CSCI317 – Final Project

Student Name: Trang Nguyen

Student Number: 6166994 – Username: tttn941

Solution 1:

(1) a short description of an improvement

ASUMPTIONS:

- I have considered using a Materialised View but since LINEITEM can be frequently updated and l_shipdate can be taken from the user input, the Materialised View would have to be frequently updated and re-compiled. Thus, I don't think using a Materialised View is a good solution in this case.
- I have also considered demoralising the database but that requires moving columns from SUPPLIER into LINEITEM. Since LINEITEM is already a large table, demoralising the database could decrease the performance of the queries rather than improving them.
- Like using a Materialised View, since LINEITEM can be frequently updated and/or inserted into, clustering the table may slow down UPDATE, INSERT and DELETE operations.
- Clustering the tables costs more than the amount of storage allowed.

However, I noticed that the VIEWS used in task1.sql are not being merged correctly (See Figure 1) so I've performed query transformation of both the query and the SELECT statement so that the Views are merged properly.

```
PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------
Plan hash value: 3365200080

-------------------------------------------------------------------------------------------------
| Id  | Operation               | Name     | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT        |          |  1817K|   204M|       | 12312   (1)| 00:00:01 |
|   1 |  MERGE JOIN             |          |  1817K|   204M|       | 12312   (1)| 00:00:01 |
|   2 |   SORT JOIN             |          |  1816K|    67M|       | 12218   (1)| 00:00:01 |
|*  3 |    VIEW                 | REVENUE  |  1816K|    67M|       | 12218   (1)| 00:00:01 |
|   4 |     WINDOW BUFFER       |          |  1816K|    83M|       | 12218   (1)| 00:00:01 |
|   5 |      HASH GROUP BY      |          |  1816K|    83M|       | 12218   (1)| 00:00:01 |
|*  6 |       FILTER            |          |       |       |       |            |          |
|*  7 |        TABLE ACCESS FULL| LINEITEM |  1816K|    83M|       | 12171   (1)| 00:00:01 |
|*  8 |   SORT JOIN             |          |  3114 |   240K|   584K|    94   (2)| 00:00:01 |
|   9 |    TABLE ACCESS FULL    | SUPPLIER |  3114 |   240K|       |    34   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------

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

   3 - filter("TOTAL_REVENUE"="ITEM_1" AND "SUPPLIER_NO">=0)
   6 - filter(TO_DATE('18-SEP-98')>=TO_DATE('28-JUN-92'))
   7 - filter("L_SHIPDATE">='28-JUN-92' AND "L_SHIPDATE"<='18-SEP-98')
   8 - access("S_SUPPKEY"="SUPPLIER_NO")
       filter("S_SUPPKEY"="SUPPLIER_NO")

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

29 rows selected.
```

Figure 1: Original query where REVENUE view is not merged correctly

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

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------
Plan hash value: 39222225

--------------------------------------------------------------------------------------------------------
| Id  | Operation                              | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                       |          |  1817K|   220M|  1825   (3)| 00:00:01 |
|*  1 |  FILTER                                |          |       |       |            |          |
|   2 |   SORT GROUP BY                        |          |  1817K|   220M|  1825   (3)| 00:00:01 |
|*  3 |    FILTER                              |          |       |       |            |          |
|*  4 |     HASH JOIN                          |          |  1817K|   220M|  1779   (1)| 00:00:01 |
|   5 |      INDEX FAST FULL SCAN              | S_IDX    |  3114 |   240K|    18   (0)| 00:00:01 |
|*  6 |      TABLE ACCESS BY INDEX ROWID BATCHED| LINEITEM |  1816K|    83M|  1756   (1)| 00:00:01 |
|   7 |       BITMAP CONVERSION TO ROWIDS      |          |       |       |            |          |
|*  8 |        BITMAP INDEX RANGE SCAN         | L_IDX    |       |       |            |          |
|   9 |   SORT AGGREGATE                       |          |     1 |    35 |  1803   (3)| 00:00:01 |
|  10 |    SORT GROUP BY                       |          |     1 |    35 |  1803   (3)| 00:00:01 |
|* 11 |     FILTER                             |          |       |       |            |          |
|  12 |      TABLE ACCESS BY INDEX ROWID BATCHED| LINEITEM |  1816K|    60M|  1756   (1)| 00:00:01 |
|  13 |       BITMAP CONVERSION TO ROWIDS      |          |       |       |            |          |
|* 14 |        BITMAP INDEX RANGE SCAN         | L_IDX    |       |       |            |          |
--------------------------------------------------------------------------------------------------------

