CSCI317 – Report1 - Samuel Ian Black – SIB979 - 6025821

Table of Contents

Original cost of query:

```
-------------------------------------------------------------------------------------------
| Id  | Operation              | Name     | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |          |   877K|    98M|       | 12273   (1)| 00:00:01 |
|   1 |  MERGE JOIN            |          |   877K|    98M|       | 12273   (1)| 00:00:01 |
|   2 |   SORT JOIN            |          |   876K|    32M|       | 12179   (1)| 00:00:01 |
|*  3 |    VIEW                | REVENUE  |   876K|    32M|       | 12179   (1)| 00:00:01 |
|   4 |     WINDOW BUFFER      |          |   876K|    40M|       | 12179   (1)| 00:00:01 |
|   5 |      HASH GROUP BY     |          |   876K|    40M|       | 12179   (1)| 00:00:01 |

PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------------------
|*  6 |       TABLE ACCESS FULL| LINEITEM |   876K|    40M|       | 12157   (1)| 00:00:01 |
|*  7 |   SORT JOIN            |          |  3046 |   234K|   568K|    94   (2)| 00:00:01 |
|   8 |    TABLE ACCESS FULL   | SUPPLIER |  3046 |   234K|       |    34   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

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:

As seen in the origin plan for the query in row id 3, the view operation is performed indicating that the view was not successfully merged.

I performed a query transformation so that the view will successfully merge.

(2) Benefits of Improvement:

The view now successfully merges with the select query and slightly reduces the cost of the processing plan which can be seen below

```
SQL> SET ECHO ON
SQL> SET FEEDBACK ON
SQL> SET LINESIZE 300
SQL> SET PAGESIZE 300
SQL>
SQL> CREATE VIEW REVENUE( S_SUPPKEY, S_NAME, S_ADDRESS, S_PHONE, TOTAL_REVENUE ) AS
  2      SELECT S_SUPPKEY, S_NAME, S_ADDRESS, S_PHONE, L_EXTENDEDPRICE * (1 - L_DISCOUNT)
  3      FROM LINEITEM, SUPPLIER
  4      WHERE L_SUPPKEY = S_SUPPKEY
  5      AND LINEITEM.L_SHIPDATE >= '08-NOV-1995'
  6      AND LINEITEM.L_SHIPDATE <= '26-MAY-2002';

View REVENUE created.

SQL>
SQL> EXPLAIN PLAN FOR
  2  SELECT S_SUPPKEY, S_NAME, S_ADDRESS, S_PHONE, SUM(TOTAL_REVENUE) AS TOTAL_REVENUE
  3  FROM REVENUE
  4  GROUP BY S_SUPPKEY, S_NAME, S_ADDRESS, S_PHONE
  5  HAVING SUM(TOTAL_REVENUE) =
  6      (SELECT MAX(SUM(R.TOTAL_REVENUE))
  7      FROM REVENUE R
  8      GROUP BY R.S_SUPPKEY)
  9  ORDER BY S_SUPPKEY;

Explained.

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

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 3144199225

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

```
| Id  | Operation            | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |          |   859K|   104M| 12215   (1)| 00:00:01 |
|*  1 |  FILTER              |          |       |       |            |          |
|   2 |   SORT GROUP BY      |          |   859K|   104M| 12215   (1)| 00:00:01 |
|*  3 |    HASH JOIN         |          |   859K|   104M| 12194   (1)| 00:00:01 |
|   4 |     TABLE ACCESS FULL| SUPPLIER |  3109 |   239K|    34   (0)| 00:00:01 |
|*  5 |     TABLE ACCESS FULL| LINEITEM |   859K|    39M| 12157   (1)| 00:00:01 |
|   6 |   SORT AGGREGATE     |          |     1 |    48 | 12179   (1)| 00:00:01 |
|   7 |    SORT GROUP BY     |          |     1 |    48 | 12179   (1)| 00:00:01 |
|*  8 |     TABLE ACCESS FULL| LINEITEM |   859K|    39M| 12157   (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 "LINEITEM"."L_SHIPDATE">=TO_DATE(' 1995-11-08
              00:00:00', 'syyyy-mm-dd hh24:mi:ss') AND
              "LINEITEM"."L_SHIPDATE"<=TO_DATE(' 2002-05-26 00:00:00', 'syyyy-mm-dd
              hh24:mi:ss') GROUP BY "L_SUPPKEY"))
   3 - access("L_SUPPKEY"="S_SUPPKEY")
   5 - filter("LINEITEM"."L_SHIPDATE">=TO_DATE(' 1995-11-08 00:00:00',
              'syyyy-mm-dd hh24:mi:ss') AND "LINEITEM"."L_SHIPDATE"<=TO_DATE('
              2002-05-26 00:00:00', 'syyyy-mm-dd hh24:mi:ss') AND "L_SUPPKEY">=0)
   8 - filter("LINEITEM"."L_SHIPDATE">=TO_DATE(' 1995-11-08 00:00:00',
              'syyyy-mm-dd hh24:mi:ss') AND "LINEITEM"."L_SHIPDATE"<=TO_DATE('
              2002-05-26 00:00:00', 'syyyy-mm-dd hh24:mi:ss'))
```

