U1M5.LW.Access and Join Methods Part 2

Report

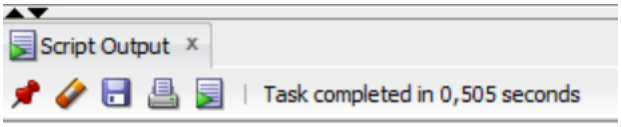
Mastykina Elizaweta

1. Auto Trace & Explain Plan

1.1. Task 1: Auto Trace configuration training

SELECT emp.* FROM emp;

1. set autotrace off




```
Autotrace Disabled
>>Query Run In:Query Result 1
```

2. set autotrace on

```
-------------------------------------------
| Id  | Operation         | Name | E-Rows |
-------------------------------------------
|   0 | SELECT STATEMENT  |      |        |
|   1 |  TABLE ACCESS FULL| EMP  |  60720 |

PLAN_TABLE_OUTPUT
-----------------------------------------------------------------------------------------
-------------------------------------------

Note
-----
   - Warning: basic plan statistics not available. These are only collected when:
       * hint 'gather_plan_statistics' is used for the statement or
       * parameter 'statistics_level' is set to 'ALL', at session or system level


Statistics
----------------------------------------------------------
               2  CPU used by this session
               5  CPU used when call started
               4  DB time
              43  Requests to/from client
              43  SQL*Net roundtrips to/from client
             601  bytes received via SQL*Net from client
          215582  bytes sent via SQL*Net to client
               2  calls to get snapshot scn: kcmgss
               4  calls to kcmgcs
              19  consistent gets
              19  consistent gets from cache
              19  consistent gets pin
              19  consistent gets pin (fastpath)
               2  cursor authentications
               2  execute count
          622592  logical read bytes from cache
              17  no work - consistent read gets
              70  non-idle wait count
               2  opened cursors cumulative
               1  opened cursors current
               2  parse count (total)
               7  process last non-idle time
              19  session logical reads
               1  sorts (memory)
            1581  sorts (rows)
              17  table scan blocks gotten
           12529  table scan disk non-IMC rows gotten
           12529  table scan rows gotten
               1  table scans (short tables)
              45  user calls
```

3. set autotrace traceonly

   Not supported on SQL Developer.

4. set autotrace traceonly explain

   Not supported on SQL Developer

5. set autotrace traceonly statistics

   Not supported on SQL Developer

6. set autotrace traceonly explain statistics

   Not supported on SQL Developer.

7. set autotrace on explain

```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
SQL_ID  a6yp0kv8k1g9s, child number 0
-------------------------------------
SELECT emp.* FROM emp

Plan hash value: 3956160932

-------------------------------------------
| Id  | Operation         | Name | E-Rows |
-------------------------------------------
|   0 | SELECT STATEMENT  |      |        |
|   1 |  TABLE ACCESS FULL| EMP  |  60720 |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
-------------------------------------------
```

8. set autotrace on statistics

```
Statistics
----------------------------------------------------------
              2  CPU used by this session
              2  CPU used when call started
             43  Requests to/from client
             43  SQL*Net roundtrips to/from client
            602  bytes received via SQL*Net from client
         212394  bytes sent via SQL*Net to client
              3  calls to get snapshot scn: kcmgss
              4  calls to kcmgcs
             19  consistent gets
             19  consistent gets from cache
             19  consistent gets pin
             19  consistent gets pin (fastpath)
              1  enqueue releases
              1  enqueue requests
              2  execute count
         622592  logical read bytes from cache
             17  no work - consistent read gets
             57  non-idle wait count
              2  opened cursors cumulative
              1  opened cursors current
              1  parse count (hard)
              2  parse count (total)
              8  process last non-idle time
              1  recursive calls
             19  session logical reads
              1  sorts (memory)
           1581  sorts (rows)
             17  table scan blocks gotten
          12529  table scan disk non-IMC rows gotten
          12529  table scan rows gotten
              1  table scans (short tables)
             45  user calls
```

9. `set autotrace on explain statistics`

   The same as set autotrace on.

