**PRACTICAL 04**

**QUERY 01**

SQL> CREATE TABLE CUSTOMERS\_COPY\_BTREE AS(SELECT \* FROM SH.CUSTOMERS);

Table created.

Elapsed: 00:00:00.50

SQL> CREATE TABLE CUSTOMERS\_COPY\_BITMAP AS(SELECT \* FROM SH.CUSTOMERS);

Table created.

Elapsed: 00:00:00.12

SQL> CREATE INDEX INDEX\_BTREE ON CUSTOMERS\_COPY\_BTREE(CUST\_GENDER);

Index created.

Elapsed: 00:00:00.10

SQL> CREATE INDEX INDEX\_BTREE1 ON CUSTOMERS\_COPY\_BTREE(CUST\_YEAR\_OF\_BIRTH);

Index created.

Elapsed: 00:00:00.05

SQL> CREATE INDEX INDEX\_BTREE2 ON CUSTOMERS\_COPY\_BTREE(CUST\_LAST\_NAME);

Index created.

Elapsed: 00:00:00.09

SQL> CREATE INDEX INDEX\_BTREE3 ON CUSTOMERS\_COPY\_BTREE(CUST\_STREET\_ADDRESS);

Index created.

Elapsed: 00:00:00.08

**QUERY 02**

SQL> CREATE BITMAP INDEX INDEX\_BITMAP ON CUSTOMERS\_COPY\_BITMAP(CUST\_GENDER);

Index created.

Elapsed: 00:00:00.05

SQL> CREATE BITMAP INDEX INDEX\_BITMA1P ON CUSTOMERS\_COPY\_BITMAP(CUST\_YEAR\_OF\_BIRTH);

Index created.

Elapsed: 00:00:00.04

SQL> CREATE BITMAP INDEX INDEX\_BITMAP2 ON CUSTOMERS\_COPY\_BITMAP(CUST\_LAST\_NAME);

Index created.

Elapsed: 00:00:00.04

SQL> CREATE BITMAP INDEX INDEX\_BITMAP3 ON CUSTOMERS\_COPY\_BITMAP(CUST\_STREET\_ADDRESS);

Index created.

Elapsed: 00:00:00.13

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Index\Columns | Gender | Year of Birth | Last Name | Street Address |
| B-Tree Index | 00:00:00:10 | 00:00:00:05 | 00:00:00:09 | 00:00:00:08 |
| Bitmap Index | 00:00:00:05 | 00:00:00:04 | 00:00:00:04 | 00:00:00:13 |

**QUERY 03**

SQL> SELECT SEGMENT\_NAME, BYTES/1024/1024 "SIZE IN MB" FROM USER\_SEGMENTS WHERE SEGMENT\_NAME ='CUSTOMERS\_COPY\_BTREE';

SEGMENT\_NAME SIZE IN MB

----------

CUSTOMERS\_COPY\_BTREE 12

Elapsed: 00:00:00.04

SQL> SELECT SEGMENT\_NAME, BYTES/1024/1024 "SIZE IN MB" FROM USER\_SEGMENTS WHERE SEGMENT\_NAME='CUSTOMERS\_COPY\_BITMAP';

SEGMENT\_NAME SIZE IN MB

----------

CUSTOMERS\_COPY\_BITMAP 12

SQL> SELECT SEGMENT\_NAME, BYTES/1024/1024 "SIZE IN MB" FROM USER\_SEGMENTS WHERE SEGMENT\_NAME ='INDEX\_BTREE';

SEGMENT\_NAME SIZE IN MB

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

INDEX\_BTREE .875

Elapsed: 00:00:00.03

SQL> SELECT SEGMENT\_NAME, BYTES/1024/1024 "SIZE IN MB" FROM USER\_SEGMENTS WHERE SEGMENT\_NAME='INDEX\_BTREE1';

