# Auto Trace & Explain Plan

|  |  |  |  |
| --- | --- | --- | --- |
| № | Auto Trace Configuration Options | Expected Results | Description |
| 1 | set autotrace off | Nothing | No AUTOTRACE report is generated. This is the default. |
| 2 | set autotrace on | STATISTICS, optimizer execution path, SQL statement execution statistics | The AUTOTRACE report includes both the optimizer execution path and the SQL statement execution statistics. |
| 3 | set autotrace traceonly | STATISTICS, optimizer execution path, SQL statement execution statistics, but nothing on screen | 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 | Optimizer execution path | The AUTOTRACE report shows only the optimizer execution path |
| 5 | set autotrace on statistics | SQL statement execution statistics | The AUTOTRACE report shows only the SQL statement execution statistics. |
| 6 | set autotrace traceonly explain statistics | Optimizer execution path, SQL statement execution statistics | The AUTOTRACE report shows the SQL statement execution statistics and the optimizer execution path |

# Join Methods

**Script for creating TABLES, INDEXES, CLUSTERS :**

--Head tables

create table small\_dept as select \* from scott.dept;

create table small\_emp as select \* from scott.emp;

--Head tables with index

create table ind\_dept as select \* from scott.dept;

create index ind\_dept\_index on ind\_dept(deptno);

create table ind\_emp as select \* from scott.emp;

create index ind\_emp\_index on ind\_emp(deptno);

--Index organized tables

create table emp\_iot

(empno, ename, job, hiredate, sal, comm, deptno,

constraint emp\_iot\_pk primary key(empno))

organization index

as

select empno, ename, job, hiredate, sal, comm, deptno

from scott.emp

CREATE TABLE DEPT\_IOT

( deptno, dname, loc,

constraint dept\_iot\_pk primary key(deptno) )

organization index

as

select \* from scott.dept;

--Index clustered tables

CREATE cluster emp\_dept\_cluster( deptno NUMBER( 2 ) )

SIZE 1024

STORAGE( INITIAL 100K NEXT 50K );

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

create table emp\_index\_cluster

(empno, ename, job, hiredate, sal, comm, deptno,

constraint emp\_index\_pk primary key(empno))

cluster emp\_dept\_cluster ( deptno )

as

select empno, ename, job, hiredate, sal, comm, deptno

from scott.emp;

CREATE TABLE dept\_index\_cluster

( deptno, dname, loc,

constraint dept\_clust\_pk primary key(deptno) )

cluster emp\_dept\_cluster ( deptno )

as

select \* from scott.dept;

--HASH clustered tables

Create cluster emp\_dept\_cluster\_hash (deptno NUMBER( 2 ))

tablespace users storage (initial 250K NEXT 50K MINEXTENTS 1 MAXEXTENTS 10 PCTINCREASE 0)

HASH IS deptno HASHKEYS 150 ;

create table emp\_hash\_cluster

(empno, ename, job, hiredate, sal, comm, deptno,

constraint emp\_hash\_pk primary key(empno))

cluster emp\_dept\_cluster\_hash ( deptno )

as

select empno, ename, job, hiredate, sal, comm, deptno

from scott.emp;

CREATE TABLE dept\_hash\_cluster

( deptno, dname, loc,

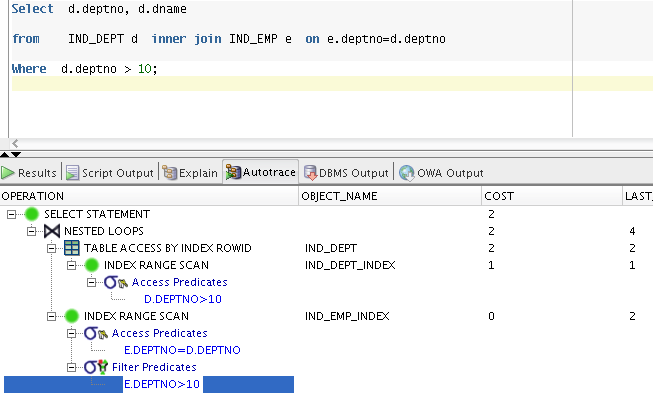
constraint dept\_hash\_pk primary key(deptno) )

cluster emp\_dept\_cluster ( deptno )

