PRACTICAL NO. 4

Q.1)

SQL> set timing on;

SQL> create index CUST\_GEN\_BTR\_IDX on customers\_btree(cust\_gender);

Index created.

Elapsed: 00:00:00.03

SQL> create index CUST\_YOB\_BTR\_IDX on customers\_btree(cust\_year\_of\_birth);

Index created.

Elapsed: 00:00:00.04

SQL> create index CUST\_LNAME\_BTR\_IDX on customers\_btree(cust\_last\_name);

Index created.

Elapsed: 00:00:00.08

SQL> create index CUST\_STRT\_ADDR\_BTR\_IDX on customers\_btree(cust\_street\_address);

Index created.

Elapsed: 00:00:00.08

Q.2)

SQL> create bitmap index CUST\_GEN\_BMP\_IDX on customers\_bitmap(cust\_gender);

Index created.

Elapsed: 00:00:00.10

SQL> create bitmap index CUST\_YOB\_BMP\_IDX on customers\_bitmap(cust\_year\_of\_birth);

Index created.

Elapsed: 00:00:00.03

SQL> create bitmap index CUST\_LNAME\_BMP\_IDX on customers\_bitmap(cust\_last\_name);

Index created.

Elapsed: 00:00:00.03

SQL> create bitmap index CUST\_STRT\_ADDR\_BMP\_IDX on customers\_bitmap(cust\_street\_address);

Index created.

Elapsed: 00:00:00.15

Q.3) (a)

SQL> select substr(segment\_name,1,30) segment\_name, bytes/1024/1024

"Size in MB"

* 1. from user\_segments
  2. where segment\_name in

('CUSTOMERS\_BTREE','CUSTOMERS\_BITMAP');

SEGMENT\_NAME Size in MB ------------------------------ ----------

CUSTOMERS\_BTREE 12

CUSTOMERS\_BITMAP 12

SQL> select substr(segment\_name,1,30) segment\_name, bytes/1024/1024

"Size in MB"

* 1. from user\_segments
  2. where segment\_name in ('CUST\_GEN\_BTR\_IDX',

'CUST\_YOB\_BTR\_IDX', 'CUST\_LNAME\_BTR\_IDX', 'CUST\_STRT\_ADDR\_BTR\_IDX');

SEGMENT\_NAME Size in MB

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

CUST\_GEN\_BTR\_IDX .875

CUST\_YOB\_BTR\_IDX 1

CUST\_LNAME\_BTR\_IDX 2

CUST\_STRT\_ADDR\_BTR\_IDX 3

(b)

SQL> select substr(segment\_name,1,30) segment\_name, bytes/1024/1024

"Size in MB"

* 1. from user\_segments
  2. where segment\_name in ('CUST\_GEN\_BMP\_IDX',

'CUST\_YOB\_BMP\_IDX', 'CUST\_LNAME\_BMP\_IDX',

'CUST\_STRT\_ADDR\_BMP\_IDX');

SEGMENT\_NAME Size in MB ------------------------------ ----------

CUST\_GEN\_BMP\_IDX .0625

CUST\_YOB\_BMP\_IDX .1875

CUST\_LNAME\_BMP\_IDX .125

CUST\_STRT\_ADDR\_BMP\_IDX 3

Q.4)

SQL> set autotrace on;

SQL> select cust\_last\_name from customers\_btree where cust\_year\_of\_birth

= 1967;

CUST\_LAST\_NAME ----------------------------------------

Abbey

Abeles

Adams Alden Alden .

.

.

Zimmerman

Zoldos

Zoldos Zwolinsky

956 rows selected.

