**Join methods**

1. Nested loop join :

* The optimizer determines the driving row source and designates it as the outer loop.

The outer loop produces a set of rows for driving the join condition. The row source can be a table accessed using an index scan, a full table scan, or any other operation that generates rows.

The number of iterations of the inner loop depends on the number of rows retrieved in the outer loop. For example, if 10 rows are retrieved from the outer table, then the database must perform 10 lookups in the inner table. If 10,000,000 rows are retrieved from the outer table, then the database must perform 10,000,000 lookups in the inner table.

* The optimizer designates the other row source as the inner loop.
* For every fetch request from the client, the basic process is as follows:

Fetch a row from the outer row source

Probe the inner row source to find rows that match the predicate criteria

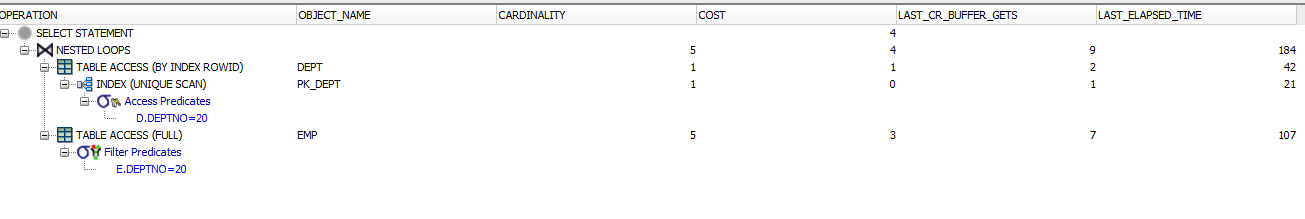
Repeat the preceding steps until all rows are obtained by the fetch request

