CSCI317 – Final Project

Student Name: Trang Nguyen

Student Number: 6166994 – Username: tttn941

Solution 5:

(1) a short description of an improvement

The unoptimized task5.java fetches PNAME of all existing parts in table PART and manually counts the number of parts whose PNAME is present in task5.txt.

To achieve the same result set, instead of retrieving all PNAMEs, including the unnecessary ones, solution5.java uses a PreparedStatement which allows for a parameterised SQL statement where PNAME is a parameter.

Then, for each line in task5.txt, solution5.java reads it into a String object using a BufferReader, then sets the string as the PNAME parameter of the SELECT statement. It then executes the query to count the number of parts with the string read from task5.txt as their names.

The result of the query – the count of satisfied parts, is then added to a counter variable which is the final count of all parts whose names match the strings from the text file.

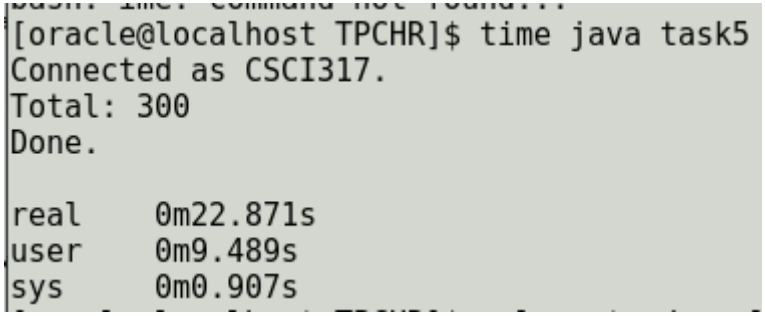
A Bitmap index on PART(PNAME) is also created server-side to speed up the query processing, as Bitmap indices are very efficient for counting.

```
CREATE BITMAP INDEX P_IDX2 ON PART(P_NAME);
```

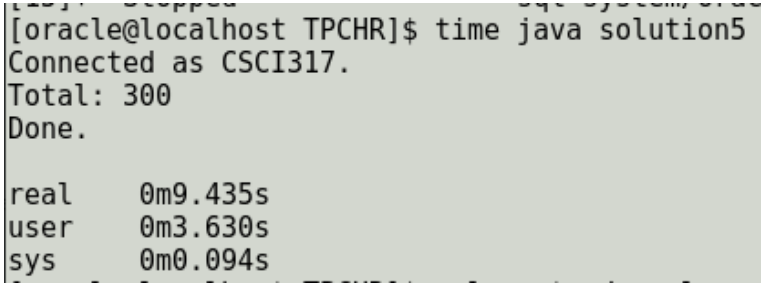
(2) information about the benefits from an improvement

- The SELECT statement uses the built-in COUNT and filter WHERE and pushes the processing of the data to the server side, which is more efficient than “reinventing the wheel” and manually iterating through the result set with nested loops.
- A PreparedStatement is so used to create a template SELECT statement so that the application does not flood the buffer cache.
- The improved statement only returns a single number for the count instead of returning everything including irrelevant information.
- For counting, a Bitmap index is a good way to speed up the query as the DBMS can traverse the index and only counts the bits (1s).

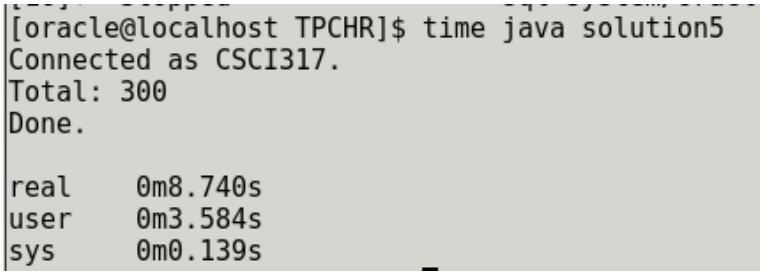
The figures below demonstrate the run time for each JDBC application:



task5.java



solution5.java without index



solution5.java with index

The query processing plans with and without the index is demonstrated below. Note the difference in the CPU Costs of the operations.

```
SQL> EXPLAIN PLAN FOR
  2  SELECT COUNT(*)
  3  FROM PART
  4  WHERE P_NAME = 'Part';

Explained.

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

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 1500225144

-----------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |     1 |    29 |   401   (1)| 00:00:01 |
|   1 |  SORT AGGREGATE    |      |     1 |    29 |            |          |
|*  2 |   TABLE ACCESS FULL| PART |    12 |   348 |   401   (1)| 00:00:01 |
-----------------------------------------------------------------------------

Predicate Information (identified by operation id):

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

   2 - filter("P_NAME"='Part')

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)
```

