ISSN: 2321-7782 (Online)
e-ISJN: A4372-3114
Impact Factor: 6.047

Volume 5, Issue 6, June 2017

International Journal of Advance Research in Computer Science and Management Studies

Research Article / Survey Paper / Case Study

Available online at: www.ijarcsms.com

Indexes reduces cost in Query Processing

Anand Harsha[1]
Assistant Professor, Aishwarya College of Education
Jai Narain Vyas University
Jodhpur – India

Roochi Harsha[2]
Instructor,
Maulana Abul Kalam Azad Muslim Pvt.ITI
Jodhpur – India

Abstract: ***This paper will help those who are dealing with hundreds of data as an administrator and wants cost effective results. Here cost reducing means that every query takes some time to be processed this time is negligible if there are few records but when we are dealing with thousands of records then this cost of processing effects on the 24X7 service providing agencies and the web services provided on the internet. The results should be on time otherwise user/listener/customer switched to the other service.***

Keywords: Query Processing, Indexing, Cost Reduction.

I. INTRODUCTION

In this paper we are considering how to calculate first the processing cost of a query with thousands of records and then by indexing reduce the cost of processing. In this paper a reader can also know about the indexes and their process to reduce the cost of query processing. The time when database are used everywhere from mobile to web, from school to colleges and from banks to small shopkeepers, applications is invented day by day to help providing services better and on time with cost reduction. This paper will try to help in this sense using Indexing.

II. USED TABLES

SQL> select * from **bill_master**;

BID	BDATE	PID	QTY
1120	11-MAY-15	1111	2
1121	25-MAY-17	1112	15
1122	03-JUL-16	1113	21
1123	25-MAY-17	1113	5
1124	25-MAY-17	1113	45
1125	25-MAY-17	1114	4
1126	25-MAY-17	1119	14

7 rows selected.

SQL> select * from product_master;

PID	PNAME
1111	BIKAJI
1112	SURF
1113	PARLE-G
1114	AMUL GHEE
1115	FINE BESAN
1116	POHA
1117	CHIL SAUSE
1118	CHIPS


```
      1119        PAPAD
      1120        VINEGAR
10 rows selected.

SQL> select * from product_detail;

       PID     PPRICE     PDESC
---------- ---------- --------------------
      1111         95        NAMKEEN BHUJIYA
      1112        150        DETERGENT
      1113         15        BISCUITS
      1114        475        COW GHEE
      1115         45        CHANA POWDER
      1116         55        FOOD PRODUCT
      1117       49.5        TOMATO SAUSE
      1118         50        POTATO CHIPS
      1119        255        NAMKEEN
      1120        435        APPLE SIDER
```

We have plan for selecting data using the three above tables with the help of this complex query. There is an query named explain which will let you know the fired query with certain key fields like number of rows retrieved, What operation is being held, how much bytes would be taken, cost with cpu percentage, and the time taken during query processing. Syntax is as follows:

Explain plan for <your query here>;

Explain plan for select bid, bdate, pname from bill_master, product_master where bill_master.pid=product_master.pid and product_master.pid in(select pid from product_detail where pprice<50) and bill_master.pid in(select pid from bill_master where qty<5);

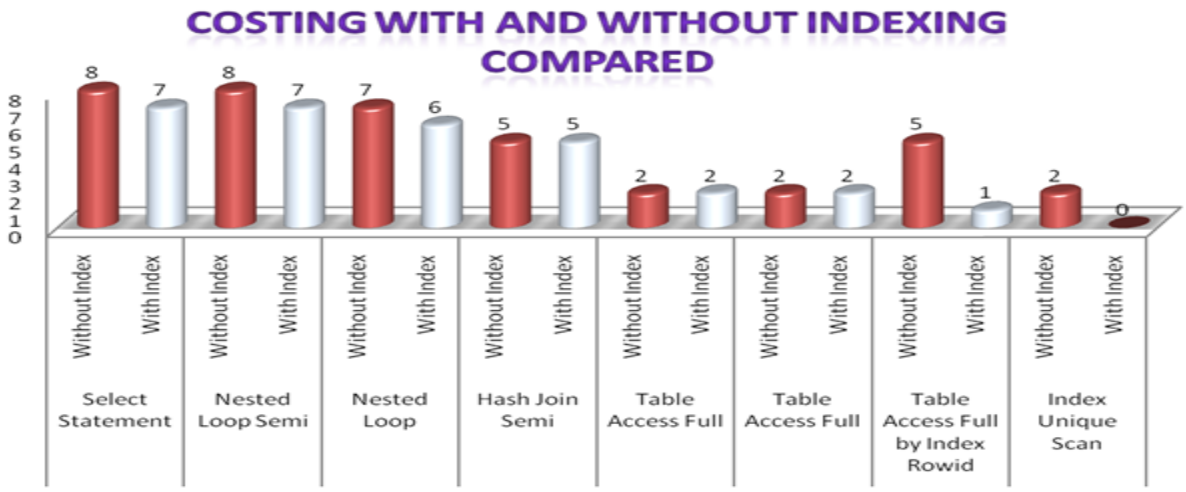
Explained.