10. `set autotrace off explain`

    Currently isn't used.

11. `set autotrace off statisticЫ`

    Currently isn't used.

12. `set autotrace off explain statistics`

    Currently isn't used.

Auto Trace Configuration Options	Results and 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	the AUTOTRACE report includes both the optimizer execution path and the SQL statement execution statistics,but suppresses the printing ofthe user's query output ! not supportedon SQLDeveloper
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	the AUTOTRACE report includes both the optimizer execution path and the SQL statement execution statistics
SET AUTOTRACE RACEONLY EXPLAIN	the AUTOTRACE report includesthe optimizer execution path,but suppresses the printing ofthe user's query output! currently not used
SET AUTOTRACE TRACEONLY STATISTICS	the AUTOTRACE report includes the SQL statement execution statistics,but suppresses the printing ofthe user's query output! not supportedon SQLDeveloper
SET AUTOTRACE TRACEONLY EXPLAINSTATISTICS	the AUTOTRACE report includes both the optimizer execution path and the SQL statement execution statistics,but suppresses the printing ofthe user's query output! not supportedon SQLDeveloper
SET AUTOTRACE OFF EXPLAIN	Is not used
SETA AUTOTRACE OFF STATISTICS	Is not used
SET AUTOTRACE OFF EXPLAIN STATISTICS	Is not used

2. Join Methods

0,18799999 seconds

Worksheet | Query Builder

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

Script Output

Task completed in 0,188 seconds

```
     EMPNO ENAME      JOB              MGR HIREDATE        SAL       COMM     DEPTNO     DEPTNO DNAME          LOC
---------- ---------- --------- ---------- -------- ---------- ---------- ---------- ---------- -------------- -------------
      7782 CLARK      MANAGER         7839 09.06.81       2450                    10         10 ACCOUNTING     NEW YORK
      7839 KING       PRESIDENT            17.11.81       5000                    10         10 ACCOUNTING     NEW YORK
      7934 MILLER     CLERK           7782 23.01.82       1300                    10         10 ACCOUNTING     NEW YORK
```

SQL Worksheet | History

48,45000076 seconds

Worksheet | Query Builder

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

Script Output

Task completed in 48,45 seconds

```
Autotrace Enabled
Displays the execution plan only.

     EMPNO ENAME      JOB              MGR HIREDATE        SAL       COMM     DEPTNO     DEPTNO DNAME          LOC
---------- ---------- --------- ---------- -------- ---------- ---------- ---------- ---------- -------------- -------------
      7782 CLARK      MANAGER         7839 09.06.81       2450                    10         10 ACCOUNTING     NEW YORK
      7839 KING       PRESIDENT            17.11.81       5000                    10         10 ACCOUNTING     NEW YORK
      7934 MILLER     CLERK           7782 23.01.82       1300                    10         10 ACCOUNTING     NEW YORK


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                |         |      5 |
|   2 |   TABLE ACCESS BY INDEX ROWID| DEPT    |      1 |
|*  3 |    INDEX UNIQUE SCAN         | PK_DEPT |      1 |
|*  4 |   TABLE ACCESS FULL          | EMP     |      5 |
-------------------------------------------------------

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

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

PLAN_TABLE_OUTPUT
```

Script Output × | Explain Plan ×

SQL | 0,055 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			3	3
NESTED LOOPS			3	3
TABLE ACCESS	DEPT	BY INDEX ROWID	1	2
INDEX	SYS_C007178	UNIQUE SCAN	1	1
Access Predicates				
D.DEPTNO=10				
TABLE ACCESS	EMP	CLUSTER	3	1
INDEX	IDXCL_EMP_DEPT	UNIQUE SCAN	1	0
Access Predicates				
E.DEPTNO=10				

The conception of work of Nested Loops Join is the following: Oracle takes the first value from the first table("outer" table is chosen by default) and compare to each value in “inner” table, in the search of match.

When all the values in the ‘inner’ table are checked Oracle takes next value in the first table and repeat this process. The same for all the rest values in the first table.

