|  |
| --- |
| EPAM Systems, RD Dep.  MTN.\*NIX.07 Oracle DB. Introduction to DWH |
| MTN.\*NIX.07 Labs - Access and Join Methods Part 2 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| REVISION HISTORY | | | | | |
| Ver. | Description of Change | Author | Date | Approved | |
| Name | Effective Date |
| 1.0 | Initial status of document | [**Kiryl Bucha**](mailto:Kiryl_bucha@epam.com) | 16-JAN-2012 |  |  |
|  |  |  |  |  |  |

# Auto Trace & Explain Plan

## Task 1: Auto Trace configuration training

Below all possible variants of SQL plus utilities autotrace:

set autotrace off

set autotrace on

set autotrace traceonly

set autotrace on explain

set autotrace on statistics

set autotrace on explain statistics

set autotrace traceonly explain

set autotrace traceonly statistics

set autotrace traceonly explain statistics

set autotrace off explain

set autotrace off statistics

set autotrace off explain statistics

**NOTE:** If you received next error: Check PLUSTRACE role is enabled. Please make next steps:

1. Run next script connected as sysdba:

# @ $ORACLE\_HOME/sqlplus/admin/plustrce.sql;

1. Grant role PLUSTRACE to $UserName$

# grant plustrace to $UserName$;

**Task Results:**

Expected:

Summary table with all result and text description of analyses this results.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Auto Trace Configuration Options | Expected Results | Description |
| 1 | set autotrace off | No AUTOTRACE report is generated. | Default |
| 2 | set autotrace on | The AUTOTRACE report includes data queried, the optimizer execution path and the SQL statement execution statistics. |  |
| 3 | set autotrace traceonly | TRACEONLY displays a trace report, but does not print query data |  |
| 4 | set autotrace on explain | EXPLAIN shows the query execution path by performing an EXPLAIN PLAN. Displays the execution plan only. |  |
| 5 | set autotrace on statistics | SQL\*Plus produces a STATISTICS report, a second connection to the database is automatically created. Displays the statistics only. |  |
| 6 | set autotrace on explain statistics | 5+6 |  |
| 7 | set autotrace traceonly explain | Displays the execution plan, but does not print query data |  |
| 8 | set autotrace traceonly statistics | Displays the statistics, but does not print query data |  |
| 9 | set autotrace traceonly explain statistics | 7+8 |  |
| 10 | set autotrace off explain | Autotrace Disabled |  |
| 11 | set autotrace off statistics | Autotrace Disabled |  |
| 11 | set autotrace off explain statistics | Autotrace Disabled |  |

# Join Methods

**The Main Task** is to create SQL and prepare execution plan of statements with join methods on Task Topics (Task 2 - 9)

**Task Results:**

There are several tasks below with the same main expected result points:

* Create SQL using next tables: scott.emp, scott.dept
* Create additional needed Tables and Indexes
* Prepare screenshots of execution plan

## Task 2: Nested Loops Joins

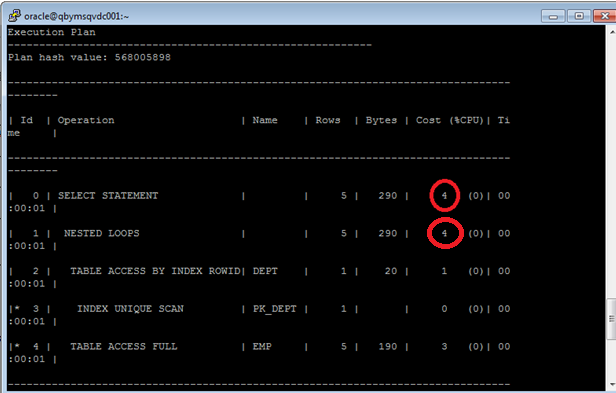
**Example:**

# SELECT \*

FROM scott.emp e, scott.dept d

WHERE e.deptno = d.deptno

AND d.deptno = 20

****

**Task:** Prepare SQL **explain plan** using: SQL plus Auto Trace Utility.

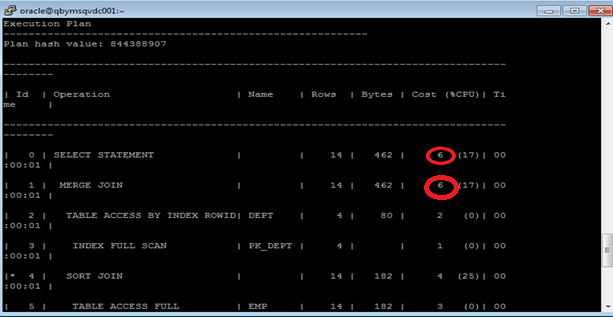
**Note:**  If you would like change in execution plan the type of join method use oracle performance hints. (USE\_NL)

