CSCI317 – Report4 - Samuel Ian Black – SIB979 - 6025821

Table of Contents

CSCI317 – Report4 - Samuel Ian Black – SIB979 - 6025821

Note that before each run of a command time java task5 the below query was run.

```
CONNECT SYSTEM/oracle;
ALTER SYSTEM FLUSH SHARED_POOL;
```

Original time of query:

```
[oracle@localhost Exam]$ time java task5

Connected as CSCI317.

Total: 300

Done.


real 0m4.325s

user 0m3.720s

sys  0m0.890s
```

Total time = 4.325 + 3.720 + 0.890 = **8.935s**

CSCI317 – Report4 - Samuel Ian Black – SIB979 - 6025821

Optimization #1 – Transformation of Select Statement

(1) Description of Improvement:

Removal of unnecessary outer while loop by using a prepared statement and transformation of select statement.

```
String query = "SELECT P_NAME " +
                           "FROM PART " +
                           "WHERE P_NAME = ?" ;


PreparedStatement stmt = conn.prepareStatement (query);


BufferedReader in = new BufferedReader(new
FileReader("task5.txt"));
      String str;
      while ( ( str = in.readLine() ) != null ) {
            stmt.setString(1, str);
            ResultSet rset = stmt.executeQuery();
            while (rset.next()) {
                count ++;
            }
       }
```

(2) Benefits of Improvement:

The original solution had an extra loop which iterated through every single row of the table `PART`, and then a nested loop which iterated through the entire task5.txt file for each row in `PART`.

This was causing m * n operations to be performed where m is the number of rows in the table `PART` and n is the number of lines in the task5.txt file.

The optimized solution performs n iterations where n is the number of lines in the task5.txt file and uses a prepared statement to query the database which avoids flooding the data buffer cache unnecessarily.

```
[oracle@localhost Exam]$ time java task5
Connected as CSCI317.
Total: 300
```

```
Done.


real 0m1.560s

user 0m1.451s

sys  0m0.072s
```

real before improvement = **4.325s**
real improvement = 4.325 – 1.560 = **2.765s**

user before improvement = **3.720s**
user improvement = 3.720 – 1.451 = **2.269s**

sys before improvement = **0.890s**
sys improvement = 0.890 – 0.072 = **0.818s**

Total time = 1.560 + 1.451 + 0.072 = **3.083s**
Total time improvement = 8.935 – 3.083 = **5.852s**

(3) Costs of Improvement:

There is no cost associated with this improvement.

(4) Report from improvement:

```
import java.sql.*;
import java.io.*;

class task5
{
  public static void main (String args [])
       throws SQLException, ClassNotFoundException
  {
    // Load the Oracle JDBC driver
    Class.forName ("oracle.jdbc.driver.OracleDriver");
    Connection conn = DriverManager.getConnection
       ("jdbc:oracle:thin:@localhost:1521:db",  "tpchr", "oracle");
      System.out.println( "Connected as CSCI317." );
  try{
     int count = 0;

          String query = "SELECT P_NAME " +
                         "FROM PART " +
                         "WHERE P_NAME = ?" ;
```

```
        PreparedStatement stmt = conn.prepareStatement (query);

        BufferedReader in = new BufferedReader( new FileReader("task5.txt") );
        String str;
        while ( ( str = in.readLine() ) != null )
        {
          stmt.setString(1, str);
          ResultSet rset = stmt.executeQuery();
          while (rset.next()) {
              count ++;
          }
        }
        in.close();

      System.out.println( "Total: " + count );
      System.out.println( "Done." );
  }
catch (SQLException e)
{
```

```
      String errmsg = e.getMessage();
      System.out.println( errmsg );
    }
    catch (IOException io )
    {
      String errmsg = io.getMessage();
      System.out.println( errmsg );
    }

  }
}
```

CSCI317 – Report4 - Samuel Ian Black – SIB979 - 6025821

Optimization #2 – Creation of B*Tree Index

(1) Description of Improvement:
Creation of an index on `P_NAME`.

```
CREATE INDEX TASK5IDX1 ON PART(P_NAME);
```

(2) Benefits of Improvement:
The creation of the index allows for the query optimizer to traverse the index vertically rather than having to access the entire table.

Time

```
[oracle@localhost Exam]$ time java task5
Connected as CSCI317.
Total: 300
Done.

real 0m0.956s
user 0m1.348s
sys  0m0.063s
```

real before improvement = 1.560s
real improvement = 1.560 – 0.956 = **0.604s**

user before improvement = 1.451s
user improvement = 1.451 – 1.348 = **0.103s**

sys before improvement = 0.072s
sys improvement = 0.072 – 0.063 = **0.009s**

Total time = 0.956 + 1.348 + 0.063 = **2.367s**
Total time improvement = 3.083 – 2.367 = **0.716s**

