|  |
| --- |
|  |
|  |
|  | **PRACTICAL NO. 4** |
|  |  |
|  |  |
|  |  |
|  |  |
|  | **Problem Statement:** |
|  |  |
|  | Write and Execute SQL indexing queries for |
|  | data warehouse. |
|  |  |
|  |  |
|  |  |
|  | /\*Q1. Create individual b-tree indexes on the following columns of the table: |
|  | customers\_copy\_btree |
|  | (a) cust\_gender |
|  | (b) cust\_year\_of\_birth |
|  | (c) cust\_last\_name |
|  | (d) cust\_street\_address |
|  | How long does it take to create the indexes?\*/ |
|  |  |
|  | create table customer\_copy\_btree as select \* from sh.customers; |
|  | create index customer\_copy\_btree\_gender\_idx on customer\_copy\_btree(cust\_gender); |
|  | Elapsed: 00:00:00.20 |
|  |  |
|  | create index customer\_copy\_btree\_yob\_idx on customer\_copy\_btree(cust\_year\_of\_birth); |
|  | Elapsed: 00:00:00.03 |
|  |  |
|  | create index customer\_copy\_btree\_lname\_idx on customer\_copy\_btree(cust\_last\_name); |
|  | Elapsed: 00:00:00.06 |
|  |  |
|  | create index customer\_copy\_btree\_stra\_idx on customer\_copy\_btree(cust\_street\_address); |
|  | Elapsed: 00:00:00.04 |
|  |  |
|  | COLUMN TIME INDEX BTREE |
|  |  |
|  | CREATION TIME Elapsed: 00:00:00.29 |
|  | GENDER Elapsed: 00:00:00.29 |
|  | YEAR OF BIRTH Elapsed: 00:00:00.23 |
|  | LAST NAME Elapsed: 00:00:00.04 |
|  | STREET ADDRESS Elapsed: 00:00:00.04 |
|  |  |
|  | /\*Q2. 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 TABLE CUSTOMER\_BITMAP AS ( SELECT \* FROM SH.CUSTOMERS); |
|  |  |
|  | Table created. |
|  |  |
|  | Elapsed: 00:00:00.29 |
|  | SQL> create bitmap index cus\_gender\_bitmap\_idx on customer\_bitmap(cust\_gender); |
|  |  |
|  | Index created. |
|  |  |
|  | Elapsed: 00:00:00.23 |
|  | SQL> create bitmap index cus\_year\_of\_birth\_bitmap\_idx on customer\_bitmap(cust\_year\_of\_birth); |
|  |  |
|  | Index created. |
|  |  |
|  | Elapsed: 00:00:00.04 |
|  | SQL> create bitmap index cus\_last\_name\_bitmap\_idx on customer\_bitmap(cust\_last\_name); |
|  |  |
|  | Index created. |
|  |  |
|  | Elapsed: 00:00:00.03 |
|  | SQL> create bitmap index cus\_street\_address\_bitmap\_idx on customer\_bitmap(cust\_street\_address); |
|  |  |
|  | Index created. |
|  |  |
|  | Elapsed: 00:00:00.04 |
|  |  |
|  | COMPARISON CHART |
|  |  |
|  | COLUMN TIME INDEX BTREE TIME INDEX BITMAP |
|  |  |
|  | CREATION TIME Elapsed: 00:00:03.06 Elapsed: 00:00:02.27 |
|  | GENDER Elapsed: 00:00:00.38 Elapsed: 00:00:00.34 |
|  | YEAR OF BIRTH Elapsed: 00:00:00.64 Elapsed: 00:00:00.03 |
|  | LAST NAME Elapsed: 00:00:00.32 Elapsed: 00:00:00.05 |
|  | STREET ADDRESS Elapsed: 00:00:01.03 Elapsed: 00:00:00.14 |
|  |  |
|  | /\*Q3. Do as directed: |
|  | (a) Find the size of each segment: customers\_copy\_bitmap and customers\_copy\_btree |
|  | (b) The b-tree index range for high and low cardinality address index. |
|  | (c) The bitmap index range for high and low cardinality address index. |
|  | \*/ |
|  |  |
|  | --A |
|  |  |