## Task 3: Sort-Merge Joins

select /\*+ USE\_MERGE (dept emp) \*/ empno, ename, dname, loc

from scott.dept, scott.emp

where emp.deptno = dept.deptno;

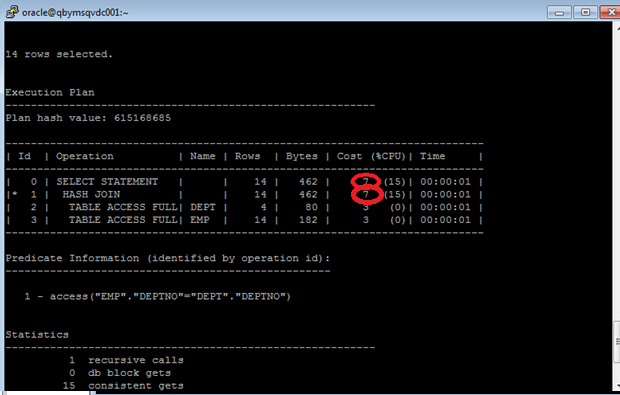


## Task 4: Hash Joins

select /\*+ USE\_HASH(dept emp) \*/ empno, ename, dname, loc

from scott.dept, scott.emp

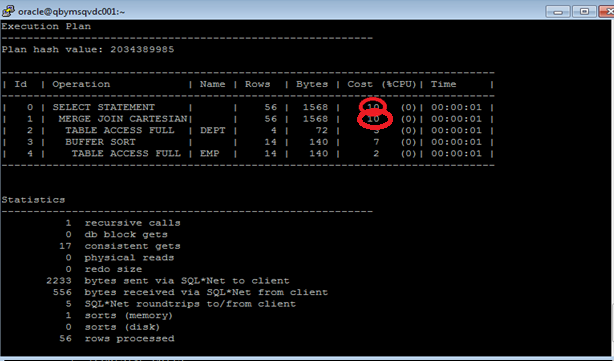
where emp.deptno = dept.deptno;



## Task 5: Cartesian Joins

select empno, ename, dname, loc

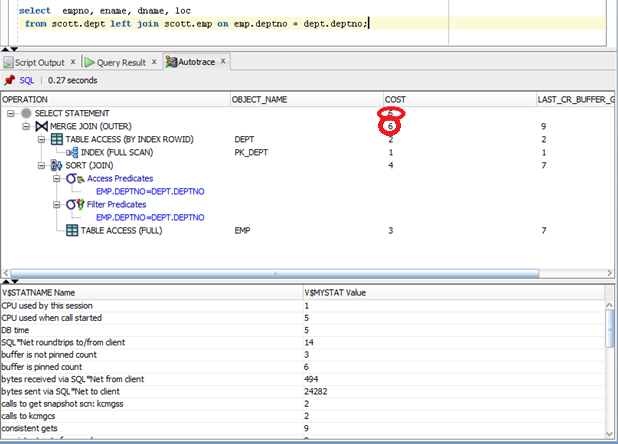
from scott.dept, scott.emp;



## Task 6: Left/Right Outer Joins

select empno, ename, dname, loc

from scott.dept left join scott.emp on emp.deptno = dept.deptno;