Explain Plan Before Creation of Index

```
SQL> SET FEEDBACK ON

SQL> SET LINESIZE 300

SQL> SET PAGESIZE 300

SQL>

SQL> EXPLAIN PLAN FOR
  2  SELECT P_NAME
  3  FROM PART
  4  WHERE P_NAME = 'turquoise yellow magenta burnished peach';


Explained.


SQL>

SQL> @showplan
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------

Plan hash value: 673417232
```

```
-------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |    12 |   348 |   401   (1)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| PART |    12 |   348 |   401   (1)| 00:00:01 |
-------------------------------------------------------------------------

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

   1 - filter("P_NAME"='turquoise yellow magenta burnished peach')
```

Explain Plan After Creation of Index

```
SQL> SET ECHO ON
SQL> SET FEEDBACK ON
SQL> SET LINESIZE 300
SQL> SET PAGESIZE 300
SQL>
SQL> CREATE INDEX TASK5IDX1 ON PART(P_NAME);
```

```
Index TASK5IDX1 created.


SQL>
SQL> EXPLAIN PLAN FOR
  2  SELECT P_NAME
  3  FROM PART
  4  WHERE P_NAME = 'turquoise yellow magenta burnished peach';

Explained.


SQL>
SQL> @showplan
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 984302744
```

```
-------------------------------------------------------------------------------
| Id  | Operation         | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |           |     1 |    29 |     3   (0)| 00:00:01 |
|*  1 |  INDEX RANGE SCAN | TASK5IDX1 |     1 |    29 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------

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

   1 - access("P_NAME"='turquoise yellow magenta burnished peach')
```

Total Cost before improvement = 401 + 401 = **802**

Total Cost after improvement = 3 + 3 = **6**

Total Cost improvement = 802 – 6 = **796**

(3) Costs of Improvement:

The cost of creating the index is 3.25MB in persistent storage.

```
SQL> select sum(bytes)/1024/1024 as "Index Size (MB)" from dba_segments where
segment_name='&INDEX_NAME';

old:select sum(bytes)/1024/1024 as "Index Size (MB)" from dba_segments where
segment_name='&INDEX_NAME'

new:select sum(bytes)/1024/1024 as "Index Size (MB)" from dba_segments where
segment_name='TASK5IDX1'


Index Size (MB)
---------------
           3.25
```

(4) Report from improvement:

```
SQL> SET ECHO ON
SQL> SET FEEDBACK ON
SQL> SET LINESIZE 300
SQL> SET PAGESIZE 300
SQL>
SQL> CREATE INDEX TASK5IDX1 ON PART(P_NAME);

Index TASK5IDX1 created.

SQL>
SQL> SELECT P_NAME
  2  FROM PART
  3  WHERE P_NAME = 'turquoise yellow magenta burnished peach';

P_NAME
----------------------------------------------------------
turquoise yellow magenta burnished peach
```

```
1 row selected.
```

CSCI317 – Report4 - Samuel Ian Black – SIB979 - 6025821

Optimization #3 –Rebuild of TASK5IDX1 In INDEX_TS_32K Tablespace

(1) Description of Improvement:
Because `TASK5IDX1` is a B*Tree index, I can take full advantage of storing it in the 32k data buffer cache which will better balance the tree and reduce physical and logical reads of the query due to “flattening” the tree over larger data blocks.

ALTER INDEX TASK5IDX1 REBUILD TABLESPACE INDEX_TS_32K;

(2) Benefits of Improvement:
The B*Tree is now more balanced which results in significantly less data blocks needed to be read resulting in less physical reads and reduced Cost of performing the query.

Time

```
[oracle@localhost Exam]$ time java task5

Connected as CSCI317.

Total: 300

Done.


real 0m0.899s

user 0m1.306s

sys  0m0.076s
```

real before improvement = 0.956s
real improvement = 0.956 – 0.899 = **0.057s**

user before improvement = 1.348s
user improvement = 1.348 – 1.306 = **0.042s**

sys before improvement = 0.063s
sys improvement = 0.063 – 0.076 = **-0.013s**

Total time = 0.899 + 1.306 + -0.013 = **2.192s**
Total time improvement = 2.367 – 2.192 = **0.175s**

Explain Plan

```
SQL> SET ECHO ON
SQL> SET FEEDBACK ON
SQL> SET LINESIZE 300
SQL> SET PAGESIZE 300
SQL>
SQL> ALTER INDEX TASK5IDX1 REBUILD TABLESPACE INDEX_TS_32K;

Index TASK5IDX1 altered.

SQL>
SQL> EXPLAIN PLAN FOR
  2  SELECT P_NAME
  3  FROM PART
  4  WHERE P_NAME = 'turquoise yellow magenta burnished peach';

Explained.

SQL>
SQL> @showplan
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 984302744

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

```
|   0 | SELECT STATEMENT |           |     1 |    29 |     1   (0)| 00:00:01 |
|*  1 |  INDEX RANGE SCAN| TASK5IDX1 |     1 |    29 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------

