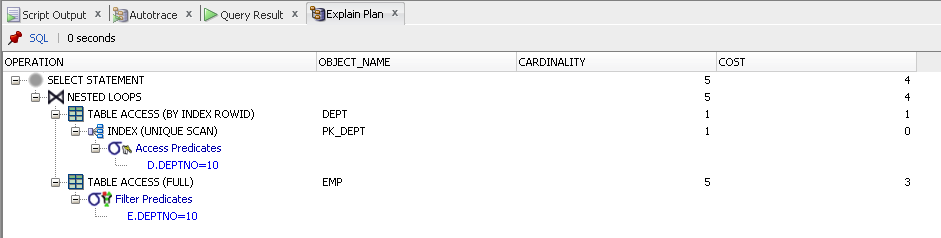
1. **Nested loop join**

select \* from scott.emp e

join scott.dept d

on e.deptno =d.deptno

where d.deptno=10;



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

Nested loops joins are useful when the database joins small subsets of data.

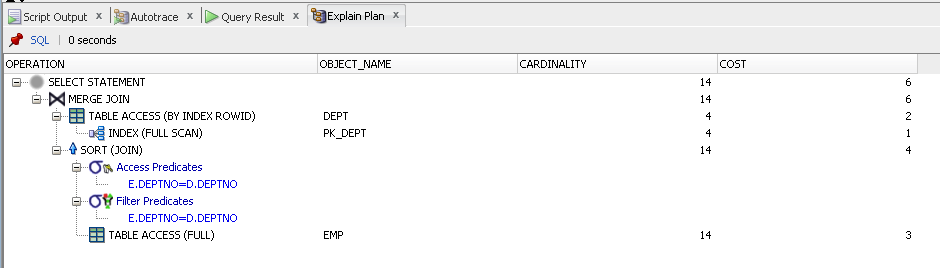
If a row source has only one row, as with an equality lookup on a primary key value then the join is a simple lookup.

1. **Sort-merge join**

select /\*+USE\_MERGE\*/\* from scott.emp e

join scott.dept d

on e.deptno =d.deptno;



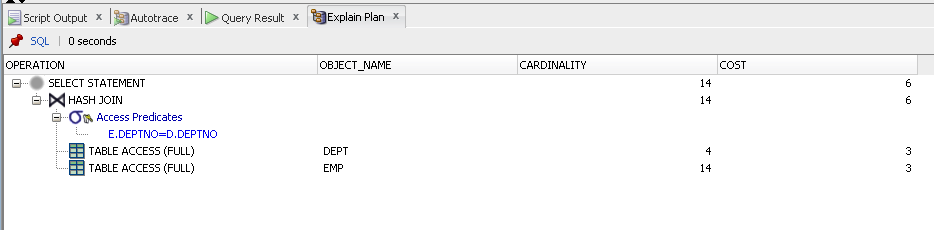
Sort-merge join reads the two tables, sorts the rows from each table in order by the join key, and then merges the sorted rows.

1. **Hash join**

select /\*+USE\_HASH(d,e)\*/\* from scott.emp e

join scott.dept d

on e.deptno =d.deptno;

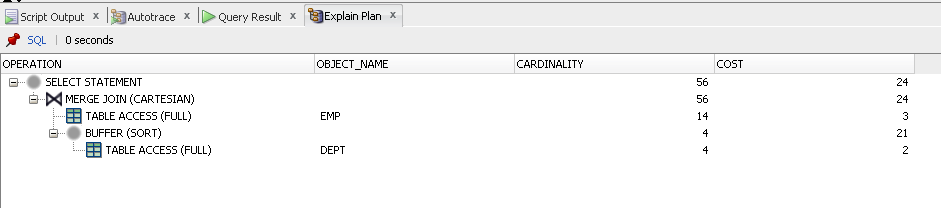


Hash join reads two tables and applies filters, less table will be cashed, then larger table will be read and the hash function is applied to the join key column to match columns.

