\*\*\* EXPERIMENT NO: 04 \*\*\*

-------------------------------------------------------------------------------

Author : Pradyot Patil

Roll No : 53 [7B]

Date : 08-SEP-2018

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Q1. Create individual b-tree indexes on the following columns of the table: customers\_copy\_btree

1. cust\_gender
2. cust\_year\_of\_birth
3. cust\_last\_name
4. cust\_street\_address

How long does it take to create the indexes?

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**SQL> CREATE INDEX btree\_cust\_gender on customer\_copy(cust\_gender );**

Index created.

**Elapsed: 00:00:00.10**

**SQL> CREATE INDEX btree\_cust\_year\_of\_birth on customer\_copy(cust\_year\_of\_birth);**

Index created.

**Elapsed: 00:00:00.04**

**SQL> CREATE INDEX btree\_cust\_last\_name on customer\_copy(cust\_last\_name);**

Index created.

**Elapsed: 00:00:00.03**

**SQL> CREATE INDEX btree\_cust\_street\_address on customer\_copy(cust\_street\_address);**

Index created.

**Elapsed: 00:00:00.04**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Query 2**: Create bitmap indexes on the above columns. How long does it take to create bitmap indexes? Compare it with the results of btree index creation.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**SQL> CREATE BITMAP INDEX bitmap\_cust\_gender on customer\_copy\_bitmap(cust\_gender );**

Index created.

**Elapsed: 00:00:00.26**

**SQL> CREATE BITMAP INDEX bitmap\_cust\_year\_of\_birth on customer\_copy\_bitmap(cust\_year\_of\_birth);**

Index created.

**Elapsed: 00:00:00.02**

**SQL> CREATE BITMAP INDEX bitmap\_cust\_last\_name on customer\_copy\_bitmap(cust\_last\_name);**

Index created.

**Elapsed: 00:00:00.01**

**SQL> CREATE BITMAP INDEX bitmap\_cust\_street\_address on customer\_copy\_bitmap(cust\_street\_address);**

Index created.

**Elapsed: 00:00:00.57**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Query 3** : Do as directed:

1. Find the size of each segment: customers\_copy\_bitmap and customers\_copy\_btree
2. The b-tree index range for high and low cardinality address index.
3. The bitmap index range for high and low cardinality address index.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**SQL> select segment\_name,bytes/1024/1024 "Size in MB" from user\_segments where segment\_name like 'BTREE%';**

SEGMENT\_NAME Size in MB

------------------ ----------

BTREE\_CUST\_GENDER .875

BTREE\_CUST\_LAST\_NAME 2

BTREE\_CUST\_STREET\_ADDRESS 3

BTREE\_CUST\_YEAR\_OF\_BIRTH 1

**4 rows selected.**

**Elapsed: 00:00:00.02**

**SQL> select segment\_name,bytes/1024/1024 "Size in MB" from user\_segments where segment\_name like 'BITMAP%';**

SEGMENT\_NAME Size in MB

------------------ ---------- BITMAP\_CUST\_GENDER 0.0625

BITMAP\_CUST\_LAST\_NAME 0.125

BITMAP\_CUST\_STREET\_ADDRESS 3

BITMAP\_CUST\_YEAR\_OF\_BIRTH 0.1875

**4 rows selected.**

**Elapsed: 00:00:00.02**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Query 4**: Use year of birth, which had 75 different values in our test data as filter column. Also show the execution plan for both indexes- btree and bitmap. Compare the cost of the execution plan for b-tree and bitmap indexes.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**SQL> select \* from customer\_copy where cust\_year\_of\_birth = 1967;**

**956 rows selected.**

**Elapsed: 00:00:00.04**

Execution Plan

----------------------------------------------------------

Plan hash value: 718019990

-----------------------------------------------------------------------------------

| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |

-----------------------------------------------------------------------------------

| 0 | SELECT STATEMENT | | 956 | 278K| 406 (1)| 00:00:05 |

|\* 1 | TABLE ACCESS FULL| CUSTOMER\_COPY | 956 | 278K| 406 (1)| 00:00:05 |

-----------------------------------------------------------------------------------