It is important to notice, that such method of join is the most ineffective. But we have sum methods to optimize it.

2.2. Task 3: Sort-Merge Joins

```
SET AUTOTRACE ON EXPLAIN;
SELECT /*+ USE_MERGE(dept emp) */ *
FROM dept, emp
WHERE emp.deptno= dept.deptno;
SELECT * FROM TABLE(dbms_xplan.display );
```

ript Output × | Explain Plan × | Query Result × | Query Result 1 ×

SQL | All Rows Fetched: 14 in 0,02 seconds

	DEPTNO	DNAME	LOC	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO_1
1	10	ACCOUNTING	NEW YORK	7782	CLARK	MANAGER	7839	09.06.81	2450	(null)	10
2	10	ACCOUNTING	NEW YORK	7839	KING	PRESIDENT	(null)	17.11.81	5000	(null)	10
3	10	ACCOUNTING	NEW YORK	7934	MILLER	CLERK	7782	23.01.82	1300	(null)	10
4	20	RESEARCH	DALLAS	7369	SMITH	CLERK	7902	17.12.80	800	(null)	20
5	20	RESEARCH	DALLAS	7566	JONES	MANAGER	7839	02.04.81	2975	(null)	20
6	20	RESEARCH	DALLAS	7788	SCOTT	ANALYST	7566	13.07.87	3000	(null)	20
7	20	RESEARCH	DALLAS	7876	ADAMS	CLERK	7788	13.07.87	1100	(null)	20
8	20	RESEARCH	DALLAS	7902	FORD	ANALYST	7566	03.12.81	3000	(null)	20
9	30	SALES	CHICAGO	7499	ALLEN	SALESMAN	7698	20.02.81	1600	300	30
10	30	SALES	CHICAGO	7521	WARD	SALESMAN	7698	22.02.81	1250	500	30
11	30	SALES	CHICAGO	7654	MARTIN	SALESMAN	7698	28.09.81	1250	1400	30
12	30	SALES	CHICAGO	7698	BLAKE	MANAGER	7839	01.05.81	2850	(null)	30
13	30	SALES	CHICAGO	7844	TURNER	SALESMAN	7698	13.07.87	1500	0	30
14	30	SALES	CHICAGO	7900	JAMES	CLERK	7698	03.12.81	950	(null)	30

```
PLAN_TABLE_OUTPUT
Plan hash value: 3809461229

-----------------------------------------------------------------------------------------
| Id  | Operation              | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |                |    14 |  1638 |     7  (15)| 00:00:01 |
|   1 |  MERGE JOIN            |                |    14 |  1638 |     7  (15)| 00:00:01 |
|   2 |   TABLE ACCESS CLUSTER| EMP            |    14 |  1218 |     2   (0)| 00:00:01 |
|   3 |    INDEX FULL SCAN     | IDXCL_EMP_DEPT |    14 |       |     1   (0)| 00:00:01 |
|*  4 |   SORT JOIN            |                |     4 |   120 |     5  (20)| 00:00:01 |
|   5 |    TABLE ACCESS FULL   | DEPT           |     4 |   120 |     4   (0)| 00:00:01 |
```

Explain Plan

SQL | 0,025 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			14	7
MERGE JOIN			14	7
TABLE ACCESS	EMP	CLUSTER	14	2
INDEX	IDXCL_EMP_DEPT	FULL SCAN	14	1
SORT		JOIN	4	5
Access Predicates				
EMP.DEPTNO=DEPT.DEPTNO				
Filter Predicates				
EMP.DEPTNO=DEPT.DEPTNO				
TABLE ACCESS	DEPT	FULL	4	4

Sort-merge joins read two tables independently and sort rows (according to WHERE statment) from each table in the order of connection key and then joins sorted rows. In case we have already sorted list we don't apply sort again. When we work with big amount of source, which don't fit memory, sort uses Operation memory.