Query processing plan of the SELECT statement *without* index

```
SQL> EXPLAIN PLAN FOR
  2  SELECT COUNT(*)
  3  FROM PART
  4  WHERE P_NAME = 'Part';

Explained.

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

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------
Plan hash value: 1947396774

-------------------------------------------------------------------------------------------
| Id  | Operation                    | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |        |     1 |    35 |     3   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE              |        |     1 |    35 |            |          |
|   2 |   BITMAP CONVERSION COUNT    |        |     1 |    35 |     3   (0)| 00:00:01 |
|*  3 |    BITMAP INDEX SINGLE VALUE| P_IDX2 |       |       |            |          |
-------------------------------------------------------------------------------------------


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

   3 - access("P_NAME"='Part')

15 rows selected.
```

Query processing plan of the SELECT statement *with* index

(3) information about the costs of an improvement (i.e., documented investments into

transient and persistent storage)

In terms of transient storage, there is an overhead in creating a Statement and in executing it for the first time. However, it is not significant to the overall application's performance. No new investment into transient storage has been made compared to task5.java.

However, an investment of 3.75MB into persistent storage is needed for the Bitmap index on `PART(P_NAME)` to speed up the query.

	SEGMENT_NAME	Index Size (MB)
1	P_IDX2	3.75

(4) a report from implementation of an improvement

```
/* Improved JDBC application – solution5.java */
import java.sql.*;
import java.io.*;

class solution5
{
  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;
      // parameterised statement
```

```
   PreparedStatement stmt = conn.prepareStatement(
    "SELECT COUNT(*) " +
       "FROM PART " +
    "WHERE P_NAME= ? "
);

// read txt file into array of String
BufferedReader in = new BufferedReader( new FileReader("task5.txt") );
String str = null;


while ( ( str = in.readLine() ) != null )
{
    // set statement's parameter
    stmt.setString(1, str);

    ResultSet rset = stmt.executeQuery();

    while ( rset.next() )
      count += rset.getInt(1);
}
```

```
      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 );
    }
  }
}
```

```
/* INDEX TO IMPROVE THE PERFORMANCE OF THE QUERY */
SQL> CREATE BITMAP INDEX P_IDX2 ON PART(P_NAME);

INDEX P_IDX2 created.

SQL>
SQL> EXPLAIN PLAN FOR
  2  SELECT COUNT(*)
  3  FROM PART
  4  WHERE P_NAME = 'Part';

Explained.

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

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------
Plan hash value: 1947396774
```

```
-------------------------------------------------------------------------------------
| Id  | Operation                    | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |        |     1 |    29 |     2   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE              |        |     1 |    29 |            |          |
|   2 |   BITMAP CONVERSION COUNT    |        |     1 |    29 |     2   (0)| 00:00:01 |
|*  3 |    BITMAP INDEX SINGLE VALUE| P_IDX2 |       |       |            |          |
-------------------------------------------------------------------------------------

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

   3 - access("P_NAME"='Part')

Note
-----

   - dynamic statistics used: dynamic sampling (level=2)

19 rows selected.
```

```
SQL> /* Find the size of the index created above */
SQL> select segment_name,
  2      sum(bytes)/1024/1024 as "Index Size (MB)"
  3  from user_segments
  4  where segment_name='P_IDX2'
  5  group by segment_name;

SEGMENT_NAME     Index Size (MB)
------------     ---------------
P_IDX2                      3.75

1 rows selected.

SQL> DROP INDEX P_IDX2;

Index P_IDX2 dropped.
```