U1M5.LW.Access and Join Methods

Part 2

Shkrabatouskaya Vera

https://github.com/VeraShkrabatouskaya/DataMola_Data-Camping-2022

1. Auto Trace & Explain Plan

1.1. Task 1: Auto Trace configuration training

Step 1: Learning all possible variants of SQL plus utilities autotrace

№	Auto Trace Configuration Options	Description
1	`set autotrace off`	No AUTOTRACE report is generated. This is the default. Disables all autotrace.
2	`set autotrace on`	The AUTOTRACE report includes both the optimizer execution path and the SQL statement execution statistics.
3	`set autotrace traceonly`	Like SET AUTOTRACE ON, but suppresses the printing of the user's query output, if any. If STATISTICS is enabled, query data is still fetched, but not printed.
4	`set autotrace on explain`	The AUTOTRACE report shows only the optimizer execution path.
5	`set autotrace on statistics`	The AUTOTRACE report shows only the SQL statement execution statistics.
6	`set autotrace on explain statistics`	The AUTOTRACE report includes both the optimizer execution path and the SQL statement execution statistics.
7	`set autotrace traceonly explain`	The AUTOTRACE report includes the optimizer execution path, but suppresses the printing of the user's query output.
8	`set autotrace traceonly statistics`	The AUTOTRACE report includes the SQL statement execution statistics, but suppresses the printing of the user's query output.
9	`set autotrace traceonly explain statistics`	The AUTOTRACE report includes both the optimizer execution path and the SQL statement execution statistics, but suppresses the printing of the user's query output.
10	`set autotrace off explain`	Autotrace function is disabled.

11	`set autotrace off statistics`	Autotrace function is disabled.
12	`set autotrace off explain statistics`	Autotrace function is disabled.

Summary: AUTOTRACE is a facility within SQL*Plus to show us the explain plan of the queries we've executed, and the resources they used. The autotrace provides instantaneous feedback including the returned rows, execution plan, and statistics. The user doesn't need to be concerned about trace file locations and formatting since the output is displayed instantly on the screen.
EXPLAIN – generate explain plan once SQL statement is finished.
STATISTICS – generate usage statistics once SQL statement is finished.
TRACEONLY – fetch all data for executed SQL but don’t display the data – very useful for testing purposes of huge queries combinations of all above options.

2. Join Methods

2.1. Task 2: Nested Loops Joins

Step 1: Script to create Oracle's "SCOTT" schema (tables EMP, DEPT, BONUS, SALGRADE)

Step 2: Data results

```
SELECT  *
    FROM emp e, dept d
   WHERE e.deptno = d.deptno
     AND d.deptno    = 10
```

Script Output | Query Result | SQL | All Rows Fetched: 3 in 4.181 seconds

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	DEPTNO_1	DNAME	LOC
1	7782	CLARK	MANAGER	7839	09-JUN-81	2450	(null)	10	10	ACCOUNTING	NEW YORK
2	7839	KING	PRESIDENT	(null)	17-NOV-81	5000	(null)	10	10	ACCOUNTING	NEW YORK
3	7934	MILLER	CLERK	7782	23-JAN-82	1300	(null)	10	10	ACCOUNTING	NEW YORK

Step 3: Using option Set Autotrace Traceonly Explain

```
set autotrace traceonly explain;
SELECT  *
    FROM scott.emp e, scott.dept d
   WHERE e.deptno = d.deptno
     AND d.deptno    = 10
```

Script Output | Query Result | Task completed in 1.028 seconds

```
PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------
SQL_ID  287skc8v2zgq0, child number 0
-------------------------------------
SELECT  *      FROM scott.emp e, scott.dept d     WHERE e.deptno =
d.deptno       AND d.deptno    = 10

Plan hash value: 568005898

-----------------------------------------------------------
| Id  | Operation                    | Name    | E-Rows |
-----------------------------------------------------------
|   0 | SELECT STATEMENT             |         |        |

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------
|   1 |  NESTED LOOPS                |         |      3 |
|   2 |   TABLE ACCESS BY INDEX ROWID| DEPT    |      1 |
|*  3 |    INDEX UNIQUE SCAN         | PK_DEPT |      1 |
|*  4 |   TABLE ACCESS FULL          | EMP     |      3 |
-----------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
```

```
set autotrace traceonly explain;
SELECT  *
     FROM scott.emp e, scott.dept d
    WHERE e.deptno = d.deptno
      AND d.deptno    = 10
```

Script Output | Query Result

Task completed in 1.028 seconds

```
Statistics
----------------------------------------------------------
          1  CPU used by this session
          1  CPU used when call started
         32  Requests to/from client
         33  SQL*Net roundtrips to/from client
          5  buffer is not pinned count
        585  bytes received via SQL*Net from client
      60831  bytes sent via SQL*Net to client
          2  calls to get snapshot scn: kcmgss
          3  calls to kcmgcs
          6  consistent gets
          2  consistent gets examination
          2  consistent gets examination (fastpath)
          6  consistent gets from cache
          4  consistent gets pin
          4  consistent gets pin (fastpath)
          2  execute count
          1  index fetch by key
     196608  logical read bytes from cache
          3  no work - consistent read gets
         33  non-idle wait count
          2  opened cursors cumulative
          2  opened cursors current
          2  parse count (total)
          1  parse time cpu
```

Set Autotrace Off Explain

In a Nested loops join, we have two tables a driving table and a secondary table. We see that the DEPT table is connected to the EMP table by Nested loops join. The DEPT table is considered first. This connection required 6 consistent gets.

```
explain plan for
SELECT  *
     FROM scott.emp e, scott.dept d
    WHERE e.deptno = d.deptno
      AND d.deptno    = 10;

select * from table (dbms_xplan.display);
```

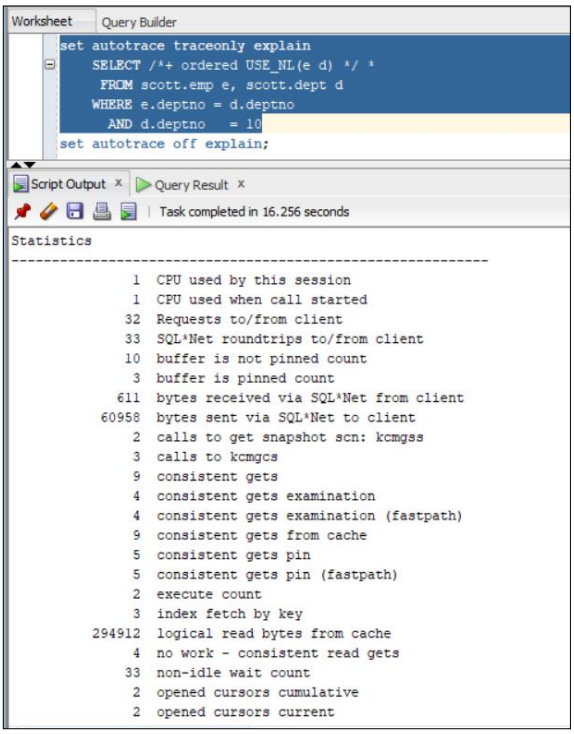
Query Result | SQL | All Rows Fetched: 17 in 0.032 seconds

```
PLAN_TABLE_OUTPUT
Plan hash value: 568005898

-------------------------------------------------------------------------------------
| Id  | Operation                    | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |         |     3 |   177 |     5   (0)| 00:00:01 |
|   1 |  NESTED LOOPS                |         |     3 |   177 |     5   (0)| 00:00:01 |
|   2 |   TABLE ACCESS BY INDEX ROWID| DEPT    |     1 |    20 |     1   (0)| 00:00:01 |
|*  3 |    INDEX UNIQUE SCAN         | PK_DEPT |     1 |       |     0   (0)| 00:00:01 |
|*  4 |   TABLE ACCESS FULL          | EMP     |     3 |   117 |     4   (0)| 00:00:01 |
-------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   3 - access("D"."DEPTNO"=10)
   4 - filter("E"."DEPTNO"=10)
```

If we use the explanation plan for select, we can notice that the cost is 5.

Let’s change in the execution plan the type of join method use oracle performance hints. Using the USE_NL hint, let’s change the order in which the tables are joined. The table Emp will be considered first.