SEGMENT\_NAME SIZE IN MB

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

INDEX\_BTREE1 1

Elapsed: 00:00:00.03

SQL> SELECT SEGMENT\_NAME, BYTES/1024/1024 "SIZE IN MB" FROM USER\_SEGMENTS WHERE SEGMENT\_NAME='INDEX\_BTREE2';

SEGMENT\_NAME SIZE IN MB

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

INDEX\_BTREE2 2

Elapsed: 00:00:00.04

SQL> SELECT SEGMENT\_NAME, BYTES/1024/1024 "SIZE IN MB" FROM USER\_SEGMENTS WHERE SEGMENT\_NAME='INDEX\_BTREE3';

SEGMENT\_NAME SIZE IN M

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

INDEX\_BTREE3 3

Elapsed: 00:00:00.04

SQL> SELECT SEGMENT\_NAME, BYTES/1024/1024 "SIZE IN MB" FROM USER\_SEGMENTS WHERE SEGMENT\_NAME='INDEX\_BITMAP';

SEGMENT\_NAME SIZE IN MB

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

INDEX\_BITMAP .0625

SQL> SELECT SEGMENT\_NAME, BYTES/1024/1024 "SIZE IN MB" FROM USER\_SEGMENTS WHERE SEGMENT\_NAME='INDEX\_BITMA1P';

SEGMENT\_NAME SIZE IN MB

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

INDEX\_BITMA1P .1875

Elapsed: 00:00:00.02

SQL> SELECT SEGMENT\_NAME, BYTES/1024/1024 "SIZE IN MB" FROM USER\_SEGMENTS WHERE SEGMENT\_NAME='INDEX\_BITMAP2';

SEGMENT\_NAME SIZE IN MB

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

INDEX\_BITMAP2 .125

Elapsed: 00:00:00.03

SQL> SELECT SEGMENT\_NAME, BYTES/1024/1024 "SIZE IN MB" FROM USER\_SEGMENTS WHERE SEGMENT\_NAME='INDEX\_BITMAP3';

SEGMENT\_NAME SIZE IN MB

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

INDEX\_BITMAP3 3

Elapsed: 00:00:00.05

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Index\Columns | Gender | Year of Birth | Last Name | Street Address |
| B-Tree Index | 0.875 | 1 | 2 | 3 |
| Bitmap Index | 0.0625 | 0.1875 | 0.125 | 3 |

**QUERY 04**

SELECT \* FROM CUSTOMERS\_COPY\_BTREE WHERE CUST\_YEAR\_OF\_BIRTH=1967;

956 rows selected.

Elapsed: 00:00:00.91

Execution Plan

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

Plan hash value: 2533345702

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

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

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

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

|\* 1 | TABLE ACCESS FULL| CUSTOMERS\_COPY\_BTREE | 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

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

0 recursive calls

0 db block gets

1519 consistent gets

0 physical reads

0 redo size

152373 bytes sent via SQL\*Net to client

1213 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 CUSTOMERS\_COPY\_BITMAP WHERE CUST\_YEAR\_OF\_BIRTH=1967;

956 rows selected.

Elapsed: 00:00:01.68

Execution Plan

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

Plan hash value: 888727555

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

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

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

| 0 | SELECT STATEMENT | | 956 | 278K| 136 (1)| 00:00:02 |

| 1 | TABLE ACCESS BY INDEX ROWID | CUSTOMERS\_COPY\_BITMAP | 956 | 278K| 136 (1)| 00:00:02 |

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

|\* 3 | BITMAP INDEX SINGLE VALUE | INDEX\_BITMA1P | | | | |

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

Predicate Information (identified by operation id):

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

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

Note

-----

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

Statistics

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

0 recursive calls

0 db block gets

805 consistent gets

0 physical reads

0 redo size

192350 bytes sent via SQL\*Net to client

1213 bytes received via SQL\*Net from client

65 SQL\*Net roundtrips to/from client

0 sorts (memory)

0 sorts (disk)

956 rows processed

**Analysis:** The Cost of execution plan of B-tree index is more than the cost of execution plan of bitmap index.

**QUERY 05**

SQL> CREATE INDEX INDEX\_BTCUSTID ON CUSTOMERS\_COPY\_BTREE(CUST\_ID);

Index created.

