

## **EXPLAIN PLAN**

### **EXECUTION PLAN**

To run an SQL query, oracle performs various operations in the background i.e. to retrieve the result directly from the database or makes it ready for the user to execute the SQL or does both. This type of method which Oracle employs is called the 'Execution plan'. The execution plan's result is stored in plan\_table (SRUNOKPN.PLAN\_TABLE). Each rows inserted into this table can be different, i.e. (1) it can specify the table access method for the given SQL statement i.e. either by full table scan or Index –scan (if the column given in the condition is indexed) or Range – scan (if there are no index or if it couldn't find the appropriate index to retrieve the data), (2) join method used for retrieving data from multiple tables and (3) the way in which ordering is performed on a dataset which can be sort or filter operation.

### **PRE REQUISITES TO WORK WITH EXECUTION PLAN**

To make sure we have access permissions to read and edit the table contents which the SQL statements accesses.

To make sure that we can insert rows into plan table if it is under a different schema and also access to view the output from plan\_table

I am using TOAD to view the results of 'Explain plan'

### **ACCESS METHOD IN ORACLE**

#### **Full table scan**

Oracle retrieves records after accessing the entire table. *Fig.1* shows 2 things, (1) that a full-table scan is done on the 'CUSTOMER' table in 'SRUNOKPN' schema to retrieve the necessary records in the table (access method). (2) The cost for running the query is NULL which means that the optimizer has used 'Rule based optimizer' to execute this query. Usually full-table scan is not advisable for large tables.

STATEMENT_ID	OPERATION	OPTIONS	OBJECT_OWNER	OBJECT_NAME	OBJECT_INSTANCE	OPTIMIZER	ID	PARENT_ID	POSITION	COST
SRUNOKPN:092208150346	9/22/2008					CHOOSE	0			
SRUNOKPN:092208150346	9/22/2008	FULL	SRUNOKPN	CUSTOMER			1	0	1	

(Fig. 1 – Full table scan result which shows that all the rows in 'customer' table has been accessed to return the result set)

## Index scan

Indexing is usually done on a column that is unique or often used. By doing this on such columns it enable SQL statement to retrieve data faster. It is one of the efficient methods used in improving the performance of the query. More explanation and examples are given in the phase -2 of this case study.

## Range scan

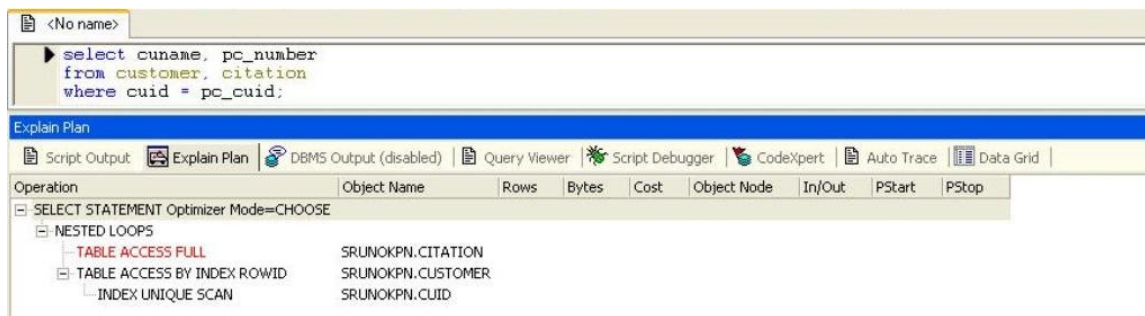
The data is accessed by using row id, which is a pseudo column in Oracle and every row in a table is associated with a unique id. A row id gives granular detail about the location of a row i.e. it leads to the location of the file, the block & finally to the slot where the data exists. If the row id is know, then this access method is the quickest way of retrieving data