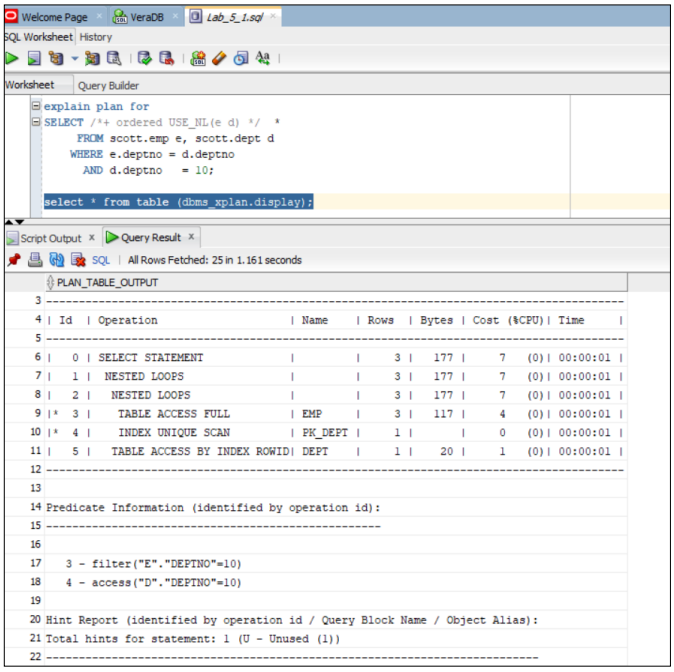
Using option Set Autotrace Traceonly Explain



```
Worksheet  Query Builder
set autotrace traceonly explain
    SELECT /*+ ordered USE_NL(e d) */ *
     FROM scott.emp e, scott.dept d
    WHERE e.deptno = d.deptno
      AND d.deptno   = 10
set autotrace off explain;

Script Output x  Query Result x
Task completed in 16.256 seconds

Statistics
----------------------------------------------------------
          1  CPU used by this session
          1  CPU used when call started
         32  Requests to/from client
         33  SQL*Net roundtrips to/from client
         10  buffer is not pinned count
          3  buffer is pinned count
        611  bytes received via SQL*Net from client
      60958  bytes sent via SQL*Net to client
          2  calls to get snapshot scn: kcmgss
          3  calls to kcmgcs
          9  consistent gets
          4  consistent gets examination
          4  consistent gets examination (fastpath)
          9  consistent gets from cache
          5  consistent gets pin
          5  consistent gets pin (fastpath)
          2  execute count
          3  index fetch by key
     294912  logical read bytes from cache
          4  no work - consistent read gets
         33  non-idle wait count
          2  opened cursors cumulative
          2  opened cursors current
```

Set Autotrace Off Explain

This type of connection requires 9 consecutive gets compared to 6 consecutive gets in the first option.



```
Welcome Page  VeraDB  Lab_5_1.sql
SQL Worksheet  History
Worksheet  Query Builder
explain plan for
SELECT /*+ ordered USE_NL(e d) */  *
     FROM scott.emp e, scott.dept d
    WHERE e.deptno = d.deptno
      AND d.deptno   = 10;

select * from table (dbms_xplan.display);

Script Output x  Query Result x
SQL | All Rows Fetched: 25 in 1.161 seconds

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------
| Id  | Operation                    | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |         |     3 |   177 |     7   (0)| 00:00:01 |
|   1 |  NESTED LOOPS                |         |     3 |   177 |     7   (0)| 00:00:01 |
|   2 |   NESTED LOOPS               |         |     3 |   177 |     7   (0)| 00:00:01 |
|*  3 |    TABLE ACCESS FULL         | EMP     |     3 |   117 |     4   (0)| 00:00:01 |
|*  4 |    INDEX UNIQUE SCAN         | PK_DEPT |     1 |       |     0   (0)| 00:00:01 |
|   5 |   TABLE ACCESS BY INDEX ROWID| DEPT    |     1 |    20 |     1   (0)| 00:00:01 |
--------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   3 - filter("E"."DEPTNO"=10)
   4 - access("D"."DEPTNO"=10)

Hint Report (identified by operation id / Query Block Name / Object Alias):
Total hints for statement: 1 (U - Unused (1))
---------------------------------------------------------------------------
```

The cost of this connection also increases - up to 7 vs. 5 in the first option. Thus, we conclude that Oracle chooses a more favorable option using Nested Loops Joins of tables in terms of cost and performance.

Summary:

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 and joining it with the matching row of the inner row source.

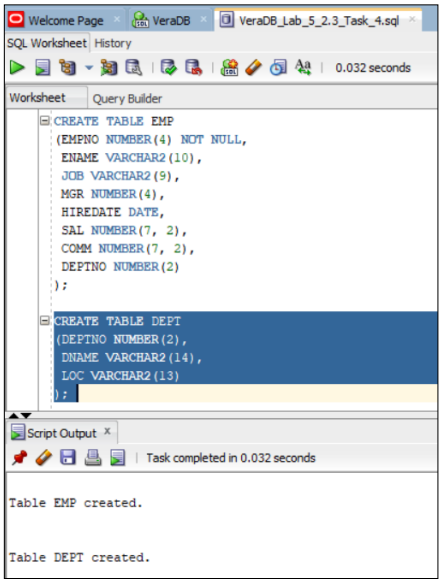
These kinds of joins are quite robust in that they use very little memory. Since row sets are built one row at a time, there is little overhead required. For that reason, they are actually good for huge result sets except for the fact that building a huge result set one row at a time can take quite a long time.

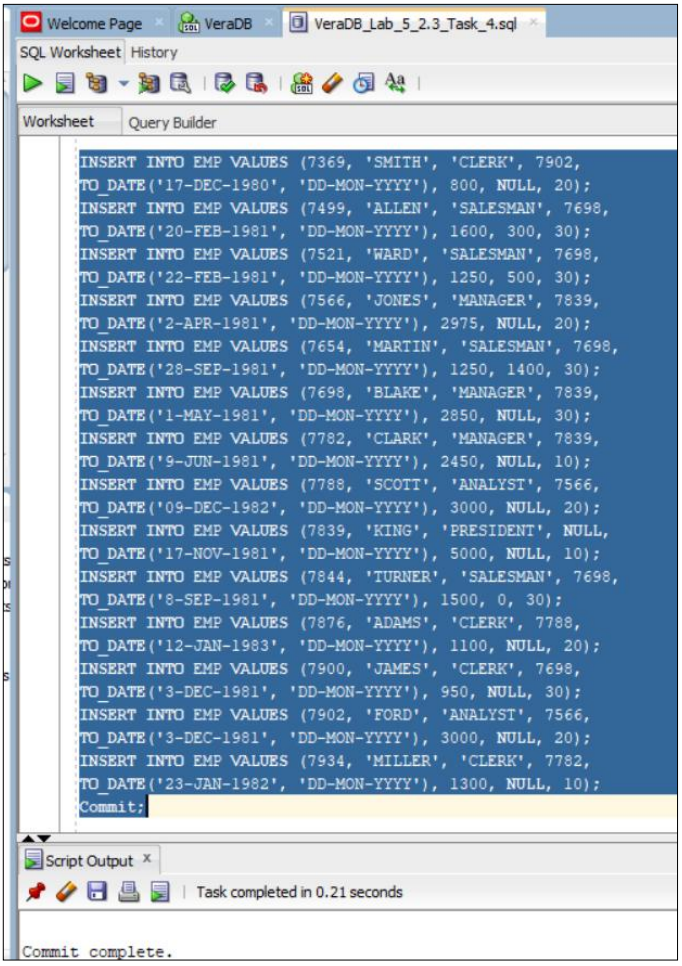
Code: Task 2

```
/*2.1. Task 2:  Nested Loops Joins*/

SELECT  *
    FROM scott.emp e, scott.dept d
   WHERE e.deptno = d.deptno
     AND d.deptno   = 10
--
set autotrace traceonly explain;
SELECT  *
    FROM scott.emp e, scott.dept d
   WHERE e.deptno = d.deptno
     AND d.deptno   = 10
set autotrace off explain;

explain plan for
SELECT   *
    FROM scott.emp e, scott.dept d
   WHERE e.deptno = d.deptno
     AND d.deptno   = 10;
select * from table (dbms_xplan.display);
--
set autotrace traceonly explain
   SELECT /*+ ordered USE_NL(e d) */ *
    FROM scott.emp e, scott.dept d
   WHERE e.deptno = d.deptno
     AND d.deptno   = 10
set autotrace off explain;

explain plan for
SELECT /*+ ordered USE_NL(e d) */  *
    FROM scott.emp e, scott.dept d
   WHERE e.deptno = d.deptno
     AND d.deptno   = 10;
select * from table (dbms_xplan.display);
--
```