Execution Plan

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

Plan hash value: 821136772

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

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

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

)| Time |

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

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

| 0 | SELECT STATEMENT | | 956 | 33460 | 146 (2

)| 00:00:02 |

|\* 1 | VIEW | index$\_join$\_001 | 956 | 33460 | 146 (2

)| 00:00:02 |

|\* 2 | HASH JOIN | | | |

| |

|\* 3 | INDEX RANGE SCAN | CUST\_YOB\_BTR\_IDX | 956 | 33460 |

2 (0

)| 00:00:01 |

| 4 | INDEX FAST FULL SCAN| CUST\_LNAME\_BTR\_IDX | 956 | 33460 | 179 (1

)| 00:00:03 |

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

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

1 - filter("CUST\_YEAR\_OF\_BIRTH"=1967) 2 - access(ROWID=ROWID)

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

Note

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

Statistics

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

9 recursive calls

0 db block gets

296 consistent gets

148 physical reads

0 redo size 20927 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 cust\_last\_name from customers\_bitmap where cust\_year\_of\_birth = 1967;

CUST\_LAST\_NAME ---------------------------------------- Abbey

Abeles Adams Alden

Alden

.

.

.

Zimmerman

Zimmerman

Zoldos

Zoldos Zwolinsky

956 rows selected.

Execution Plan

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

Plan hash value: 502087595

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

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

| Id | Operation | Name | Rows | Bytes | Cos t (%CPU)| Time |

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

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

| 0 | SELECT STATEMENT | | 956 |

33460 |

13 (16)| 00:00:01 |

|\* 1 | VIEW | index$\_join$\_001 | 956 |

33460 |

13 (16)| 00:00:01 | |\* 2 | HASH JOIN | | |

|

| |

| 3 | BITMAP CONVERSION TO ROWIDS| | 956 |

33460 |

1 (0)| 00:00:01 | |\* 4 | BITMAP INDEX SINGLE VALUE | CUST\_YOB\_BMP\_IDX | |

|

| |

| 5 | BITMAP CONVERSION TO ROWIDS| | 956 |

33460 |

10 (0)| 00:00:01 |

| 6 | BITMAP INDEX FULL SCAN | CUST\_LNAME\_BMP\_IDX | |

|

| |

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

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

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

1 - filter("CUST\_YEAR\_OF\_BIRTH"=1967) 2 - access(ROWID=ROWID) 4 - access("CUST\_YEAR\_OF\_BIRTH"=1967)

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

Statistics

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

32 recursive calls

0 db block gets 142 consistent gets 19 physical reads

0 redo size 20927 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> set autotrace off;

Q.5) (a)

SQL> set timing on;

SQL> create index custid\_btr\_idx on customers\_btree(cust\_id);

Index created.

Elapsed: 00:00:00.07

SQL> create bitmap index custid\_bmp\_idx on customers\_bitmap(cust\_id);

Index created.

Elapsed: 00:00:00.10

SQL> select substr(segment\_name,1,30) segment\_name, bytes/1024/1024

"Size in MB"

* 1. from user\_segments
  2. where segment\_name in ('CUSTID\_BTR\_IDX','CUSTID\_BMP\_IDX');

SEGMENT\_NAME Size in MB

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

CUSTID\_BMP\_IDX 2

CUSTID\_BTR\_IDX 2

(b)

SQL> declare

* + 1. upd\_cust\_id number(5);
    2. cust\_yob number(5);
    3. begin
    4. for i in 1 .. 5000 loop
    5. upd\_cust\_id := dbms\_random.value(1, 55000);
    6. cust\_yob := dbms\_random.value(1900, 2000);
    7. update customers\_btree
    8. set cust\_year\_of\_birth = cust\_yob
    9. where cust\_id = upd\_cust\_id;
    10. commit;
    11. end loop;
    12. end;
    13. /

PL/SQL procedure successfully completed.

SQL> declare

* + 1. upd\_cust\_id number(5);
    2. cust\_yob number(5);
    3. begin
    4. for i in 1 .. 5000 loop
    5. upd\_cust\_id := dbms\_random.value(1, 55000);
    6. cust\_yob := dbms\_random.value(1900, 2000);
    7. update customers\_bitmap
    8. set cust\_year\_of\_birth = cust\_yob
    9. where cust\_id = upd\_cust\_id;
    10. commit;
    11. end loop;
    12. end;
    13. /

PL/SQL procedure successfully completed.