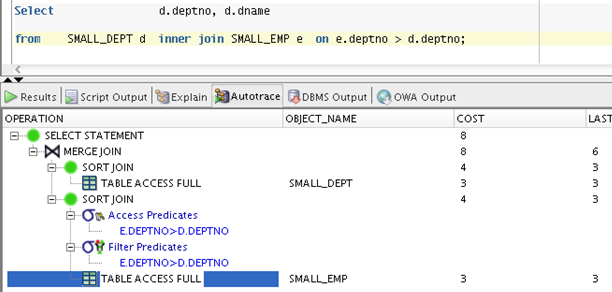
as

select \* from scott.dept;

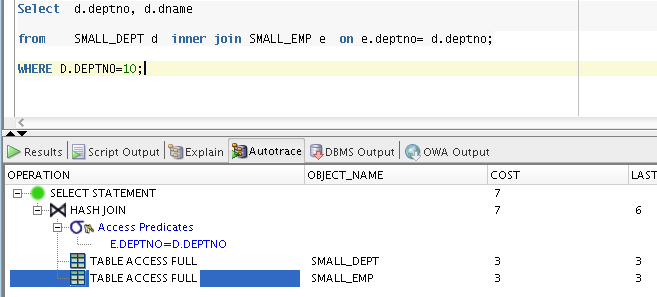
## Task 2: Nested Loops Joins

****

## Task 3: Sort-Merge Joins

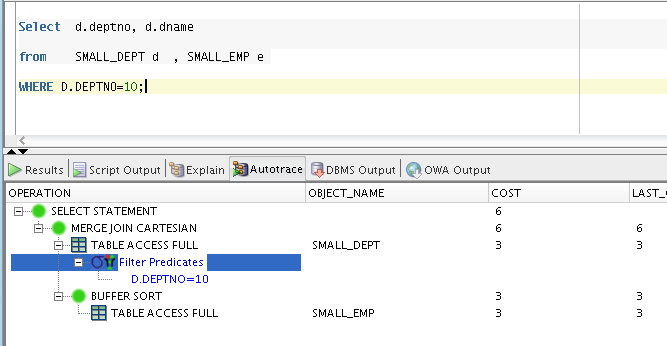


## Task 4: Hash Joins

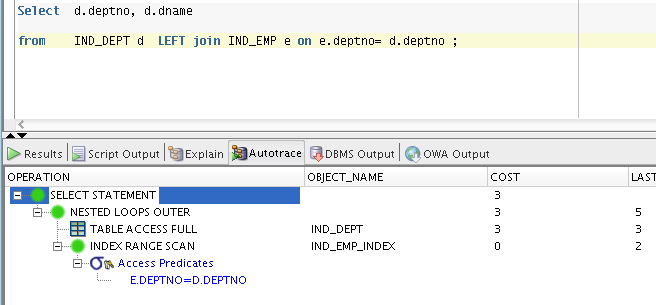


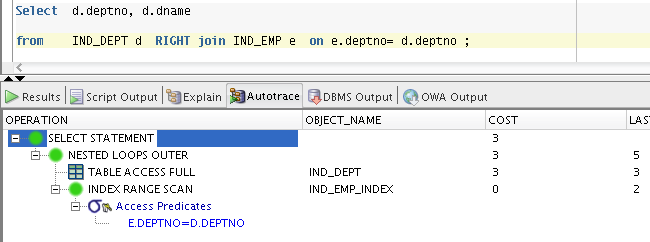
## 

## Task 5: Cartesian Joins

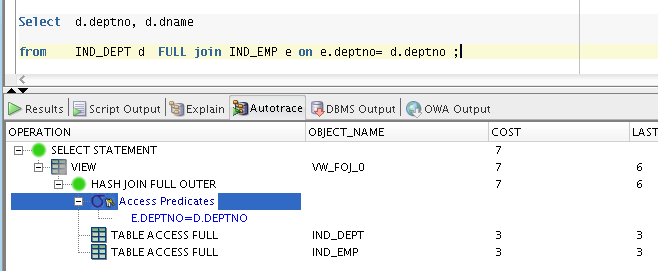


## Task 6: Left/Right Outer Joins

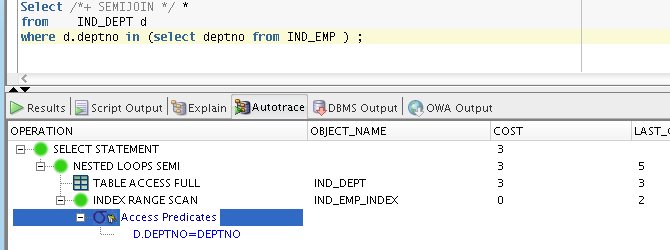
****

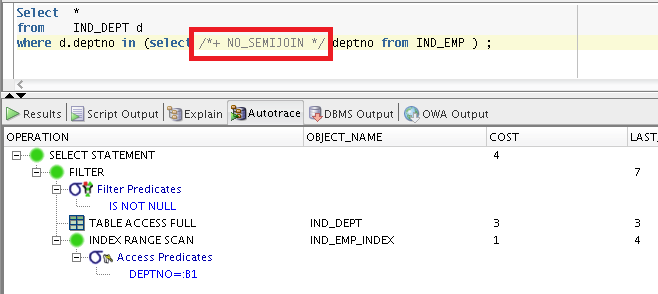
****

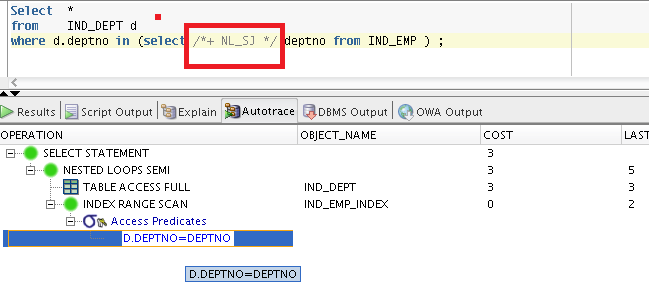
## Task 7: Full Outer Join

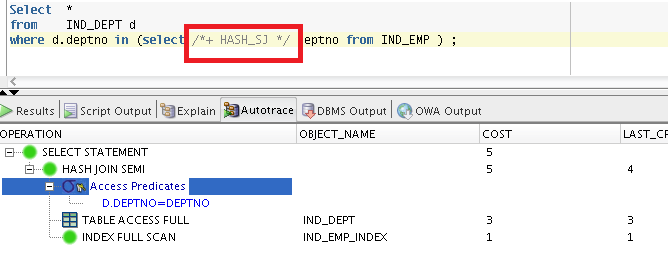


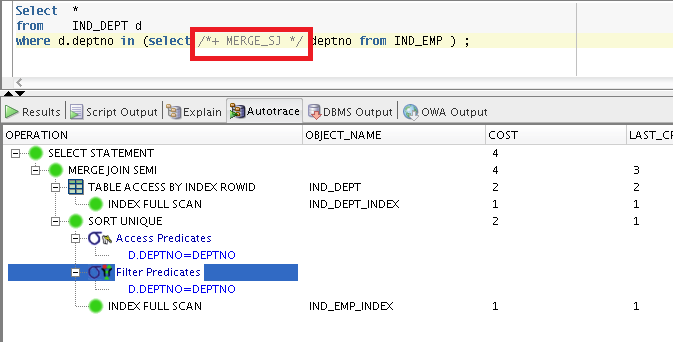
## Task 8: Semi Joins



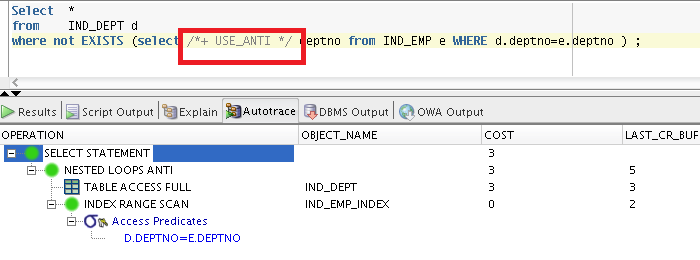


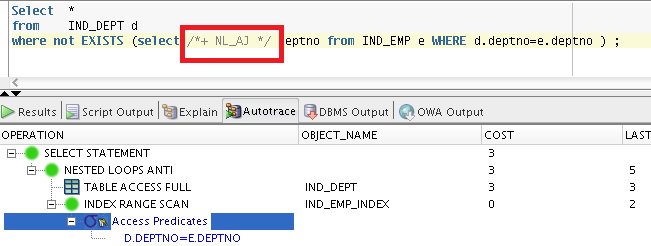


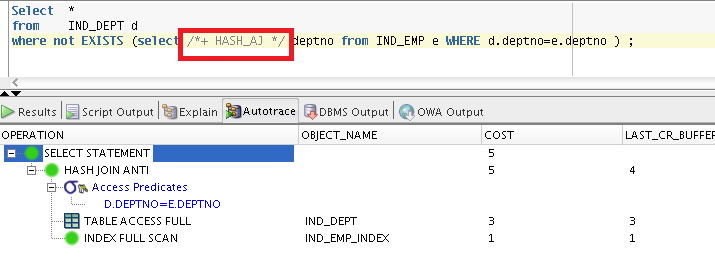


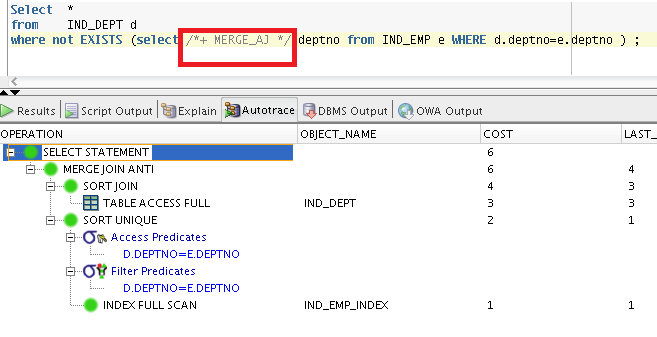


## Task 9: Anti Joins









## 

## Task 10: Prepare summary table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Join Access “A” | Join Access “B” | Nested Loop | Hash Join | Sort-Merge Join | Anti-Join | Semi-Join |
| small\_dept | small\_emp | -required indexes for FK columns | Have good performance, when we haven’t got indexes for FK | Without indexes work rather slow | Performance as in simple hash join query. | Without any indexes the speed is not very fast. |
| small\_dept | ind\_emp | Rather effective if we have indexes for columns(we have index for one FK), used in join. Use very little memory. | Unused by default(nested loop more faster)  Hash joins can efficiently process large, unsorted, non indexed inputs. | Indexes improves Sort-merge join. Very convenient to sort data by index. | Rather fast, as we need to read only index to find no match in it. Table is unused. | Indexes in sub query improve the speed very well. When the first much in found in index, we don’t need to search more. |
| small\_dept | emp\_iot | -required indexes for FK columns | A little bit faster, (I guess, it more faster to build hash in IOT) | Using IOT is good for performance, but it faster to sort data by index. We need to read ALL data in table in my query and in IOT we should read all index structure. | Not so fast in sub query table, as simple index, as we should read in fast full scan all index structure. | Using IOT is improves results, no doubt, but we don’t need to read data, we should only find match in index structure |