Predicate Information (identified by operation id):

---------------------------------------------------

1 - filter("CUST\_YEAR\_OF\_BIRTH"=1967)

Note

-----

- dynamic sampling used for this statement (level=2)

Statistics

----------------------------------------------------------

412 recursive calls

0 db block gets

1637 consistent gets

7 physical reads

0 redo size

152373 bytes sent via SQL\*Net to client

1212 bytes received via SQL\*Net from client

65 SQL\*Net roundtrips to/from client

0 sorts (memory)

0 sorts (disk)

956 rows processed

**SQL> select \* from customer\_copy\_bitmap where cust\_year\_of\_birth = 1967;**

**956 rows selected.**

**Elapsed: 00:00:00.03**

Execution Plan

----------------------------------------------------------

Plan hash value: 399721698

----------------------------------------------------------------------------------------------------------

| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |

----------------------------------------------------------------------------------------------------------

| 0 | SELECT STATEMENT | | 956 | 278K| 130 (0)| 00:00:02 |

| 1 | TABLE ACCESS BY INDEX ROWID | CUSTOMER\_COPY\_BITMAP | 956 | 278K| 130 (0)| 00:00:02 |

| 2 | BITMAP CONVERSION TO ROWIDS| | | | | |

|\* 3 | BITMAP INDEX SINGLE VALUE | BITMAP\_CUST\_YEAR\_OF\_BIRTH | | | | |

----------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):

---------------------------------------------------

3 - access("CUST\_YEAR\_OF\_BIRTH"=1967)

Note

-----

- dynamic sampling used for this statement (level=2)

Statistics

----------------------------------------------------------

412 recursive calls

0 db block gets

928 consistent gets

7 physical reads

0 redo size

192350 bytes sent via SQL\*Net to client

1212 bytes received via SQL\*Net from client

65 SQL\*Net roundtrips to/from client

0 sorts (memory)

0 sorts (disk)

956 rows processed \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Query 5**: Show that update to the bitmap indexed column takes a bit longer than the b-tree indexed column.

1. Create new indexes on cust\_id column of btree and bitmap table.
2. Set the timing on
3. Write a PL/SQL procedure for each table as directed:
4. Create new columns- upd\_cust\_id and cust\_yob\_value with number format.
5. In loop of 500 allot random values to both columns
6. set cust\_year\_of\_birth = cust\_yob\_value and consider cust\_id = upd\_cust\_id

What is the size of the indexes compared to the size as before the updates took place.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**SQL> create INDEX cust\_id\_btree ON customer\_copy(CUST\_ID) ;**

**Index created.**

**Elapsed: 00:00:00.08**

**SQL> create BITMAP INDEX cust\_id\_bitmap ON customer\_copy\_bitmap(CUST\_ID) ;**

**Index created.**

**Elapsed: 00:00:00.11**

**SQL> declare**

**2 upd\_cust\_id number(5);**

**3 cust\_yob\_value number(4);**

**4 begin**

**5 for i in 1 .. 500 loop**

**6 upd\_cust\_id := dbms\_random.value(1,55000);**

**7 cust\_yob\_value := dbms\_random.value(1900,2000);**

**8 update customer\_copy**

**9 set cust\_year\_of\_birth = cust\_yob\_value**

**10 where cust\_id = upd\_cust\_id;**

**11 commit;**

**12 end loop;**

**13 end;**

**14 /**

**PL/SQL procedure successfully completed.**

**Elapsed: 00:00:00.07**

**SQL> declare**

**2 upd\_cust\_id number(5);**

**3 cust\_yob\_value number(4);**

**4 begin**

**5 for i in 1 .. 500 loop**

**6 upd\_cust\_id := dbms\_random.value(1,55000);**

**7 cust\_yob\_value := dbms\_random.value(1900,2000);**

**8 update customer\_copy\_bitmap**

**9 set cust\_year\_of\_birth = cust\_yob\_value**

**10 where cust\_id = upd\_cust\_id;**

**11 commit;**

**12 end loop;**

**13 end;**

**14 /**

**PL/SQL procedure successfully completed.**

**Elapsed: 00:00:00.10**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Query 6**: Q6. Comparison of time for index creation for normal bitmap index and join bitmap index.