Elapsed: 00:00:00.06

SQL> CREATE INDEX INDEX\_BICUSTID1 ON CUSTOMERS\_COPY\_BITMAP(CUST\_ID);

Index created.

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 CUSTOMERS\_COPY\_BTREE SET CUST\_YEAR\_OF\_BIRTH =

CUST\_YOB\_VALUE WHERE CUST\_ID = UPD\_CUST\_ID;

9 COMMIT;

10 END LOOP;

11 END;

12 /

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 CUSTOMERS\_COPY\_BITMAP SET CUST\_YEAR\_OF\_BIRTH = CUST\_YOB\_VALUE WHERE CUST\_ID = UPD\_CUST\_ID;

9 COMMIT;

10 END LOOP;

11 END;

12 /

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.17

SQL> SELECT SEGMENT\_NAME,BYTES/1024/1024 "SIZE IN MB" FROM USER\_SEGMENTS WHERE SEGMENT\_NAME='INDEX\_BTCUSTID';

SEGMENT\_NAME SIZE IN MB

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

INDEX\_BTCUSTID 12

Elapsed: 00:00:00.03

**Analysis:** Updating columns cause no effects on size of segments, both indexes requires same space as before update.

**QUERY 06**

SQL> CREATE TABLE CUSTOMERS\_BIJX\_TEST\_BITMAP AS (SELECT \* FROM SH.CUSTOMERS);

Table created.

SQL> CREATE TABLE SALES\_BIJX\_TEST\_BITMAP AS (SELECT \* FROM SH.SALES);

Table created.

SQL> SET TIMING ON;

SQL> CREATE BITMAP INDEX SALES\_BIJX\_TEST\_BITMAP\_BIX1 ON SALES\_BIJX\_TEST\_BITMAP(CUST\_ID);

Index created.

Elapsed: 00:00:00.28

SQL> CREATE BITMAP INDEX CUST\_BIJX\_TEST\_BITMAP\_BIX1 ON CUSTOMERS\_BIJX\_TEST\_BITMAP(CUST\_LAST\_NAME);

Index created.

Elapsed: 00:00:00.04

SQL> CREATE TABLE CUSTOMERS\_BIJX\_TEST\_BITJOIN AS (SELECT \* FROM SH.CUSTOMERS);

Table created.

Elapsed: 00:00:00.10

SQL> ALTER TABLE CUSTOMERS\_BIJX\_TEST\_BITJOIN ADD CONSTRAINTS PRIMARY\_KEY\_ID PRIMARY KEY(CUST\_ID);

Table altered.

Elapsed: 00:00:00.15

SQL> CREATE TABLE SALES\_BIJX\_TEST\_BITJOIN AS (SELECT \* FROM SH.SALES);

Table created.

Elapsed: 00:00:00.29

SQL> CREATE BITMAP INDEX sales\_bijx\_test\_bitjoin\_bjx1 ON

2 SALES\_BIJX\_TEST\_BITJOIN(CUSTOMERS\_BIJX\_TEST\_BITJOIN.CUST\_ID)

3 FROM SALES\_BIJX\_TEST\_BITJOIN,CUSTOMERS\_BIJX\_TEST\_BITJOIN

4 WHERE SALES\_BIJX\_TEST\_BITJOIN.CUST\_ID = CUSTOMERS\_BIJX\_TEST\_BITJOIN.CU

ST\_ID;

Index created.

