**Alexander Korytko**

Task 5

# 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 | Disables all autotrace | No AUTOTRACE report is generated. This is the default |
| 2 | set autotrace on | Shows the execution plan as well as statistics of the statement. | The AUTOTRACE report includes both the optimizer execution path and the SQL statement execution statistics. |
| 3 | set autotrace traceonly | Displays the execution plan and the statistics (as set autotrace on does), but doesn't print a query's result. | Similar to 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 | Displays the optimizer execution plan only. | The AUTOTRACE report shows only the optimizer execution path. |
| 5 | set autotrace on statistics | Displays the execution statistics only. | The AUTOTRACE report shows only the SQL statement execution statistics. |
| 6 | set autotrace on explain statistics | Display the execution plan, execution statistics and data queried | The AUTOTRACE report includes data queried, the optimizer execution path and the SQL statement execution statistics |
|  |  |  |  |
| 7 | set autotrace traceonly explain | Display execution plan without executing the query | Shows the execution plan, but does not execute the query |
| 8 | set autotrace traceonly statistics | Display statistics only without executing the query | Shows the execution statistics. Note: This runs the query, but does not display the results. |
| 9 | set autotrace traceonly explain statistics | Display execution plan and statistics without executing the query | Shows the execution plan and statistic, runs the query, but does not display the result of this query |
|  |  |  |  |
| 10 | set autotrace off explain | Turns off execution plan reporting |  |
| 11 | set autotrace off statistics | Turns off execution statistics reporting |  |
| 12 | set autotrace off explain statistics | Turns off execution plan reporting and execution statistics reporting |  |

# 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

**IOT\_TABLE:**

CREATE TABLE iot\_emp ( empno PRIMARY KEY, ename, job )

ORGANIZATION INDEX TABLESPACE USeRS

AS SELECT emp.empno, emp.ename, emp.job

FROM emp;

**Index Cluster Tables:**

*-- Create CLuster*

CREATE cluster dept\_emp\_cluster( deptno NUMBER( 2 ) )

SIZE 1024

STORAGE( INITIAL 100K NEXT 50K );

*-- CREATE index*

CREATE INDEX idxcl\_emp\_dept on cluster dept\_emp\_cluster;

*--CREATE Table 1 in our cluster*

CREATE TABLE ict\_dept

(

deptno NUMBER( 2 ) PRIMARY KEY

, dname VARCHAR2( 14 )

, loc VARCHAR2( 13 )

)

cluster dept\_emp\_cluster ( deptno ) ;

*--CREATE Table 2 in our cluster*

CREATE TABLE ict\_emp

(

empno NUMBER PRIMARY KEY

, ename VARCHAR2( 10 )

, job VARCHAR2( 9 )

, deptno NUMBER( 2 ) REFERENCES dept( deptno )

)

cluster dept\_emp\_cluster ( deptno ) ;

*--Fiil in our Table 1*

INSERT INTO ict\_dept( deptno , dname , loc)

SELECT deptno , dname , loc

FROM scott.dept;

*--Fiil in our Table 2*

INSERT INTO ict\_emp ( empno, ename, job, deptno )

SELECT rownum, ename, job, deptno

FROM scott.emp

**Index Cluster Tables:**

*-- Create Hash\_CLuster*

CREATE CLUSTER hash\_cluster (deptno NUMBER( 2 ))

SIZE 1024

STORAGE (INITIAL 100K NEXT 50K

MINEXTENTS 1 MAXEXTENTS 3

PCTINCREASE 0)

HASH IS deptno HASHKEYS 150;

*--CREATE Table 1 in our hash\_cluster*

CREATE TABLE hash\_dept

(

deptno NUMBER( 2 ) PRIMARY KEY

, dname VARCHAR2( 14 )

, loc VARCHAR2( 13 )

)

cluster hash\_cluster ( deptno ) ;

*--CREATE Table 2 in our hash\_cluster*

CREATE TABLE hash\_emp

(

empno NUMBER PRIMARY KEY

, ename VARCHAR2( 10 )

, job VARCHAR2( 9 )

, deptno NUMBER( 2 ) REFERENCES dept( deptno )

)

cluster hash\_cluster ( deptno ) ;

*--Fiil in our Table 1*

INSERT INTO hash\_dept( deptno , dname , loc)

SELECT deptno , dname , loc

FROM scott.dept;

*--Fiil in our Table 2*

INSERT INTO hash\_emp ( empno, ename, job, deptno )