Do as directed:

1. Create table customers\_bijx\_test\_bitmap from customers & sales\_bijx\_test\_bitmap as from sales
2. create bitmap index sales\_bijx\_test\_bitmap\_bix1 on sales\_bijx\_test\_bitmap table and cust\_id column, and bitmap index cust\_bijx\_test\_bitmap\_bix1 on customers\_bijx\_test\_bitmap table and cust\_last\_name column.

What is the elapsed time for each index creation?

1. Create table customers\_bijx\_test\_bitjoin from customers and Create table sales\_bijx\_test\_bitjoin from Sales and add constraint of primary key to cust\_id column of cust\_bijx\_test\_bitjoin table.
2. (a) create bitmap index named sales\_bijx\_test\_bitjoin\_bjx1 using sales\_bijx\_test\_bitjoin & customers\_bijx\_test\_bitjoin.cust\_id tables.

(b) create bitmap index named sales\_bijx\_test\_bitjoin\_bjx2 using tables sales\_bijx\_test\_bitjoin and customers\_bijx\_test\_bitjoin.cust\_last\_name

Conclude which index creation takes more time.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**SQL> CREATE TABLE customers\_bijx\_test\_bitmap AS SELECT \* FROM SH.CUSTOMERS**

Table created.

**Elapsed: 00:00:00.12**

**SQL> CREATE TABLE sales\_bijx\_test\_bitmap AS SELECT \* FROM SH.SALES ;**

Table created.

**Elapsed: 00:00:02.62**

**SQL> CREATE BITMAP INDEX sales\_bijx\_test\_bitmap\_bix1 ON sales\_bijx\_test\_bitmap(CUST\_ID) ;**

Index created.

**Elapsed: 00:00:00.22**

**SQL> CREATE BITMAP INDEX cust\_bijx\_test\_bitmap\_bix1 ON customers\_bijx\_test\_bitmap(cust\_last\_name) ;**

Index created.

**Elapsed: 00:00:00.01**

**SQL> CREATE TABLE sales\_bijx\_test\_bitmap\_bitjoin AS SELECT \* FROM SH.SALES ;**

Table created.

**SQL> CREATE TABLE customers\_bijx\_bitmap\_bitjoin AS SELECT \* FROM SH.CUSTOMERS ;**

Table created.

**Elapsed: 00:00:00.08**

**SQL> CREATE TABLE sales\_bijx\_bitmap\_bitjoin AS SELECT \* FROM SH.SALES ;**

Table created.

**SQL> ALTER TABLE customers\_bijx\_bitmap\_bitjoin**

**2 add CONSTRAINT customers\_bijx\_bitmap\_pk primary key(cust\_id) ;**

Table altered.

**SQL> create bitmap index sales\_bijx\_bitjoin\_bjx1 on sales\_bijx\_bitmap\_bitjoin(customers\_bijx\_bitmap\_bitjoin.cust\_id)**

**2 from sales\_bijx\_bitmap\_bitjoin, customers\_bijx\_bitmap\_bitjoin**

**3 where sales\_bijx\_bitmap\_bitjoin.cust\_id = customers\_bijx\_bitmap\_bitjoin.cust\_id;**

Index created.

**Elapsed: 00:00:01.15**

**SQL> create bitmap index sales\_bijx\_bitjoin\_bjx2 on sales\_bijx\_bitmap\_bitjoin(customers\_bijx\_bitmap\_bitjoin.cust\_last\_name)**

**2 from sales\_bijx\_bitmap\_bitjoin, customers\_bijx\_bitmap\_bitjoin**

**3 where sales\_bijx\_bitmap\_bitjoin.cust\_id = customers\_bijx\_bitmap\_bitjoin.cust\_id;**

Index created.