Elapsed: 00:00:01.01

SQL>

SQL> CREATE BITMAP INDEX sales\_bijx\_test\_bitjoin\_bjx2 ON

2 SALES\_BIJX\_TEST\_BITJOIN(CUSTOMERS\_BIJX\_TEST\_BITJOIN.CUST\_LAST\_NAME)

3 FROM SALES\_BIJX\_TEST\_BITJOIN,CUSTOMERS\_BIJX\_TEST\_BITJOIN

4 WHERE SALES\_BIJX\_TEST\_BITJOIN.CUST\_ID = CUSTOMERS\_BIJX\_TEST\_BITJOIN.CU

ST\_ID;

Index created.

Elapsed: 00:00:01.55

SQL>

**QUERY 07**

SQL> CREATE TABLE STUDENT(

2 STUDID NUMBER(7) PRIMARY KEY,

3 STUDNAME VARCHAR2(30));

Table created.

SQL> INSERT INTO STUDENT VALUES(1,'KIRAN PIMPALKAR');

1 row created.

SQL> INSERT INTO STUDENT VALUES(2,'ABC XYZ');

1 row created.

SQL> INSERT INTO STUDENT VALUES(3,'GHJ LKJ');

1 row created.

SQL> INSERT INTO STUDENT VALUES(4,'WER ASD');

1 row created.

SQL> INSERT INTO STUDENT VALUES(5,'JULIAN CLARA');

1 row created.

SQL> INSERT INTO STUDENT VALUES(6,'ABID MALIK');

1 row created.

SQL> INSERT INTO STUDENT VALUES(7,'JIA CRACKER');

1 row created.

SQL> INSERT INTO STUDENT VALUES(8,'KIM TAN');

1 row created.

SQL> CREATE INDEX STUD\_STUDNAME\_IDX ON STUDENT(STUDNAME);

Index created.

Elapsed: 00:00:00.01

SQL> SELECT COMPRESSION,LEAF\_BLOCKS,

2 ROUND(NUM\_ROWS/DECODE(LEAF\_BLOCKS,0,1,LEAF\_BLOCKS)) "ROWS PER BLOCK", DISTINCT\_KEYS,

3 NUM\_ROWS,NUM\_ROWS-DISTINCT\_KEYS DUP\_ROWS

4 FROM USER\_INDEXES

5 WHERE

6 INDEX\_NAME = 'STUD\_STUDNAME\_IDX';

COMPRESS LEAF\_BLOCKS ROWS PER BLOCK DISTINCT\_KEYS NUM\_ROWS DUP\_ROWS

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

DISABLED 1 8 8 8 0

Elapsed: 00:00:00.02

SQL> DECLARE V\_A NUMBER;

2 BEGIN

3 V\_A := 9;

4 WHILE V\_A < 10000

5 LOOP INSERT INTO STUDENT VALUES(V\_A,'SMITH');

6 V\_A := V\_A + 1;

7 END LOOP;

8 END;

9 /

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.51

SQL> EXEC DBMS\_STATS.gather\_table\_stats('SS','STUDENT');

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.10

SQL> SELECT COMPRESSION,LEAF\_BLOCKS,

2 ROUND(NUM\_ROWS/DECODE(LEAF\_BLOCKS,0,1,LEAF\_BLOCKS)) "ROWS PER BLOCK", DISTINCT\_KEYS,

3 NUM\_ROWS,NUM\_ROWS-DISTINCT\_KEYS DUP\_ROWS

4 FROM USER\_INDEXES

5 WHERE

6 INDEX\_NAME = 'STUD\_STUDNAME\_IDX';

COMPRESS LEAF\_BLOCKS ROWS PER BLOCK DISTINCT\_KEYS NUM\_ROWS DUP\_ROWS

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

DISABLED 41 244 9 9999 9990

Elapsed: 00:00:00.01

SQL> DROP INDEX STUD\_STUDNAME\_IDX;

Index dropped.

Elapsed: 00:00:00.21

SQL> CREATE INDEX STUD\_STUDNAME\_IDX ON STUDENT(STUDNAME)COMPRESS;

Index created.