SELECT rownum, ename, job, deptno

FROM scott.emp

* Prepare screenshots of execution plan

## Task 2: Nested Loops Joins

**Example:**

SELECT \*

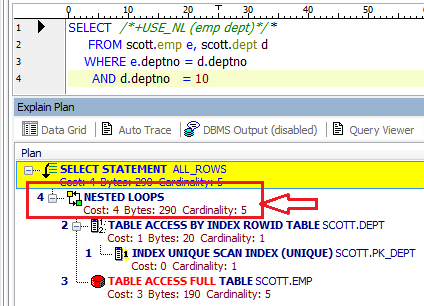
FROM scott.emp e, scott.dept d

WHERE e.deptno = d.deptno

AND d.deptno = 10

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

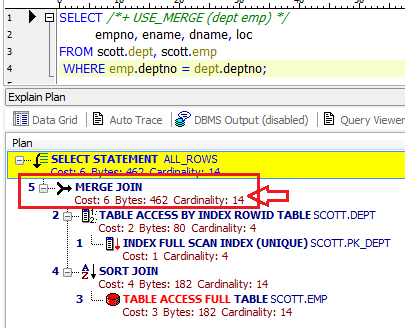
**Note:**  If you would like change in execution plan the type of join method use oracle performance hints. (USE\_NL)



## Task 3: Sort-Merge Joins

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

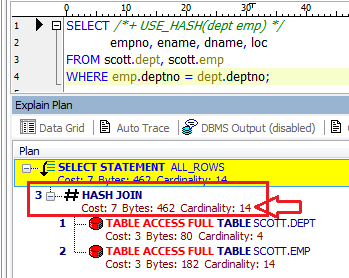
**Note:**  If you would like change in execution plan the type of join method use oracle performance hints. (USE\_MERGE)



## Task 4: Hash Joins

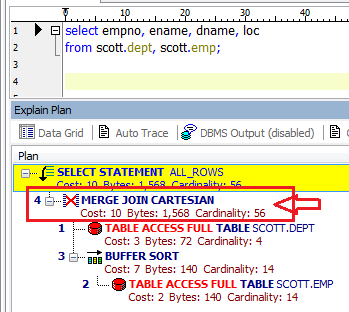
**Task:** Prepare SQL **explain plan** using software: Oracle SQL Developer.

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



## Task 5: Cartesian Joins

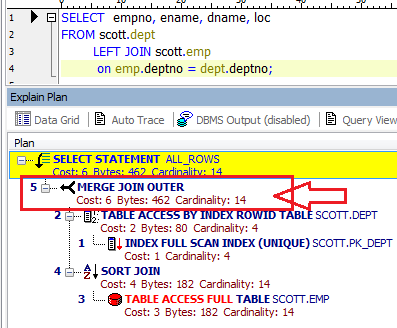
**Task:** Prepare SQL **explain plan** using software: Oracle SQL Developer.



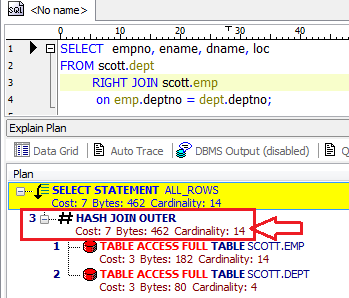
## Task 6: Left/Right Outer Joins

**Tasks:**

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

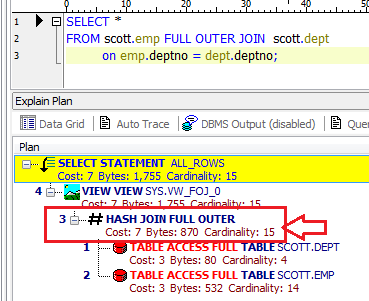


* Create SQL right outer join



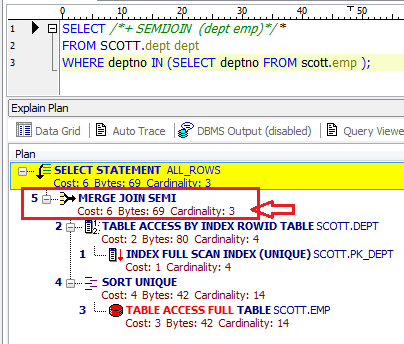
## Task 7: Full Outer Join

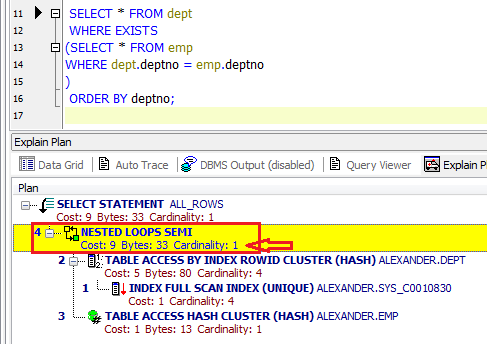
**Task:** Prepare SQL **explain plan** using: SQL plus Auto Trace Utility.