1. **Cartesian join**

select /\*+ leading(e,d) use\_nl(e) \*/ \* from scott.emp e,

scott.dept d;



Cartesian joins are used when all rows from one table should be joined to all rows from other table.

1. **Left/Right outer joins**

**ANSI**

select \* from scott.emp e

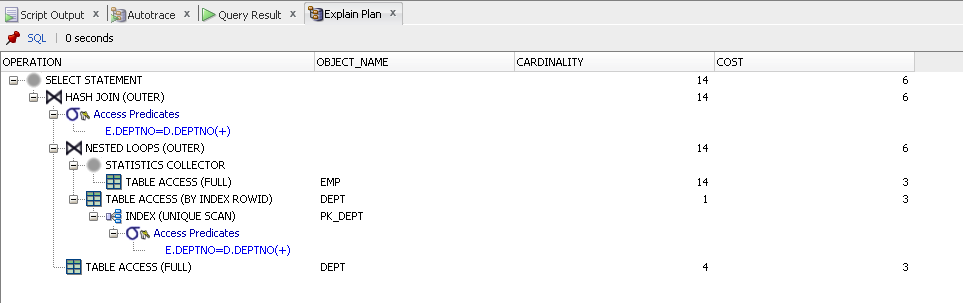
left join scott.dept d

on e.deptno=d.deptno;

**Oracle syntax**

select \* from scott.emp e, scott.dept d

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



Left join is used when we need to see all rows from the left table and relevant rows from the right table.

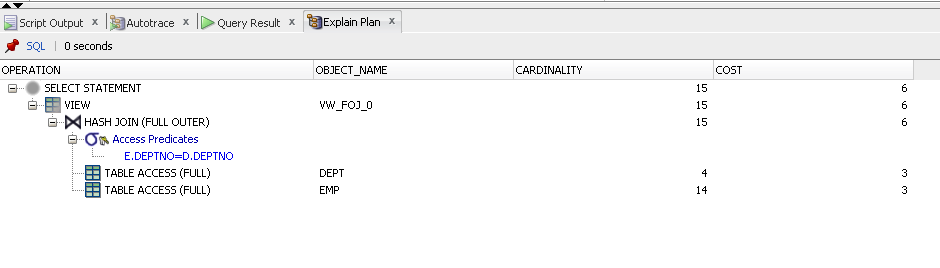
1. **Full outer join**

**ANSI:**

select \* from scott.emp e

full join scott.dept d

on e.deptno=d.deptno;



**Oracle syntax:**

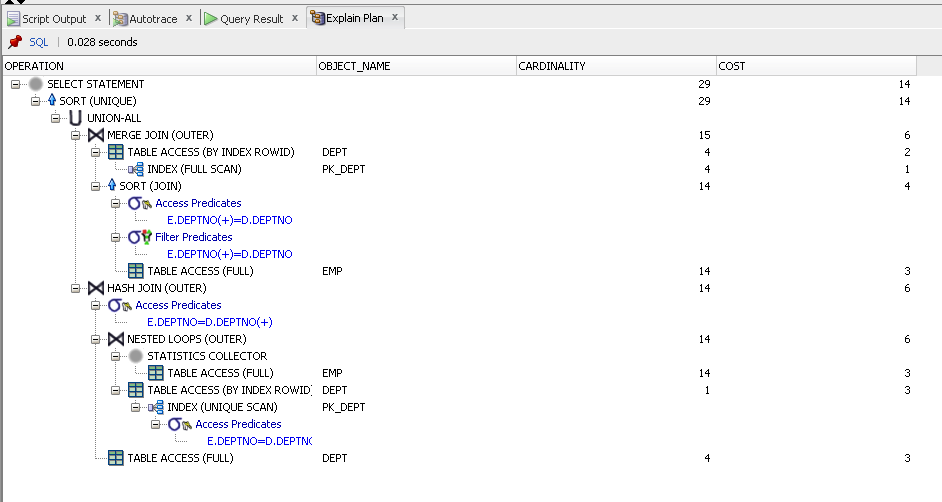
select \* from scott.emp e,scott.dept d

where e.deptno(+)=d.deptno

union

select \* from scott.emp e,scott.dept d

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



Full outer join returns all rows fromboth tables that match plus the rows that are unique to each table.