Elapsed: 00:00:00.06

SQL> EXEC DBMS\_STATS.gather\_table\_stats('SS','STUDENT');

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.15

SQL> SELECT COMPRESSION,LEAF\_BLOCKS,

2 ROUND(NUM\_ROWS/DECODE(LEAF\_BLOCKS,0,1,LEAF\_BLOCKS)) "ROWS PER BLOCK", DISTINCT\_KEYS,

3 NUM\_ROWS,NUM\_ROWS-DISTINCT\_KEYS DUP\_ROWS

4 FROM USER\_INDEXES

5 WHERE

6 INDEX\_NAME = 'STUD\_STUDNAME\_IDX';

COMPRESS LEAF\_BLOCKS ROWS PER BLOCK DISTINCT\_KEYS NUM\_ROWS DUP\_ROWS

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

ENABLED 16 625 9 9999 9990

Elapsed: 00:00:00.00

**QUERY 08**

CREATE INDEX EMP\_TOTAL\_SAL\_IDX ON HR.EMPLOYEES (12 \* salary \* commission\_pct, salary, commission\_pct);

Index created.

SQL> SELECT EMPLOYEE\_ID, LAST\_NAME, FIRST\_NAME, 12\*SALARY\*COMMISSION\_PCT AS "ANNUAL SAL" FROM EMPLOYEES WHERE (12 \* SALARY \* COMMISSION\_PCT) < 50000;

EMPLOYEE\_ID LAST\_NAME FIRST\_NAME ANNUAL SAL

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

173 Kumar Sundita 7320

167 Banda Amit 7440

179 Johnson Charles 7440

166 Ande Sundar 7680

165 Lee David 8160

164 Marvins Mattea 8640

155 Tuvault Oliver 12600

178 Grant Kimberely 12600

172 Bates Elizabeth 13140

171 Smith William 13320

163 Greene Danielle 17100

154 Cambrault Nanette 18000

153 Olsen Christopher 19200

177 Livingston Jack 20160

176 Taylor Jonathon 20640

161 Sewall Sarath 21000

170 Fox Tayler 23040

169 Bloom Harrison 24000

149 Zlotkey Eleni 25200

175 Hutton Alyssa 26400

160 Doran Louise 27000

152 Hall Peter 27000

151 Bernstein David 28500

159 Smith Lindsey 28800

162 Vishney Clara 31500

168 Ozer Lisa 34500

150 Tucker Peter 36000

158 McEwen Allan 37800

148 Cambrault Gerald 39600

174 Abel Ellen 39600

157 Sully Patrick 39900

156 King Janette 42000

147 Errazuriz Alberto 43200

146 Partners Karen 48600

34 rows selected.

SQL> CREATE INDEX EMP\_FNAME\_UPPERCASE\_IDX ON EMPLOYEES ( UPPER(FIRST\_NAME) );

Index created.

Elapsed: 00:00:00.19

SQL> SELECT \* FROM EMPLOYEES WHERE UPPER(FIRST\_NAME) = 'ALLAN';

EMPLOYEE\_ID FIRST\_NAME LAST\_NAME EMAIL PHONE\_NUMBER HIRE\_DATE JOB\_ID SALARY COMMISSION\_PCT MANAGER\_ID DEPARTMENT\_ID

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

158 Allan McEwen AMCEWEN 011.44.1345.829268 01-AUG-04 SA\_REP 9000 .35 146 80

Elapsed: 00:00:00.00

Execution Plan

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

Plan hash value: 3842569606

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

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

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

| 0 | SELECT STATEMENT | | 1 | 69 | 2 (0)| 00:00:01 |

| 1 | TABLE ACCESS BY INDEX ROWID| EMPLOYEES | 1 | 69 | 2 (0)| 00:00:01 |

|\* 2 | INDEX RANGE SCAN | EMP\_FNAME\_UPPERCASE\_IDX | 1 | | 1 (0)| 00:00:01 |

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

Predicate Information (identified by operation id):

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

2 - access(UPPER("FIRST\_NAME")='ALLAN')

SQL> CREATE INDEX EMP\_FNAME\_LOWERCASE\_IDX

2 ON EMPLOYEES ( LOWER(FIRST\_NAME) );

Index created.