2.3. Task 4: Hash Joins

```
SET AUTOTRACE ON EXPLAIN;
SELECT /*+ USE_HASH(dept emp) */ *
FROM dept, emp
WHERE emp.deptno= dept.deptno;
SELECT * FROM TABLE(dbms_xplan.display );
```

	DEPTNO	DNAME	LOC	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO_1
1	10	ACCOUNTING	NEW YORK	7782	CLARK	MANAGER	7839	09.06.81	2450	(null)	10
2	10	ACCOUNTING	NEW YORK	7839	KING	PRESIDENT	(null)	17.11.81	5000	(null)	10
3	10	ACCOUNTING	NEW YORK	7934	MILLER	CLERK	7782	23.01.82	1300	(null)	10
4	20	RESEARCH	DALLAS	7369	SMITH	CLERK	7902	17.12.80	800	(null)	20
5	20	RESEARCH	DALLAS	7566	JONES	MANAGER	7839	02.04.81	2975	(null)	20
6	20	RESEARCH	DALLAS	7788	SCOTT	ANALYST	7566	13.07.87	3000	(null)	20
7	20	RESEARCH	DALLAS	7876	ADAMS	CLERK	7788	13.07.87	1100	(null)	20
8	20	RESEARCH	DALLAS	7902	FORD	ANALYST	7566	03.12.81	3000	(null)	20
9	30	SALES	CHICAGO	7499	ALLEN	SALESMAN	7698	20.02.81	1600	300	30
0	30	SALES	CHICAGO	7521	WARD	SALESMAN	7698	22.02.81	1250	500	30
1	30	SALES	CHICAGO	7654	MARTIN	SALESMAN	7698	28.09.81	1250	1400	30
2	30	SALES	CHICAGO	7698	BLAKE	MANAGER	7839	01.05.81	2850	(null)	30
3	30	SALES	CHICAGO	7844	TURNER	SALESMAN	7698	13.07.87	1500	0	30
4	30	SALES	CHICAGO	7900	JAMES	CLERK	7698	03.12.81	950	(null)	30

```
PLAN_TABLE_OUTPUT
1  Plan hash value: 2553673404
2
3  ---------------------------------------------------------------------------------------
4  | Id  | Operation              | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
5  ---------------------------------------------------------------------------------------
6  |   0 | SELECT STATEMENT       |                |    14 |  1638 |     6   (0)| 00:00:01 |
7  |*  1 |  HASH JOIN             |                |    14 |  1638 |     6   (0)| 00:00:01 |
8  |   2 |   TABLE ACCESS FULL    | DEPT           |     4 |   120 |     4   (0)| 00:00:01 |
9  |   3 |   TABLE ACCESS CLUSTER| EMP            |    14 |  1218 |     2   (0)| 00:00:01 |
10 |   4 |    INDEX FULL SCAN     | IDXCL_EMP_DEPT |    14 |       |     1   (0)| 00:00:01 |
```

Explain Plan

SQL | 0,021 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			14	6
HASH JOIN			14	6
Access Predicates				
EMP.DEPTNO=DEPT.DEPTNO				
TABLE ACCESS	DEPT	FULL	4	4
TABLE ACCESS	EMP	CLUSTER	14	2
INDEX	IDXCL_EMP_DEPT	FULL SCAN	14	1
Other XML				

Hash joins reads two tables and applies WHERE condition. Base on the table statistic, the table which returns the least number of rows hashing in the memory. This hash table includes all table row data and is loaded into hash blocks based on a randomization function that converts the join key to a hash value. As long as there is enough memory, this hash table will remain in memory. However, if there is not enough available memory, the hash table can be written to temporary disk space. The next step is to read another larger table and apply a hash function to the join key column. This hash value is then used to check the smaller in-memory table for the corresponding hash bucket

containing the row data for the first table. Each bin has a list (represented by a bitmap) of the rows in that bin. This list is checked against the test string. If a match is found, the string is returned, otherwise it is discarded. The large line is read only once and each line is checked for a match. This is different from Nested Loops Joins, where the inner table is read multiple times. So actually in this case the big table is the leading table since they are only read once and the smaller hash table is read many times.

