CSCI317 – Final Project

Student Name: Trang Nguyen

Student Number: 6166994 – Username: tttn941

Solution 3:

(1) a short description of an improvement

The query processing of task3.sql is as follows:

```
PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------------------------------------
Plan hash value: 2676312324

-------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name          | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |               |     1 |       |       |  1621   (4)| 00:00:01 |
|   1 |  SORT AGGREGATE          |               |     1 |       |       |            |          |
|   2 |   VIEW                   |               |  1944K|       |       |  1621   (4)| 00:00:01 |
|*  3 |    FILTER                |               |       |       |       |            |          |
|   4 |     HASH GROUP BY        |               |  1944K|    24M|       |  1621   (4)| 00:00:01 |
|   5 |      INDEX FAST FULL SCAN| LINEITEM_PKEY |  1944K|    24M|       |  1571   (1)| 00:00:01 |

PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------------------------------------
|   6 |     HASH UNIQUE          |               |  1944K|       |  7632K|  5295   (1)| 00:00:01 |
|   7 |      INDEX FAST FULL SCAN| LINEITEM_PKEY |  1944K|       |       |  1571   (1)| 00:00:01 |
-------------------------------------------------------------------------------------------------------

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

   3 - filter(COUNT(*)> (SELECT DISTINCT "TSIZE"() FROM "LINEITEM" "LINEITEM"))

Note
-----

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

23 rows selected.
```

Query processing plan of the SELECT statement in task3.sql

Since LINEITEM is a large table (254.5MB), table scans of LINEITEM are expensive. Although an INDEX FAST FULL SCAN is used by the query instead of accessing LINEITEM table, the index on its primary key is still large (1,944,000 rows), making the operation expensive.

We notice how the SELECT statement in task3.sql calls TSIZE function for as many times as there are rows in LINEITEM, and then it uses the function's results to filter out the result set in the outer SELECT statement. The effect of this is similar to that of a nested loop, which is very expensive for a large table like LINEITEM. This explains why the SELECT statement never ends.

Since TSIZE only returns a single number no matter what table the SELECT statement used to call the function is, the "FROM LINEITEM" clause becomes redundant. Thus, we can resolve this problem by replacing the "FROM LINEITEM" clause by "FROM DUAL", a dummy table created by the DBMS that only has one row [1], which allows for the function's return value to be selected only once. It allows for the statement to end and its performance to be improved.

The improved SELECT statement is as follows:

```
SELECT COUNT(*)
FROM ( SELECT L_ORDERKEY
       FROM LINEITEM
       GROUP BY L_ORDERKEY
       HAVING COUNT(*) > (SELECT TSIZE
                          FROM DUAL));
```

To further improve the query, a Bitmap index is created on LINEITEM(L_ORDERKEY) as Bitmap is efficient for

```
CREATE BITMAP INDEX L_IDX ON LINEITEM(L_ORDERKEY);
```

(2) information about the benefits from an improvement

- Using DUAL improves the query in that instead of calling TSIZE function as many times as there are rows in LINEITEM, DUAL allows us to call TSIZE only once as it only has one row.
- The Bitmap index speeds up the processing of COUNT, as shown in the figures below (compared to the figure in part (1)).

```
PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------
Plan hash value: 4154509521

-------------------------------------------------------------------------------------------
| Id  | Operation                          | Name  | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |       |     1 |       |  1597   (4)| 00:00:01 |
|   1 |  SORT AGGREGATE                    |       |     1 |       |            |          |
|   2 |   VIEW                             |       |  1941K|       |  1597   (4)| 00:00:01 |
|*  3 |    FILTER                          |       |       |       |            |          |
|   4 |     HASH GROUP BY                  |       |  1941K|    24M|  1597   (4)| 00:00:01 |
|   5 |      BITMAP CONVERSION TO ROWIDS   |       |  1941K|    24M|  1547   (1)| 00:00:01 |
|   6 |       BITMAP INDEX FAST FULL SCAN| L_IDX |       |       |            |          |
|   7 |     FAST DUAL                      |       |     1 |       |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------

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

   3 - filter(COUNT(*)> (SELECT "TSIZE"() FROM "SYS"."DUAL" "DUAL"))

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

23 rows selected.
```

Query processing plan of the improved solution3.sql

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

In terms of the transient storage, no further investment has been made compared to task3.sql.

However, an investment of 13.5MB was made into the Bitmap index to speed up the query as Bitmap indices are known to be efficient when it comes to counting.

	SEGMENT_NAME	Index Size (MB)
1	L_IDX	13.5

(4) a report from implementation of an improvement

```
SQL> CREATE BITMAP INDEX L_IDX ON LINEITEM(L_ORDERKEY);


INDEX L_IDX created.


SQL> CREATE OR REPLACE FUNCTION TSIZE RETURN NUMBER
  2  AS
  3  TS NUMBER;
  4  BEGIN
  5    SELECT AVG(COUNT(*))
  6    INTO TS
  7    FROM LINEITEM
  8    GROUP BY L_ORDERKEY;
  9
```

```
 10    RETURN TS;
 11  END;
 12  /

Function TSIZE compiled

SQL>
SQL> show errors
SQL>
SQL> explain plan for
  2  SELECT COUNT(*)
  3  FROM ( SELECT L_ORDERKEY
  4         FROM LINEITEM
  5         GROUP BY L_ORDERKEY
  6         HAVING COUNT(*) > (SELECT TSIZE
  7                            FROM DUAL));

Explained.
```

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

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------
Plan hash value: 4154509521

-----------------------------------------------------------------------------------------
| Id  | Operation                          | Name  | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |       |     1 |       |  1597   (4)| 00:00:01 |
|   1 |  SORT AGGREGATE                    |       |     1 |       |            |          |
|   2 |   VIEW                             |       |  1941K|       |  1597   (4)| 00:00:01 |
|*  3 |    FILTER                          |       |       |       |            |          |
|   4 |     HASH GROUP BY                  |       |  1941K|    24M|  1597   (4)| 00:00:01 |
|   5 |      BITMAP CONVERSION TO ROWIDS   |       |  1941K|    24M|  1547   (1)| 00:00:01 |
|   6 |       BITMAP INDEX FAST FULL SCAN| L_IDX |       |       |            |          |
|   7 |     FAST DUAL                      |       |     1 |       |     2   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------
```

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

   3 - filter(COUNT(*)> (SELECT "TSIZE"() FROM "SYS"."DUAL" "DUAL"))

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

23 rows selected.

SQL> /* Retrieve index's size */
SQL> select segment_name,
  2      sum(bytes)/1024/1024 as "Index Size (MB)"
  3  from user_segments
  4  where segment_name='L_IDX'
  5  group by segment_name;

SEGMENT_NAME     Index Size (MB)
------------     ---------------
L_IDX                      13.5
```

```
1 rows selected.


SQL>
SQL> DROP FUNCTION TSIZE;

Function TSIZE dropped.

SQL> DROP INDEX L_IDX;

Index L_IDX dropped.
```

Reference

[1] Docs.oracle.com. 2021. *Selecting from the DUAL Table*. [online] Available at: <https://docs.oracle.com/cd/B19306_01/server.102/b14200/queries009.htm> [Accessed 18 June 2021].