SQL> exec dbms\_stats.gather\_table\_stats('GARI',

'CUSTOMERS\_BTREE');

PL/SQL procedure successfully completed.

SQL> exec dbms\_stats.gather\_table\_stats('GARI',

'CUSTOMERS\_BITMAP');

PL/SQL procedure successfully completed.

(c)

SQL> select substr(segment\_name,1,30) segment\_name, bytes/1024/1024

"Size in MB"

* 1. from user\_segments
  2. where segment\_name in ('CUSTID\_BTR\_IDX','CUSTID\_BMP\_IDX');

SEGMENT\_NAME Size in MB

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

CUSTID\_BMP\_IDX 2

CUSTID\_BTR\_IDX 2

Q.6) (a)

SQL> set timing on

SQL> create table customers\_bijx\_test\_bitmap as (select \* from sh.customers);

Table created.

Elapsed: 00:00:00.19

SQL> create table sales\_bijx\_test\_bitmap as (select \* from sh.sales);

Table created.

Elapsed: 00:00:02.44

(b)

SQL> create bitmap index sales\_bijx\_test\_bitmap\_bix1 on sales\_bijx\_test\_bitmap(cust\_id);

Index created.

Elapsed: 00:00:00.79

SQL> create bitmap index cust\_bijx\_test\_bitmap\_bix1 on customers\_bijx\_test\_bitmap(cust\_last\_name);

Index created.

Elapsed: 00:00:00.09

(c)

SQL> create table customers\_bijx\_test\_bitjoin as (select \* from sh.customers);

Table created.

Elapsed: 00:00:00.87

SQL> create table sales\_bijx\_test\_bitjoin as (select \* from sh.sales);

Table created.

Elapsed: 00:00:01.78

SQL> alter table customers\_bijx\_test\_bitjoin add constraint cid\_pk primary key(cust\_id);

Table altered.

Elapsed: 00:00:00.17

(d)(A)

SQL> create bitmap index sales\_bijx\_test\_bitjoin\_bjx1 on sales\_bijx\_test\_bitjoin(c.cust\_id)

* 1. from sales\_bijx\_test\_bitjoin s, customers\_bijx\_test\_bitjoin c
  2. where s.cust\_id = c.cust\_id;

Index created.

Elapsed: 00:00:01.86

(B)

SQL> create bitmap index sales\_bijx\_test\_bitjoin\_bjx2 on sales\_bijx\_test\_bitjoin(c.cust\_last\_name)

1. from sales\_bijx\_test\_bitjoin s, customers\_bijx\_test\_bitjoin c
2. where s.cust\_id = c.cust\_id;

Index created.

Elapsed: 00:00:01.66

SQL> set timing off;

COMPRESSED INDEX:

1. CREATE TABLE STUDENT(STUDID, STUDNAME)

SQL> create table student (

* 1. stud\_id number(5),
  2. stud\_name varchar2(20));

Table created.

1. ADD 10 ROWS

SQL> insert into student values(&stud\_id, '&stud\_name');

Enter value for stud\_id: 1

Enter value for stud\_name: ABC

old 1: insert into student values(&stud\_id, '&stud\_name') new 1: insert into student values(1, 'ABC’)

1 row created.

SQL> /

Enter value for stud\_id: 2

Enter value for stud\_name: DEF

old 1: insert into student values(&stud\_id, '&stud\_name') new 1: insert into student values(2, 'DEF’)

1 row created.

SQL> /

Enter value for stud\_id: 3

Enter value for stud\_name: GHI

old 1: insert into student values(&stud\_id, '&stud\_name') new 1: insert into student values(3, 'GHI')

1 row created.

SQL> /

Enter value for stud\_id: 4

Enter value for stud\_name: JKL

old 1: insert into student values(&stud\_id, '&stud\_name') new 1: insert into student values(4, 'JKL’)

1 row created.