|  | select segment\_name, |
|  | bytes/1024/1024 "Size in MB" |
|  | from user\_segments |
|  | where segment\_name like '%CUSTOMER\_COPY%'; |
|  |  |
|  | --B |
|  |  |
|  | select segment\_name, |
|  | bytes/1024/1024 "Size in MB" |
|  | from user\_segments |
|  | where segment\_name like '%BTREE%'; |
|  |  |
|  | SEGMENT\_NAME Size in MB |
|  | --------------------------------------------------------------------------------- ---------- |
|  | CUS\_GENDER\_BTREE\_IDX .875 |
|  | CUS\_YEAR\_OF\_BIRTH\_BTREE\_IDX 1 |
|  | CUS\_LAST\_NAME\_BTREE\_IDX 2 |
|  | CUS\_GENDER\_BTREE\_IDXX .875 |
|  | CUS\_YEAR\_OF\_BIRTH\_BTREE\_IDXX 1 |
|  | CUS\_LAST\_NAME\_BTREE\_IDXX 2 |
|  |  |
|  |  |
|  | --C |
|  |  |
|  | select segment\_name, |
|  | bytes/1024/1024 "Size in MB" |
|  | from user\_segments |
|  | where segment\_name like '%BITMAP%'; |
|  |  |
|  |  |
|  | SEGMENT\_NAME Size in MB |
|  | --------------------------------------------------------------------------------- ---------- |
|  | CUSTOMER\_BITMAPS 12 |
|  | CUS\_GENDER\_BITMAP\_IDX .0625 |
|  | CUS\_YEAR\_OF\_BIRTH\_BITMAP\_IDX .1875 |
|  | CUS\_LAST\_NAME\_BITMAP\_IDX .125 |
|  | CUS\_STREET\_ADDRESS\_BITMAP\_IDX 3 |
|  |  |
|  | Elapsed: 00:00:00.06 |
|  |  |
|  | /\*Q4. 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.\*/ |
|  |  |
|  | set autotrace on; |
|  |  |
|  | select \* from customer\_copy\_btree where cust\_year\_of\_birth=1967; |
|  |  |
|  | 956 rows selected. |
|  |  |
|  | Elapsed: 00:00:12.15 |
|  |  |
|  | Execution Plan |
|  | ---------------------------------------------------------- |
|  | Plan hash value: 3388583990 |
|  |  |
|  | ---------------------------------------------------------------------------------- |
|  | | Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time | |
|  | ---------------------------------------------------------------------------------- |
|  | | 0 | SELECT STATEMENT | | 956 | 278K| 406 (1)| 00:00:05 | |
|  | |\* 1 | TABLE ACCESS FULL| CUSTOMERTEST | 956 | 278K| 406 (1)| 00:00:05 | |
|  | ---------------------------------------------------------------------------------- |
|  |  |
|  | select \* from customer\_copy\_bitmap where cust\_year\_of\_birth=1967; |
|  |  |
|  | Execution Plan |
|  | ---------------------------------------------------------- |
|  | Plan hash value: 2649141227 |
|  |  |
|  | --------------------------------------------------------------------------------------------------------- |
|  | | Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time | |
|  | --------------------------------------------------------------------------------------------------------- |
|  | | 0 | SELECT STATEMENT | | 956 | 278K| 98 (0)| 00:00:02 | |
|  | | 1 | TABLE ACCESS BY INDEX ROWID | CUSTOMER\_COPY\_BITMAP | 956 | 278K| 98 (0)| 00:00:02 | |
|  | | 2 | BITMAP CONVERSION TO ROWIDS| | | | | | |
|  | |\* 3 | BITMAP INDEX SINGLE VALUE | CUST\_COPY\_BITMAP\_YOB\_IDX | | | | | |