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

   1 - filter(SUM("L_EXTENDEDPRICE"*(1-"L_DISCOUNT"))= (SELECT
              MAX(SUM("L_EXTENDEDPRICE"*(1-"L_DISCOUNT"))) FROM "TPCHR"."LINEITEM" "LINEITEM" WHERE
              TO_DATE('18-SEP-98')>=TO_DATE('28-JUN-92') AND "L_SHIPDATE"<='18-SEP-98' AND
              "L_SHIPDATE">='28-JUN-92' GROUP BY :B1))
   3 - filter(TO_DATE('18-SEP-98')>=TO_DATE('28-JUN-92'))
   4 - access("L_SUPPKEY"="S_SUPPKEY")
   6 - filter("L_SUPPKEY">=0)
   8 - access("L_SHIPDATE">='28-JUN-92' AND "L_SHIPDATE"<='18-SEP-98')
  11 - filter(TO_DATE('18-SEP-98')>=TO_DATE('28-JUN-92'))
  14 - access("L_SHIPDATE">='28-JUN-92' AND "L_SHIPDATE"<='18-SEP-98')

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

39 rows selected.
```

Figured 2: solution1.sql where REVENUE is merged correctly to the SELECT statement

The transformation is performed by predicate pushing. In this case, the predicate `" l_shipdate >= '28-JUN-92' and l_shipdate <= '18-SEP-98' "` is pushed down the syntax tree before the `JOIN` is performed. This helps reducing the size of the view REVENUE before it being joined with SUPPLIER; thus, the transformed query costs less than the original query.

The transformed query is as follows:

```
create view revenue(
        supplier_no,
        total_revenue
    ) as
    select
        l_suppkey,
        (l_extendedprice * (1 - l_discount))
    from
        lineitem
    where
        l_shipdate >= '28-JUN-92'
        and l_shipdate <= '18-SEP-98';


select
    s_suppkey,
    s_name,
    s_address,
    s_phone,
    sum(total_revenue)
from
    revenue, supplier
where
    supplier_no = s_suppkey
group by
    s_suppkey,
    s_name,
    s_address,
    s_phone
having sum(total_revenue) = (select max(sum(total_revenue)) from revenue group by s_suppkey)
order by
    s_suppkey;
```

To further improve the performance of the queries, two indices were created, including:

- A Bitmap index on LINEITEM(L_SHIPDATE) and
- A B*-Tree index on SUPPLIER(S_SUPPKEY,S_NAME,S_ADDRESS,S_PHONE)

The two indices allow the query to traverse the index which only has the necessary columns, without the need to access the entire tables LINEITEM and SUPPLIER.

```
create bitmap index L_IDX on lineitem(l_shipdate);
create index S_IDX on supplier(s_suppkey,s_name,s_address,s_phone);
```

(2) information about the benefits from an improvement

- As shown in Figure 1 and Figure 2 above, the transformation is performed by predicate pushing, which helps reducing the size of the view REVENUE before it being joined with SUPPLIER; thus, the transformed query costs less than the original query.
- The two indices further improve the performance of the SELECT statement. They allow for two index scans rather than accessing to the entire tables.

The figures below demonstrate the query processing plans with and without the two indices:

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

PLAN_TABLE_OUTPUT
-----------------------------------------------------------------------------------------------------------
Plan hash value: 39222225

-----------------------------------------------------------------------------------------------------------
| Id  | Operation                              | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                       |          |  1817K|   220M|  1825   (3)| 00:00:01 |
|*  1 |  FILTER                                |          |       |       |            |          |
|   2 |   SORT GROUP BY                        |          |  1817K|   220M|  1825   (3)| 00:00:01 |
|*  3 |    FILTER                              |          |       |       |            |          |
|*  4 |     HASH JOIN                          |          |  1817K|   220M|  1779   (1)| 00:00:01 |
|   5 |      INDEX FAST FULL SCAN              | S_IDX    |  3114 |   240K|    18   (0)| 00:00:01 |
|*  6 |      TABLE ACCESS BY INDEX ROWID BATCHED| LINEITEM |  1816K|    83M|  1756   (1)| 00:00:01 |
|   7 |       BITMAP CONVERSION TO ROWIDS      |          |       |       |            |          |
|*  8 |        BITMAP INDEX RANGE SCAN         | L_IDX    |       |       |            |          |
|   9 |   SORT AGGREGATE                       |          |     1 |    35 |  1803   (3)| 00:00:01 |
|  10 |    SORT GROUP BY                       |          |     1 |    35 |  1803   (3)| 00:00:01 |
|* 11 |     FILTER                             |          |       |       |            |          |
|  12 |      TABLE ACCESS BY INDEX ROWID BATCHED| LINEITEM |  1816K|    60M|  1756   (1)| 00:00:01 |
|  13 |       BITMAP CONVERSION TO ROWIDS      |          |       |       |            |          |
|* 14 |        BITMAP INDEX RANGE SCAN         | L_IDX    |       |       |            |          |
-----------------------------------------------------------------------------------------------------------

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

   1 - filter(SUM("L_EXTENDEDPRICE"*(1-"L_DISCOUNT"))= (SELECT
              MAX(SUM("L_EXTENDEDPRICE"*(1-"L_DISCOUNT"))) FROM "TPCHR"."LINEITEM" "LINEITEM" WHERE
              TO_DATE('18-SEP-98')>=TO_DATE('28-JUN-92') AND "L_SHIPDATE"<='18-SEP-98' AND
              "L_SHIPDATE">='28-JUN-92' GROUP BY :B1))
   3 - filter(TO_DATE('18-SEP-98')>=TO_DATE('28-JUN-92'))
   4 - access("L_SUPPKEY"="S_SUPPKEY")
   6 - filter("L_SUPPKEY">=0)
   8 - access("L_SHIPDATE">='28-JUN-92' AND "L_SHIPDATE"<='18-SEP-98')
  11 - filter(TO_DATE('18-SEP-98')>=TO_DATE('28-JUN-92'))
  14 - access("L_SHIPDATE">='28-JUN-92' AND "L_SHIPDATE"<='18-SEP-98')

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

39 rows selected.
```