2.4. Task 5: Cartesian Joins

```
SET AUTOTRACE ON EXPLAIN;
SELECT * FROM dept, emp;
SELECT * FROM TABLE(dbms_xplan.display );
```

	DEPTNO	DNAME	LOC	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO_1
1	10	ACCOUNTING	NEW YORK	7369	SMITH	CLERK	7902	17.12.80	800	(null)	20
2	10	ACCOUNTING	NEW YORK	7499	ALLEN	SALESMAN	7698	20.02.81	1600	300	30
3	10	ACCOUNTING	NEW YORK	7521	WARD	SALESMAN	7698	22.02.81	1250	500	30
4	10	ACCOUNTING	NEW YORK	7566	JONES	MANAGER	7839	02.04.81	2975	(null)	20
5	10	ACCOUNTING	NEW YORK	7654	MARTIN	SALESMAN	7698	28.09.81	1250	1400	30
6	10	ACCOUNTING	NEW YORK	7698	BLAKE	MANAGER	7839	01.05.81	2850	(null)	30
7	10	ACCOUNTING	NEW YORK	7782	CLARK	MANAGER	7839	09.06.81	2450	(null)	10
8	10	ACCOUNTING	NEW YORK	7788	SCOTT	ANALYST	7566	13.07.87	3000	(null)	20
9	10	ACCOUNTING	NEW YORK	7839	KING	PRESIDENT	(null)	17.11.81	5000	(null)	10
10	10	ACCOUNTING	NEW YORK	7844	TURNER	SALESMAN	7698	13.07.87	1500	0	30
11	10	ACCOUNTING	NEW YORK	7876	ADAMS	CLERK	7788	13.07.87	1100	(null)	20
12	10	ACCOUNTING	NEW YORK	7900	JAMES	CLERK	7698	03.12.81	950	(null)	30
13	10	ACCOUNTING	NEW YORK	7902	FORD	ANALYST	7566	03.12.81	3000	(null)	20
14	10	ACCOUNTING	NEW YORK	7934	MILLER	CLERK	7782	23.01.82	1300	(null)	10
15	20	RESEARCH	DALLAS	7369	SMITH	CLERK	7902	17.12.80	800	(null)	20
16	20	RESEARCH	DALLAS	7499	ALLEN	SALESMAN	7698	20.02.81	1600	300	30
17	20	RESEARCH	DALLAS	7521	WARD	SALESMAN	7698	22.02.81	1250	500	30

```
PLAN_TABLE_OUTPUT
Plan hash value: 2034389985

-----------------------------------------------------------------------------
| Id  | Operation            | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |      |    56 |  6552 |    12   (0)| 00:00:01 |
|   1 |  MERGE JOIN CARTESIAN|      |    56 |  6552 |    12   (0)| 00:00:01 |
|   2 |   TABLE ACCESS FULL  | DEPT |     4 |   120 |     4   (0)| 00:00:01 |
|   3 |   BUFFER SORT        |      |    14 |  1218 |     8   (0)| 00:00:01 |
|   4 |    TABLE ACCESS FULL | EMP  |    14 |  1218 |     2   (0)| 00:00:01 |
```

Explain Plan

SQL | 0,016 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			56	12
MERGE JOIN		CARTESIAN	56	12
TABLE ACCESS	DEPT	FULL	4	4
BUFFER		SORT	14	8
TABLE ACCESS	EMP	FULL	14	2
Other XML				