|  | --------------------------------------------------------------------------------------------------------- |
|  |  |
|  |  |
|  | For btree indexed table,a full table scan is run. |
|  | The cost of the execution plan against the bitmap indexed table |
|  | is shown above. |
|  |  |
|  |  |
|  | /\*Q5. Show that update to the bitmap indexed column takes a bit longer than the b-tree indexed |
|  | column. |
|  | (a) Create new indexes on cust\_id column of btree and bitmap table. |
|  | (b) Set the timing on |
|  | (c) Write a PL/SQL procedure for each table as directed: |
|  | i. Create new columns- upd\_cust\_id and cust\_yob\_value with number format. |
|  | ii. In loop of 500 allot random values to both columns |
|  | iii. set cust\_year\_of\_birth = cust\_yob\_value and consider cust\_id = upd\_cust\_id |
|  | (d) What is the size of the indexes compared to the size as before the updates took place. |
|  | \*/ |
|  |  |
|  |  |
|  | --A |
|  | create index test\_cid\_btree on customertest(cust\_id); |
|  | create index test\_cid\_bitmap on customer\_copy\_bitmaps(cust\_id); |
|  |  |
|  | SQL> create index test\_cid\_btree on customertest(cust\_id); |
|  | Index created. |
|  |  |
|  | Elapsed: 00:00:00.78 |
|  |  |
|  | SQL> create index test\_cid\_bitmap on customer\_bitmaps(cust\_id); |
|  | Index created. |
|  |  |
|  | Elapsed: 00:00:00.38 |
|  |  |
|  | --B |
|  | SET TIMING ON |
|  |  |
|  | --C.1 |
|  | declare |
|  | upd\_cust\_id number(5); |
|  | upd\_yob number(4); |
|  | begin |
|  | for i in 1 .. 500 loop |
|  | upd\_cust\_id := dbms\_random.value(1,55000); |
|  | upd\_yob := dbms\_random.value(1900,2000); |
|  | update customer\_bitmaps |
|  | set cust\_year\_of\_birth=upd\_yob |
|  | where cust\_id = upd\_cust\_id; |
|  | commit; |
|  | end loop; |
|  | end; |
|  | / |
|  | PL/SQL procedure successfully completed. |
|  |  |
|  | Elapsed: 00:00:00.76 |
|  |  |
|  | --C.2 |
|  | declare |
|  | upd\_cust\_id number(5); |
|  | upd\_yob number(4); |
|  | begin |
|  | for i in 1 .. 500 loop |
|  | upd\_cust\_id := dbms\_random.value(1,55000); |
|  | upd\_yob := dbms\_random.value(1900,2000); |
|  | update customertest |
|  | set cust\_year\_of\_birth=upd\_yob |
|  | where cust\_id = upd\_cust\_id; |
|  | commit; |
|  | end loop; |
|  | end; |
|  | / |
|  |  |
|  | PL/SQL procedure successfully completed. |
|  |  |
|  | Elapsed: 00:00:01.04 |
|  |  |
|  | /\*Q6. Comparison of time for index creation for normal bitmap index and join bitmap index. |
|  | Do as directed: |
|  | a) Create table customers\_bijx\_test\_bitmap from customers & sales\_bijx\_test\_bitmap as |
|  | from sales |
|  | b) 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? |
|  | c) 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. |
|  | d) (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. |
|  | \*/ |
|  |  |
|  | --A |
|  | create table customers\_bijx\_test\_bitmap as select \* from sh.customers; |