2.2. Task 3: Sort-Merge Joins

Step 1: Create Tables and Insert Values





```
INSERT INTO DEPT VALUES (10, 'ACCOUNTING', 'NEW YORK');
INSERT INTO DEPT VALUES (20, 'RESEARCH', 'DALLAS');
INSERT INTO DEPT VALUES (30, 'SALES', 'CHICAGO');
INSERT INTO DEPT VALUES (40, 'OPERATIONS', 'BOSTON');
Commit;
```

Script Output | Task completed in 0.075 seconds

```
1 row inserted.

Commit complete.
```

Step 2: Data results

```
SELECT  *
    FROM emp e, dept d
   WHERE e.deptno = d.deptno;
```

Script Output | Explain Plan | Query Result | SQL | All Rows Fetched: 14 in 0.007 seconds

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	DEPTNO_1	DNAME	LOC
1	7369	SMITH	CLERK	7902	17-DEC-80	800	(null)	20	20	RESEARCH	DALLAS
2	7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30	30	SALES	CHICAGO
3	7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30	30	SALES	CHICAGO
4	7566	JONES	MANAGER	7839	02-APR-81	2975	(null)	20	20	RESEARCH	DALLAS
5	7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30	30	SALES	CHICAGO
6	7698	BLAKE	MANAGER	7839	01-MAY-81	2850	(null)	30	30	SALES	CHICAGO
7	7782	CLARK	MANAGER	7839	09-JUN-81	2450	(null)	10	10	ACCOUNTING	NEW YORK
8	7788	SCOTT	ANALYST	7566	09-DEC-82	3000	(null)	20	20	RESEARCH	DALLAS
9	7839	KING	PRESIDENT	(null)	17-NOV-81	5000	(null)	10	10	ACCOUNTING	NEW YORK
10	7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30	30	SALES	CHICAGO
11	7876	ADAMS	CLERK	7788	12-JAN-83	1100	(null)	20	20	RESEARCH	DALLAS
12	7900	JAMES	CLERK	7698	03-DEC-81	950	(null)	30	30	SALES	CHICAGO
13	7902	FORD	ANALYST	7566	03-DEC-81	3000	(null)	20	20	RESEARCH	DALLAS
14	7934	MILLER	CLERK	7782	23-JAN-82	1300	(null)	10	10	ACCOUNTING	NEW YORK

Step 3: Using software: SQL plus Auto Trace Utility

Explain Plan:

```
set autotrace traceonly explain;
SELECT  *
    FROM emp e, dept d
   WHERE e.deptno = d.deptno;
set autotrace off explain;
```

Script Output | Query Result | Task completed in 0.543 seconds

```
SQL_ID  c3krmnjs4x144, child number 0
-------------------------------------
SELECT  *      FROM emp e, dept d     WHERE e.deptno = d.deptno

Plan hash value: 615168685

-----------------------------------------------------------------------------
| Id  | Operation          | Name | E-Rows |  OMem |  1Mem | Used-Mem |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |        |       |       |          |
|*  1 |  HASH JOIN         |      |     14 |   801K|   801K|  954K (0)|

PLAN_TABLE_OUTPUT
-----------------------------------------------------------------------------
|   2 |   TABLE ACCESS FULL| DEPT |      4 |       |       |          |
|   3 |   TABLE ACCESS FULL| EMP  |     14 |       |       |          |
-----------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("E"."DEPTNO"="D"."DEPTNO")
```

```
set autotrace traceonly explain;
SELECT  *
     FROM emp e, dept d
    WHERE e.deptno = d.deptno;
set autotrace off explain;
```

Script Output | Query Result | Task completed in 0.532 seconds

```
Statistics
----------------------------------------------------------
          1  CPU used by this session
          1  CPU used when call started
          1  DB time
         32  Requests to/from client
         33  SQL*Net roundtrips to/from client
          4  buffer is not pinned count
        547  bytes received via SQL*Net from client
      61690  bytes sent via SQL*Net to client
          3  calls to get snapshot scn: kcmgss
          6  calls to kcmgcs
         10  consistent gets
         10  consistent gets from cache
         10  consistent gets pin
         10  consistent gets pin (fastpath)
          1  enqueue releases
          1  enqueue requests
          3  execute count
     327680  logical read bytes from cache
          6  no work - consistent read gets
         33  non-idle wait count
```

We can see that Oracle joined tables with Hash Joins. Let's change the type of join method use oracle performance hints. (USE_MERGE)

```
set autotrace traceonly explain;
SELECT  /*+ ordered USE_MERGE (d e) */ *
     FROM emp e, dept d
    WHERE e.deptno = d.deptno;
set autotrace off explain;
```

Script Output | Query Result | SQL | All Rows Fetched: 14 in 0.01 seconds

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	DEPTNO_1	DNAME	LOC
1	7782	CLARK	MANAGER	7839	09-JUN-81	2450	(null)	10	10	ACCOUNTING	NEW YORK
2	7839	KING	PRESIDENT	(null)	17-NOV-81	5000	(null)	10	10	ACCOUNTING	NEW YORK
3	7934	MILLER	CLERK	7782	23-JAN-82	1300	(null)	10	10	ACCOUNTING	NEW YORK
4	7566	JONES	MANAGER	7839	02-APR-81	2975	(null)	20	20	RESEARCH	DALLAS
5	7902	FORD	ANALYST	7566	03-DEC-81	3000	(null)	20	20	RESEARCH	DALLAS
6	7876	ADAMS	CLERK	7788	12-JAN-83	1100	(null)	20	20	RESEARCH	DALLAS
7	7369	SMITH	CLERK	7902	17-DEC-80	800	(null)	20	20	RESEARCH	DALLAS
8	7788	SCOTT	ANALYST	7566	09-DEC-82	3000	(null)	20	20	RESEARCH	DALLAS
9	7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30	30	SALES	CHICAGO
10	7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30	30	SALES	CHICAGO
11	7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30	30	SALES	CHICAGO
12	7900	JAMES	CLERK	7698	03-DEC-81	950	(null)	30	30	SALES	CHICAGO
13	7698	BLAKE	MANAGER	7839	01-MAY-81	2850	(null)	30	30	SALES	CHICAGO
14	7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30	30	SALES	CHICAGO

```
SELECT  /*+ ordered USE_MERGE (d e) */ *      FROM emp e, dept d
WHERE e.deptno = d.deptno

Plan hash value: 3406566467

-------------------------------------------------------------------------------
| Id  | Operation           | Name | E-Rows |  OMem |  1Mem | Used-Mem |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |      |        |       |       |          |

PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------
|   1 |  MERGE JOIN         |      |     14 |       |       |          |
|   2 |   SORT JOIN         |      |     14 |  2048 |  2048 | 2048  (0)|
|   3 |    TABLE ACCESS FULL| EMP  |     14 |       |       |          |
|*  4 |   SORT JOIN         |      |      4 |  2048 |  2048 | 2048  (0)|
|   5 |    TABLE ACCESS FULL| DEPT |      4 |       |       |          |
-------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
```

```
set autotrace traceonly explain;
SELECT  /*+ ordered USE_MERGE (d e) */ *
     FROM emp e, dept d
    WHERE e.deptno = d.deptno;
set autotrace off explain;
```

Script Output × | Query Result ×

Task completed in 0.543 seconds

```
Statistics
----------------------------------------------------------
              1  CPU used by this session
              1  CPU used when call started
             32  Requests to/from client
             33  SQL*Net roundtrips to/from client
              4  buffer is not pinned count
            578  bytes received via SQL*Net from client
          61520  bytes sent via SQL*Net to client
              6  calls to get snapshot scn: kcmgss
              8  calls to kcmgcs
             12  consistent gets
             12  consistent gets from cache
             12  consistent gets pin
             12  consistent gets pin (fastpath)
              1  enqueue releases
              1  enqueue requests
              4  execute count
         393216  logical read bytes from cache
              6  no work - consistent read gets
```

The consistent gets parameter is higher when using the Merge Joins for this query, so Oracle chooses a more favorable option using Hush Joins of tables in terms of performance

Summary:

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.