(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 VIEW REVENUE( S_SUPPKEY, S_NAME, S_ADDRESS, S_PHONE, TOTAL_REVENUE ) AS
  2      SELECT S_SUPPKEY, S_NAME, S_ADDRESS, S_PHONE, L_EXTENDEDPRICE * (1 - L_DISCOUNT)
  3      FROM LINEITEM, SUPPLIER
  4      WHERE L_SUPPKEY = S_SUPPKEY
  5      AND LINEITEM.L_SHIPDATE >= '08-NOV-1995'
  6      AND LINEITEM.L_SHIPDATE <= '26-MAY-2002';

View REVENUE created.
```

```
SQL>
SQL> SELECT S_SUPPKEY, S_NAME, S_ADDRESS, S_PHONE, SUM(TOTAL_REVENUE) AS TOTAL_REVENUE
  2  FROM REVENUE
  3  GROUP BY S_SUPPKEY, S_NAME, S_ADDRESS, S_PHONE
  4  HAVING SUM(TOTAL_REVENUE) =
  5      (SELECT MAX(SUM(R.TOTAL_REVENUE))
  6      FROM REVENUE R
  7      GROUP BY R.S_SUPPKEY)
  8  ORDER BY S_SUPPKEY;

 S_SUPPKEY S_NAME                S_ADDRESS                    S_PHONE         TOTAL_REVENUE
---------- ------------------ ------------------------------- -------------- -------------
2993       Supplier#000002993 CDRN7azuEWTawl9G                30-541-514-5637 12267625.5

1 row selected.
```

Optimization #2 – Creation of Bitmap Index

(1) Description of Improvement:

I created a bitmap index over the `L_SHIPDATE` column;
the total number of rows in a table `LINEITEM` is 1,800,093
the total number of distinct rows of a column `L_SHIPDATE` is 2,526
meaning that the cardinality of `L_SHIPDATE =` (2,526/1,800,093) * 100 = 0.14% making the `L_SHIPDATE` a good column for a bitmap to be created.

```
CREATE BITMAP INDEX TASK1IDX1 ON LINEITEM(L_SHIPDATE);
```

(2) Benefits of Improvement:

The bitmap index significantly improves the performance of the query since the LINEITEM table is so large.
Rather than having to traverse the entire LINEITEM table, the query optimizer can instead traverse the bitmap index.

```
SQL> SET ECHO ON
SQL> SET FEEDBACK ON
SQL> SET LINESIZE 300
SQL> SET PAGESIZE 300
SQL>
SQL> CREATE BITMAP INDEX TASK1IDX1 ON LINEITEM(L_SHIPDATE);

INDEX TASK1IDX1 created.
SQL>
SQL> EXPLAIN PLAN FOR
  2  SELECT S_SUPPKEY, S_NAME, S_ADDRESS, S_PHONE, SUM(TOTAL_REVENUE) AS TOTAL_REVENUE
  3  FROM REVENUE
  4  GROUP BY S_SUPPKEY, S_NAME, S_ADDRESS, S_PHONE
  5  HAVING SUM(TOTAL_REVENUE) =
  6      (SELECT MAX(SUM(R.TOTAL_REVENUE))
  7      FROM REVENUE R
  8      GROUP BY R.S_SUPPKEY)
  9  ORDER BY S_SUPPKEY;

Explained.

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

PLAN_TABLE_OUTPUT
----------------------------------------------------------------------------------------------------
Plan hash value: 3552806918