|  | create bitmap index ccust\_bijx\_test\_bitmap\_bix1 on customers\_bijx\_test\_bitmap(cust\_last\_name); |
|  | Elapsed: 00:00:00.01 |
|  |  |
|  |  |
|  | --B |
|  | create table sales\_bijx\_test\_bitmap as select \* from sh.sales; |
|  | create bitmap index sales\_bijx\_test\_bitmap\_bix1 on sales\_bijx\_test\_bitmap(cust\_id); |
|  | Elapsed: 00:00:00.39 |
|  |  |
|  |  |
|  | --C |
|  | create table customers\_bijx\_test\_bitjoin as select \* from(customers); |
|  | alter table customers\_bijx\_test\_bitjoin add constraint pk\_cust PRIMARY KEY(cust\_id); |
|  | create table sales\_bijx\_test\_bitjoin as select \* from(sh.sales); |
|  |  |
|  | --D.A |
|  | create bitmap index sales\_bijx\_test\_bitjoin\_bjx1 on sales\_bijx\_test\_bitjoin(customers\_bijx\_test\_bitjoin.cust\_id) |
|  | from sales\_bijx\_test\_bitjoin,customers\_bijx\_test\_bitjoin |
|  | where sales\_bijx\_test\_bitjoin.cust\_id=customers\_bijx\_test\_bitjoin.cust\_id; |
|  | Elapsed: 00:00:01.09 |
|  |  |
|  |  |
|  | --D.B |
|  | create bitmap index sales\_bijx\_test\_bitjoin\_bjx2 on sales\_bijx\_test\_bitjoin(customers\_bijx\_test\_bitjoin.cust\_last\_name) |
|  | from sales\_bijx\_test\_bitjoin,customers\_bijx\_test\_bitjoin |
|  | where sales\_bijx\_test\_bitjoin.cust\_id=customers\_bijx\_test\_bitjoin.cust\_id; |
|  | Elapsed: 00:00:01.05 |
|  |  |
|  |  |
|  |  |
|  | ---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\*/ |
|  |  |
|  | --1. |
|  | create table student( |
|  | studid int, |
|  | studname varchar2(30)); |
|  |  |
|  | --2. |
|  | insert into student values(1,'A'); |
|  | insert into student values(2,'B'); |
|  | insert into student values(3,'C'); |
|  | insert into student values(4,'D'); |
|  | insert into student values(5,'E'); |
|  | insert into student values(6,'F'); |
|  | insert into student values(7,'G'); |
|  | insert into student values(8,'H'); |
|  | insert into student values(9,'I'); |
|  |  |
|  | --3. |
|  | CREATE INDEX STUDENT\_BTREE\_INDEX ON STUDENT(studname); |
|  | Index created. |
|  | Elapsed: 00:00:00.02 |
|  |  |
|  | --4 |
|  | SELECT |
|  | COMPRESSION, |
|  | LEAF\_BLOCKS, |
|  | Round(NUM\_ROWS/Decode(LEAF\_BLOCKS,0,1,LEAF\_BLOCKS)) "ROWS PER BLOCK", DISTINCT\_KEYS, |
|  | NUM\_ROWS,NUM\_ROWS-DISTINCT\_KEYS DUP\_ROWS |
|  | FROM |
|  | USER\_INDEXES |
|  | WHERE |
|  | INDEX\_NAME = 'STUDENT\_BTREE\_INDEX'; |
|  |  |
|  |  |
|  | COMPRESS LEAF\_BLOCKS ROWS PER BLOCK DISTINCT\_KEYS NUM\_ROWS DUP\_ROWS |
|  | -------- ----------- -------------- ------------- ---------- ---------- |
|  | DISABLED 1 9 9 9 0 |
|  |  |
|  | SELECT t.blocks, t.num\_rows, i.clustering\_factor |
|  | FROM user\_tables t, user\_indexes i |
|  | WHERE t.table\_name = i.table\_name AND i.index\_name='STUDENT\_BTREE\_INDEX'; |
|  |  |
|  | BLOCKS NUM\_ROWS CLUSTERING\_FACTOR |
|  | ---------- ---------- ----------------- |
|  | 1 |
|  | --5 |
|  | DECLARE v\_a NUMBER; |
|  | BEGIN |
|  | v\_a := 11; |
|  | WHILE v\_a < 10000 |
|  | LOOP |
|  | INSERT INTO STUDENT VALUES(v\_a,'Smith'); |