Now after this Explanation Completed the result of the query fired will be shown by the below query.

SQL> select * from table(dbms_xplan.display());

Without index the result in cost versus with index cost can be compared in the below table

Select Statement		Nested Loop Semi		Nested Loop		Hash Join Semi		Table Access Full		Table Access Full		Table Access Full by Index Rowid		Index Unique Scan	
Without Index	With Index	Without Index	With Index	Without Index	With Index	Without Index	With Index	Without Index	With Index	Without Index	With Index	Without Index	With Index	Without Index	With Index
8	7	8	7	7	6	5	5	2	2	2	2	5	1	2	0



III. WITHOUT INDEX RESULT

Explain plan for select bid, bdate, pname from bill_master, product_master where bill_master.pid=product_master.pid and product_master.pid in(select pid from product_detail where pprice<50) and bill_master.pid in(select pid from bill_master where qty<5);

SQL> select * from table(dbms_xplan.display());

```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 3518860643
-----------------------------------------------------------------------------------------------
| Id  | Operation                      | Name           | Rows  | Bytes | Cost (%CPU)| Time|
-----------------------------------------------------------------------------------------------
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |                |    1 |    42 |    8 (13)| 00:00:01 |
|   1 | NESTED LOOPS                   |                |    1 |    42 |    8  (13)| 00:00:01 |
|   2 |   MERGE JOIN CARTESIAN         |                |    6 |   186|    7 (15)| 00:00:01 |
|* 3 |    HASH JOIN SEMI              |                |    1 |    23 |    5  (20)| 00:00:01 |
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   4 |     TABLE ACCESS FULL          | BILL_MASTER    |    7 |   112 |    2(0)| 00:00:01   |
|* 5 |     TABLE ACCESS FULL          | BILL_MASTER    |    1 |    7  |    2(0)| 00:00:01   |
|   6 |    BUFFER SORT                 |                |    4 |    32 |  5 (20)| 00:00:01   |
|   7 |     SORT UNIQUE                |                |    4 |    32 |      2(0)| 00:00:01 |
|* 8 |      TABLE ACCESS FULL         | PRODUCT_DETAIL |    4 |    32 |     2(0)| 00:00:01  |
|   9 |   TABLE ACCESS BY INDEX ROWID  | PRODUCT_MASTER |    1 |    11 |     1(0)| 00:00:01 |
|* 10 |    INDEX UNIQUE SCAN           | PIDPK          |    1 |       |     0(0)| 00:00:01 |
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   3 - access("BILL_MASTER"."PID"="PID")
   5 - filter("QTY"<5)
   8 - filter("PPRICE"<50)
  10 - access("PRODUCT_MASTER"."PID"="PID")
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
filter("BILL_MASTER"."PID"="PRODUCT_MASTER"."PID")
```

26 rows selected.

THIS CAN BE EASILY UNDERSTAND BY THIS BELOW TABLE AND WILL SOON COMPARED WITH INDEX RESULT VALUE TABLE.

Id	Operation	Name	Rows	Bytes	Cost(%CPU)	Time
0	SELECT STATEMENT		1	42	8 (13)	00:00:01
1	NESTED LOOPS		1	42	8 (13)	00:00:01
2	MERGE JOIN CARTESIAN		6	186	7 (15)	00:00:01
*3	HASH JOIN SEMI		1	23	5 (20)	00:00:01
4	TABLE ACCESS FULL	BILL_MASTER	7	112	2 (0)	00:00:01
*5	TABLE ACCESS FULL	BILL_MASTER	1	7	2 (0)	00:00:01
6	BUFFER SORT		4	32	5 (20)	00:00:01
7	SORT UNIQUE		4	32	2 (0)	00:00:01
*8	TABLE ACCESS FULL	PRODUCT_DETAIL	4	32	2 (0)	00:00:01
9	TABLE ACESS BY INDEX ROWID	PRODUCT_MASTER	1	11	1 (0)	00:00:01
*10	INDEX UNIQUE SCAN	PIDPK	1		0 (0)	00:00:01