SQL> /

Enter value for stud\_id: 5

Enter value for stud\_name: MNO

old 1: insert into student values(&stud\_id, '&stud\_name') new 1: insert into student values(5, ‘MNO’)

1 row created.

SQL> /

Enter value for stud\_id: 6

Enter value for stud\_name: PQR

old 1: insert into student values(&stud\_id, '&stud\_name') new 1: insert into student values(6, ‘PQR’)

1 row created.

SQL> /

Enter value for stud\_id: 7

Enter value for stud\_name: STU

old 1: insert into student values(&stud\_id, '&stud\_name') new 1: insert into student values(7, ‘STU’)

1 row created.

SQL> /

Enter value for stud\_id: 8

Enter value for stud\_name: VWX

old 1: insert into student values(&stud\_id, '&stud\_name') new 1: insert into student values(8, ‘VWX’)

1 row created.

SQL> /

Enter value for stud\_id: 9

Enter value for stud\_name: YZ

old 1: insert into student values(&stud\_id, '&stud\_name') new 1: insert into student values(9, 'YZ’)

1 row created.

SQL> /

Enter value for stud\_id: 10

Enter value for stud\_name: BAAC

old 1: insert into student values(&stud\_id, '&stud\_name') new 1: insert into student values(10, 'BAAC')

1 row created.

1. DEFINE INDEX ON STUDNAME(FIRST NAME AND LAST NAME)

SQL> set timing on

SQL> create index stud\_name\_idx on student(stud\_name);

Index created.

Elapsed: 00:00:00.00

1. GET THE STATISTICS OF INDEX

SQL> SELECT

* 1. COMPRESSION,
  2. LEAF\_BLOCKS,
  3. Round(NUM\_ROWS/Decode(LEAF\_BLOCKS,0,1,LEAF\_BLOCKS))

"ROWS PER BLOCK", DISTINCT\_KEYS,

* 1. NUM\_ROWS,NUM\_ROWS-DISTINCT\_KEYS DUP\_ROWS
  2. FROM
  3. USER\_INDEXES
  4. WHERE
  5. INDEX\_NAME = 'STUD\_NAME\_IDX';

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

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

DISABLED 1 10 10 10

0

Elapsed: 00:00:00.01

1. NOW ADD ABOUT 10000 ROWS THAT WILL HAVE SAME LAST NAME

SQL> DECLARE v\_a NUMBER;

* 1. BEGIN
  2. v\_a := 11;
  3. WHILE v\_a <= 10000
  4. LOOP
  5. INSERT INTO STUDENT VALUES(v\_a,'CAAB TIH’);
  6. v\_a := v\_a + 1;
  7. END LOOP;
  8. COMMIT;
  9. END;
  10. /

PL/SQL procedure successfully completed.

Elapsed: 00:00:01.01

1. GET THE STATISTICS OF INDEX

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

PL/SQL procedure successfully completed.

Elapsed: 00:00:01.15

SQL> SELECT

* 1. COMPRESSION,
  2. LEAF\_BLOCKS,
  3. Round(NUM\_ROWS/Decode(LEAF\_BLOCKS,0,1,LEAF\_BLOCKS)) "ROWS

PER BLOCK", DISTINCT\_KEYS,

* 1. NUM\_ROWS,NUM\_ROWS-DISTINCT\_KEYS DUP\_ROWS
  2. FROM
  3. USER\_INDEXES
  4. WHERE
  5. INDEX\_NAME = 'STUDENTNAME\_IDX';

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

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

DISABLED 41 244 11 10000

9989

Elapsed: 00:00:00.00

1. DROP INDEX

SQL> drop index stud\_name\_idx;

Index dropped.

Elapsed: 00:00:00.01

1. CREATE COMPRESSED INDEX

SQL> CREATE INDEX STUD\_NAME\_IDX ON STUDENT(STUD\_NAME)COMPRESS

TABLESPACE USERS;

Index created.