|  | v\_a := v\_a + 1; |
|  | END LOOP; |
|  | COMMIT; |
|  | END; |
|  | / |
|  |  |
|  | PL/SQL procedure successfully completed. |
|  |  |
|  | --6 |
|  | EXEC DBMS\_STATS.gather\_table\_stats('ANSHIT', 'STUDENT'); |
|  |  |
|  | SELECT |
|  | COMPRESSION, |
|  | LEAF\_BLOCKS, |
|  | Round(NUM\_ROWS/Decode(LEAF\_BLOCKS,0,1,LEAF\_BLOCKS)) "ROWS PER BLOCK", DISTINCT\_KEYS, |
|  | NUM\_ROWS,NUM\_ROWS-DISTINCT\_KEYS DUP\_ROWS |
|  | FROM |
|  | USER\_INDEXES |
|  | WHERE |
|  | INDEX\_NAME = 'STUDENT\_BTREE\_INDEX'; |
|  |  |
|  | COMPRESS LEAF\_BLOCKS ROWS PER BLOCK DISTINCT\_KEYS NUM\_ROWS DUP\_ROWS |
|  | -------- ----------- -------------- ------------- ---------- ---------- |
|  | DISABLED 36 278 10 9998 9988 |
|  |  |
|  |  |
|  | SQL> SELECT t.blocks, t.num\_rows, i.clustering\_factor |
|  | 2 FROM user\_tables t, user\_indexes i |
|  | 3 WHERE t.table\_name = i.table\_name AND i.index\_name='STUDENT\_BTREE\_INDEX'; |
|  |  |
|  | BLOCKS NUM\_ROWS CLUSTERING\_FACTOR |
|  | ---------- ---------- ----------------- |
|  | 28 9998 21 |
|  |  |
|  | --7 |
|  | DROP INDEX STUDENT\_BTREE\_INDEX; |
|  |  |
|  | Index dropped. |
|  |  |
|  | --8 |
|  | CREATE INDEX EMP\_EMPNAME\_IDX ON STUDENT(studname)COMPRESS TABLESPACE USERS; |
|  |  |
|  | Index Created. |
|  |  |
|  | --9 |
|  | SELECT |
|  | COMPRESSION, |
|  | LEAF\_BLOCKS, |
|  | Round(NUM\_ROWS/Decode(LEAF\_BLOCKS,0,1,LEAF\_BLOCKS)) "ROWS PER BLOCK", DISTINCT\_KEYS, |
|  | NUM\_ROWS,NUM\_ROWS-DISTINCT\_KEYS DUP\_ROWS |
|  | FROM |
|  | USER\_INDEXES |
|  | WHERE |
|  | INDEX\_NAME = 'STUDNAME\_IDX'; |
|  |  |
|  | COMPRESS LEAF\_BLOCKS ROWS PER BLOCK DISTINCT\_KEYS NUM\_ROWS DUP\_ROWS |
|  | -------- ----------- -------------- ------------- ---------- ---------- |
|  | ENABLED 16 625 10 9998 9988 |
|  |  |
|  | SELECT t.blocks, t.num\_rows, i.clustering\_factor |
|  | FROM user\_tables t, user\_indexes i |
|  | WHERE t.table\_name = i.table\_name AND i.index\_name='STUDNAME\_IDX'; |
|  |  |
|  | BLOCKS NUM\_ROWS CLUSTERING\_FACTOR |
|  | ---------- ---------- ----------------- |
|  | 28 9998 21 |
|  |  |
|  | --10 |
|  |  |
|  | When compression is enabled, less leaf blocks are created and |
|  | rows per block gets increased. |
|  |  |
|  | --FUNCTION BASED INDEX |
|  | /\*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. |
|  | 2. Create function based index on employee name for Upper and lower function. |
|  | 3. Create user table with attributes (UserId, UserName, Gender) |
|  | 4. Insert 10000 records in user table |
|  | 5. Build regular index on Username |
|  | 6. Build function based index on user name based on Upper function |
|  | 7. Compare the response time and comment. |
|  | \*/ |
|  |  |
|  |  |
|  | --1 |
|  | CREATE TABLE HR AS ( SELECT \* FROM HR.EMPLOYEES); |
|  | SQL> CREATE TABLE HR AS ( SELECT \* FROM HR.EMPLOYEES); |
|  |  |
|  | Table created. |
|  |  |
|  | Elapsed: 00:00:00.13 |
|  |  |
|  | CREATE INDEX INDEX\_FBI\_HR ON HR(COMMISSION\_PCT\*SALARY); |