**Elapsed: 00:00:00.68**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Query 7**: **Compressed Index:**

1. Create table Student(StudId, StudName)
2. Add 10 Rows
3. Define Index on StudName(First Name and Last Name)
4. Get the Statistics of Index
5. Now add about 10000 rows that will have same last name
6. Get the Statistics of Index
7. Drop Index
8. Create Compressed Index
9. Get the Statistics of Index
10. Compare statics and give your comments

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**SQL> CREATE TABLE student(**

**2 studid NUMBER(5),**

**3 studname VARCHAR2(25) );**

**Table created.**

**SQL> INSERT INTO student VALUES(1,'Pradyot Patil') ;**

**1 row created.**

**SQL> INSERT INTO student VALUES(2,'Nishant Khanorkar') ;**

**1 row created.**

**SQL> INSERT INTO student VALUES(3,'Prathamesh Deshpande') ;**

**1 row created.**

**SQL> INSERT INTO student VALUES(4,'Purvesh Baghele') ;**

**1 row created.**

**SQL> INSERT INTO student VALUES(5,'Varad Panch') ;**

**1 row created.**

**SQL> INSERT INTO student VALUES(6,'Yash Ghatole') ;**

**1 row created.**

**SQL> INSERT INTO student VALUES(7,'Ayush Khare') ;**

**1 row created.**

**SQL> INSERT INTO student VALUES(8,'Arpit Bhaskarwar') ;**

**1 row created.**

**SQL> INSERT INTO student VALUES(9,'Anshika Choudhary') ;**

**1 row created.**

**SQL> INSERT INTO student VALUES(10,'Rushali Sindurkar') ;**

**1 row created.**

**SQL> CREATE INDEX btree\_name on student(studname);**

Index created.

**SQL> select num\_rows, blevel, leaf\_blocks from user\_indexes**

**2 where index\_name='BTREE\_NAME';**

**NUM\_ROWS BLEVEL LEAF\_BLOCKS**

**---------- ---------- -----------**

**10 0 1**

**1 row selected.**

**SQL> declare**

**2 begin**

**3 for i in 11 .. 10000 loop**

**4 INSERT INTO student VALUES(i,'Pradyot Patil') ;**

**5 commit;**

**6 end loop;**

**7 end;**

**8 /**

**PL/SQL procedure successfully completed.**

**SQL> select num\_rows, blevel, leaf\_blocks from user\_indexes**

**2 where index\_name='BTREE\_NAME';**

**NUM\_ROWS BLEVEL LEAF\_BLOCKS**

**---------- ---------- -----------**

**10 0 1**

**1 row selected.**

**SQL> EXEC DBMS\_STATS.gather\_index\_stats('cs753', 'btree\_name');**

**SQL> select num\_rows, blevel, leaf\_blocks from user\_indexes**

**2 where index\_name='BTREE\_NAME';**

**NUM\_ROWS BLEVEL LEAF\_BLOCKS**

**---------- ---------- -----------**

**10000 1 56**

**1 row selected.**

**SQL> DROP INDEX btree\_name ;**

Index dropped.

**SQL> CREATE INDEX comp\_name on student(studname) COMPRESS;**

**Index created.**

**Elapsed: 00:00:00.09**

**SQL> select num\_rows, blevel, leaf\_blocks from user\_indexes**

**2 where index\_name='COMP\_NAME';**

**NUM\_ROWS BLEVEL LEAF\_BLOCKS**

**---------- ---------- -----------**

**10000 1 16**

**1 row selected.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Query 8**: **Function Based Indexes:**

1. Create function based index on Employee table of HR schema. Function should be on salary attribute based on commission percentage.

Find out list of employees having commission percentage less than 50000.