Elapsed: 00:00:00.03

1. GET THE STATISTICS OF INDEX

SQL> SELECT

* + 1. COMPRESSION,
    2. LEAF\_BLOCKS,
    3. Round(NUM\_ROWS/Decode(LEAF\_BLOCKS,0,1,LEAF\_BLOCKS)) "ROWS

PER BLOCK", DISTINCT\_KEYS,

* + 1. NUM\_ROWS,NUM\_ROWS-DISTINCT\_KEYS DUP\_ROWS
    2. FROM
    3. USER\_INDEXES
    4. WHERE
    5. INDEX\_NAME = 'STUD\_NAME\_IDX';

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

DUP\_ROWS

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

ENABLED 16 625 10 10000

9990

Elapsed: 00:00:00.00

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.

SQL> create table emp as (select \* from hr.employees);

Table created.

Elapsed: 00:00:00.08

SQL> select count(employee\_id) from emp where

(12\*salary\*commission\_pct) < 50000;

COUNT(EMPLOYEE\_ID) ------------------ 34

Elapsed: 00:00:00.00

SQL> select employee\_id, first\_name, last\_name from emp where

(12\*salary\*commission\_pct) < 50000;

EMPLOYEE\_ID FIRST\_NAME LAST\_NAME ----------- -------------------- ------------------------- 146 Karen Partners 147 Alberto Errazuriz

148 Gerald Cambrault 149 Eleni Zlotkey

1. Peter Tucker
2. David Bernstein
3. Peter Hall
4. Christopher Olsen
5. Nanette Cambrault
6. Oliver Tuvault 156 Janette King

.

.

.

EMPLOYEE\_ID FIRST\_NAME LAST\_NAME ----------- -------------------- ------------------------- 179 Charles Johnson

34 rows selected.

Elapsed: 00:00:00.02

SQL> CREATE INDEX emp\_total\_sal\_idx

2 ON emp (12 \* salary \* commission\_pct, salary, commission\_pct);

Index created.

Elapsed: 00:00:00.01

SQL> select employee\_id, first\_name, last\_name from emp where

(12\*salary\*commission\_pct) < 50000;

EMPLOYEE\_ID FIRST\_NAME LAST\_NAME ----------- -------------------- -------------------------

173 Sundita Kumar

167 Amit Banda

179 Charles Johnson

166 Sundar Ande

165 David Lee 164 Mattea Marvins

155 Oliver Tuvault

178 Kimberely Grant

172 Elizabeth Bates 171 William Smith 163 Danielle Greene

.

.

.

34 rows selected.

Elapsed: 00:00:00.03

1. CREATE USER TABLE WITH ATTRIBUTES (USERID, USERNAME, GENDER)

SQL> CREATE TABLE user\_data (

* 1. id NUMBER(10) NOT NULL,
  2. first\_name VARCHAR2(40) NOT NULL,
  3. last\_name VARCHAR2(40) NOT NULL,
  4. gender VARCHAR2(1));

Table created.

Elapsed: 00:00:00.02

1. INSERT 10000 RECORDS IN USER TABLE

SQL> BEGIN

* 1. FOR cur\_rec IN 1 .. 2000 LOOP
  2. IF MOD(cur\_rec, 2) = 0 THEN
  3. INSERT INTO user\_data
  4. VALUES (cur\_rec, 'John' || cur\_rec, 'Doe', 'M');
  5. ELSE
  6. INSERT INTO user\_data
  7. VALUES (cur\_rec, 'Jayne' || cur\_rec, 'Doe', 'F');
  8. END IF;
  9. COMMIT;
  10. END LOOP;
  11. END;
  12. /

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.10

SQL> EXEC DBMS\_STATS.gather\_table\_stats(USER, 'user\_data', cascade

=> TRUE);

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.05

1. BUILD REGULAR INDEX ON USERNAME

SQL> CREATE INDEX name\_idx ON user\_data (first\_name);

Index created.

Elapsed: 00:00:00.00

SQL> EXEC DBMS\_STATS.gather\_table\_stats(USER, 'user\_data', cascade

=> TRUE);

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.16

SQL> set autotrace on;

SQL> SELECT \* from user\_data where UPPER(first\_name) like 'JOHN%';

ID FIRST\_NAME ---------- ---------------------------------------- LAST\_NAME G ---------------------------------------- -

938 John938 Doe M

940 John940 Doe M

942 John942 Doe M .

.