|  | SQL> CREATE INDEX INDEX\_FBI\_HR ON HR(COMMISSION\_PCT\*SALARY); |
|  |  |
|  | Index created. |
|  |  |
|  | Elapsed: 00:00:00.03 |
|  |  |
|  |  |
|  | SELECT \* FROM HR |
|  | WHERE SALARY\*COMMISSION\_PCT < 50000; |
|  |  |
|  | SQL> SELECT \* FROM HR WHERE (COMMISSION\_PCT\*SALARY) <50000; |
|  |  |
|  | EMPLOYEE\_ID FIRST\_NAME LAST\_NAME EMAIL PHONE\_NUMBER HIRE\_DATE JOB\_ID SALARY COMMISSION\_PCT MANAGER\_ID DEPARTMENT\_ID |
|  | ----------- -------------------- ------------------------- ------------------------- -------------------- --------- ---------- ---------- -------------- ---------- ------------- |
|  | 173 Sundita Kumar SKUMAR 011.44.1343.329268 21-APR-08 SA\_REP 6100 .1 148 |
|  | 80 |
|  | 167 Amit Banda ABANDA 011.44.1346.729268 21-APR-08 SA\_REP 6200 .1 147 |
|  | 80 |
|  | 179 Charles Johnson CJOHNSON 011.44.1644.429262 04-JAN-08 SA\_REP 6200 .1 149 |
|  | 80 |
|  | 166 Sundar Ande SANDE 011.44.1346.629268 24-MAR-08 SA\_REP 6400 .1 147 |
|  | 80 |
|  | 165 David Lee DLEE 011.44.1346.529268 23-FEB-08 SA\_REP 6800 .1 147 |
|  | 80 |
|  | 164 Mattea Marvins MMARVINS 011.44.1346.329268 24-JAN-08 SA\_REP 7200 .1 147 |
|  | 80 |
|  | 155 Oliver Tuvault OTUVAULT 011.44.1344.486508 23-NOV-07 SA\_REP 7000 .15 145 |
|  | 80 |
|  | 178 Kimberely Grant KGRANT 011.44.1644.429263 24-MAY-07 SA\_REP 7000 .15 149 |
|  | 172 Elizabeth Bates EBATES 011.44.1343.529268 24-MAR-07 SA\_REP 7300 .15 148 |
|  | 80 |
|  | 171 William Smith WSMITH 011.44.1343.629268 23-FEB-07 SA\_REP 7400 .15 148 |
|  | 80 |
|  | 163 Danielle Greene DGREENE 011.44.1346.229268 19-MAR-07 SA\_REP 9500 .15 147 |
|  | 80 |
|  | 154 Nanette Cambrault NCAMBRAU 011.44.1344.987668 09-DEC-06 SA\_REP 7500 .2 145 |
|  | 80 |
|  | 153 Christopher Olsen COLSEN 011.44.1344.498718 30-MAR-06 SA\_REP 8000 .2 145 |
|  | 80 |
|  | 177 Jack Livingston JLIVINGS 011.44.1644.429264 23-APR-06 SA\_REP 8400 .2 149 |
|  | 80 |
|  | 176 Jonathon Taylor JTAYLOR 011.44.1644.429265 24-MAR-06 SA\_REP 8600 .2 149 |
|  | 80 |
|  | 161 Sarath Sewall SSEWALL 011.44.1345.529268 03-NOV-06 SA\_REP 7000 .25 146 |
|  | 80 |
|  | 170 Tayler Fox TFOX 011.44.1343.729268 24-JAN-06 SA\_REP 9600 .2 148 |
|  | 80 |
|  | 169 Harrison Bloom HBLOOM 011.44.1343.829268 23-MAR-06 SA\_REP 10000 .2 148 |
|  | 80 |
|  | 149 Eleni Zlotkey EZLOTKEY 011.44.1344.429018 29-JAN-08 SA\_MAN 10500 .2 100 |
|  | 80 |
|  | 175 Alyssa Hutton AHUTTON 011.44.1644.429266 19-MAR-05 SA\_REP 8800 .25 149 |
|  | 80 |
|  | 152 Peter Hall PHALL 011.44.1344.478968 20-AUG-05 SA\_REP 9000 .25 145 |
|  | 80 |
|  | 160 Louise Doran LDORAN 011.44.1345.629268 15-DEC-05 SA\_REP 7500 .3 146 |
|  | 80 |
|  | 151 David Bernstein DBERNSTE 011.44.1344.345268 24-MAR-05 SA\_REP 9500 .25 145 |
|  | 80 |
|  | 159 Lindsey Smith LSMITH 011.44.1345.729268 10-MAR-05 SA\_REP 8000 .3 146 |
|  | 80 |
|  | 162 Clara Vishney CVISHNEY 011.44.1346.129268 11-NOV-05 SA\_REP 10500 .25 147 |
|  | 80 |