1. Create function based index on employee name for Upper and lower function.
2. Create user table with attributes (UserId, UserName, Gender)
3. Insert 10000 records in user table
4. Build regular index on Username
5. Build function based index on user name based on Upper function
6. Compare the response time and comment.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**CREATE TABLE tp(**

**user\_id NUMBER(5),**

**user\_name VARCHAR2(25),**

**gender CHAR(1) ) ;**

Table created.

**declare**

**begin**

**for i in 1 .. 10000 loop**

**INSERT INTO tp VALUES(i,'Pradyot Patil','M') ;**

**commit;**

**end loop;**

**end;**

**/**

**CREATE INDEX name\_u on tp(user\_name);**

Index created.

**SELECT COUNT(\*) FROM tp where user\_name='Pradyot Patil' ;**

COUNT(\*)

----------

10000

1 row selected.

Elapsed: 00:00:00.02

Execution Plan

----------------------------------------------------------

Plan hash value: 450564413

---------------------------------------------------------------------------

| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |

---------------------------------------------------------------------------

| 0 | SELECT STATEMENT | | 1 | 14 | 11 (0)| 00:00:01 |

| 1 | SORT AGGREGATE | | 1 | 14 | | |

|\* 2 | TABLE ACCESS FULL| TP | 10000 | 136K| 11 (0)| 00:00:01 |

---------------------------------------------------------------------------

Predicate Information (identified by operation id):

---------------------------------------------------

2 - filter("USER\_NAME"='Pradyot Patil')

Note

-----

- dynamic sampling used for this statement (level=2)

Statistics

----------------------------------------------------------

9 recursive calls

0 db block gets

88 consistent gets

12 physical reads

0 redo size

526 bytes sent via SQL\*Net to client

519 bytes received via SQL\*Net from client

2 SQL\*Net roundtrips to/from client

0 sorts (memory)

0 sorts (disk)

1 rows processed

**CREATE INDEX nameu\_u on tp(UPPER(user\_name));**

Index created.

**SELECT COUNT(\*) FROM tp where UPPER(user\_name)='PRADYOT PATIL' ;**

Elapsed: 00:00:00.02

COUNT(\*)

----------

10000

1 row selected.

Elapsed: 00:00:00.01

Execution Plan

----------------------------------------------------------

Plan hash value: 2531064493

-----------------------------------------------------------------------------

| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |

-----------------------------------------------------------------------------

| 0 | SELECT STATEMENT | | 1 | 14 | 10 (0)| 00:00:01 |

| 1 | SORT AGGREGATE | | 1 | 14 | | |

|\* 2 | INDEX RANGE SCAN| NAMEU\_U | 100 | 1400 | 35 (0)| 00:00:01 |

-----------------------------------------------------------------------------

Predicate Information (identified by operation id):

---------------------------------------------------

2 - access(UPPER("USER\_NAME")='PRADYOT PATIL')

Note

-----

- dynamic sampling used for this statement (level=2)

Statistics

----------------------------------------------------------

28 recursive calls

0 db block gets

78 consistent gets

35 physical reads

0 redo size

526 bytes sent via SQL\*Net to client

519 bytes received via SQL\*Net from client

2 SQL\*Net roundtrips to/from client

0 sorts (memory)

0 sorts (disk)

1 rows processed

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Query 9**: **Index Organized Table**

1. Create an IOT look\_ups with the attributes (lookup\_code, lookup\_value, lookup\_description) in tablespace ts\_lookup.

Constraint: lookup\_code should be primary key

PctThreshold is 20 and and lookup\_description should be in overflow area.

Overflow should be in ts\_overflow tablespace.