## Task 8: Semi Joins

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



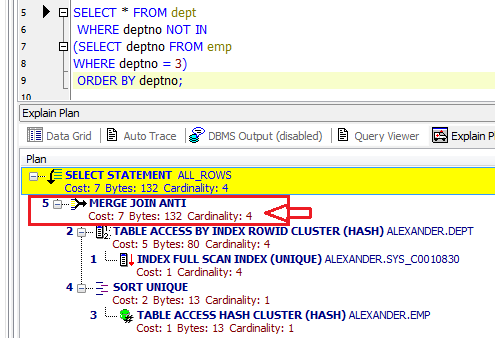


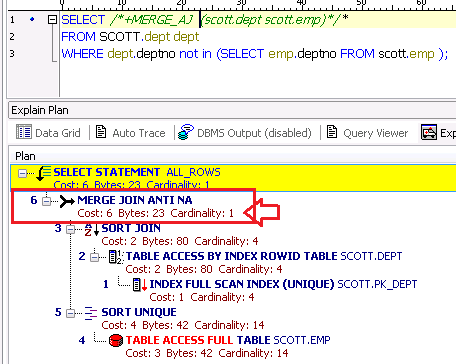
**Note:**  If you would like change in execution plan the type of join method use oracle performance hints.

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)

## 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

**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 | Very slow operation on these 2 types of tables (without indexes) Cost=3425 (100000 rows) Operation is slow because we need to read full Tables and for 1 row from table 1 trying to find match in table 2. Good type of Join when we need to get a first N rows in result | Have a best performance, than NL JOIN if we need to read the entire result sets. Because it’s takes time to create Hash table. Cost =425 Because we do only Range Scan for 1 table, Than Access by RowID to this table and full Table Access for Table 2 | Work the same like Hash Join by Cost. Good performance when we have index in out tables. All Tables should be orderd by join key and that’s can take some time. Can be useful for a Range Joins (>,<) Hash Joins available only for = | Work the same like Hash Join by Cost. | I think It’s working the slow with tables without sorted key or Index |
| Small Table | Index Small Table | Working faster then previous table types (without index) because we can use index key in this case | I thins it’s better to use Nested Loop or Merge join (because we have index). Index not help as to improve more speed for result sets | Good type of Join because we can sort columsn by index and it’s save a lot of time for processing result | Working slowly, we need to find only these items wich doesn\t have matches and it’s take time | Good choose wen we have index in EXIST query. So we don\t need to search next identical item if we found one, all other items will be skipped |
| Small Table | IOT Table | Good performance, It’s easy to take RowID in this case (if index is a join key) | Not really faster then in Index table. We still need to create hash Table for one table and use hash key to check the same values from both tables | Really good performance in this case. It’s easy to sort Index in this type of tables. | Slow speed if we have big data set. We need to match all rows from subquery and check if they exist in another table | Working better then with Index Small table. |
| Small Table | Index Cluster Table | Not a good perfomance, because Inner because table are not in one cluster | Not a good perfomance, because Inner table is in a cluster So we need to create a Hash Table and match the data with data in different blocks | It’s working better than with Index hash cluster table because it’s easy to sort data in the same blocks. But any way it’s working slowly than with IOT Table | The same like Anti-Join (in case of speed performance). Inner table are stored in cluster and Small table not that’s why we don’t have all advantages from cluster structure | It has good performance, because we have index cluster table in subquery. But probably, it’s better to use regular index table in this case, because in our case both our tables not in one cluster |
| Small Table | Index Hash Cluster Table | It’s working slowly because tables are not in one cluster. It’s better to use this type of join with tables in one cluster | It’s better to use this type of Join with unsorted big sets of data. | It’s not a good type of Join for this Tables type, because for Merge join we need to sort our columns by data. Data stored in cluster and it’s slow down the check speed, because tables are stored in different blocks | I think it’s working good for subquery because hash speed up the search results but it’s working slow in scope with Small not indexed Table | It’s hag a good speed because in this case we need read hash key to find matches in both tables. |