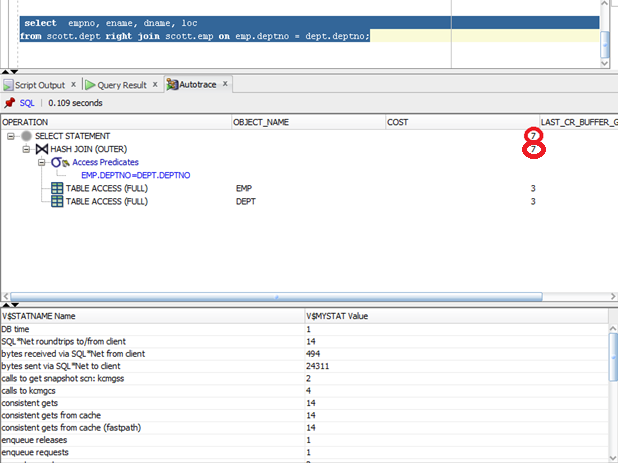
****

**Tasks:**

* Prepare SQL **trace protocol** using software: Oracle SQL Developer.
* Create SQL left outer join
* Create SQL right outer join

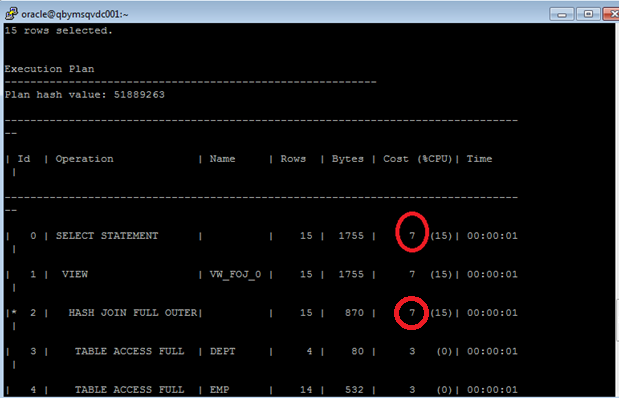
select empno, ename, dname, loc

from scott.dept right join scott.emp on emp.deptno = dept.deptno;



## Task 7: Full Outer Join

select \* from scott.emp full outer join scott.dept on (emp.deptno = dept.deptno);



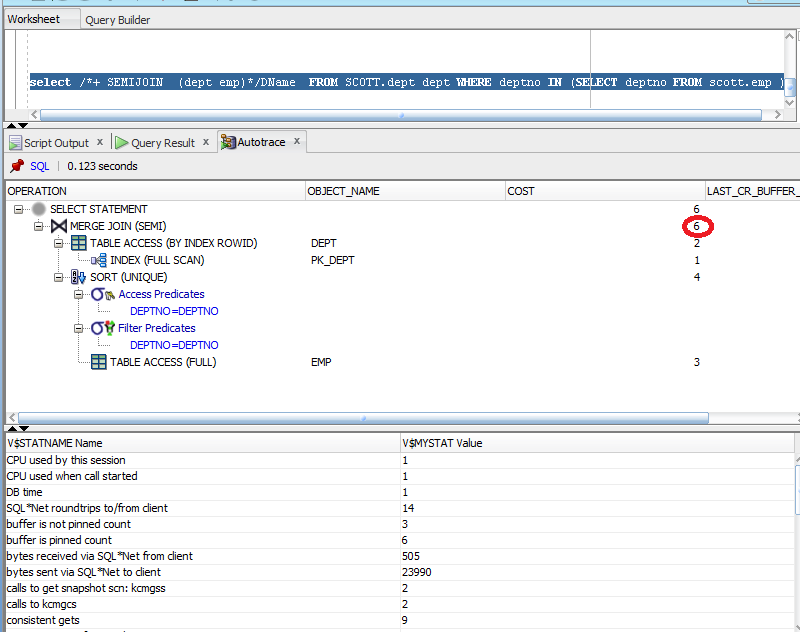
## Task 8: Semi Joins

**Task:** Prepare All possible variants of SEMI JOIN SQL **explain plan** using: SQL plus Auto Trace Utility.

**Note:**  If you would like change in execution plan the type of join method use oracle performance hints.

1. SEMIJOIN – perform a semi-join (the optimizer gets to pick which kind)
2. NO\_SEMIJOIN – obviously means don’t perform a semi-join
3. NL\_SJ – perform a nested loops semi-join (deprecated as of 10g)
4. HASH\_SJ – perform a hash semi-join (deprecated as of 10g)
5. MERGE\_SJ – perform a merge semi-join (deprecated as of 10g)

select /\*+ SEMIJOIN (dept emp)\*/DName FROM SCOTT.dept dept WHERE deptno IN (SELECT deptno FROM scott.emp );