.

1000 rows selected.

Elapsed: 00:00:01.40

Execution Plan

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

Plan hash value: 2489064024

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

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

| Id | Operation | Name | Rows | Bytes | Cost

(%CPU)| Time |

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

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

| 0 | SELECT STATEMENT | | 100 | 1900 | 5 (0)| 00:00:01 |

|\* 1 | TABLE ACCESS FULL| USER\_DATA | 100 | 1900 | 5

(0)| 00:00:01 |

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

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

Predicate Information (identified by operation id): --------------------------------------------------- 1 - filter(UPPER("FIRST\_NAME") LIKE 'JOHN%')

Statistics

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

1 recursive calls

0 db block gets 83 consistent gets 0 physical reads

0 redo size 30238 bytes sent via SQL\*Net to client 1246 bytes received via SQL\*Net from client

68 SQL\*Net roundtrips to/from client 0 sorts (memory)

0 sorts (disk)

1000 rows processed

5. BUILD FUNCTION BASED INDEX ON USER NAME BASED ON UPPER FUNCTION

SQL> drop index name\_idx;

Index dropped.

Elapsed: 00:00:00.01

SQL> CREATE INDEX name\_idx ON user\_data (UPPER(first\_name));

Index created.

Elapsed: 00:00:00.18

SQL> EXEC DBMS\_STATS.gather\_table\_stats(USER, 'user\_data', cascade

=> TRUE);

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.05

SQL> set autotrace on;

SQL> SELECT \* from user\_data where UPPER(first\_name) like 'JOHN%';

ID FIRST\_NAME ---------- ---------------------------------------- LAST\_NAME G ---------------------------------------- - 10 John10

Doe M

100 John100 Doe M

1000 John1000 Doe M

.

.

.

1000 rows selected.

Elapsed: 00:00:00.85

Execution Plan

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

Plan hash value: 2250899927

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

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

--------- | Id | Operation | Name | Rows | Bytes | Cost (%CPU)| T ime |

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

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

---------

| 0 | SELECT STATEMENT | | 1 | 28 | 3 (0)| 0

0:00:01 |

| 1 | TABLE ACCESS BY INDEX ROWID| USER\_DATA | 1 | 28 | 3 (0)| 0

0:00:01 |

|\* 2 | INDEX RANGE SCAN | NAME\_IDX | 1 | | 2 (0)| 0

0:00:01 |

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

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

---------

Predicate Information (identified by operation id): --------------------------------------------------- 2 - access(UPPER("FIRST\_NAME") LIKE 'JOHN%') filter(UPPER("FIRST\_NAME") LIKE 'JOHN%')

Statistics

---------------------------------------------------------- 1 recursive calls 0 db block gets

303 consistent gets 0 physical reads 0 redo size

30238 bytes sent via SQL\*Net to client 1246 bytes received via SQL\*Net from client 68 SQL\*Net roundtrips to/from client

0 sorts (memory) 0 sorts (disk)

1000 rows processed

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.

SQL> create tablespace ts\_lookup datafile '/temp/ts\_lookup.dbf' size 10M;

Tablespace created.

Elapsed: 00:00:00.09

SQL>

SQL> CREATE TABLE iot\_lookups(

* + 1. lookup\_code NUMBER(3),
    2. lookup\_value NUMBER(3),
    3. lookup\_desc VARCHAR2(40),
    4. CONSTRAINT iot\_pk PRIMARY KEY (lookup\_code))
    5. ORGANIZATION index
    6. TABLESPACE ts\_lookup
    7. PCTTHRESHOLD 20
    8. INCLUDING lookup\_value
    9. OVERFLOW TABLESPACE users;

Table created.