**Tables for exercise 7:**

CREATE TABLE **big\_table\_with\_index** AS

SELECT trunc(DBMS\_RANDOM.VALUE\*1000000) id, rpad( rownum, 100 ) rown, rpad('big table', 100) text

FROM dual

CONNECT BY rownum < 1000000;

CREATE INDEX bug\_idx ON big\_table\_with\_index ( id );

CREATE TABLE **big\_table\_without\_index** AS

SELECT trunc(DBMS\_RANDOM.VALUE\*1000000) id, rpad( rownum, 100 ) rown, rpad('big table', 100) text

FROM dual

CONNECT BY rownum < 1000000;

create table **small\_table\_with\_index** as

SELECT trunc(DBMS\_RANDOM.VALUE\*100) id, rownum rown, 'small table' text

FROM dual

CONNECT BY rownum < 20;

create index sm\_idx on small\_table\_with\_index (id);

create table **small\_table\_without\_index** as

SELECT trunc(DBMS\_RANDOM.VALUE\*100) id, rownum rown, 'small table' text

FROM dual

CONNECT BY rownum < 20;

create table **small\_table\_without\_index2** as

SELECT trunc(DBMS\_RANDOM.VALUE\*100) id, rownum rown, 'small table' text

FROM dual

CONNECT BY rownum < 20;

CREATE TABLE **megium\_table\_without\_index** AS

SELECT trunc(DBMS\_RANDOM.VALUE\*100) id, rpad( rownum, 20 ) rown, rpad('mediun table', 20) text

FROM dual

CONNECT BY rownum < 300;

CREATE TABLE **megium\_table\_with\_index** AS

SELECT trunc(DBMS\_RANDOM.VALUE\*100) id, rpad( rownum, 20 ) rown, rpad('mediun table', 20) text

FROM dual

CONNECT BY rownum < 300;

create index med\_idx on megium\_table\_with\_index (id);

|  |  |  |  |
| --- | --- | --- | --- |
| **Table A** | **Table B** | **Query** | **Join type description** |
| big\_table\_with\_index | big\_table\_without\_index | select \* from big\_table\_with\_index bw  join big\_table\_without\_index bo  on bw.id=bo.id; | **Hash join**  This join type was chosen because of large amount of data and equijoin. |
| small\_table\_with\_index | Small\_table\_without\_index | select \* from small\_table\_with\_index si  join small\_table\_without\_index so  on si.id=so.id; | **Merge join**  This join type was chosen because of small amoun of data and using index. So sorting for merge join is more effective than loop join |
| Small\_table\_without\_index | Small\_table\_without\_index | select \* from small\_table\_without\_index s1  join small\_table\_without\_index2 s2  on s1.id=s2.id; | **Hash join**  This join type was chosen because of the lack of indexes |
| Big\_table\_with\_index | Small\_table\_with\_index | select \* from big\_table\_with\_index bi  join small\_table\_with\_index si  on bi.id=so.id; | **Hash join(Nested loop inside)**  This join type was chosen because of large amount of data and indexes |
| Big\_table\_without\_index | Small\_table\_without\_index | select \* from big\_table\_without\_index bo  join small\_table\_without\_index so  on bo.id=so.id; | **Hash join**  This join type was chosen because of big amount of data and lack of index |
| Big\_table\_without\_index | Big\_table\_without\_index2 | select \* from big\_table\_without\_index b1  join big\_table\_without\_index2 b2  on bo.id=so.id  order by bo.id; | **Merge join**  This join type was chosen because of order by statement |
| Small\_table\_with\_index | Small\_table\_with\_index | select \* from small\_table\_with\_index s1  join small\_table\_with\_index2 s2  on s1.id=s2.id  where s2.id=1; | **Merge join(cartesian)**  This join type was chosen because of small tables and filter with = |