Code: Task 3

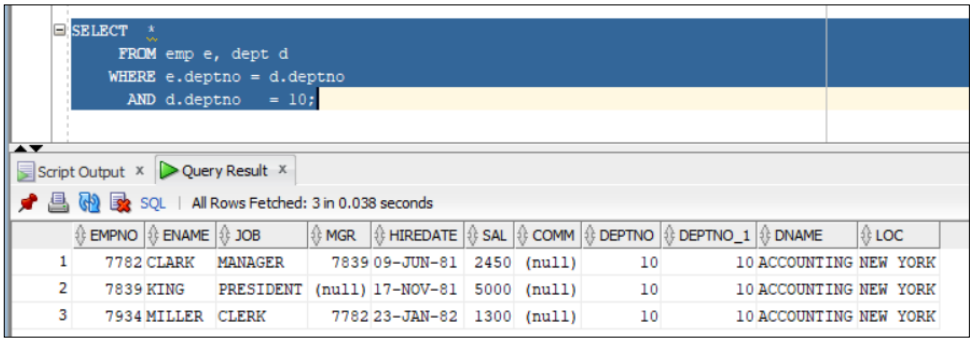
```
/*2.2. Task 3: Sort-Merge Joins*/

CREATE TABLE EMP
(EMPNO NUMBER(4) NOT NULL,
 ENAME VARCHAR2(10),
 JOB VARCHAR2(9),
 MGR NUMBER(4),
 HIREDATE DATE,
 SAL NUMBER(7, 2),
 COMM NUMBER(7, 2),
 DEPTNO NUMBER(2)
);

CREATE TABLE DEPT
(DEPTNO NUMBER(2),
 DNAME VARCHAR2(14),
 LOC VARCHAR2(13)
);

INSERT INTO EMP VALUES (7369, 'SMITH', 'CLERK', 7902,
TO_DATE('17-DEC-1980', 'DD-MON-YYYY'), 800, NULL, 20);
INSERT INTO EMP VALUES (7499, 'ALLEN', 'SALESMAN', 7698,
TO_DATE('20-FEB-1981', 'DD-MON-YYYY'), 1600, 300, 30);
INSERT INTO EMP VALUES (7521, 'WARD', 'SALESMAN', 7698,
TO_DATE('22-FEB-1981', 'DD-MON-YYYY'), 1250, 500, 30);
INSERT INTO EMP VALUES (7566, 'JONES', 'MANAGER', 7839,
TO_DATE('2-APR-1981', 'DD-MON-YYYY'), 2975, NULL, 20);
INSERT INTO EMP VALUES (7654, 'MARTIN', 'SALESMAN', 7698,
TO_DATE('28-SEP-1981', 'DD-MON-YYYY'), 1250, 1400, 30);
INSERT INTO EMP VALUES (7698, 'BLAKE', 'MANAGER', 7839,
TO_DATE('1-MAY-1981', 'DD-MON-YYYY'), 2850, NULL, 30);
INSERT INTO EMP VALUES (7782, 'CLARK', 'MANAGER', 7839,
TO_DATE('9-JUN-1981', 'DD-MON-YYYY'), 2450, NULL, 10);
INSERT INTO EMP VALUES (7788, 'SCOTT', 'ANALYST', 7566,
TO_DATE('09-DEC-1982', 'DD-MON-YYYY'), 3000, NULL, 20);
INSERT INTO EMP VALUES (7839, 'KING', 'PRESIDENT', NULL,
TO_DATE('17-NOV-1981', 'DD-MON-YYYY'), 5000, NULL, 10);
INSERT INTO EMP VALUES (7844, 'TURNER', 'SALESMAN', 7698,
TO_DATE('8-SEP-1981', 'DD-MON-YYYY'), 1500, 0, 30);
INSERT INTO EMP VALUES (7876, 'ADAMS', 'CLERK', 7788,
TO_DATE('12-JAN-1983', 'DD-MON-YYYY'), 1100, NULL, 20);
INSERT INTO EMP VALUES (7900, 'JAMES', 'CLERK', 7698,
TO_DATE('3-DEC-1981', 'DD-MON-YYYY'), 950, NULL, 30);
INSERT INTO EMP VALUES (7902, 'FORD', 'ANALYST', 7566,
TO_DATE('3-DEC-1981', 'DD-MON-YYYY'), 3000, NULL, 20);
INSERT INTO EMP VALUES (7934, 'MILLER', 'CLERK', 7782,
TO_DATE('23-JAN-1982', 'DD-MON-YYYY'), 1300, NULL, 10);
Commit;
```

```
INSERT INTO DEPT VALUES (10, 'ACCOUNTING', 'NEW YORK');
INSERT INTO DEPT VALUES (20, 'RESEARCH', 'DALLAS');
INSERT INTO DEPT VALUES (30, 'SALES', 'CHICAGO');
INSERT INTO DEPT VALUES (40, 'OPERATIONS', 'BOSTON');
Commit;

SELECT  *
    FROM emp e, dept d
   WHERE e.deptno = d.deptno;

set autotrace traceonly explain;
SELECT  *
    FROM emp e, dept d
   WHERE e.deptno = d.deptno;
set autotrace off explain;

set autotrace traceonly explain;
SELECT  /*+ ordered USE_MERGE (d e) */ *
    FROM emp e, dept d
   WHERE e.deptno = d.deptno;
set autotrace off explain;

Drop table emp;
Drop table dept;
select segment_name, segment_type from user_segments;
PURGE RECYCLEBIN;
select segment_name, segment_type from user_segments;
```

2.3. Task 4: Hash Joins

Step 1: Create Emp and Dept Tables, Insert Values

Step 2: Data results



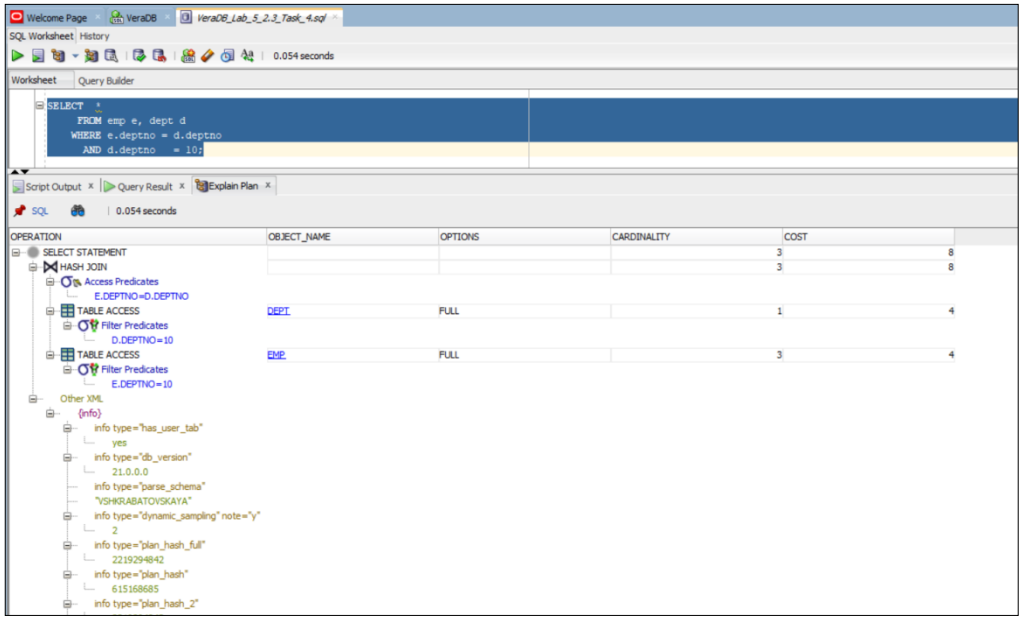
```
SELECT *
  FROM emp e, dept d
 WHERE e.deptno = d.deptno
   AND d.deptno    = 10;
```

Script Output | Query Result | SQL | All Rows Fetched: 3 in 0.038 seconds

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	DEPTNO_1	DNAME	LOC
1	7782	CLARK	MANAGER	7839	09-JUN-81	2450	(null)	10	10	ACCOUNTING	NEW YORK
2	7839	KING	PRESIDENT	(null)	17-NOV-81	5000	(null)	10	10	ACCOUNTING	NEW YORK
3	7934	MILLER	CLERK	7782	23-JAN-82	1300	(null)	10	10	ACCOUNTING	NEW YORK

Step 3: Using software: Oracle SQL Developer

Explain Plan:



Script Output | Query Result | Explain Plan | 0.054 seconds

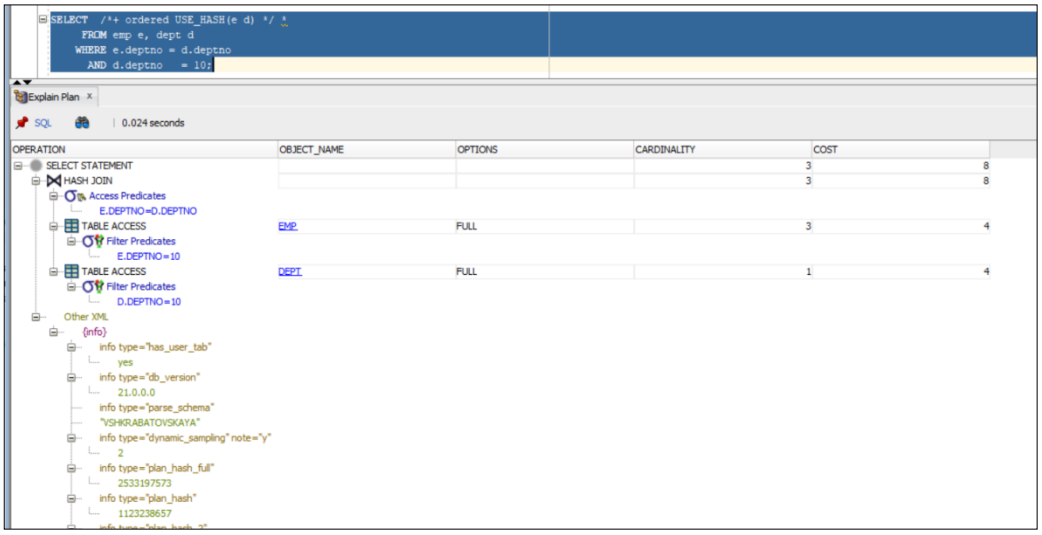
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			3	8
HASH JOIN			3	8
Access Predicates: E.DEPTNO=D.DEPTNO				
TABLE ACCESS	DEPT	FULL	1	4
Filter Predicates: D.DEPTNO=10				
TABLE ACCESS	EMP	FULL	3	4
Filter Predicates: E.DEPTNO=10				

Other XML

```
{info}
  info type="has_user_tab"
    yes
  info type="db_version"
    21.0.0.0
  info type="parse_schema"
    "VSHKRABATOVSKAYA"
  info type="dynamic_sampling" note="y"
    2
  info type="plan_hash_full"
    2219294842
  info type="plan_hash"
    615168685
  info type="plan_hash_2"
```

We see that in the hash join plan, the smaller hash table DEPT is listed first and the EMP table is listed second. The decision as to which table is smallest depends not just on the number of rows but the size of those rows as well, since the entire row must be stored in the hash table.

If we change the order in which the tables are joined, we see that the cost does not change.



Summary:

HASH joins are the usual choice of the Oracle optimizer when the memory is set up to accommodate them. In a HASH join, Oracle accesses one table (usually the smaller of the joined results) and builds a hash table on the join key in memory. It then scans the other table in the join (usually the larger one) and probes the hash table for matches to it. Hash joins are only possible if the join is an equi-join.

Code: Task 4

```
/*2.3. Task 4: Hash Joins*/

SELECT  *
    FROM emp e, dept d
   WHERE e.deptno = d.deptno
     AND d.deptno   = 10;

SELECT  /*+ ordered USE_HASH(e d) */ *
    FROM emp e, dept d
   WHERE e.deptno = d.deptno
     AND d.deptno   = 10;

Drop table emp;
Drop table dept;
select segment_name, segment_type from user_segments;
PURGE RECYCLEBIN;
select segment_name, segment_type from user_segments;
```

2.4. Task 5: Cartesian Joins

Step 1: Data results

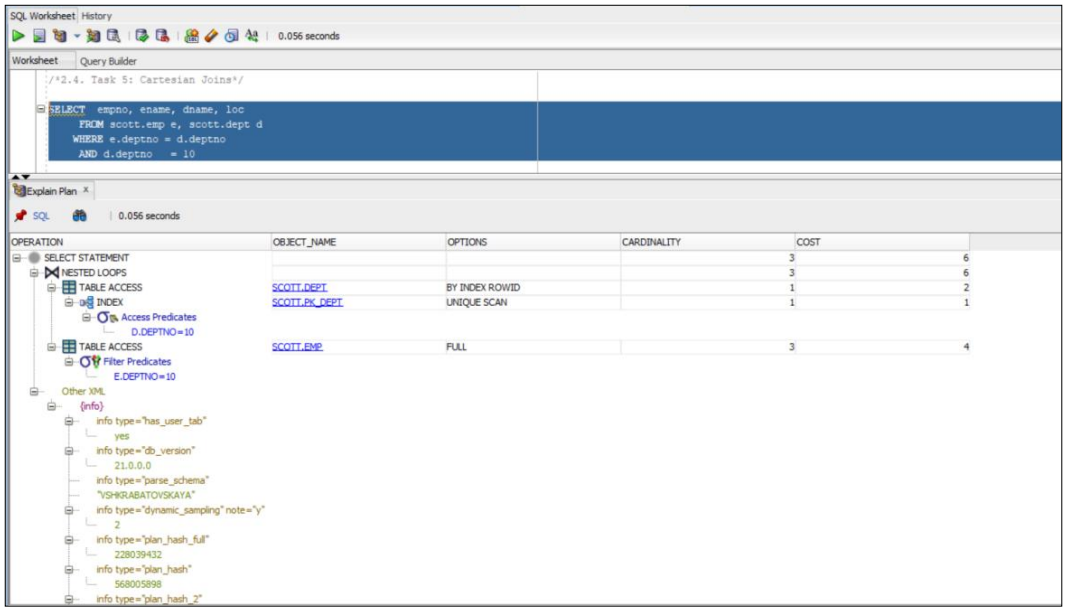
```
SELECT  empno, ename, dname, loc
    FROM scott.emp e, scott.dept d
   WHERE e.deptno = d.deptno
     AND d.deptno  = 10
```

Script Output | Query Result | SQL | All Rows Fetched: 3 in 0.011 seconds

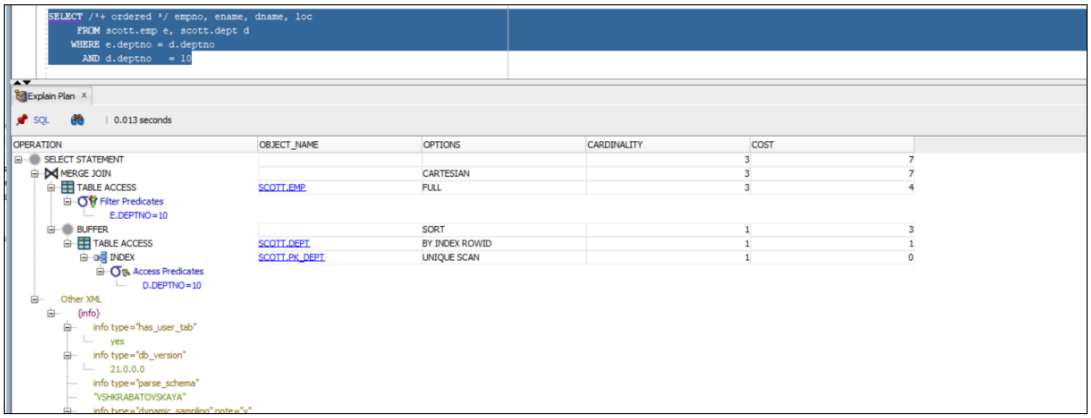
	EMPNO	ENAME	DNAME	LOC
1	7782	CLARK	ACCOUNTING	NEW YORK
2	7839	KING	ACCOUNTING	NEW YORK
3	7934	MILLER	ACCOUNTING	NEW YORK

Step 2: Using software: Oracle SQL Developer

Explain Plan:



In this query, Oracle suggests using Nested loops joins.
Let's change in the execution plan the type of join method use oracle performance hints. (USE_MERGE)



The cost parameter is higher when using the Merge Cartesian Joins for this query.

Summary:

Cartesian joins occur when all the rows from one table are joined to all the rows of another table.

Therefore, the total number of rows resulting from the join equals the number of rows from one table (A) multiplied by the number of rows in the other table (B) such that A x B = total rows in the result

set. Cartesian joins often occur when a join condition is overlooked or left out such that there isn't a specified join column so the only operation possible is to simply join everything from one row source to everything from the other.