## **JOINS**

### Nested loop

Nested loop is used by SQL statements that access multiple columns from different tables to retrieve results. When two tables are used in SQL statement, then all the rows in the first table satisfying the given 'where' condition is returned and then the result of the first table probes the second table to return result satisfying each row in the first table. The result for the second table is obtained by performing either full table scan (FTS) or by index. An example of the nested loop is given by the following SQL statement which returns all the customers who are both in customer and citation table.

```
select cuid, cuname, pc_number
from customer, citation
where cuid = pc_cuid
and substr(pc_number,1,2) = '07';
```



(Fig. 2 shows the two tables that are nested i.e. customer & citation table. 'CUID' is the primary key for customer table and 'PC\_CUID' is the foreign key in citation table)

#### Understanding the explain plan for the above diagram,

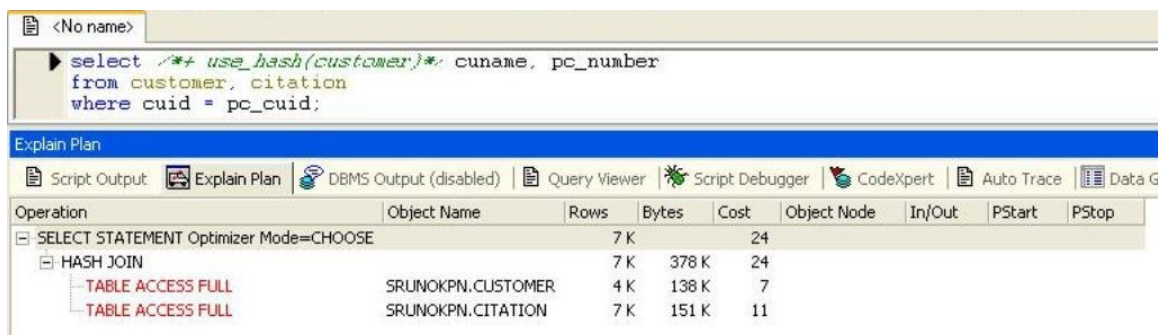
- Nested loops means there exists a join condition between the citation and customer table
- Table access full (SRUNOKPN.CITATION) means full table scan is done for citation in SRUNOKPN schema
- But the customer table is accessed returns records based on the unique value of the table. In customer table 'CUID' field is the primary key which is always unique and the customer table retrieves distinct row when using this unique column and the 'Table access by index rowid (SRUNOKPN.CUSTOMER)' means the table returns

the result set based on the records returned by 'INDEX UNIQUE SCAN' which is the 'CUID' field

### **Hash Join**

When a large table is joined to a small table, the efficiency in retrieving the might be a big question. To avoid such situations Oracle's hash join is used. When the SQL statement is executed, hash join takes every single row of the first table (small table) and produces a hash value for each row using hash algorithm and stores it in a hash table. It performs the same action for the second (larger table). Then it looks for a matching hash value for the large table with the previously generated hash value of the small table, if it matches then the result is produced. For example, the following SQL statement returns all customers who have received citations. A customer can receive 'n' number of citation. So there exist 'one-to-many' relationships between these tables.

```
select /*+ use_hash(customer)*/ cuname, pc_number
from customer, citation
where cuid = pc_cuid;
```



Operation	Object Name	Rows	Bytes	Cost	Object Node	In/Out	PStart	PStop
SELECT STATEMENT Optimizer Mode=CHOOSE		7 K		24				
HASH JOIN		7 K	378 K	24				
TABLE ACCESS FULL	SRUNOKPN.CUSTOMER	4 K	138 K	7				
TABLE ACCESS FULL	SRUNOKPN.CITATION	7 K	151 K	11				