The transformed query processing plan *with* the two indices

```
PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------------------
Plan hash value: 3780874396

-------------------------------------------------------------------------------------
| Id  | Operation             | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |          |  1817K|   220M| 12243   (1)| 00:00:01 |
|*  1 |  FILTER               |          |       |       |            |          |
|   2 |   SORT GROUP BY       |          |  1817K|   220M| 12243   (1)| 00:00:01 |
|*  3 |    FILTER             |          |       |       |            |          |
|*  4 |     HASH JOIN         |          |  1817K|   220M| 12197   (1)| 00:00:01 |
|   5 |      TABLE ACCESS FULL| SUPPLIER |  3114 |   240K|    34   (0)| 00:00:01 |
|*  6 |      TABLE ACCESS FULL| LINEITEM |  1816K|    83M| 12158   (1)| 00:00:01 |
|   7 |   SORT AGGREGATE      |          |     1 |    35 | 12218   (1)| 00:00:01 |
|   8 |    SORT GROUP BY      |          |     1 |    35 | 12218   (1)| 00:00:01 |
|*  9 |     FILTER            |          |       |       |            |          |
|* 10 |      TABLE ACCESS FULL| LINEITEM |  1816K|    60M| 12171   (1)| 00:00:01 |
-------------------------------------------------------------------------------------

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

   1 - filter(SUM("L_EXTENDEDPRICE"*(1-"L_DISCOUNT"))= (SELECT
              MAX(SUM("L_EXTENDEDPRICE"*(1-"L_DISCOUNT"))) FROM "TPCHR"."LINEITEM"
              "LINEITEM" WHERE TO_DATE('18-SEP-98')>=TO_DATE('28-JUN-92') AND
              "L_SHIPDATE">='28-JUN-92' AND "L_SHIPDATE"<='18-SEP-98' GROUP BY :B1))
   3 - filter(TO_DATE('18-SEP-98')>=TO_DATE('28-JUN-92'))
   4 - access("L_SUPPKEY"="S_SUPPKEY")
   6 - filter("L_SUPPKEY">=0 AND "L_SHIPDATE">='28-JUN-92' AND
              "L_SHIPDATE"<='18-SEP-98')
   9 - filter(TO_DATE('18-SEP-98')>=TO_DATE('28-JUN-92'))
  10 - filter("L_SHIPDATE">='28-JUN-92' AND "L_SHIPDATE"<='18-SEP-98')
```

The query processing plan *without* the two indices

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

In terms of transient storage, no new investment into transient storage has been made compared to task1.sql.

However, an investment of a total of 7.0MB into persistent storage is needed for the two indices:

- Bitmap index on `LINEITEM(L_SHIPDATE)`
- B*-tree index on SUPPLIER(S_SUPPKEY, S_NAME, S_ADDRESS, S_PHONE)

speed up the query.