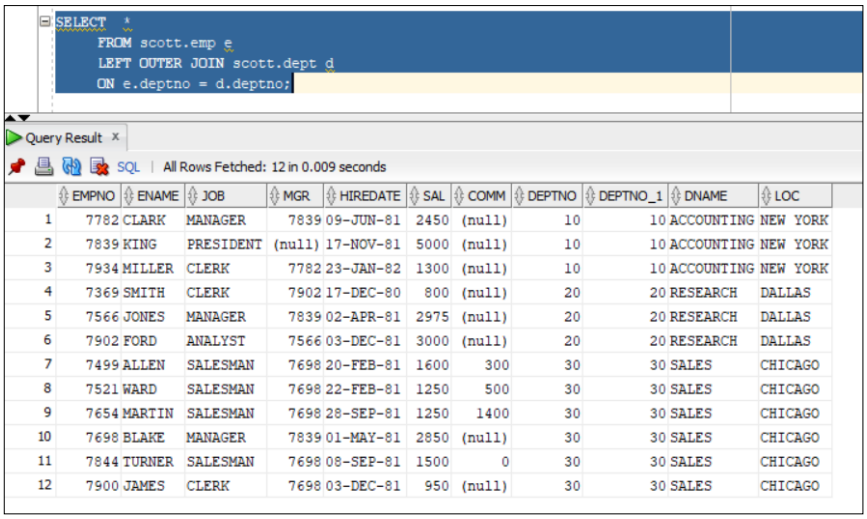
Code: Task 5

```
/*2.4. Task 5: Cartesian Joins*/

SELECT  empno, ename, dname, loc
    FROM scott.emp e, scott.dept d
   WHERE e.deptno = d.deptno
    AND d.deptno   = 10

SELECT /*+ ordered */ empno, ename, dname, loc
    FROM scott.emp e, scott.dept d
   WHERE e.deptno = d.deptno
     AND d.deptno   = 10
```

2.5. Task 6: Left/Right Outer Joins

Step 1: Script to create Oracle's "SCOTT" schema (tables EMP, DEPT, BONUS, SALGRADE)

Step 2: Data results/ LEFT OUTER JOIN



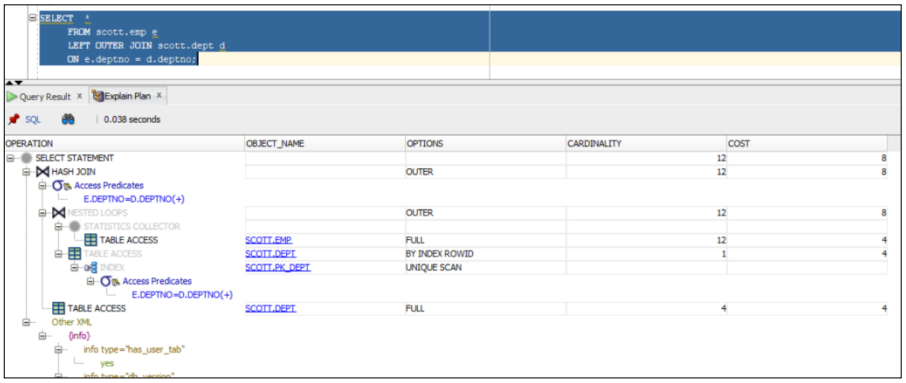
```
SELECT *
    FROM scott.emp e
    LEFT OUTER JOIN scott.dept d
    ON e.deptno = d.deptno;
```

Query Result | SQL | All Rows Fetched: 12 in 0.009 seconds

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	DEPTNO_1	DNAME	LOC
1	7782	CLARK	MANAGER	7839	09-JUN-81	2450	(null)	10	10	ACCOUNTING	NEW YORK
2	7839	KING	PRESIDENT	(null)	17-NOV-81	5000	(null)	10	10	ACCOUNTING	NEW YORK
3	7934	MILLER	CLERK	7782	23-JAN-82	1300	(null)	10	10	ACCOUNTING	NEW YORK
4	7369	SMITH	CLERK	7902	17-DEC-80	800	(null)	20	20	RESEARCH	DALLAS
5	7566	JONES	MANAGER	7839	02-APR-81	2975	(null)	20	20	RESEARCH	DALLAS
6	7902	FORD	ANALYST	7566	03-DEC-81	3000	(null)	20	20	RESEARCH	DALLAS
7	7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30	30	SALES	CHICAGO
8	7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30	30	SALES	CHICAGO
9	7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30	30	SALES	CHICAGO
10	7698	BLAKE	MANAGER	7839	01-MAY-81	2850	(null)	30	30	SALES	CHICAGO
11	7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30	30	SALES	CHICAGO
12	7900	JAMES	CLERK	7698	03-DEC-81	950	(null)	30	30	SALES	CHICAGO

Step 3: Using software: Oracle SQL Developer

Explain Plan:

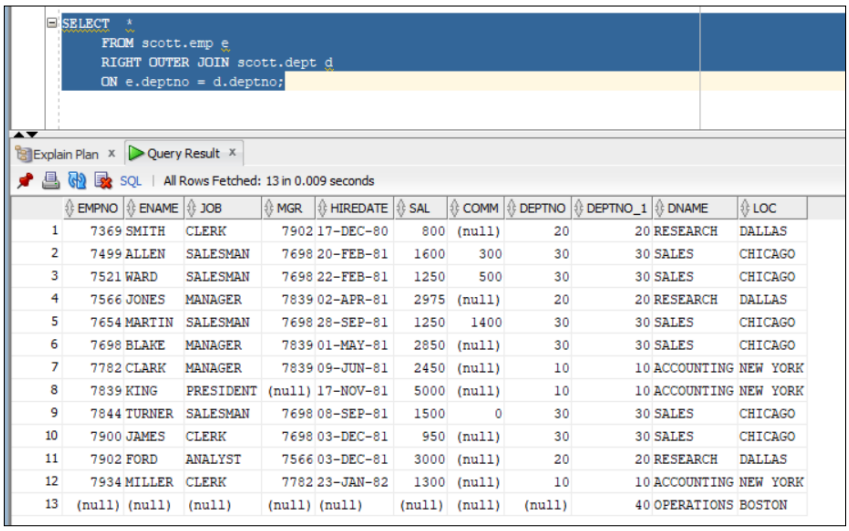


```
SELECT *
    FROM scott.emp e
    LEFT OUTER JOIN scott.dept d
    ON e.deptno = d.deptno;
```

Query Result | Explain Plan | SQL | 0.038 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			12	8
HASH JOIN		OUTER	12	8
Access Predicates				
E.DEPTNO=D.DEPTNO(+)				
NESTED LOOPS		OUTER	12	8
STATISTICS COLLECTOR				
TABLE ACCESS	SCOTT.EMP	FULL	12	4
TABLE ACCESS	SCOTT.DEPT	BY INDEX ROWID	1	4
INDEX	SCOTT.PK_DEPT	UNIQUE SCAN		
Access Predicates				
E.DEPTNO=D.DEPTNO(+)				
TABLE ACCESS	SCOTT.DEPT	FULL	4	4
Other XML				
(info)				
info type="has_user_tab"				
yes				

Step 4: Data results/ RIGHT OUTER JOIN

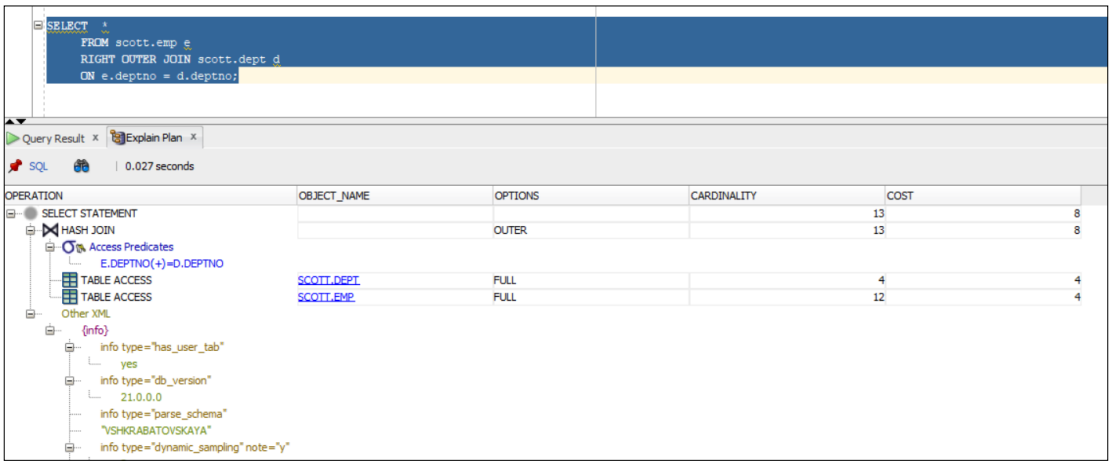


```
SELECT *
    FROM scott.emp e
    RIGHT OUTER JOIN scott.dept d
    ON e.deptno = d.deptno;
```