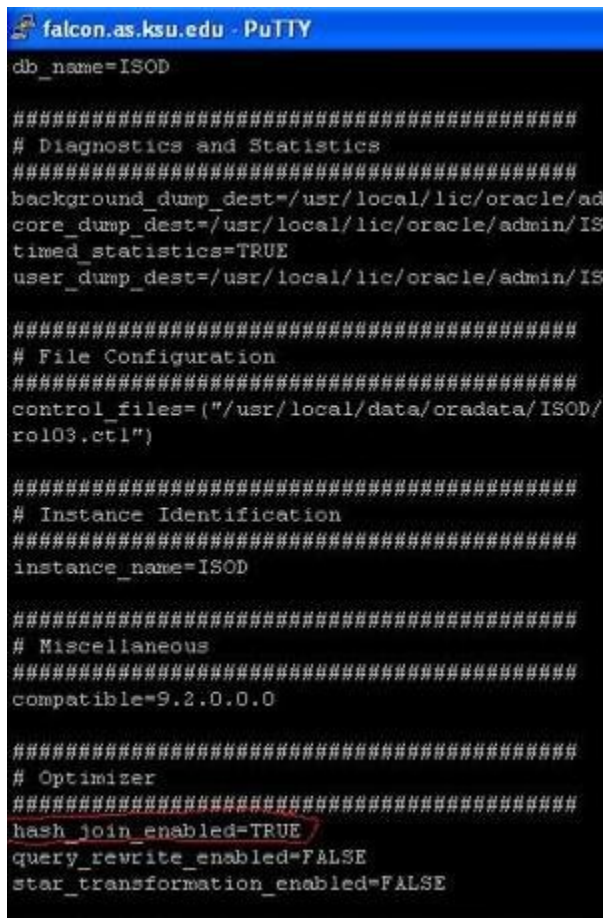
(Fig. 3 Shows the Hash join between two tables)

### **Understanding the explain plan for the above diagram,**

- Table access full (SRUNOKPN.CUSTOMER) means full table scan is performed on customer table.

- Table access full (SRUNOKPN.CITATION) means full table scan is performed on citation table and all the citations matching the customer records in the customer table is returned as the result.
- Hash join specifies that the small table in this query 'customer' table joins the large table i.e. 'citation' table.

The hash join can be used only if the 'hash\_join\_enabled' feature is set *TRUE* in init.ora file. The following figure shows that that property has been set true in my init.ora file.



```

falcon.as.ksu.edu - PuTTY
db_name=ISOD

#####
# Diagnostics and Statistics
#####
background_dump_dest=/usr/local/lic/oracle/ad
core_dump_dest=/usr/local/lic/oracle/admin/IS
timed_statistics=TRUE
user_dump_dest=/usr/local/lic/oracle/admin/IS

#####
# File Configuration
#####
control_files=("/usr/local/data/oradata/ISOD/
rol03.ctl")

#####
# Instance Identification
#####
instance_name=ISOD

#####
# Miscellaneous
#####
compatible=9.2.0.0

#####
# Optimizer
#####
hash_join_enabled=TRUE
query_rewrite_enabled=FALSE
star_transformation_enabled=FALSE

```

## ORDERING OF RECORDS

There are various operations to perform ordering operations, i.e. either by sort or filter

## Sort

The screenshot shows the TOAD for Oracle SQL Editor interface. The main window displays the following SQL query:

```
select cuname, pc_number
from customer, citation
where cuid = pc_cuid
and cuname like '%BEL%'
order by cuid;
```

Below the query editor, the 'Explain Plan' tab is active, showing the execution plan for the query. The plan is as follows:

Operation	Object Name	Rows	Bytes	Cost	Object Node	In/Out	PStart	PStop
SELECT STATEMENT Optimizer Mode=CHOOSE								
SORT ORDER BY								
NESTED LOOPS								
TABLE ACCESS FULL	SRUNOKPN.CITATION							
TABLE ACCESS BY INDEX ROWID	SRUNOKPN.CUSTOMER							
INDEX UNIQUE SCAN	SRUNOKPN.CUID							

(Fig 5 shows that the records returned are ordered based on the sort operation. Sorting operation should not be used for large tables because it affects the performance of a query.)