	SEGMENT_NAME	Index Size (MB)
1	L_IDX	6.5
2	S_IDX	0.5

(4) a report from implementation of an improvement

```
SQL> /* IMPROVED STATEMENTS */
SQL> create bitmap index L_IDX on lineitem(l_shipdate);


Index L_IDX created.


SQL> create index S_IDX on supplier(s_suppkey,s_name,s_address,s_phone);


Index S_IDX created.
```

```
SQL> create view revenue(
  2          supplier_no,
  3          total_revenue
  4      ) as
  5   select
  6         l_suppkey,
  7         (l_extendedprice * (1 - l_discount))
  8   from
  9         lineitem
 10   where
 11          l_shipdate >= '28-JUN-92'
 12         and l_shipdate <= '18-SEP-98';

View REVENUE created.
```

```
SQL> explain plan for
  2  select
  3   s_suppkey,
  4   s_name,
  5   s_address,
  6   s_phone,
  7   sum(total_revenue)
  8  from
  9   revenue, supplier
 10  where
 11      supplier_no = s_suppkey
 12  group by
 13      s_suppkey,
 14      s_name,
 15   s_address,
 16   s_phone
 17  having sum(total_revenue) = (select max(sum(total_revenue)) from revenue group by s_suppkey)
 18  order by
 19      s_suppkey;

Explained.
```

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

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------
Plan hash value: 39222225

-----------------------------------------------------------------------------------------------------
| Id  | Operation                                | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |          |  1817K|   220M|  1825   (3)| 00:00:01 |
|*  1 |  FILTER                                  |          |       |       |            |          |
|   2 |   SORT GROUP BY                          |          |  1817K|   220M|  1825   (3)| 00:00:01 |
|*  3 |    FILTER                                |          |       |       |            |          |
|*  4 |     HASH JOIN                            |          |  1817K|   220M|  1779   (1)| 00:00:01 |
|   5 |      INDEX FAST FULL SCAN                | S_IDX    |  3114 |   240K|    18   (0)| 00:00:01 |
|*  6 |      TABLE ACCESS BY INDEX ROWID BATCHED| LINEITEM |  1816K|    83M|  1756   (1)| 00:00:01 |
|   7 |       BITMAP CONVERSION TO ROWIDS        |          |       |       |            |          |
|*  8 |        BITMAP INDEX RANGE SCAN           | L_IDX    |       |       |            |          |
|   9 |   SORT AGGREGATE                         |          |     1 |    35 |  1803   (3)| 00:00:01 |
```

```
|  10 |    SORT GROUP BY                          |          |     1 |    35 |  1803   (3)| 00:00:01 |
|* 11 |     FILTER                                |          |       |       |            |          |
|  12 |      TABLE ACCESS BY INDEX ROWID BATCHED| LINEITEM |  1816K|    60M|  1756   (1)| 00:00:01 |
|  13 |       BITMAP CONVERSION TO ROWIDS         |          |       |       |            |          |
|* 14 |        BITMAP INDEX RANGE SCAN            | L_IDX    |       |       |            |          |
-------------------------------------------------------------------------------------------------

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

   1 - filter(SUM("L_EXTENDEDPRICE"*(1-"L_DISCOUNT"))= (SELECT
              MAX(SUM("L_EXTENDEDPRICE"*(1-"L_DISCOUNT"))) FROM "TPCHR"."LINEITEM" "LINEITEM" WHERE
              TO_DATE('18-SEP-98')>=TO_DATE('28-JUN-92') AND "L_SHIPDATE"<='18-SEP-98' AND
              "L_SHIPDATE">='28-JUN-92' GROUP BY :B1))
   3 - filter(TO_DATE('18-SEP-98')>=TO_DATE('28-JUN-92'))
   4 - access("L_SUPPKEY"="S_SUPPKEY")
   6 - filter("L_SUPPKEY">=0)
   8 - access("L_SHIPDATE">='28-JUN-92' AND "L_SHIPDATE"<='18-SEP-98')
  11 - filter(TO_DATE('18-SEP-98')>=TO_DATE('28-JUN-92'))
  14 - access("L_SHIPDATE">='28-JUN-92' AND "L_SHIPDATE"<='18-SEP-98')
```

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

39 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      or segment_name='S_IDX'
  6  group by segment_name;

SEGMENT_NAME     Index Size (MB)
------------     ---------------
L_IDX                        6.5
S_IDX                        0.5

2 rows selected.
```

```
SQL> drop view revenue;

View REVENUE dropped.

SQL> drop index L_IDX;

Index L_IDX dropped.

SQL> drop index S_IDX;

Index S_IDX dropped.
```