2.5. Task 6: Left/Right Outer Joins

Worksheet | Query Builder

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

Script Output

Task completed in 0,183 seconds

```
     EMPNO ENAME      JOB              MGR HIREDATE        SAL       COMM     DEPTNO     DEPTNO DNAME          LOC
---------- ---------- --------- ---------- -------- ---------- ---------- ---------- ---------- -------------- -------------
      7782 CLARK      MANAGER         7839 09.06.81       2450                    10         10 ACCOUNTING     NEW YORK
      7839 KING       PRESIDENT            17.11.81       5000                    10         10 ACCOUNTING     NEW YORK
      7934 MILLER     CLERK           7782 23.01.82       1300                    10         10 ACCOUNTING     NEW YORK
      7369 SMITH      CLERK           7902 17.12.80        800                    20         20 RESEARCH       DALLAS
      7566 JONES      MANAGER         7839 02.04.81       2975                    20         20 RESEARCH       DALLAS
      7788 SCOTT      ANALYST         7566 19.04.87       3000                    20         20 RESEARCH       DALLAS
      7876 ADAMS      CLERK           7788 23.05.87       1100                    20         20 RESEARCH       DALLAS
      7902 FORD       ANALYST         7566 03.12.81       3000                    20         20 RESEARCH       DALLAS
      7499 ALLEN      SALESMAN        7698 20.02.81       1600        300         30         30 SALES          CHICAGO
      7521 WARD       SALESMAN        7698 22.02.81       1250        500         30         30 SALES          CHICAGO
      7654 MARTIN     SALESMAN        7698 28.09.81       1250       1400         30         30 SALES          CHICAGO

     EMPNO ENAME      JOB              MGR HIREDATE        SAL       COMM     DEPTNO     DEPTNO DNAME          LOC
---------- ---------- --------- ---------- -------- ---------- ---------- ---------- ---------- -------------- -------------
      7698 BLAKE      MANAGER         7839 01.05.81       2850                    30         30 SALES          CHICAGO
      7844 TURNER     SALESMAN        7698 08.09.81       1500          0         30         30 SALES          CHICAGO
      7900 JAMES      CLERK           7698 03.12.81        950                    30         30 SALES          CHICAGO

14 rows selected.
```

Cartesian Joins occur when all rows in one table are joined with all rows in another table. Therefore, the total number of rows resulting from the join is equal to the number of rows from one table (A) times the number of rows from the other table (B), so A x B = total number of rows in the result set. Cartesian joins often occur when a join condition is omitted or a specified join column is ignored, so the only possible operation is to simply join all rows from one table with all rows from another table.

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

Plan hash value: 3387915970

-----------------------------------------------------------------------------------------------------------
| Id  | Operation          | Name | Starts | E-Rows | A-Rows |   A-Time    | Buffers |  OMem |  1Mem | Used-Mem |
-----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |      1 |        |     14 |00:00:00.01 |       8 |       |       |          |

PLAN_TABLE_OUTPUT
-----------------------------------------------------------------------------------------------------------
|*  1 |  HASH JOIN OUTER   |      |      1 |     14 |     14 |00:00:00.01 |       8 |   604K|   604K|  627K (0)|
|   2 |   TABLE ACCESS FULL| EMP  |      1 |     14 |     14 |00:00:00.01 |       4 |       |       |          |
|   3 |   TABLE ACCESS FULL| DEPT |      1 |      4 |      4 |00:00:00.01 |       4 |       |       |          |
-----------------------------------------------------------------------------------------------------------

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

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

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			14	8
HASH JOIN		OUTER	14	8
Access Predicates				
E.DEPTNO=D.DEPTNO(+)				
NESTED LOOPS		OUTER	14	8
STATISTICS COLLECTOR				
TABLE ACCESS	SCOTT.EMP	FULL	14	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				

2.6. Task 7: Full Outer Join

SQL Worksheet History

8,53299999 seconds

Worksheet Query Builder

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

Script Output

Task completed in 8,533 seconds

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	DEPTNO	DNAME	LOC
7369	SMITH	CLERK	7902	17.12.80	800		20	20	RESEARCH	DALLAS
7499	ALLEN	SALESMAN	7698	20.02.81	1600	300	30	30	SALES	CHICAGO
7521	WARD	SALESMAN	7698	22.02.81	1250	500	30	30	SALES	CHICAGO
7566	JONES	MANAGER	7839	02.04.81	2975		20	20	RESEARCH	DALLAS
7654	MARTIN	SALESMAN	7698	28.09.81	1250	1400	30	30	SALES	CHICAGO
7698	BLAKE	MANAGER	7839	01.05.81	2850		30	30	SALES	CHICAGO
7782	CLARK	MANAGER	7839	09.06.81	2450		10	10	ACCOUNTING	NEW YORK
7788	SCOTT	ANALYST	7566	19.04.87	3000		20	20	RESEARCH	DALLAS
7839	KING	PRESIDENT		17.11.81	5000		10	10	ACCOUNTING	NEW YORK
7844	TURNER	SALESMAN	7698	08.09.81	1500	0	30	30	SALES	CHICAGO
7876	ADAMS	CLERK	7788	23.05.87	1100		20	20	RESEARCH	DALLAS

