# CSCI317 – Report3 - Samuel Ian Black – SIB979 - 6025821

## Table of Contents

# CSCI317 – Report3 - Samuel Ian Black – SIB979 - 6025821

## Original cost of query

N/A will not return result/s due to nested loops

# CSCI317 – Report3 - Samuel Ian Black – SIB979 - 6025821

## Optimization #1 – Transformation of Select Statement

Note that I decided against using Materialized Views as we're unsure of whether or not the client's DB has designated downtime to mitigate the more expensive insertion, and if the client doesn't have a designated downtime for the DB to perform inserts and commits to update the materialized views and instead wants the views to remain up to date and therefore, updated on insert, if the table LINEITEM is frequently inserted to which is what I assume since It's the largest table, the performance of the entire DB will greatly suffer which is why I've decided not to use Materialized Views.

(1) Description of Improvement:
The query was previously selecting the function `TSIZE` from the table `LINEITEM`
e.g. `SELECT DISTINCT TSIZE FROM LINEITEM` which is incorrect because the TSIZE function already queries the LINEITEM table therefore, this can be solved by changing the query to: `SELECT TSIZE FROM DUAL.`

(2) Benefits of Improvement:
The query will now end and return the correct result.

```
SQL> SET ECHO ON
SQL> SET FEEDBACK ON
SQL> SET LINESIZE 300
SQL> SET PAGESIZE 300
SQL>
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, COUNT(*)
  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: 4190422380

-------------------------------------------------------------------------------
| Id  | Operation               | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT        |               |     1 |       |  1621   (4)| 00:00:01 |
|   1 |  SORT AGGREGATE         |               |     1 |       |            |          |
|   2 |   VIEW                  |               |  1940K|       |  1621   (4)| 00:00:01 |
```

```
|*  3 |    FILTER                |                |       |       |            |          |
|   4 |     HASH GROUP BY        |                |  1940K|    24M|  1621   (4)| 00:00:01 |
|   5 |      INDEX FAST FULL SCAN| LINEITEM_PKEY  |  1940K|    24M|  1571   (1)| 00:00:01 |
|   6 |     FAST DUAL            |                |     1 |       |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------


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


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

Total Cost before improvement = **N/A**

Total Cost after improvement = 1,621 + 1,621 + 1,621 + 1,571 + 2 = **6,436**

Total Cost improvement = **N/A**

(3) Costs of Improvement:
There is no cost associated with this improvement.

(4) Report from improvement:

```
SQL> SET ECHO ON
SQL> SET FEEDBACK ON
SQL> SET LINESIZE 300
SQL> SET PAGESIZE 300
SQL>
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> SELECT COUNT(*)
  2  FROM ( SELECT L_ORDERKEY, COUNT(*)
  3         FROM LINEITEM
  4         GROUP BY L_ORDERKEY
  5         HAVING COUNT(*) > (SELECT TSIZE
  6                            FROM DUAL) );

  COUNT(*)
----------
    193031

1 row selected.
```

# CSCI317 – Report3 - Samuel Ian Black – SIB979 - 6025821

## Optimization #2 – Creation of Bitmap Index

(1) Description of Improvement:

I created a bitmap index over the `L_ORDERKEY` column;
The total number of rows in a table `LINEITEM` is 1,800,093
the total number of distinct rows of a column `L_ORDERKEY` is 450,000
meaning that the cardinality of `L_ORDERKEY` = (450,000/1,800,093) * 100 = 24.99% making the `L_ORDERKEY` a reasonable column for a bitmap to be created.

```
CREATE BITMAP INDEX TASK3IDX1 ON ORDERS(O_ORDERDATE);
```

(2) Benefits of Improvement:

The query now scans the Bitmap index `TASK3IDX1` rather than a primary key which is slightly faster.

```
SQL> SET ECHO ON
SQL> SET FEEDBACK ON
SQL> SET LINESIZE 300
SQL> SET PAGESIZE 300
SQL>
SQL> CREATE BITMAP INDEX TASK3IDX1 ON LINEITEM(L_ORDERKEY);

INDEX TASK3IDX1 created.
```

```
SQL>
SQL> EXPLAIN PLAN FOR
  2  SELECT COUNT(*)
  3  FROM ( SELECT L_ORDERKEY, COUNT(*)
  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: 2915800778
```

```
-------------------------------------------------------------------------------------------
| Id  | Operation                          | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |           |     1 |       |  1597   (4)| 00:00:01 |
|   1 |  SORT AGGREGATE                    |           |     1 |       |            |          |
|   2 |   VIEW                             |           |  1940K|       |  1597   (4)| 00:00:01 |
|*  3 |    FILTER                          |           |       |       |            |          |
|   4 |     HASH GROUP BY                  |           |  1940K|    24M|  1597   (4)| 00:00:01 |
|   5 |      BITMAP CONVERSION TO ROWIDS   |           |  1940K|    24M|  1547   (1)| 00:00:01 |
|   6 |       BITMAP INDEX FAST FULL SCAN| TASK3IDX1 |       |       |            |          |
|   7 |     FAST DUAL                      |           |     1 |       |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------

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

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

Total Cost before improvement = **6,436**
Total Cost after improvement = 1,597 + 1,597 + 1,597 + 1,547 + 2 = **6,340**

Total Cost improvement = 6,436 - 6,340 = **96**

(3) Costs of Improvement:
The cost of creating the index is 13.5MB 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='TASK3IDX1'


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

(4) Report from improvement:

```
SQL> SET ECHO ON
SQL> SET FEEDBACK ON
SQL> SET LINESIZE 300
SQL> SET PAGESIZE 300
SQL>
SQL> CREATE BITMAP INDEX TASK3IDX1 ON LINEITEM(L_ORDERKEY);

INDEX TASK3IDX1 created.

SQL>
SQL> SELECT COUNT(*)
  2  FROM ( SELECT L_ORDERKEY, COUNT(*)
  3         FROM LINEITEM
  4         GROUP BY L_ORDERKEY
  5         HAVING COUNT(*) > (SELECT TSIZE
  6                            FROM DUAL) );

  COUNT(*)
```

```
----------

    193031


1 row selected.
```

# CSCI317 – Report3 - Samuel Ian Black – SIB979 - 6025821

## Total Costs

Persistent Storage: 86.25MB 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
5. TASK3IDX1 = 13.5MB

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