Explain Plan | Query Result | SQL | All Rows Fetched: 13 in 0.009 seconds

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	DEPTNO_1	DNAME	LOC
1	7369	SMITH	CLERK	7902	17-DEC-80	800	(null)	20	20	RESEARCH	DALLAS
2	7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30	30	SALES	CHICAGO
3	7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30	30	SALES	CHICAGO
4	7566	JONES	MANAGER	7839	02-APR-81	2975	(null)	20	20	RESEARCH	DALLAS
5	7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30	30	SALES	CHICAGO
6	7698	BLAKE	MANAGER	7839	01-MAY-81	2850	(null)	30	30	SALES	CHICAGO
7	7782	CLARK	MANAGER	7839	09-JUN-81	2450	(null)	10	10	ACCOUNTING	NEW YORK
8	7839	KING	PRESIDENT	(null)	17-NOV-81	5000	(null)	10	10	ACCOUNTING	NEW YORK
9	7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30	30	SALES	CHICAGO
10	7900	JAMES	CLERK	7698	03-DEC-81	950	(null)	30	30	SALES	CHICAGO
11	7902	FORD	ANALYST	7566	03-DEC-81	3000	(null)	20	20	RESEARCH	DALLAS
12	7934	MILLER	CLERK	7782	23-JAN-82	1300	(null)	10	10	ACCOUNTING	NEW YORK
13	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	40	OPERATIONS	BOSTON

Step 5: Using software: Oracle SQL Developer



The cost parameter doesn't change when using the Left Joins or the Right Joins for this query.

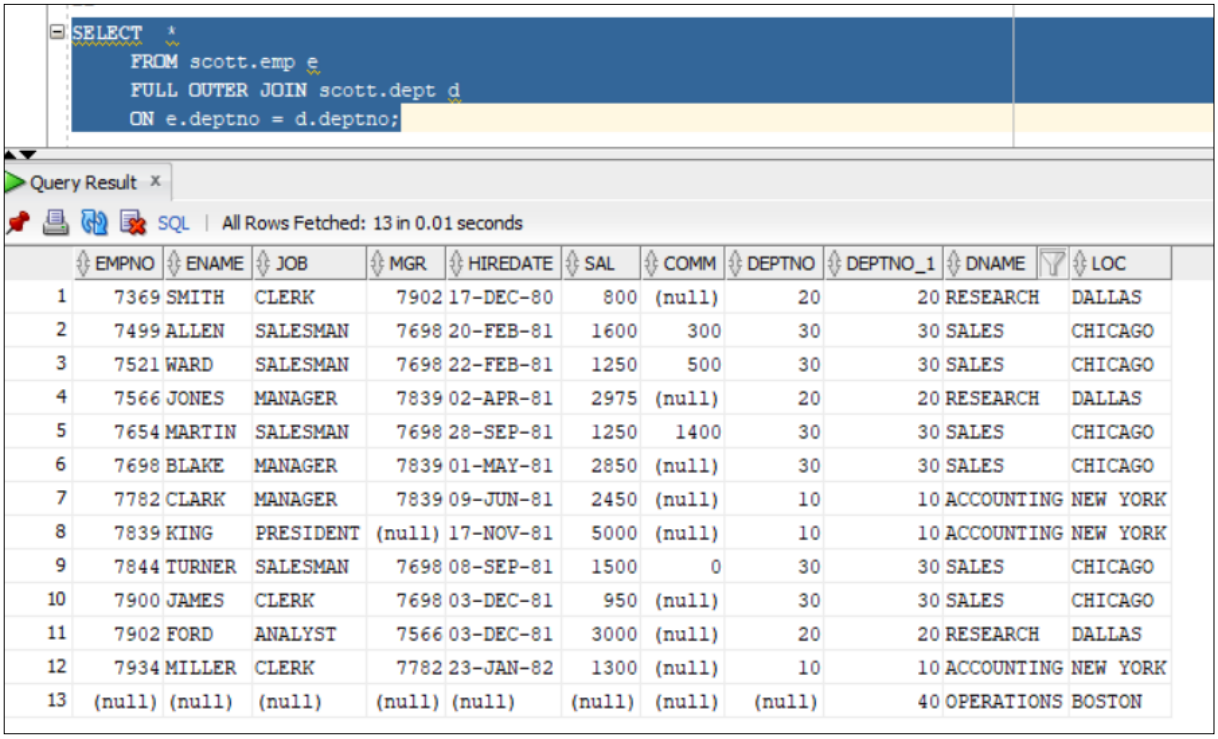
Code: Task 6

```
/*2.5. Task 6: Left/Right Outer Joins*/
--Script to create Oracle's "SCOTT" schema (tables EMP, DEPT, BONUS, SALGRADE)

SELECT  *
    FROM scott.emp e
    LEFT OUTER JOIN scott.dept d
    ON e.deptno = d.deptno;

SELECT  *
    FROM scott.emp e
    RIGHT OUTER JOIN scott.dept d
    ON e.deptno = d.deptno;
```

2.6. Task 7: Full Outer Join

Step 1: Script to create Oracle's "SCOTT" schema (tables EMP, DEPT, BONUS, SALGRADE)

Step 2: Data results

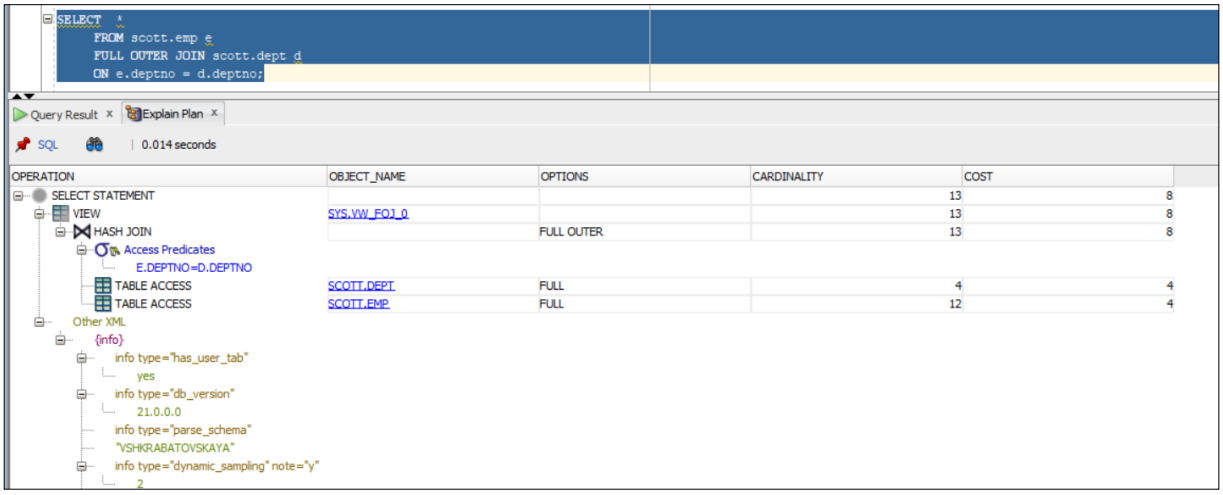


```
SELECT  *
    FROM scott.emp e
    FULL OUTER JOIN scott.dept d
    ON e.deptno = d.deptno;
```

Query Result | SQL | All Rows Fetched: 13 in 0.01 seconds

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	DEPTNO_1	DNAME	LOC
1	7369	SMITH	CLERK	7902	17-DEC-80	800	(null)	20	20	RESEARCH	DALLAS
2	7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30	30	SALES	CHICAGO
3	7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30	30	SALES	CHICAGO
4	7566	JONES	MANAGER	7839	02-APR-81	2975	(null)	20	20	RESEARCH	DALLAS
5	7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30	30	SALES	CHICAGO
6	7698	BLAKE	MANAGER	7839	01-MAY-81	2850	(null)	30	30	SALES	CHICAGO
7	7782	CLARK	MANAGER	7839	09-JUN-81	2450	(null)	10	10	ACCOUNTING	NEW YORK
8	7839	KING	PRESIDENT	(null)	17-NOV-81	5000	(null)	10	10	ACCOUNTING	NEW YORK
9	7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30	30	SALES	CHICAGO
10	7900	JAMES	CLERK	7698	03-DEC-81	950	(null)	30	30	SALES	CHICAGO
11	7902	FORD	ANALYST	7566	03-DEC-81	3000	(null)	20	20	RESEARCH	DALLAS
12	7934	MILLER	CLERK	7782	23-JAN-82	1300	(null)	10	10	ACCOUNTING	NEW YORK
13	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	40	OPERATIONS	BOSTON