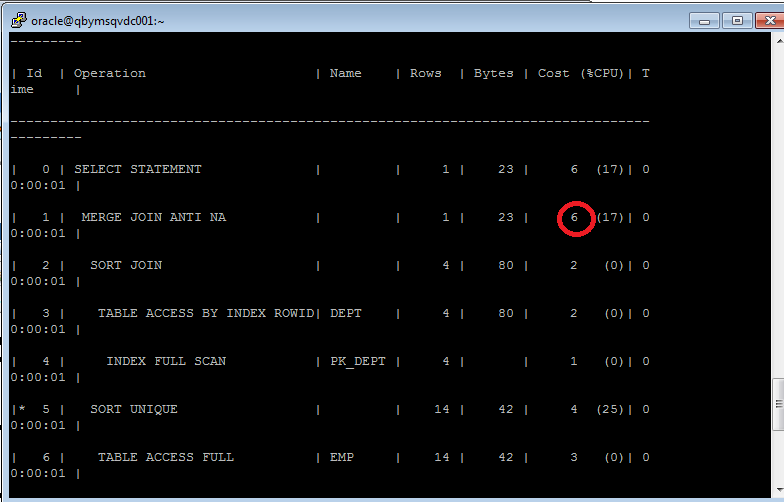


## Task 9: Anti Joins

**Task:** Prepare All possible variants of ANTI JOIN SQL **explain plan** using: SQL plus Auto Trace Utility.

**Note:**  If you would like change in execution plan the type of join method use oracle performance hints.

1. ANTIJOIN – perform an anti-join (the optimizer gets to pick which kind)
2. USE\_ANTI – older version of ANTIJOIN hint
3. NL\_AJ – perform a NESTED LOOPS anti-join (deprecated as of 10g)
4. HASH\_AJ – perform a HASH anti-join (deprecated as of 10g)
5. MERGE\_AJ – perform a MERGE anti-join (deprecated as of 10g)



## Task 10: Prepare summary table

In every type of join in our examples oracle did the operation of access of each table. So we can make the conclusion that in tables with indexes, iot tables, index clustered tables access full will cost fewer and will work faster. In Nested Loop, Sort-Merge Join, Anti-Join, Semi-Join joins we also need to make scan of column by which we connect two tables. So it is better to have unique index for it column.

**Task:** Make comparison of 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 | Anti-Join | Semi-Join |
| Small Table | Small Table | fast  because we need only full scan of tables, nested loop which will cost very few on small tables in case of unique row\_id and select statement | Fast  Because we need only access full of both tables, counting hash for each value. As we have every value unique, so it allow some time. | good | good | good |
| Small Table | Indexed Small Table | super fast  because we need only full access of 1table, index full access,unique index scan, nested loop which will cost very few on small tables in case of unique row\_id and select statement  All operations with indexes work very fast if we have not very large tables. | Fast  Because we need access full of both tables, counting hash for each value. As we have every value unique, so it allow some time. | Not good varian, we have different by size tables and, we have to make full scan of 1 table, index full scan of 2 table, sort them, merge and then select | Used for clauses with not exist, not in. Oracle make a lot of operations to make such select.  Full access, sort, table access, sort again and then join and select. | Something like anti-join. |
| Heap table | Iot table | We can use it for middle by size tables, it will be fast if we have index. But in case of huge table, loop programming will take a lot of time. | It is good variant if the iot table is small and big heap table, because oracle build bitmap for iot and then very fast compose rows from heap table. But in case when we have very large both tables and id will be unique for each row it can take some time | It is good variant If tables are not very huge and have the same id graduation one row from 1 table corresponds to one row of the second. | -//-  With growing tables will take more and more time. Hard operations.  But if we know that it will return not very much rows it can be used. | -//-  Something like anti-join. |
| Heap table | Hash clustered table | Bad variant, because we have no advantages from clustering. | Super, because our values of each deptno is stored in different blocks so oracle built hash for one value in block and jump to another block. | Very good variant, because our values of each deptno is stored in different blocks, so they have been already sorted. | It will be work better here, because  because our values of each deptno is stored in different blocks | -//- |