| small\_dept | emp\_index\_cluster | Tables don’t store at the same cluster, have no any benefits compared with ind\_emp table and small\_dept table | Unused by default(nested loop more faster)  Hash joins can efficiently process large, unsorted, non indexed inputs. | Hash clusters KILLED Sort-Merge Join, because we can’t sort hash, we can sort data. | Index cluster in sub query improves speed, reduce number of reading blocks, as they all are near each other for indexed fiels | Creating simple indexes for table in sub-query is more effective, I don’t see sense for creating index cluster for Semi-join. |
| small\_dept | emp\_hash\_cluster | Tables don’t store at the same cluster, have no any benefits compared with ind\_emp table and small\_dept table | Unused by default(nested loop more faster)  Hash joins can efficiently process large, unsorted, non indexed inputs. | Hash clusters KILLED Sort-Merge Join, because we can’t sort hash, we can sort data. | Oracle use by default merge-join, when we use index clusters, so we need to avoid sorting in query plan or don’t create hash clusters. | Hash cluster is a good practice, when it created for table in sub-query, it can improves performance a lot, as it fast to read hash to find match. |
| ind\_dept | ind\_emp | Effective, because we have indexes for all FK in query columns, used in join. Use very little memory. | Unused by default(nested loop more faster)  Hash joins can efficiently process large, unsorted, non indexed inputs. | Have best performance from all tables. Using The max COST have sort operation, that’s why for best performance whe should think what will be the best for sorting data | Rather fast, as we need to read only index to find no match in it. Table is unused. | Indexes in sub query improve the speed very well. When the first much in found in index, we don’t need to search more. |