Elapsed: 00:00:00.03

SQL>

SQL> SELECT \* FROM EMPLOYEES WHERE LOWER(FIRST\_NAME) = 'allan';

EMPLOYEE\_ID FIRST\_NAME LAST\_NAME EMAIL PHONE\_NUMBER HIRE\_DATE JOB\_ID SALARY COMMISSION\_PCT MANAGER\_ID DEPARTMENT\_ID

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

158 Allan McEwen AMCEWEN 011.44.1345.829268 01-AUG-04 SA\_REP 9000 .35 146 80

Elapsed: 00:00:00.00

Execution Plan

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

Plan hash value: 671893953

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

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

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

| 0 | SELECT STATEMENT | | 1 | 69 | 2 (0)| 00:00:01 |

| 1 | TABLE ACCESS BY INDEX ROWID| EMPLOYEES | 1 | 69 | 2 (0)| 00:00:01 |

|\* 2 | INDEX RANGE SCAN | EMP\_FNAME\_LOWERCASE\_IDX | 1 | | 1 (0)| 00:00:01 |

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

Predicate Information (identified by operation id):

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

2 - access(LOWER("FIRST\_NAME")='allan')

SQL> CREATE TABLE USER\_DATA (

2 id NUMBER(10) NOT NULL,

3 username VARCHAR2(40) NOT NULL,

4 gender VARCHAR2(1)

5 );

Table created.

Elapsed: 00:00:00.27

SQL> BEGIN

2 FOR cur\_rec IN 1 .. 10000 LOOP

3 IF MOD(cur\_rec, 2) = 0 THEN

4 INSERT INTO user\_data

5 VALUES (cur\_rec, 'Jeremy Gilbert', 'F');

6 ELSE

7 INSERT INTO user\_data

8 VALUES (cur\_rec, 'Bonnie Bennette', 'F');

9 END IF;

10 COMMIT;

11 END LOOP;

12 END;

13 /

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.75

SQL> SELECT COUNT(\*) FROM USER\_DATA;

COUNT(\*)

----------

10000

Elapsed: 00:00:00.00

SQL> CREATE INDEX USERDATA\_NAME\_BTREE\_IDX ON USER\_DATA(username);

Index created.

Elapsed: 00:00:00.03

SQL> CREATE INDEX USERDATA\_NAME\_FBI\_IDX

2 ON USER\_DATA ( UPPER(username) );

Index created.

Elapsed: 00:00:00.01

Analysis:Function Based Index, creates a matching index that exactly matches the predicates within the SQL *where* clause. This ensures that the query is retrieved with a minimal amount of disk I/O and the fastest possible speed.

**QUERY 09**

SQL> CREATE TABLESPACE TS\_LOOKUP DATAFILE '/TEMP/TS\_LOOKUP.DBF' SIZE 10M;

Tablespace created.

SQL> CREATE TABLE IOT\_LOOKUPS(

2 LOOKUP\_CODE NUMBER(3),

3 LOOKUP\_VALUE NUMBER(3),

4 LOOKUP\_DESCR VARCHAR2(40),

5 CONSTRAINT IOT\_PK PRIMARY KEY(LOOKUP\_CODE))

6 ORGANIZATION INDEX

7 TABLESPACE TS\_LOOKUP

8 PCTTHRESHOLD 20

9 INCLUDING LOOKUP\_VALUE

10 OVERFLOW TABLESPACE USERS;

Table created.