|  | 168 Lisa Ozer LOZER 011.44.1343.929268 11-MAR-05 SA\_REP 11500 .25 148 |
|  | 80 |
|  | 150 Peter Tucker PTUCKER 011.44.1344.129268 30-JAN-05 SA\_REP 10000 .3 145 |
|  | 80 |
|  | 158 Allan McEwen AMCEWEN 011.44.1345.829268 01-AUG-04 SA\_REP 9000 .35 146 |
|  | 80 |
|  | 148 Gerald Cambrault GCAMBRAU 011.44.1344.619268 15-OCT-07 SA\_MAN 11000 .3 100 |
|  | 80 |
|  | 174 Ellen Abel EABEL 011.44.1644.429267 11-MAY-04 SA\_REP 11000 .3 149 |
|  | 80 |
|  | 157 Patrick Sully PSULLY 011.44.1345.929268 04-MAR-04 SA\_REP 9500 .35 146 |
|  | 80 |
|  | 156 Janette King JKING 011.44.1345.429268 30-JAN-04 SA\_REP 10000 .35 146 |
|  | 80 |
|  | 147 Alberto Errazuriz AERRAZUR 011.44.1344.429278 10-MAR-05 SA\_MAN 12000 .3 100 |
|  | 80 |
|  | 146 Karen Partners KPARTNER 011.44.1344.467268 05-JAN-05 SA\_MAN 13500 .3 100 |
|  | 80 |
|  | 145 John Russell JRUSSEL 011.44.1344.429268 01-OCT-04 SA\_MAN 14000 .4 100 |
|  | 80 |
|  |  |
|  | 35 rows selected. |
|  |  |
|  | Elapsed: 00:00:00.05 |
|  |  |
|  | --2 |
|  |  |
|  | CREATE INDEX EMPNAME\_INDEX ON HR(UPPER(FIRST\_NAME) || LOWER(LAST\_NAME)); |
|  | SQL> CREATE INDEX EMPNAME\_INDEX ON HR(UPPER(FIRST\_NAME) || LOWER(LAST\_NAME)); |
|  |  |
|  | Index created. |
|  |  |
|  | Elapsed: 00:00:00.09 |
|  |  |
|  | --3 |
|  |  |
|  | CREATE TABLE user\_data ( |
|  | userid NUMBER(10) NOT NULL, |
|  | username VARCHAR2(40) NOT NULL, |
|  | gender VARCHAR2(1) |
|  | ); |
|  |  |
|  | Table created. |
|  |  |
|  | Elapsed: 00:00:00.31 |
|  |  |
|  | --4 |
|  | BEGIN |
|  | FOR userid IN 1 .. 100000 LOOP |
|  | IF MOD(userid, 2) = 0 THEN |
|  | INSERT INTO user\_data |
|  | VALUES (userid, 'John', 'M'); |
|  | ELSE |
|  | INSERT INTO user\_data |
|  | VALUES (userid, 'Jayne', 'F'); |
|  | END IF; |
|  | COMMIT; |
|  | END LOOP; |
|  | END; |
|  | / |
|  |  |
|  | PL/SQL procedure successfully completed. |
|  |  |
|  | Elapsed: 00:00:19.29 |
|  |  |
|  | --5 |
|  |  |
|  | SQL> CREATE INDEX INDEX\_REGULAR ON USER\_DATA(USERNAME); |
|  |  |
|  | Index created. |
|  |  |
|  | \*\*\*\*\*\*\*\*\* |
|  | Elapsed: 00:00:00.67 |
|  | \*\*\*\*\*\*\*\*\* |
|  | SELECT COUNT(\*) FROM USER\_DATA; |
|  | Elapsed: 00:00:00.67 |
|  | SQL> |
|  |  |
|  | --6 |
|  | CREATE INDEX INDEX\_FBI\_USERNAME ON USER\_DATA(UPPER(USERNAME)); |
|  |  |
|  | SQL> CREATE INDEX INDEX\_FBI\_USERNAME ON USER\_DATA(UPPER(USERNAME)); |
|  |  |
|  | Index created. |
|  |  |
|  | SQL> SELECT COUNT(\*) FROM USER\_DATA; |
|  |  |
|  | COUNT(\*) |
|  | ---------- |
|  | 100000 |
|  |  |
|  | Elapsed: 00:00:00.10 |
|  |  |
|  | SQL> |
|  |  |
|  |  |
|  | --7 |
|  | Function based index gives a faster retrieval than normal BTree index. |
|  |  |
|  | --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. |
|  | 2. Create a Index Organized Table(IOT) emp\_iot based on hr.employees |
|  | 3. Create a Index Organized Table(IOT) emp101\_emp based on hr.employees. Place the |
|  | column hiredate in overflow area. |