1. Create a Index Organized Table(IOT) emp\_iot based on hr.employees
2. Create a Index Organized Table(IOT) emp101\_emp based on hr.employees. Place the column hiredate in overflow area.
3. Compare the timings of executing select all from employees,emp\_iot, and emp101\_iot. Comment on your observations.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**CREATE TABLESPACE ts\_lookup DATAFILE '/temp/tse\_tbs.dbf' SIZE 10M;**

**CREATE TABLESPACE ts\_overflow DATAFILE '/temp/tsf\_tbs.dbf' SIZE 10M;**

**CREATE TABLE look**

**(lookup\_code NUMBER(5) NOT NULL,**

**lookup\_value NUMBER(5) NOT NULL,**

**description VARCHAR2(50) NOT NULL,**

**CONSTRAINT pk\_locations PRIMARY KEY (lookup\_code)**

**)**

**ORGANIZATION INDEX**

**TABLESPACE ts\_lookup**

**PCTTHRESHOLD 20**

**INCLUDING lookup\_value**

**OVERFLOW TABLESPACE ts\_overflow;**

Table created.

**CREATE TABLE emp**

**(emp\_id NUMBER(5) ,**

**f\_name VARCHAR2(20) ,**

**l\_name VARCHAR2(25) NOT NULL,**

**hire\_date DATE,**

**CONSTRAINT pk\_emp PRIMARY KEY (emp\_id)**

**)**

**ORGANIZATION INDEX**

**TABLESPACE ts\_lookup**

**PCTTHRESHOLD 20**

**INCLUDING l\_name**

**OVERFLOW TABLESPACE ts\_overflow;**

Table created.

**INSERT INTO emp (emp\_id,f\_name,l\_name,hire\_date)**

**SELECT EMPLOYEE\_ID,FIRST\_NAME,LAST\_NAME,HIRE\_DATE**

**FROM EMPLOYEES**

**WHERE 1=1;**

107 rows created.

**DROP TABLE emp;**

**DROP TABLE emp2 ;**

**CREATE TABLE EMP\_IOT**

**(employee\_id primary key,first\_name,last\_name,email,phone\_number,hire\_date,job\_id,salary,commission\_pct,manager\_id,department\_id) ORGANIZATION INDEX AS (SELECT \* FROM HR.EMPLOYEES);**

Table created.

**CREATE TABLE EMP\_IOT\_overflow**

**(employee\_id primary key,first\_name,last\_name,email,phone\_number,hire\_date,job\_id,salary,commission\_pct,manager\_id,department\_id) ORGANIZATION INDEX including phone\_number overflow AS (SELECT \* FROM HR.EMPLOYEES);**

Table created.

**SELECT \* FROM EMP\_IOT\_OVERFLOW ;**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Query 9**: **Clusters**

1. Create a cluster PERSONNEL containing copy\_emp and copy\_dept tables. Cluster key is deptno.
2. create a index on cluster PERSONNEL.
3. Populate copy\_emp and copy\_dept with data from emp and dept tables of scott respectively.
4. Drop cluster PERSONNEL.

Create a hash cluster named hash\_emp containing table copy101\_emp. Create 10 hashkeys and use the hash function (empno mod 100).

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**CREATE CLUSTER personnel (deptno NUMBER(2)) ;**

Cluster created.

**CREATE INDEX idx\_personnel on CLUSTER PERSONNEL ;**

Index Created.

**CREATE TABLE COPY\_DEPT(deptno PRIMARY KEY,DNAME,LOC) CLUSTER PERSONNEL(deptno) AS (SELECT \* FROM SCOTT.DEPT) ;**

Table created.

**CREATE TABLE COPY\_EMP(EMPNO PRIMARY KEY,deptno,ENAME) CLUSTER PERSONNEL(deptno) AS (SELECT EMPNO,DEPTNO,ENAME FROM SCOTT.EMP) ;**

Table created.

**DROP CLUSTER personnel;**

Cluster dropped.

**CREATE CLUSTER hash\_emp (empno NUMBER(4))**

**TABLESPACE users**

**STORAGE (INITIAL 250K NEXT 50K**

**MINEXTENTS 1 MAXEXTENTS 3**

**PCTINCREASE 0)**

**HASHKEYS 10**

**HASH IS MOD(empno, 100);**

Cluster created.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*