Step 3: Using software: Oracle SQL Developer



```
SELECT  *
    FROM scott.emp e
    FULL OUTER JOIN scott.dept d
    ON e.deptno = d.deptno;
```

Query Result | Explain Plan | SQL | 0.014 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			13	8
VIEW	SYS.VW_FOJ_0		13	8
HASH JOIN		FULL OUTER	13	8
Access Predicates				
E.DEPTNO=D.DEPTNO				
TABLE ACCESS	SCOTT.DEPT	FULL	4	4
TABLE ACCESS	SCOTT.EMP	FULL	12	4
Other XML				
{info}				
info type="has_user_tab"				
yes				
info type="db_version"				
21.0.0.0				
info type="parse_schema"				
"VSHKRABATOVSKAYA"				
info type="dynamic_sampling" note="y"				
2				

Code: Task 7

```
/*2.6. Task 7: Full Outer Join*/

--Script to create Oracle's "SCOTT" schema (tables EMP, DEPT, BONUS, SALGRADE)
SELECT  *
    FROM scott.emp e
    FULL OUTER JOIN scott.dept d
    ON e.deptno = d.deptno;
```

2.7. Task 8: Semi Joins

Step 1: Create Emp and Dept Tables, Insert Values

Step 2: Data results

```
select /* using in */ ename, job
from emp e
where deptno IN (select deptno from dept d);
```

Explain Plan | Script Output | Query Result

SQL | All Rows Fetched: 14 in 0.007 seconds

	ENAME	JOB
1	CLARK	MANAGER
2	KING	PRESIDENT
3	MILLER	CLERK
4	SMITH	CLERK
5	JONES	MANAGER
6	SCOTT	ANALYST
7	ADAMS	CLERK
8	FORD	ANALYST
9	ALLEN	SALESMAN
10	WARD	SALESMAN
11	MARTIN	SALESMAN
12	BLAKE	MANAGER
13	TURNER	SALESMAN
14	JAMES	CLERK

Step 3: Using software: SQL plus Auto Trace Utility

Worksheet | Query Builder

```
set autotrace traceonly explain;
select /* using in */ ename, job
from emp e
where deptno IN (select deptno from dept d);
set autotrace off explain;
```

Script Output | Query Result

Task completed in 0.446 seconds

```
SQL_ID  7y9vandpv4srv, child number 0
-------------------------------------
select /* using in */ ename, job from emp e where deptno IN (select
deptno from dept d)

Plan hash value: 230627304

-----------------------------------------------------------------------------
| Id  | Operation          | Name | E-Rows |  OMem |  1Mem | Used-Mem |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |        |       |       |          |

PLAN_TABLE_OUTPUT
-----------------------------------------------------------------------------
|*  1 |  HASH JOIN SEMI    |      |     14 |   863K|   863K|  982K (0)|
|   2 |   TABLE ACCESS FULL| EMP  |     14 |       |       |          |
|   3 |   TABLE ACCESS FULL| DEPT |      4 |       |       |          |
-----------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("DEPTNO"="DEPTNO")

Note
```

Summary: A 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.

The requirements for Oracle's cost based optimizer to decide to use a semi-join:

- The statement must use either the keyword IN (= ANY) or the keyword EXISTS
- The statement must have a subquery in the IN or EXISTS clause
- If the statement uses the EXISTS syntax, it must use a correlated subquery (to get the expected results)
- The IN or EXISTS clause may not be contained inside an OR branch

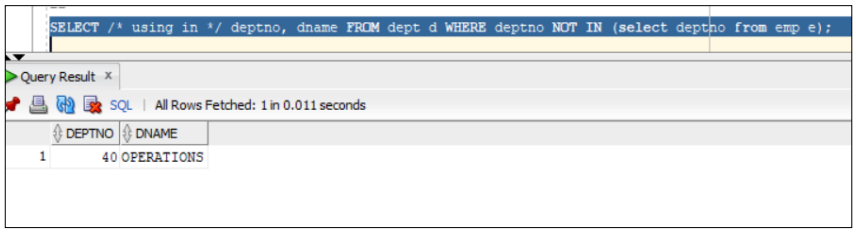
Code: Task 8

```
/*2.7. Task 8: Semi Joins*/
select /* using in */ ename, job
from emp e
where deptno IN (select deptno from dept d);

set autotrace traceonly explain;
select /* using in */ ename, job
from emp e
where deptno IN (select deptno from dept d);
set autotrace off explain;
```

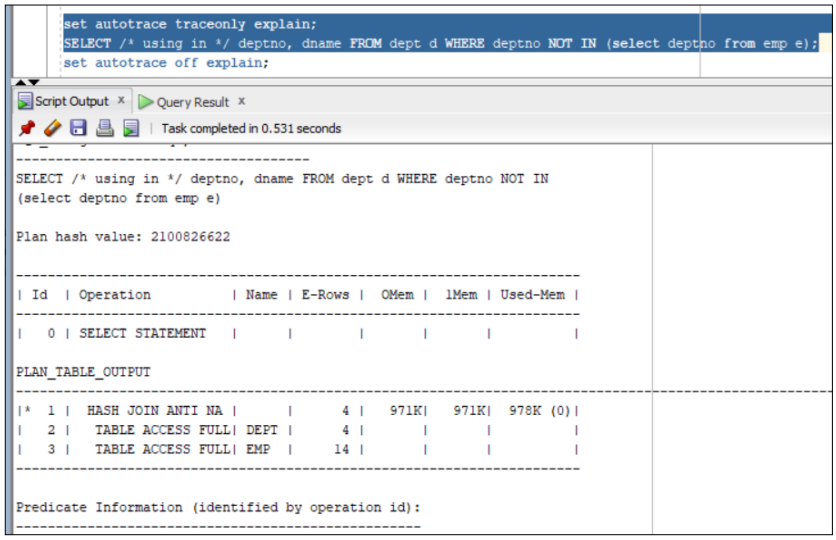
2.8. Task 9: Anti Joins

Step 1: Create Emp and Dept Tables, Insert Values

Step 2: Data results



Step 3: Using software: SQL plus Auto Trace Utility



Summary: 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. However, they are the opposite of semi-joins in terms of the data they return.

Code: Task 9

```
/*2.8. Task 9: Anti Joins*/
SELECT /* using in */ deptno, dname FROM dept d WHERE deptno NOT IN (select deptno from emp e);

set autotrace traceonly explain;
SELECT /* using in */ deptno, dname FROM dept d WHERE deptno NOT IN (select deptno from emp e);
set autotrace off explain;
```

2.9. Task 10: Prepare summary table

Compare 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
Small Table	Small Table	+	+ (if the join is an equi-join)	+
Small Table	Indexed Small Table	+	+ (if the join is an equi-join)	+
Indexed Small Table	Indexed Small Table	+	+ (if the join is an equi-join)	+
Big Table	Big Table	+/- (quite a long time)	+ (if the table can fit in memory)	+/- (for large row sources can be quite memory and time-consuming to complete)
Big Table	Small Table	+/- (quite a long time)	+/-(if the smaller table can fit in memory)	+/- (for large row sources can be quite memory and time-consuming to complete)
Big Table	Indexed Small Table	+/- (quite a long time)	+/-(if the smaller table can fit in memory)	+/- (for large row sources can be quite memory and time-consuming to complete)
Big Table	Indexed Big Table	+/- (quite a long time)	+ (if the table can fit in memory)	+/- (for large row sources can be quite memory and time-consuming to complete)
Indexed Big Table	Small Table	+/- (quite a long time)	+/-(if the smaller table can fit in memory)	+/- (for large row sources can be quite memory and time-consuming to complete)
Indexed Big Table	Indexed Small Table	+/- (quite a long time)	+/-(if the smaller table can fit in memory)	+
Indexed Big Table	Indexed Big Table	+/- (quite a long time)	+ (if the table can fit in memory)	+