|  | 4. Compare the timings of executing select all from employees,emp\_iot, and emp101\_iot. |
|  | Comment on your observations.\*/ |
|  |  |
|  |  |
|  | --1 |
|  | CREATE TABLESPACE LOOKUPS DATAFILE 'C:\Users\admin\Desktop\DWM' SIZE 10M; |
|  | SQL> CREATE TABLESPACE LOOKUPTEST DATAFILE 'C:\Users\admin\Desktop\DWM\lookuptest.dbf' SIZE 10M; |
|  |  |
|  | Tablespace created. |
|  |  |
|  | Elapsed: 00:00:01.51 |
|  | SQL> |
|  |  |
|  | CREATE TABLE IOT\_LOOKUPS( |
|  | lookup\_code NUMBER(10), |
|  | lookup\_value NUMBER(10), |
|  | lookup\_description VARCHAR2(40), |
|  | CONSTRAINT LOOKUP\_PK PRIMARY KEY(LOOKUP\_CODE)) |
|  | ORGANIZATION INDEX |
|  | PCTTHRESHOLD 20 |
|  | INCLUDING LOOKUP\_DESCRIPTION |
|  | OVERFLOW TABLESPACE LOOKUPTEST; |
|  |  |
|  | Table created. |
|  |  |
|  | BEGIN |
|  | FOR lookup\_code IN 1 .. 100000 LOOP |
|  | IF MOD(lookup\_code, 2) = 0 THEN |
|  | INSERT INTO IOT\_LOOKUPS |
|  | VALUES (lookup\_code, lookup\_code+1, 'String'); |
|  | ELSE |
|  | INSERT INTO IOT\_LOOKUPS |
|  | VALUES (lookup\_code, lookup\_code+2, 'String\_2'); |
|  | END IF; |
|  | COMMIT; |
|  | END LOOP; |
|  | END; |
|  | / |
|  |  |
|  | PL/SQL procedure successfully completed. |
|  |  |
|  | Elapsed: 00:00:06.98 |
|  |  |
|  | --2 |
|  |  |
|  | CREATE TABLE EMP\_IOT |
|  | (EMP\_NO NUMBER, |
|  | EMP\_NAME VARCHAR2(20), |
|  | EMP\_DEPT NUMBER, |
|  | EMP\_ADDRESS VARCHAR2(500), |
|  | EMP\_HIST VARCHAR2(1000), |
|  | CONSTRAINT EMP\_PK PRIMARY KEY(EMP\_NO)) |
|  | ORGANIZATION INDEX |
|  | INCLUDING EMP\_NAME |
|  | TABLESPACE LOOKUP\_DESCRIPTION |
|  | OVERFLOW TABLESPACE LOOKUPTEST; |
|  |  |
|  | Tablespace Created. |
|  |  |
|  | BEGIN |
|  | FOR EMP\_NO IN 1 .. 100000 LOOP |
|  | IF MOD(EMP\_NO, 2) = 0 THEN |
|  | INSERT INTO EMP\_IOT |
|  | VALUES (EMP\_NO,'SAM', 101, 'MYADDRESS', 'OLDTEXT'); |
|  | ELSE |
|  | INSERT INTO EMP\_IOT |
|  | VALUES (EMP\_NO,'JAM', 501, 'MYADDRESS', 'OLDTEXT'); |
|  | END IF; |
|  | COMMIT; |
|  | END LOOP; |
|  | END; |
|  | / |
|  |  |
|  | PL/SQL procedure successfully completed. |
|  |  |
|  | Elapsed: 00:00:13.96 |
|  | --3 |
|  |  |
|  | CREATE TABLE EMP\_IOT101 |
|  | (EMP\_NO NUMBER, |
|  | EMP\_NAME VARCHAR2(20), |
|  | EMP\_DEPT NUMBER, |
|  | EMP\_ADDRESS VARCHAR2(500), |
|  | EMP\_HIST VARCHAR2(1000), |
|  | HIREDATE DATE, |
|  | CONSTRAINT EMP\_PUK PRIMARY KEY(EMP\_NO)) |
|  | ORGANIZATION INDEX |
|  | INCLUDING HIREDATE |
|  | TABLESPACE LOOKUPTEST |
|  | OVERFLOW TABLESPACE LOOKUPTEST; |
|  |  |
|  | Table created. |
|  |  |
|  | BEGIN |
|  | FOR EMP\_NO IN 1 .. 100000 LOOP |
|  | IF MOD(EMP\_NO, 2) = 0 THEN |
|  | INSERT INTO EMP\_IOT101 |
|  | VALUES (EMP\_NO,'SAM', 101, 'MYADDRESS', 'OLDTEXT', '1-4-2005'); |
|  | ELSE |
|  | INSERT INTO EMP\_IOT101 |
|  | VALUES (EMP\_NO,'JAM', 501, 'MYADDRESS', 'OLDTEXT','1-4-2005'); |
|  | END IF; |
|  | COMMIT; |
|  | END LOOP; |
|  | END; |
|  | / |
|  |  |
|  | PL/SQL procedure successfully completed. |
|  |  |
|  | Elapsed: 00:00:20.16 |
|  |  |