| ind\_dept | emp\_iot | -In queries used HASH join | A little bit faster, (I guess, it more faster to build hash in IOT) | Indexes improves Sort-merge join. Very convenient to sort data by index | Not so fast in sub query table, as simple index, as we should read in fast full scan all index structure. | Using IOT is improves results, no doubt, but we don’t need to read data, we should only find match in index structure |
| ind\_dept | emp\_index\_cluster | Tables don’t store at the same cluster, have no any benefits compared with ind\_emp table and small\_dept table | Unused by default(nested loop more faster)  Hash joins can efficiently process large, unsorted, non indexed inputs. | Indexes improves Sort-merge join. Very convenient to sort data by index.  Index cluster improves performance not better than simple index. | Index cluster in sub query improves speed, reduce number of reading blocks, as they all are near each other for indexed fiels | Creating simple indexes for table in sub-query is more effective, I don’t see sense for creating index cluster for Semi-join. |
| ind\_dept | emp\_hash\_cluster | Tables don’t store at the same cluster, have no any benefits compared with ind\_emp table and small\_dept table | Have a good results, but in clause, that we build hash cluster, using | Hash clusters KILLED Sort-Merge Join, because we can’t sort hash, we can sort data. | Oracle use by default merge-join, when we use index clusters, so we need to avoid sorting in query plan or don’t create hash clusters. | Hash cluster is a good practice, when it created for table in sub-query, it can improves performance a lot, as it fast to read hash to find match. |
| dept\_iot | emp\_iot | Have EXCELLENT performance, IOT makes loop search of FK very convinient | Unused by default(nested loop more faster)  Hash joins can efficiently process large, unsorted, non indexed inputs. | Index cluster and IOT improves performance not better than simple index. | Not so fast in sub query table, as simple index, as we should read in fast full scan all index structure. | Using IOT is improves results, no doubt, but we don’t need to read data, we should only find match in index structure. |
| dept\_iot | emp\_index\_cluster | Have the same excellent results. | Unused by default(nested loop more faster) | Index cluster and IOT improves performance not better than simple index. | Index cluster in sub query improves speed, reduce number of reading blocks, as they all are near each other for indexed fiels | Creating simple indexes for table in sub-query is more effective, I don’t see sense for creating index cluster for Semi-join. |