Elapsed: 00:00:00.50

SQL> SELECT table\_name, iot\_type, iot\_name FROM user\_tables where table\_name='IOT\_LOOKUPS';

TABLE\_NAME IOT\_TYPE IOT\_NAME

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

IOT\_LOOKUPS IOT

Elapsed: 00:00:00.11

SQL> CREATE TABLE emp\_iot(

2 employee\_id NUMBER(6),

3 first\_name VARCHAR2(20),

4 last\_name VARCHAR2(25)CONSTRAINT emp\_last\_name\_nn NOT NULl,

5 email VARCHAR2(25)CONSTRAINT emp\_email\_nn NOT NULL,

6 phone\_number VARCHAR2(20),

7 hire\_date DATE CONSTRAINT emp\_hire\_date\_nn NOT NULL,

8 job\_id VARCHAR2(10)CONSTRAINT emp\_job\_nn NOT NULL,

9 salary NUMBER(8,2),

10 commission\_pct NUMBER(2,2),

11 manager\_id NUMBER(6),

12 department\_id NUMBER(4),CONSTRAINT emp\_salary\_min CHECK (salary > 0),

13 CONSTRAINT emp\_email\_uk

14 UNIQUE (email), CONSTRAINT emp\_id\_pk PRIMARY KEY (employee\_id) )

15 ORGANIZATION INDEX;

Table created.

Elapsed: 00:00:00.30

SQL> SELECT table\_name, iot\_type, iot\_name FROM user\_tables where table\_name='EMP\_IOT';

TABLE\_NAME IOT\_TYPE IOT\_NAME

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

EMP\_IOT IOT

Elapsed: 00:00:00.01

SQL> CREATE TABLE emp\_iot\_101

2 (employee\_id NUMBER(6),

3 first\_name VARCHAR2(20),

4 last\_name VARCHAR2(25)CONSTRAINT emp\_last\_name\_nnn NOT NULL,

5 email VARCHAR2(25)CONSTRAINT emp\_email\_nnn NOT NULL,

6 phone\_number VARCHAR2(20),

7 hire\_date DATE CONSTRAINT emp\_hire\_date\_nnn NOT NULL,

8 job\_id VARCHAR2(10)CONSTRAINT emp\_job\_nnn NOT NULL,

9 salary NUMBER(8,2),

10 commission\_pct NUMBER(2,2),

11 manager\_id NUMBER(6),

12 department\_id NUMBER(4), CONSTRAINT emp\_salary\_minn CHECK (salary > 0),

13 CONSTRAINT emp\_emaill\_uk UNIQUE (email),

14 CONSTRAINT emp\_idd\_pk PRIMARY KEY (employee\_id) )

15 ORGANIZATION INDEX

16 INCLUDING phone\_number

17 OVERFLOW TABLESPACE USERS;

Table created.

Elapsed: 00:00:00.09

SQL> SELECT table\_name, iot\_type, iot\_name FROM user\_tables where table\_name='EMP\_IOT\_101';

TABLE\_NAME IOT\_TYPE IOT\_NAME

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

EMP\_IOT\_101 IOT

Elapsed: 00:00:00.02

SQL> INSERT INTO EMP\_IOT\_101 SELECT \* FROM HR.EMPLOYEES;

107 rows created.

SQL> INSERT INTO EMP\_IOT SELECT \* FROM HR.EMPLOYEES;

107 rows created.

SQL> SELECT \* FROM EMP\_IOT;

107 rows selected.

Elapsed: 00:00:00.07

Execution Plan

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

Plan hash value: 2754052484

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

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

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

| 0 | SELECT STATEMENT | | 107 | 14231 | 3 (0)| 00:00:01 |

| 1 | INDEX FAST FULL SCAN| EMP\_ID\_PK | 107 | 14231 | 3 (0)| 00:00:01 |

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

Note

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

SQL> SELECT \* FROM EMP\_IOT\_101;

Execution Plan

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

Plan hash value: 2964137647

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

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

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

| 0 | SELECT STATEMENT | | 107 | 133 | 802 (0)| 00:00:10 |

| 1 | INDEX FAST FULL SCAN| EMP\_IDD\_PK | 107 | 133 | 802 (0)| 00:00:10 |

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

Note

-----

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

**Analysis:** There’s a very vast difference in the cost of creation between emp\_iot and emp\_iot\_101. For emp\_iot it is 3 whereas for emp\_iot\_101 it is 802

**QUERY 10**

SQL> CREATE CLUSTER PERSONNEL (DEPTNO NUMBER(2));

Cluster created.

