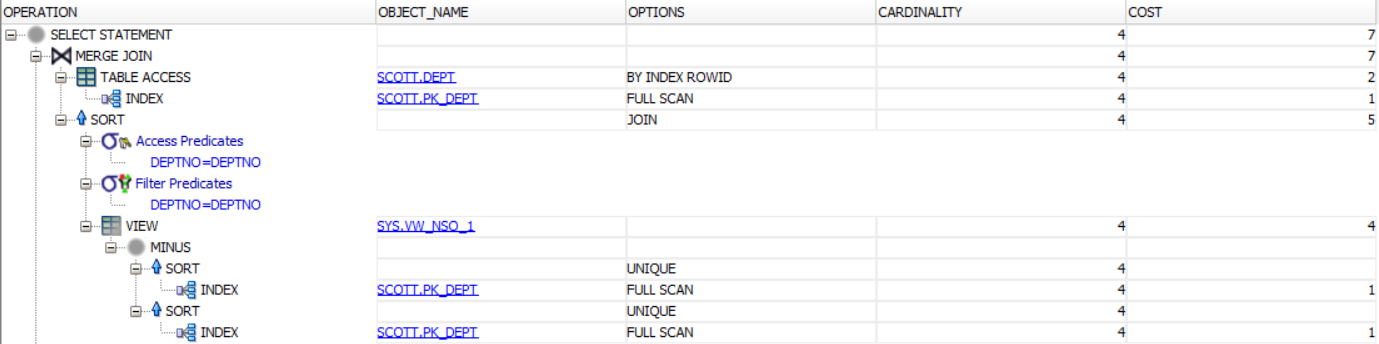
from scott.dept

where deptno in

(select deptno from scott.dept minus

select deptno from scott.dept);





*Note*. The Oracle MINUS operator is used to return all rows in the first SELECT statement that are not returned by the second SELECT statement. Each SELECT statement will define a dataset. The MINUS operator retrieve all records from the first dataset and then remove from the results all records from the second dataset.

1. Left outer anti join

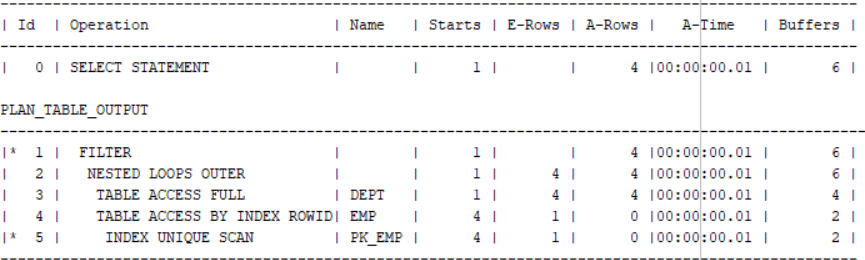
set autotrace on explain;

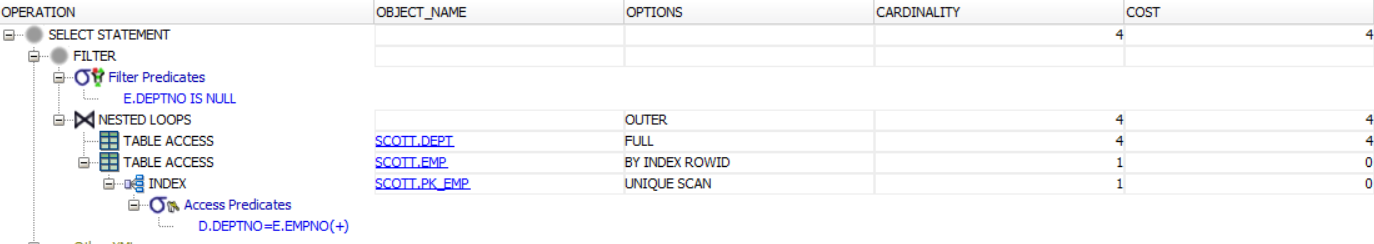
select /\*+ gather\_plan\_statistics left outer \*/ dname

from scott.dept d, scott.emp e

where d.deptno = e.empno(+)

and e.deptno is null;





Picture 9.1 -Query Result

## Task 10: Summary table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Join Access “A” | Join Access “B” | Nested Loop | Hash Join | Sort-Merge Join | Anti-Join | Semi-Join |
| Small Table | Small Table | Good | Ineffective | Ineffective | Contextual | |
| Small Table | Small Table(Indexed) |
| Small Table(Indexed) | Small Table(Indexed) | Good |
| Big table | Big table | Awful | Good | Ineffective |
| Big table(Indexed) | Big table |
| Big table(Indexed) | Big table(Indexed) | Good |
| Small | Big | Ineffective | |

## Laboratory work summary:

We used many join methods to see which has better performance in different situations.

At the table above and insights from each task (or sub – task) you can see some results and conclusion.