Elapsed: 00:00:00.02

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: 00:00:00.03

1. CREATE A INDEX ORGANIZED TABLE(IOT) EMP\_IOT BASED ON HR.EMPLOYEES

SQL> CREATE TABLE emp\_iot

* 1. ( employee\_id NUMBER(6)
  2. , first\_name VARCHAR2(20)
  3. , last\_name VARCHAR2(25)
  4. CONSTRAINT emp\_last\_name\_nn NOT NULL
  5. , email VARCHAR2(25)
  6. CONSTRAINT emp\_email\_nn NOT NULL
  7. , phone\_number VARCHAR2(20)
  8. , hire\_date DATE
  9. CONSTRAINT emp\_hire\_date\_nn NOT NULL
  10. , job\_id VARCHAR2(10)
  11. CONSTRAINT emp\_job\_nn NOT NULL
  12. , salary NUMBER(8,2)
  13. , commission\_pct NUMBER(2,2)
  14. , manager\_id NUMBER(6)
  15. , department\_id NUMBER(4)
  16. , CONSTRAINT emp\_salary\_min
  17. CHECK (salary > 0)
  18. , CONSTRAINT emp\_email\_uk
  19. UNIQUE (email), CONSTRAINT emp\_id\_pk PRIMARY KEY

(employee\_id) )

* 1. ORGANIZATION INDEX
  2. ;

Table created.

Elapsed: 00:00:00.06

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: 00:00:00.00

1. CREATE A INDEX ORGANIZED TABLE(IOT) EMP101\_EMP BASED ON HR.EMPLOYEES. PLACE THE COLUMN HIREDATE IN OVERFLOW AREA.

SQL> CREATE TABLE emp\_iot\_101

* 1. ( employee\_id NUMBER(6)
  2. , first\_name VARCHAR2(20)
  3. , last\_name VARCHAR2(25)
  4. CONSTRAINT emp\_last\_name\_nn NOT NULL
  5. , email VARCHAR2(25)
  6. CONSTRAINT emp\_email\_nn NOT NULL
  7. , phone\_number VARCHAR2(20)
  8. , hire\_date DATE
  9. CONSTRAINT emp\_hire\_date\_nn NOT NULL
  10. , job\_id VARCHAR2(10)
  11. CONSTRAINT emp\_job\_nn NOT NULL
  12. , salary NUMBER(8,2)
  13. , commission\_pct NUMBER(2,2)
  14. , manager\_id NUMBER(6)
  15. , department\_id NUMBER(4)
  16. , CONSTRAINT emp\_salary\_min
  17. CHECK (salary > 0)
  18. , CONSTRAINT emp\_email\_uk
  19. UNIQUE (email), CONSTRAINT emp\_id\_pk PRIMARY KEY

(employee\_id) )

* 1. ORGANIZATION INDEX
  2. INCLUDING phone\_number
  3. OVERFLOW TABLESPACE USERS;

Table created.

Elapsed: 00:00:00.03

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: 00:00:00.00

CLUSTERS

1. CREATE A CLUSTER PERSONNEL CONTAINING COPY\_EMP AND COPY\_DEPT TABLES. CLUSTER KEY IS DEPTNO.

SQL> create cluster personnel (deptno number(2));

Cluster created.

Elapsed: 00:00:00.02

SQL>

SQL> create table copy\_emp (

* 1. empno number(4) PRIMARY KEY,
  2. ename varchar2(10),
  3. job varchar2(9),
  4. mgr number(4),
  5. hiredate date,
  6. sal number(7, 2),
  7. comm number(7, 2),
  8. deptno number(2))
  9. cluster personnel(deptno);

Table created.

Elapsed: 00:00:00.06

SQL> create table copy\_dept (

* 1. deptno number(2) PRIMARY KEY,
  2. dname varchar2(14),
  3. loc varchar2(13))
  4. cluster personnel(deptno);

Table created.

Elapsed: 00:00:00.01

1. CREATE AN INDEX ON CLUSTER PERSONNEL.

SQL> create index personnel\_idx on cluster personnel;

Index created.