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

   1 - access("P_NAME"='turquoise yellow magenta burnished peach')
```

Total Cost before improvement = 3 + 3 = **6**

Total Cost after improvement = 1 + 1 = **2**

Total Cost improvement = 6 – 2 = **4**

utlbstat/utlestat (Screenshots used because of spacing issues)

Before improvement:

TABLE_SPACE	FILE_NAME	READS	BLKS_READ	READ_TIME	WRITES	BLKS_WRT	WRITE_TIME	MEGABYTES	AVG_RT	blocks/rd
INDEX_TS_32K	/opt/oracle/oradata/DB/32k_tbs.dbf	0	0	0	0	0	0	67	0	0
SYSAUX	/opt/oracle/oradata/DB/sysaux01.dbf	0	0	0	0	0	0	692	0	0
SYSTEM	/opt/oracle/oradata/DB/system01.dbf	19	19	0	0	0	0	954	0	1
TPCHR	/opt/oracle/oradata/DB/tpchr.dbf	30	277	0	0	0	0	3146	0	9.23
UNDOTBS1	/opt/oracle/oradata/DB/undotbs01.dbf	0	0	0	0	0	0	357	0	0
USERS	/opt/oracle/oradata/DB/users01.dbf	0	0	0	0	0	0	31	0	0

TPCHR + INDEX_TS_32K READS = 30

TPCHR + INDEX_TS_32K BLKS_READ = 277

After improvement:

TABLE_SPACE	FILE_NAME	READS	BLKS_READ	READ_TIME	WRITES	BLKS_WRT	WRITE_TIME	MEGABYTES	AVG_RT	blocks/rd
INDEX_TS_32K	/opt/oracle/oradata/DB/32k_tbs.dbf	30	33	0	0	0	0	67	0	1.1
SYSAUX	/opt/oracle/oradata/DB/sysaux01.dbf	4	7	0	0	0	0	692	0	1.75
SYSTEM	/opt/oracle/oradata/DB/system01.dbf	18	18	0	0	0	0	954	0	1
TPCHR	/opt/oracle/oradata/DB/tpchr.dbf	0	0	0	0	0	0	3146	0	0
UNDOTBS1	/opt/oracle/oradata/DB/undotbs01.dbf	0	0	0	0	0	0	357	0	0
USERS	/opt/oracle/oradata/DB/users01.dbf	0	0	0	0	0	0	31	0	0

TPCHR + INDEX_TS_32K READS = 30

TPCHR + INDEX_TS_32K BLKS_READ = 33

Total READS improvement = 30 – 30 = **0**

Total BLKS_READ improvement = 277 – 33 = **244**

(3) Costs of Improvement:

Rebuilding of the index TASK5IDX1 does not add additional persistent storage, the index is simply rebuilt on the 32K Tablespace.

```
SQL> select sum(bytes)/1024/1024 as "Index Size (MB)" from dba_segments where segment_name='&INDEX_NAME';
old:select sum(bytes)/1024/1024 as "Index Size (MB)" from dba_segments where segment_name='&INDEX_NAME'
new:select sum(bytes)/1024/1024 as "Index Size (MB)" from dba_segments where segment_name='TASK5IDX1'

Index Size (MB)
---------------
           3.25
```

(4) Report from improvement:

```
SQL> SET ECHO ON
SQL> SET FEEDBACK ON
SQL> SET LINESIZE 300
SQL> SET PAGESIZE 300
SQL>
SQL> ALTER INDEX TASK5IDX1 REBUILD TABLESPACE INDEX_TS_32K;

Index TASK5IDX1 altered.

SQL>
SQL> SELECT P_NAME
  2  FROM PART
  3  WHERE P_NAME = 'turquoise yellow magenta burnished peach';

P_NAME
-----------------------------------------------------
turquoise yellow magenta burnished peach

1 row selected.
```

Buffer Cache Hit Ratio

Because I've been altering transient storage, It's important that I ensure the hit rate is at least 85% but preferably above 95%.

Source:[https://docs.oracle.com/database/121/TGDBA/tune_buffer_cache.htm#TGDBA536]

```
SQL> SELECT name, value
  2  FROM V$SYSSTAT
  3  WHERE name IN ('db block gets from cache', 'consistent gets from cache',
  4  'physical reads cache');

NAME                                                                  VALUE
---------------------------------------------------------------- ----------
db block gets from cache                                           23137
consistent gets from cache                                       6889631
physical reads cache                                              742706
```

```
Buffer Cache Hit Ratio = 1 - (('physical reads cache') / ('consistent gets from cache' + 'db block gets from cache'))
```

= 1 – ((742706) / (6889631+ 23137)) = 0.8925602595 or 89.25%

This result was obtained after running every query and every java script twice, indicating that after normal use will stabilize at a higher hit rate which is an indicator that the buffer cache has been used appropriately to optimize the Database.

CSCI317 – Report4 - Samuel Ian Black – SIB979 - 6025821

Total Costs

Persistent Storage: 87.75MB of 300MB

1. TASK1IDX1 = 6.5MB
2. TASK1IDX2 = 0.5MB
3. TASK2IDX1 = 1.75MB
4. INDEX_TS_32K = 64MB (Size of Tablespace used to calculate Persistent Storage)
    - TASK2IDX2 = 60MB
    - TASK5IDX1 3.25MB
5. TASK3IDX1 = 13.5MB
6. TASK4IDX1 = 1.5MB

Transient Memory: 88MB of 100MB

1. db_32K_cache_size = 64M
2. db_cache_size = 208
    - Originally 184 + 24 from allocated 100MB Transient Storage expansion