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	DEPTNO	DNAME	LOC
7900	JAMES	CLERK	7698	03.12.81	950		30	30	SALES	CHICAGO
7902	FORD	ANALYST	7566	03.12.81	3000		20	20	RESEARCH	DALLAS
7934	MILLER	CLERK	7782	23.01.82	1300		10	10	ACCOUNTING	NEW YORK
								40	OPERATIONS	BOSTON

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

Plan hash value: 51889263

-----------------------------------------------------------------------------------------------------------
| Id  | Operation              | Name     | Starts | E-Rows | A-Rows |   A-Time    | Buffers |  OMem |  1Mem | Used-Mem |
-----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |          |      1 |        |     15 |00:00:00.01 |       8 |       |       |          |

PLAN_TABLE_OUTPUT
-----------------------------------------------------------------------------------------------------------
|   1 |  VIEW                  | VW_FOJ_0 |      1 |     15 |     15 |00:00:00.01 |       8 |       |       |          |
|*  2 |   HASH JOIN FULL OUTER|          |      1 |     15 |     15 |00:00:00.01 |       8 |  801K|  801K|  902K (0)|
|   3 |    TABLE ACCESS FULL   | DEPT     |      1 |      4 |      4 |00:00:00.01 |       4 |       |       |          |
|   4 |    TABLE ACCESS FULL   | EMP      |      1 |     14 |     14 |00:00:00.01 |       4 |       |       |          |
-----------------------------------------------------------------------------------------------------------

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

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

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			15	8
VIEW	SYS.VW_FOJ_0		15	8
HASH JOIN		FULL OUTER	15	8
Access Predicates				
E.DEPTNO=D.DEPTNO				
TABLE ACCESS	SCOTT.DEPT	FULL	4	4
TABLE ACCESS	SCOTT.EMP	FULL	14	4
Other XML				

2.7. Task 8: Semi Joins

SQL Worksheet | History

0,19 seconds

Worksheet | Query Builder

```
SELECT /* using in */ deptno, dname FROM scott.dept d
WHERE deptno IN (select deptno from scott.emp e);
```

Script Output

Task completed in 0,19 seconds

```
    DEPTNO DNAME
---------- --------------
        10 ACCOUNTING
        20 RESEARCH
        30 SALES
```

```
SELECT /* using in */ deptno, dname FROM scott.dept d WHERE deptno IN
(select deptno from scott.emp e)

Plan hash value: 1090737117

-------------------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name    | Starts | E-Rows | A-Rows |   A-Time    | Buffers |  OMem |  1Mem | Used-Mem |
-------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |         |      1 |        |      3 |00:00:00.01 |       6 |       |       |          |

PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------------------------------------
|   1 |  MERGE JOIN SEMI             |         |      1 |      3 |      3 |00:00:00.01 |       6 |       |       |          |
|   2 |   TABLE ACCESS BY INDEX ROWID| DEPT    |      1 |      4 |      4 |00:00:00.01 |       2 |       |       |          |
|   3 |    INDEX FULL SCAN           | PK_DEPT |      1 |      4 |      4 |00:00:00.01 |       1 |       |       |          |
|*  4 |   SORT UNIQUE                |         |      4 |     14 |      3 |00:00:00.01 |       4 |  2048 |  2048 | 2048  (0)|
|   5 |    TABLE ACCESS FULL         | EMP     |      1 |     14 |     14 |00:00:00.01 |       4 |       |       |          |
-------------------------------------------------------------------------------------------------------------

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

   4 - access("DEPTNO"="DEPTNO")

PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------------------------------------
       filter("DEPTNO"="DEPTNO")
```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			3	7
MERGE JOIN		SEMI	3	7
TABLE ACCESS	SCOTT.DEPT	BY INDEX ROWID	4	2
INDEX	SCOTT.PK_DEPT	FULL SCAN	4	1
SORT		UNIQUE	14	5
Access Predicates				
DEPTNO=DEPTNO				
Filter Predicates				
DEPTNO=DEPTNO				
TABLE ACCESS	SCOTT.EMP	FULL	14	4
Other XML				