Elapsed: 00:00:00.00

SQL> CREATE TABLE EMP (

2 EMPNO NUMBER(4) PRIMARY KEY,

3 ENAME VARCHAR2(10),

4 JOB VARCHAR2(9),

5 MGR NUMBER(4),

6 HIREDATE DATE,

7 SAL NUMBER(7, 2),

8 COMM NUMBER(7, 2),

9 DEPTNO NUMBER(2))

10 CLUSTER PERSONNEL(DEPTNO);

Table created.

Elapsed: 00:00:00.17

SQL> CREATE TABLE DEPT (

2 DEPTNO NUMBER(2) PRIMARY KEY,

3 DNAME VARCHAR2(14),

4 LOC VARCHAR2(13))

5 CLUSTER PERSONNEL(DEPTNO);

Table created.

Elapsed: 00:00:00.04

SQL> CREATE INDEX PERSONNEL\_IDX ON CLUSTER PERSONNEL;

Index created.

Elapsed: 00:00:00.00

SQL> BEGIN

2 INSERT INTO EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)

3 SELECT EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO FROM SCOTT.EMP;

4 COMMIT;

5 END

6 ;

7 /

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.10

SQL> BEGIN

2 INSERT INTO DEPT(DEPTNO, DNAME, LOC)

3 SELECT DEPTNO, DNAME, LOC FROM SCOTT.DEPT;

4 COMMIT;

5 END;

6 /

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.09

SQL> SELECT \* FROM TAB WHERE TNAME IN ('EMP','DEPT');

TNAME TABTYPE CLUSTERID

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

DEPT TABLE 2

EMP TABLE 1

Elapsed: 00:00:00.06

SQL> SELECT PCT\_FREE,BUFFER\_POOL, TABLESPACE\_NAME,CLUSTER\_TYPE FROM USER\_CLUSTERS

2 WHERE CLUSTER\_NAME='PERSONNEL';

PCT\_FREE BUFFER\_ TABLESPACE\_NAME CLUST

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

10 DEFAULT USERS INDEX

Elapsed: 00:00:00.07

SQL> DROP CLUSTER PERSONNEL INCLUDING TABLES;

Cluster dropped.

Elapsed: 00:00:00.49

SQL> CREATE CLUSTER HASH\_EMP (EMPNO NUMBER(4))

2 HASHKEYS 10

3 HASH IS MOD(EMPNO, 100);

Cluster created.

Elapsed: 00:00:00.06

SQL> CREATE TABLE COPY101 (

2 EMPNO NUMBER(4) PRIMARY KEY,

3 ENAME VARCHAR2(10),

4 JOB VARCHAR2(9),

5 MGR NUMBER(4),

6 HIREDATE DATE,

7 SAL NUMBER(7, 2),

8 COMM NUMBER(7, 2),

9 DEPTNO NUMBER(2))

10 CLUSTER HASH\_EMP(EMPNO);

Table created.

Elapsed: 00:00:00.08

SQL> SELECT \* FROM TAB WHERE TNAME IN ('COPY101');

TNAME TABTYPE CLUSTERID

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

COPY101 TABLE 1

Elapsed: 00:00:00.01

SQL> BEGIN

2 INSERT INTO COPY101 (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)

3 SELECT EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO FROM SCOTT.EMP;

4 COMMIT;

5 END;

6 /

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.05

SQL> SELECT PCT\_FREE,BUFFER\_POOL, TABLESPACE\_NAME,CLUSTER\_TYPE FROM USER\_CLUSTERS

2 WHERE CLUSTER\_NAME='HASH\_EMP';

PCT\_FREE BUFFER\_ TABLESPACE\_NAME CLUST

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

10 DEFAULT USERS HASH