| dept\_iot | emp\_hash\_cluster | Rather fast, but instead of hash cluster, index will be better | Unused by default(nested loop more faster) | Hash clusters KILLED Sort-Merge Join, because we can’t sort hash, we can sort data. | Oracle use by default merge-join, when we use index clusters, so we need to avoid sorting in query plan or don’t create hash clusters. | Hash cluster is a good practice, when it created for table in sub-query, it can improves performance a lot, as it fast to read hash to find match. |
| dept\_index\_cluster | emp\_index\_cluster | Rather fast, but not faster, then IOT. | Hash joins can efficiently process large, unsorted, non indexed inputs. | Performance of IOTs is better than Index cluster in Sort-merge join. | Index cluster in sub query improves speed, reduce number of reading blocks, as they all are near each other for indexed fiels | Creating simple indexes for table in sub-query is more effective, I don’t see sense for creating index cluster for Semi-join. |
| dept\_index\_cluster | emp\_hash\_cluster | Good results, but hash cluster slows query. | Unused by default(nested loop more faster) | Hash clusters KILLED Sort-Merge Join, because we can’t sort hash, we can sort data. | Oracle use by default merge-join, when we use index clusters, so we need to avoid sorting in query plan or don’t create hash clusters. | Hash cluster is a good practice, when it created for table in sub-query, it can improves performance a lot, as it fast to read hash to find match. |
| dept\_hash\_cluster | emp\_hash\_cluster | Rather slow, creating hash cluster have no sense | Hash joins can efficiently process large, unsorted, non indexed inputs. | Hash clusters KILLED Sort-Merge Join, because we can’t sort hash, we can sort data. | Oracle use by default merge-join, when we use index clusters, so we need to avoid sorting in query plan or don’t create hash clusters. | Hash cluster is a good practice, when it created for table in sub-query, it can improves performance a lot, as it fast to read hash to find match. |