As we can see the inner join is not functionally equivalent to the semijoin because of the number of rows returned. The main difference between a normal inner join and a semijoin is that, with a semijoin, each record in the first set is returned only once, regardless of how many matches there are in the second set.

2.8. Task 9: Anti Joins

SQL Worksheet | History

0,193 seconds

Worksheet | Query Builder

```
SELECT /* using in */ deptno, dname FROM scott.dept d
WHERE deptno NOT IN (select deptno from scott.emp e);
```

Script Output

Task completed in 0,193 seconds

```
    DEPTNO DNAME
---------- --------------
        40 OPERATIONS
```

```
SELECT /* using in */ deptno, dname FROM scott.dept d WHERE deptno NOT
IN (select deptno from scott.emp e)

Plan hash value: 218628244

-----------------------------------------------------------------------------------------------------------------------
| Id  | Operation                          | Name    | Starts | E-Rows | A-Rows |   A-Time    | Buffers |  OMem |  1Mem | Used-Mem |
-----------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |         |      1 |        |      1 |00:00:00.01 |       7 |       |       |          |

PLAN_TABLE_OUTPUT
-----------------------------------------------------------------------------------------------------------------------
|   1 |  MERGE JOIN ANTI NA                |         |      1 |      1 |      1 |00:00:00.01 |       7 |       |       |          |
|   2 |   SORT JOIN                        |         |      1 |      4 |      4 |00:00:00.01 |       0 |  2048 |  2048 | 2048  (0)|
|   3 |    TABLE ACCESS BY INDEX ROWID BATCHED| DEPT    |      1 |      4 |      4 |00:00:00.01 |       2 |       |       |          |
|   4 |     INDEX FULL SCAN                | PK_DEPT |      1 |      4 |      4 |00:00:00.01 |       1 |       |       |          |
|*  5 |   SORT UNIQUE                      |         |      5 |     14 |      3 |00:00:00.01 |       0 |  2048 |  2048 | 2048  (0)|
|   6 |    TABLE ACCESS FULL               | EMP     |      1 |     14 |     14 |00:00:00.01 |       5 |       |       |          |
-----------------------------------------------------------------------------------------------------------------------

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


PLAN_TABLE_OUTPUT
-----------------------------------------------------------------------------------------------------------------------
   5 - access("DEPTNO"="DEPTNO")
       filter("DEPTNO"="DEPTNO")
```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	7
MERGE JOIN		ANTI NA	1	7
SORT		JOIN	4	2
TABLE ACCESS	SCOTT.DEPT	BY INDEX ROWID...	4	2
INDEX	SCOTT.PK_DEPT	FULL SCAN	4	1
SORT		UNIQUE	14	5
Access Predicates				
DEPTNO=DEPTNO				
Filter Predicates				
DEPTNO=DEPTNO				
TABLE ACCESS	SCOTT.EMP	FULL	14	4
Other XML				

2.9. Task 10: Prepare summary tableAnti-/Semi- Joins are applied depending

on the desired result(without repetition) and the selected Join type.

Join Access “A”	Join Access “B”	Nested Loop	Hash Join	Sort-Merge Join
Small Table	Small Table			
Small Table	Indexed Small Table			
Indexed Small	Indexed Small		ok, but may be not as effective as others	
Small	Big		ok, but hash table should be built for a smaller table	ok if tables are sorted
Big	Big		ok If the hash table can fit in memory	
Big	Indexed Big		ok If the hash table can fit in memory	
Indexed Big	Indexed Big		ok If the hash table can fit in memory	