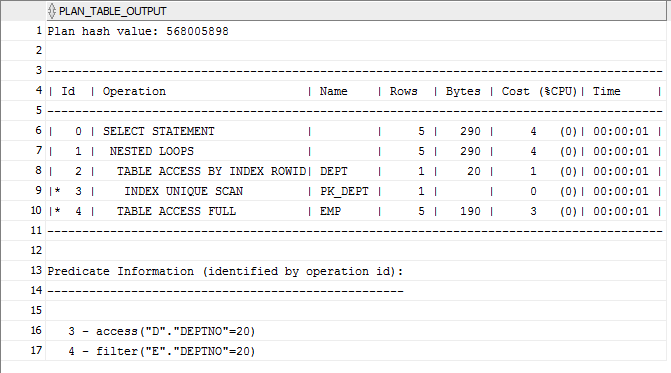
SELECT \*

FROM scott.emp e, scott.dept d

WHERE e.deptno = d.deptno

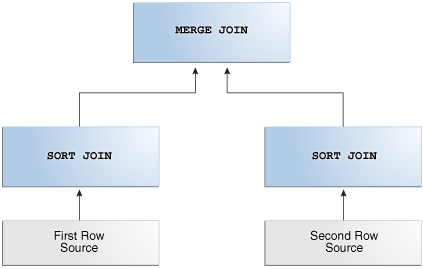
AND d.deptno = 20

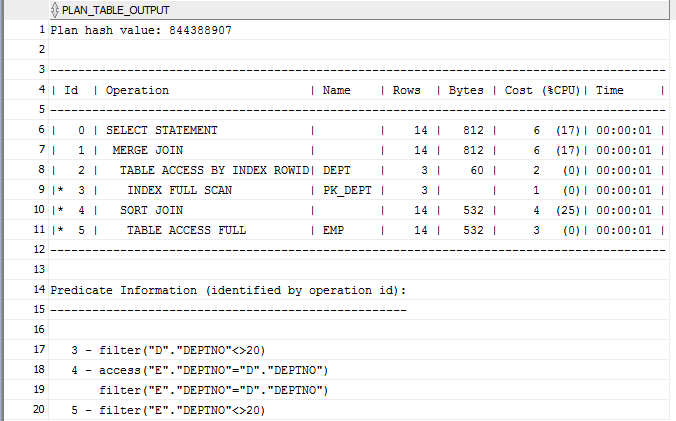


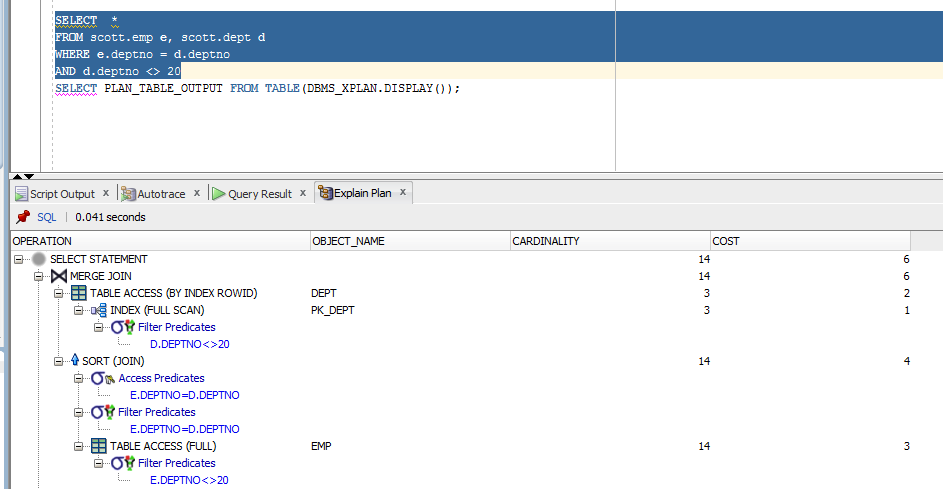


1. Sort-Merge Join

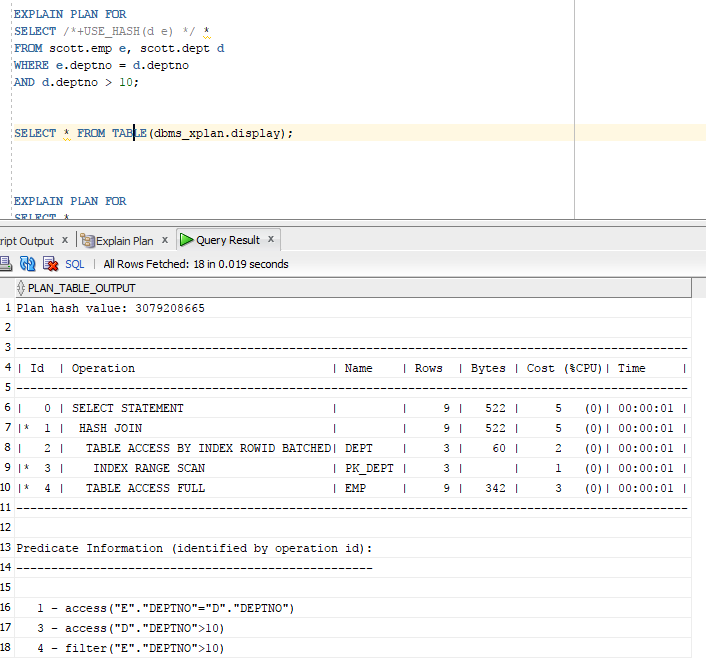
If the two data sets in the join are not already sorted, then the database sorts them. These are the SORT JOIN operations. For each row in the first data set, the database probes the second data set for matching rows and joins them, basing its start position on the match made in the previous iteration. This is the MERGE JOIN operation.



