Report

Subject: OLAP Optimisation queries.

Students: CHARFAOUI Younes, BOURBAI Ismail.

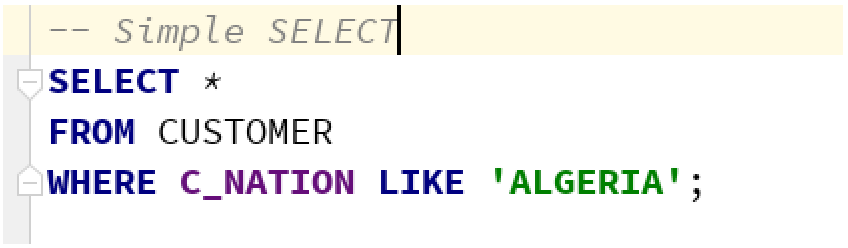
Github Repository: Click Here

Step One:

We've installed the oracle 11g desktop class in our computers, and we have used the SQL*Loader to load our prepared data into the tables

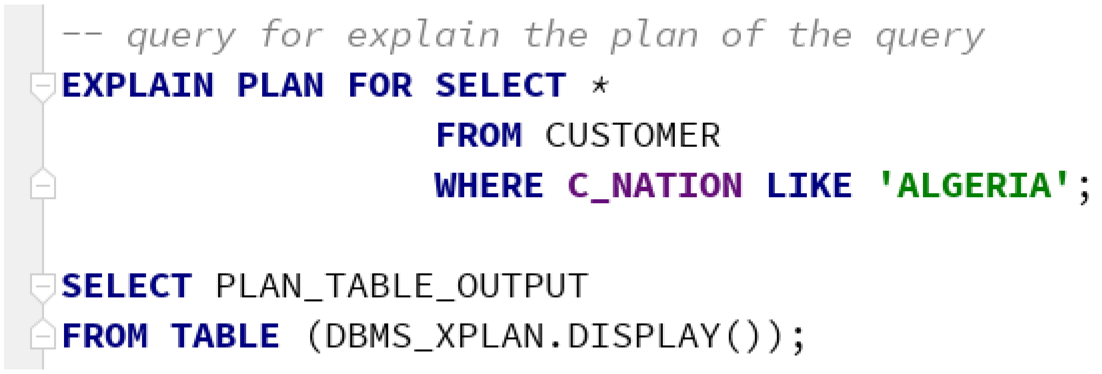
Step Two:

In this step we've test the following query in our DBMS:

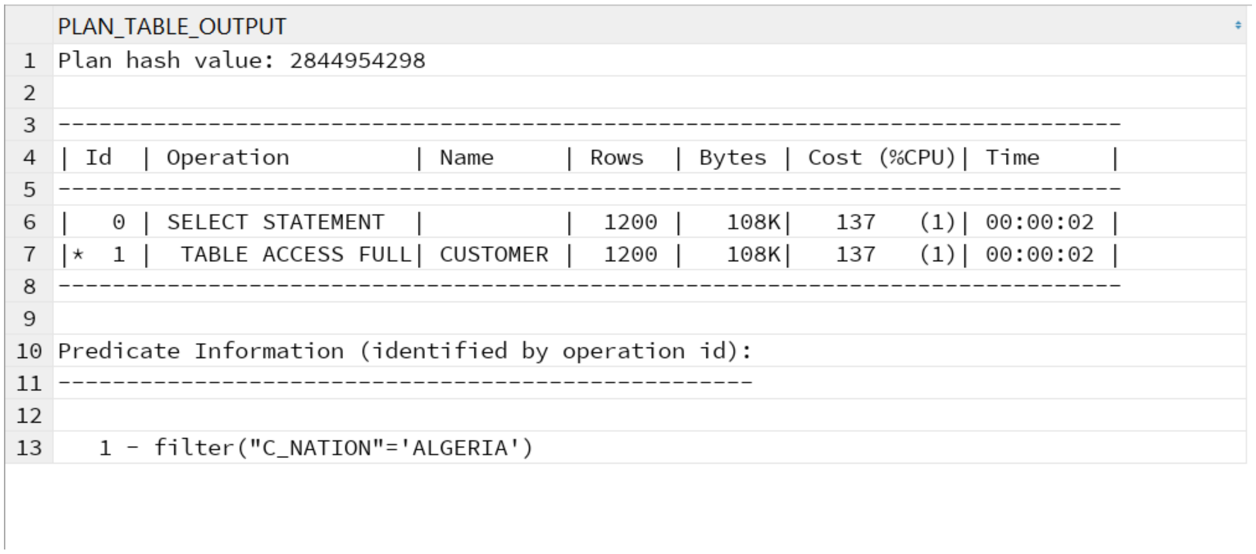


```sql
-- Simple SELECT
SELECT *
FROM CUSTOMER
WHERE C_NATION LIKE 'ALGERIA';
```