IV. WITH INDEX RESULT

(Bill_master(bid) index created bm, product_detail(pid) index created pd)

SQL> select * from table(dbms_xplan.display());


```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 1741765567
-------------------------------------------------------------------------------------------------
| Id  | Operation                          | Name                | Rows | Bytes | Cost (%CPU)| Time |
-------------------------------------------------------------------------------------------------
 PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |                     |    1 |    42 |     7(15)| 00:00:01 |
|   1 |  NESTED LOOPS SEMI                 |                     |    1 |    42 |     7 (15)| 00:00:01 |
|   2 |   NESTED LOOPS                     |                     |    1 |    34 |     6(17)| 00:00:01 |
|*  3 |    HASH JOIN SEMI                  |                     |    1 |    23 |     5(20)| 00:00:01 |
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   4 |     TABLE ACCESS FULL              | BILL_MASTER         |    7 |   112 |     2 (0)  | 00:00:01 |
|* 5 |     TABLE ACCESS FULL              | BILL_MASTER         |    1 |     7 |     2 (0)  | 00:00:01 |
|   6 |    TABLE ACCESS BY INDEX ROWID     | PRODUCT_MASTER|    1 |    11 |     1 (0)  | 00:00:01 |
|* 7 |     INDEX UNIQUE SCAN              | PIDPK               |    1 |       |     0 (0)  | 00:00:01 |
|* 8 |    TABLE ACCESS BY INDEX ROWID     | PRODUCT_DETAIL |    2 |    16 |     1 (0)  | 00:00:01 |
|* 9 |     INDEX RANGE SCAN               | PD                  |    1 |       |     0 (0)  | 00:00:01 |
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   3 - access("BILL_MASTER"."PID"="PID")
   5 - filter("QTY"<5)
   7 - access("BILL_MASTER"."PID"="PRODUCT_MASTER"."PID")
   8 - filter("PPRICE"<50)
   9 - access("PRODUCT_MASTER"."PID"="PID")

25 rows selected.

SQL>
```

This can be explained in the tabular format

Id	Operation	Name	Rows	Bytes	Cost(%CPU0	Time
0	SELECT STATEMENT		1	42	7 (15)	00:00:01
1	NESTED LOOP SEMI		1	42	7 (15)	00:00:01
2	NESTED LOOPS		1	34	6 (17)	00:00:01
*3	HASH JOIN SEMI		1	23	5 (20)	00:00:01
4	TABLE ACCESS FULL	BILL_MASTER	7	112	2 (0)	00:00:01
*5	TABLE ACCESS FULL	BILL_MASTER	1	7	2 (0)	00:00:01
6	TABLE ACCESS BY INDEX ROWID	PRODUCT_MASTER	1	11	1 (0)	00:00:01
7	INDEX UNIQUE SCAN	PIDPK	1		0 (0)	00:00:01
*8	TABLE ACCESS BY INDEX ROWID	PRODUCT_DETAIL	2	16	1 (0)	00:00:01
9	INDEX RANGE SCAN	PD	1		0 (0)	00:00:01

V. Conclusion

Although this paper is just an analytical work which uses few records and reduces cost with indexing. But whenever and wherever it is applied in organization on thousands of records will save cost in context of CPU uses and the Time. Multiple times the testing is done, Indexes applied and dropped, comparison made using tables and designed in the form of chart so can be easily understandable for the viewers and oracle administrator and programmers.

References

1. http://use-the-index-luke.com/sql/explain-plan/oracle/getting-an-execution-plan.
2. https://docs.oracle.com/cd/B19306_01/server.102/b14211/ex_plan.htm#g42231.

AUTHOR(S) PROFILE

Anand Harsha, received the MCA degree in Computer Science & Applications from Lucky Institute of Professional Studies affiliated with Jai Narain Vyas University, Jodhpur Rajasthan in 2007 and M.A. degree in Political Science from S.B.K Govt. PG. College, Jaisalmer Affiliated with MDSU, Ajmer in 2004. During 2015-2017, he stayed in Aiswarya college of Management and Research Center as an Assistant professor in Department of Computer Science.

Roochi Harsha, received the MSC(CS) degree from Govt. Bangur PG College Pali affiliated with M.D.S.U, Ajmer, Rajasthan in 2010 and BCA degree from sajjan international college pali affiliated with M.D.S.U,Ajmer, Rajasthan, in 2007. During 2012-2017, she stayed in Maulana Abul Kalam Azad Muslim Pvt.ITI as an Instructor in Department of Electric Trade.