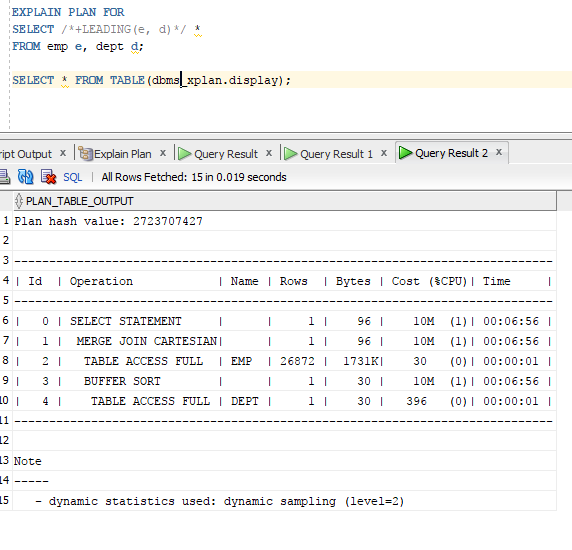


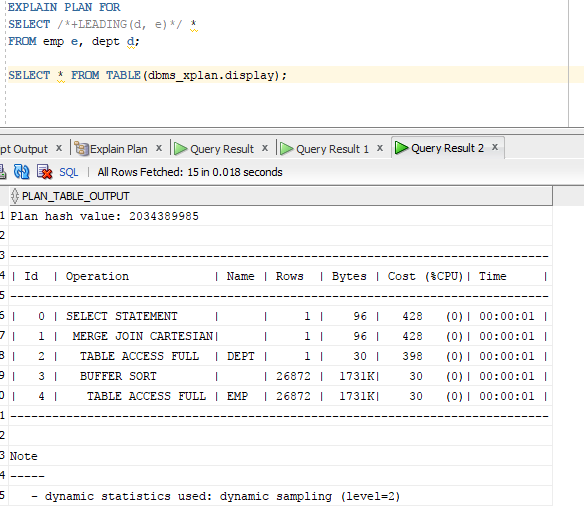


**Task 4 Hash Join**



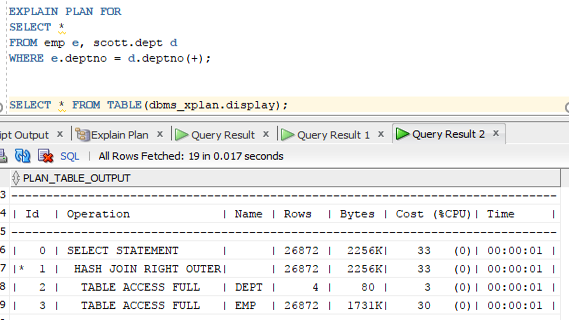
**Task 5 Cartesian Join**

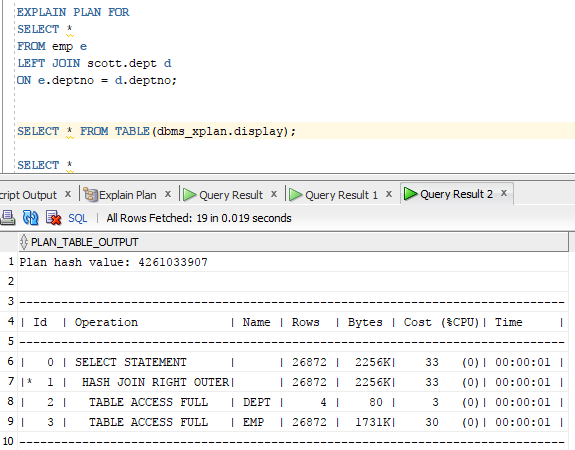




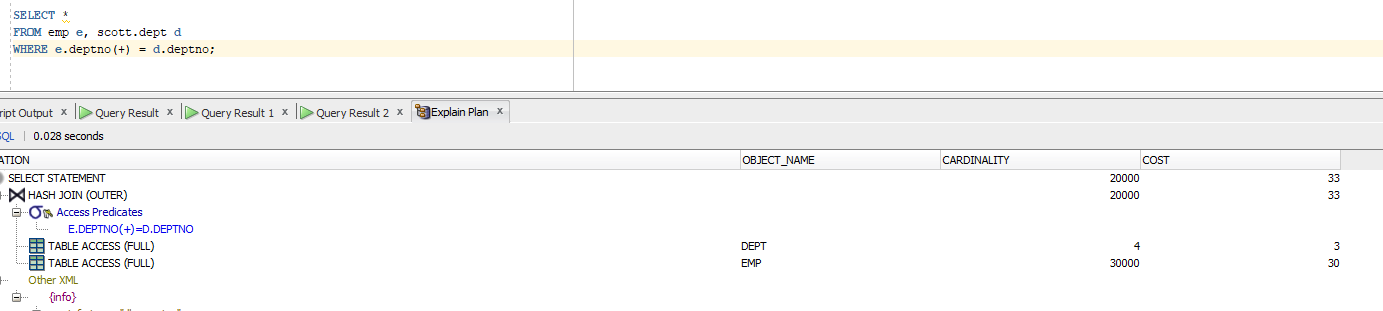
**Task 6 Outer Join**

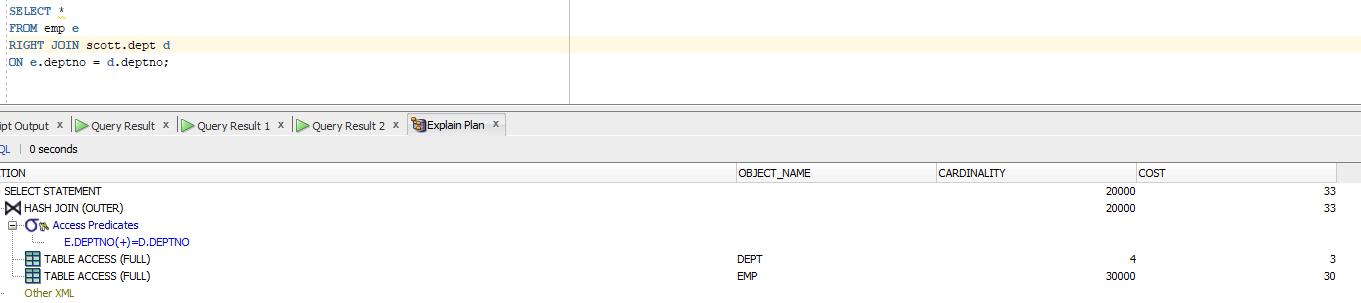
1. Left Join





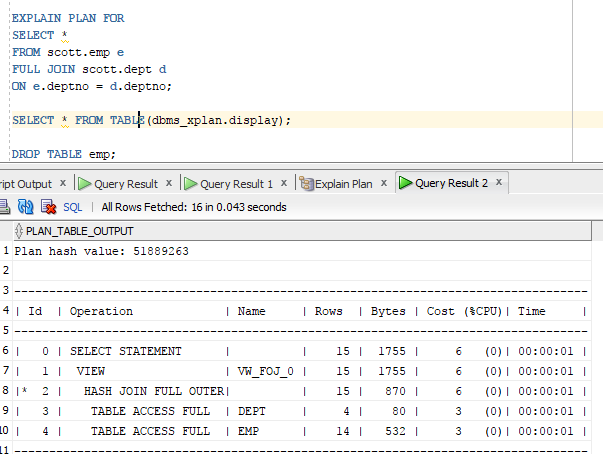
b) Right Join



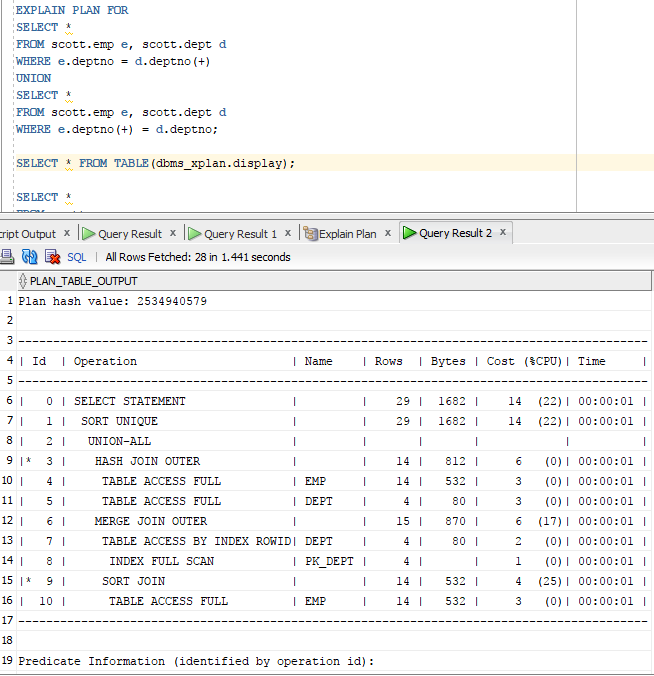


**Task 7 Full Outer Join**

1. ANSI



1. Oracle syntax



**Task 8**

|  |  |  |
| --- | --- | --- |
| Table “A” | Table “B” | Join type description |
| Small Table w/o index on join field | Small Table w/ index on join field | Nested loop(small tables and index) |
| Small Table w/o index on join field | IOT | Merge Join(join with keys, tables approximately the same in size ) |
| Large Table w/o index on join field | Small Table w/ index on join table | Nested Loop ( PK – required condition)  Small Table is limited by 1 value, so Nested Loop could be used effective to go in one cycle through all large table |
| Medium table without index on join field | Medium table without index on join field | Hash Join(  The joining in this case is better done by hashing one of the tables, because sorting is too expensive) |
| Medium table w/ index on join field | Small table w/ index on join field | With hint /\* + ORDERED / Oracle takes larger table, and after that smaller. It understands that Nested Loop doesn’t execute and choose Cartesian Join instead of it. |
| Any size table | Any size table | Cartesian Join (With no condition of joining every time will be Cartesian join). |
| Large table with index on join key | Large table with index on join key | Hash Join(  The joining in this case is better done by hashing one of the tables, because sorting is too expensive) |
| Large table without index on join field | Small table without index on join field | Hash Join (move small table to hash and turn to it from the large table) |
| Large table with index on join field | Small table with index on join field | Merge Join (one table is sorted by its PK so there is no need to turn to manual sort. This table can be easily proceed by large table in merge join method |
| IOT | IOT | Hash Join(Joining with key and move smaller table to hash and) |