The time taken by this query was 406ms. If we show the plan executed by the DBMS we get the following plan:



```sql
-- query for explain the plan of the query
EXPLAIN PLAN FOR SELECT *
                 FROM CUSTOMER
                 WHERE C_NATION LIKE 'ALGERIA';

SELECT PLAN_TABLE_OUTPUT
FROM TABLE (DBMS_XPLAN.DISPLAY());
```



```
PLAN_TABLE_OUTPUT
Plan hash value: 2844954298

------------------------------------------------------------------------------
| Id  | Operation           | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |           |  1200 |   108K|   137   (1)| 00:00:02 |
|*  1 |  TABLE ACCESS FULL| CUSTOMER |  1200 |   108K|   137   (1)| 00:00:02 |
------------------------------------------------------------------------------

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

   1 - filter("C_NATION"='ALGERIA')
```

Step Three:

Now we have to choose the field of which we are going to make in index to help optimize the query result, of course it's going to be the selection criteria in our query which is C_NATION.

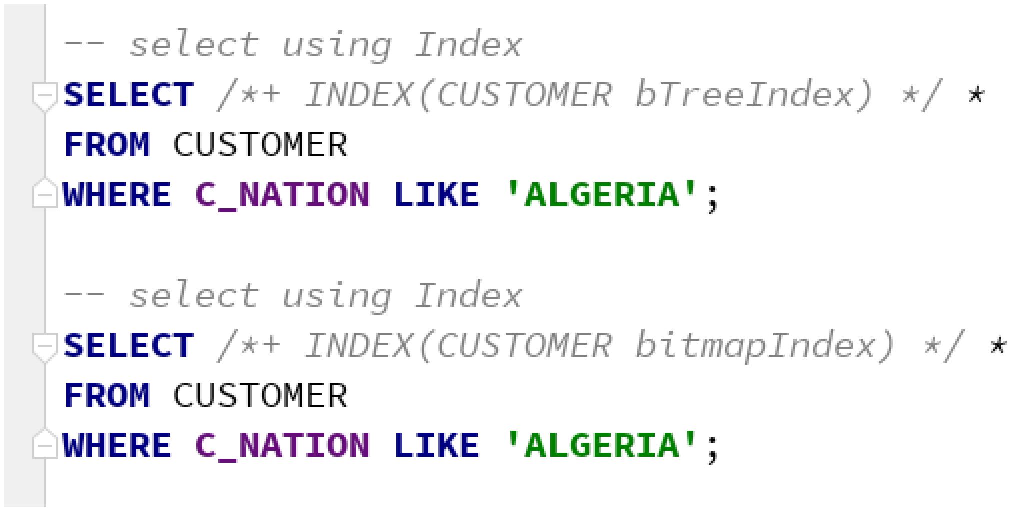
Step Four:

For the index types and for the sack of simplicity we are going to test with Bitmap, B-tree (which used by default). We can declare them as Following:

```sql
-- Simple B-Tree Index Creation.
CREATE INDEX bTreeIndex ON CUSTOMER (C_NATION);

-- Bitmap B-Tree Index Creation.
CREATE BITMAP INDEX bitmapIndex ON CUSTOMER (C_NATION);
```