Elapsed: 00:00:00.01

SQL> select \* from tab where tname in ('COPY\_EMP','COPY\_DEPT');

TNAME TABTYPE CLUSTERID ------------------------------ ------- ---------- COPY\_DEPT TABLE 2 COPY\_EMP TABLE 1

Elapsed: 00:00:00.07

SQL> select pct\_free,buffer\_pool, tablespace\_name,cluster\_type from user\_clusters where cluster\_name='PERSONNEL';

PCT\_FREE BUFFER\_ TABLESPACE\_NAME CLUST ---------- ------- ------------------------------ ----- 10 DEFAULT USERS INDEX

Elapsed: 00:00:00.02

1. POPULATE COPY\_EMP AND COPY\_DEPT WITH DATA FROM EMP AND DEPT TABLES OF SCOTT RESPECTIVELY.

SQL> begin

* 1. insert into copy\_emp (empno, ename, job, mgr, hiredate, sal, comm, deptno)
  2. select empno, ename, job, mgr, hiredate, sal, comm, deptno from scott.emp; 4 commit;
  3. end;
  4. /

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.00

SQL>

SQL> begin

* 1. insert into copy\_dept (deptno, dname, loc)
  2. select deptno, dname, loc from scott.dept;
  3. commit;
  4. end;
  5. /

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.01

1. CREATE A DUMMY TABLE "DUMMY" WHICH REFERENCES EMPNO OF COPY\_EMP.

SQL> create table dummy (

* 1. dummy\_val number(4) constraint dfk references copy\_emp(empno));

Table created.

Elapsed: 00:00:00.00

1. DROP CLUSTER PERSONNEL**.**

SQL> drop cluster personnel including tables; drop cluster personnel including tables

\*

ERROR at line 1:

ORA-02449: unique/primary keys in table referenced by foreign keys

Elapsed: 00:00:00.02

SQL> drop table dummy;

Table dropped.

Elapsed: 00:00:00.05

SQL> drop cluster personnel including tables;

Cluster dropped.

Elapsed: 00:00:00.09

**6.** CREATE A HASH CLUSTER NAMED HASH\_EMP CONTAINING TABLE COPY101\_EMP. CREATE 10 HASHKEYS AND USE THE HASH FUNCTION (EMPNO MOD 100).

SQL> create cluster hash\_emp (empno number(4))

1. HASHKEYS 10
2. HASH IS MOD(empno, 100);

Cluster created.

Elapsed: 00:00:00.06

SQL> create table copy101\_emp (

1. empno number(4) PRIMARY KEY,
2. ename varchar2(10),
3. job varchar2(9),
4. mgr number(4),
5. hiredate date,
6. sal number(7, 2),
7. comm number(7, 2),
8. deptno number(2))
9. cluster hash\_emp(empno);

Table created.

Elapsed: 00:00:00.01

SQL> select \* from tab where tname in ('COPY101\_EMP');

TNAME TABTYPE CLUSTERID ------------------------------ ------- ---------- COPY101\_EMP TABLE 1

Elapsed: 00:00:00.00

SQL> begin

* 1. insert into copy101\_emp (empno, ename, job, mgr, hiredate, sal, comm, deptno)
  2. select empno, ename, job, mgr, hiredate, sal, comm, deptno from scott.emp; 4 commit;
  3. end;
  4. /

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.01

SQL> select pct\_free,buffer\_pool, tablespace\_name,cluster\_type from user\_clusters where cluster\_name='HASH\_EMP';

PCT\_FREE BUFFER\_ TABLESPACE\_NAME CLUST ---------- ------- ------------------------------ ----- 10 DEFAULT USERS HASH

Elapsed: 00:00:00.00

SQL> select pct\_free,buffer\_pool, tablespace\_name,cluster\_type from user\_clusters where cluster\_name='HASH\_EMP';

PCT\_FREE BUFFER\_ TABLESPACE\_NAME CLUST ---------- ------- ------------------------------ ----- 10 DEFAULT USERS HASH

Elapsed: 00:00:00.00