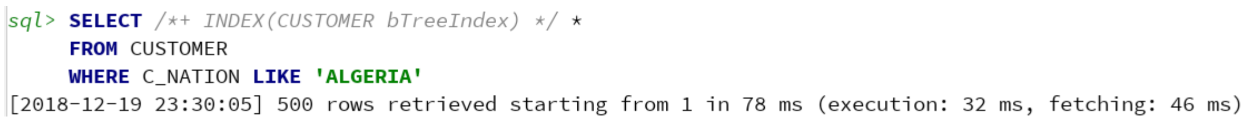
Step Five:

Now we are in the phase of testing the result of adding indexes on the C_NATION columns, we will use both of them and compare the results.



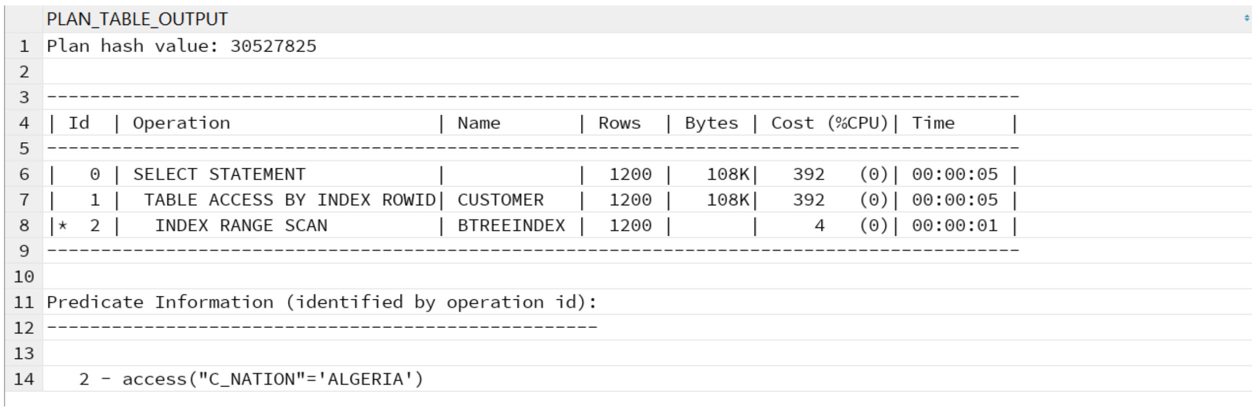
```sql
-- select using Index
SELECT /*+ INDEX(CUSTOMER bTreeIndex) */ *
FROM CUSTOMER
WHERE C_NATION LIKE 'ALGERIA';

-- select using Index
SELECT /*+ INDEX(CUSTOMER bitmapIndex) */ *
FROM CUSTOMER
WHERE C_NATION LIKE 'ALGERIA';
```

The result of using B-Tree Index was 78ms.



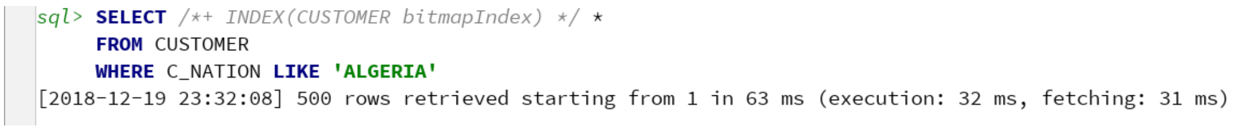
```
sql> SELECT /*+ INDEX(CUSTOMER bTreeIndex) */ *
     FROM CUSTOMER
     WHERE C_NATION LIKE 'ALGERIA'
[2018-12-19 23:30:05] 500 rows retrieved starting from 1 in 78 ms (execution: 32 ms, fetching: 46 ms)
```

Here is the plan of the query execution:



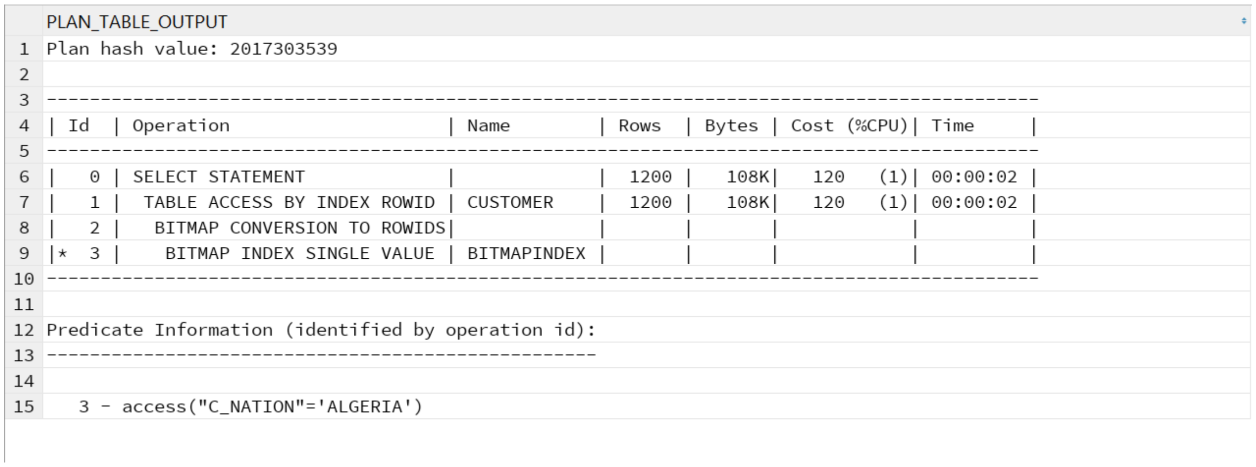
```
PLAN_TABLE_OUTPUT
Plan hash value: 30527825

--------------------------------------------------------------------------------------------
| Id  | Operation                    | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |             |  1200 |  108K|   392   (0)| 00:00:05 |
|   1 |  TABLE ACCESS BY INDEX ROWID| CUSTOMER    |  1200 |  108K|   392   (0)| 00:00:05 |
|*  2 |   INDEX RANGE SCAN           | BTREEINDEX  |  1200 |       |     4   (0)| 00:00:01 |
--------------------------------------------------------------------------------------------

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

   2 - access("C_NATION"='ALGERIA')
```

The result of using Bitmap Index was 63ms.



```
sql> SELECT /*+ INDEX(CUSTOMER bitmapIndex) */ *
     FROM CUSTOMER
     WHERE C_NATION LIKE 'ALGERIA'
[2018-12-19 23:32:08] 500 rows retrieved starting from 1 in 63 ms (execution: 32 ms, fetching: 31 ms)
```

Here is the plan of the query execution:



```
PLAN_TABLE_OUTPUT
Plan hash value: 2017303539

---------------------------------------------------------------------------------------------
| Id  | Operation                    | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |             |  1200 |   108K|   120   (1)| 00:00:02 |
|   1 |  TABLE ACCESS BY INDEX ROWID | CUSTOMER    |  1200 |   108K|   120   (1)| 00:00:02 |
|   2 |   BITMAP CONVERSION TO ROWIDS|             |       |       |            |          |
|*  3 |    BITMAP INDEX SINGLE VALUE | BITMAPINDEX |       |       |            |          |
---------------------------------------------------------------------------------------------

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

   3 - access("C_NATION"='ALGERIA')
```

Yeah it's not a big difference because it isn't that much lot of data we talk about optimization when the volume of the data is really huge.

Step Six:

In This last step, we have tested some hints in the oracle hints page but we did not get that much good result and we thing that because of the data is not that much bigger, we could use more complicated queries but the concept is the same.

Conclusion:

In This Practical word we saw that Optimizing and Tuning DBMS parameter is not that much easier task like creating table and database and making handy foreign keys, at this time comes the Database administrator, hi is specialist in this parameter by testing a lot of combination to come with the best one.