----------------------------------------------------------------------------------------------------
| Id  | Operation                              | Name       | Rows  | Bytes | Cost  (%CPU)| Time     |
----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                       |            |   859K|   104M|  1406    (2)| 00:00:01 |
|*  1 |  FILTER                                |            |       |       |             |          |
|   2 |   SORT GROUP BY                        |            |   859K|   104M|  1406    (2)| 00:00:01 |
|*  3 |    HASH JOIN                           |            |   859K|   104M|  1385    (1)| 00:00:01 |
|   4 |     TABLE ACCESS FULL                  | SUPPLIER   |  3109 |   239K|    34    (0)| 00:00:01 |
|*  5 |     TABLE ACCESS BY INDEX ROWID BATCHED| LINEITEM  |   859K|    39M|  1348   (0)| 00:00:01 |
|   6 |      BITMAP CONVERSION TO ROWIDS       |           |       |       |            |          |
|*  7 |       BITMAP INDEX RANGE SCAN          | TASK1IDX1 |       |       |            |          |
|   8 |   SORT AGGREGATE                       |           |     1 |    48 |  1370   (2)| 00:00:01 |
|   9 |    SORT GROUP BY                       |           |     1 |    48 |  1370   (2)| 00:00:01 |
|  10 |     TABLE ACCESS BY INDEX ROWID BATCHED| LINEITEM  |   859K|    39M|  1348   (0)| 00:00:01 |
|  11 |      BITMAP CONVERSION TO ROWIDS       |           |       |       |            |          |
|* 12 |       BITMAP INDEX RANGE SCAN          | TASK1IDX1 |       |       |            |          |
-----------------------------------------------------------------------------------------------------

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
              "LINEITEM"."L_SHIPDATE"<=TO_DATE(' 2002-05-26 00:00:00', 'syyyy-mm-dd hh24:mi:ss') AND
              "LINEITEM"."L_SHIPDATE">=TO_DATE(' 1995-11-08 00:00:00', 'syyyy-mm-dd hh24:mi:ss')
GROUP BY
              "L_SUPPKEY"))
   3 - access("L_SUPPKEY"="S_SUPPKEY")
   5 - filter("L_SUPPKEY">=0)
   7 - access("LINEITEM"."L_SHIPDATE">=TO_DATE(' 1995-11-08 00:00:00', 'syyyy-mm-dd
              hh24:mi:ss') AND "LINEITEM"."L_SHIPDATE"<=TO_DATE(' 2002-05-26 00:00:00', 'syyyy-mm-dd
              hh24:mi:ss'))
  12 - access("LINEITEM"."L_SHIPDATE">=TO_DATE(' 1995-11-08 00:00:00', 'syyyy-mm-dd
              hh24:mi:ss') AND "LINEITEM"."L_SHIPDATE"<=TO_DATE(' 2002-05-26 00:00:00', 'syyyy-mm-dd
              hh24:mi:ss'))
```

(3) Costs of Improvement:

The cost of creating the index is 6.5MB 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='TASK1IDX1'


Index Size (MB)
---------------
            6.5
```

(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 TASK1IDX1 ON LINEITEM(L_SHIPDATE);

INDEX TASK1IDX1 created.
```

```
SQL>
SQL> SELECT S_SUPPKEY, S_NAME, S_ADDRESS, S_PHONE, SUM(TOTAL_REVENUE) AS TOTAL_REVENUE
  2  FROM REVENUE
  3  GROUP BY S_SUPPKEY, S_NAME, S_ADDRESS, S_PHONE
  4  HAVING SUM(TOTAL_REVENUE) =
  5      (SELECT MAX(SUM(R.TOTAL_REVENUE))
  6      FROM REVENUE R
  7      GROUP BY R.S_SUPPKEY)
  8  ORDER BY S_SUPPKEY;

S_SUPPKEY S_NAME                 S_ADDRESS                     S_PHONE         TOTAL_REVENUE
--------- ---------------------- ----------------------------- --------------- -------------
     2993 Supplier#000002993     CDRN7azuEWTawl9G              30-541-514-5637 12267625.5

1 row selected.
```

Optimization #3 – Creation of B*Tree Index

(1) Description of Improvement:

I created a B*Tree index over the `S_SUPPKEY`, `S_NAME`, `S_ADDRESS` and `S_PHONE` columns.

```
CREATE INDEX TASK1IDX2 ON SUPPLIER(S_SUPPKEY, S_NAME, S_ADDRESS, S_PHONE);
```

(2) Benefits of Improvement:

Rather than accessing the entire SUPPLIER table, the query optimizer can traverse the index vertically.

```
SQL> SET ECHO ON
SQL> SET FEEDBACK ON
SQL> SET LINESIZE 300
SQL> SET PAGESIZE 300
SQL>
SQL> CREATE INDEX TASK1IDX2 ON SUPPLIER(S_SUPPKEY, S_NAME, S_ADDRESS, S_PHONE);

Index TASK1IDX2 created.
SQL>
SQL> EXPLAIN PLAN FOR
  2  SELECT S_SUPPKEY, S_NAME, S_ADDRESS, S_PHONE, SUM(TOTAL_REVENUE) AS TOTAL_REVENUE
  3  FROM REVENUE
  4  GROUP BY S_SUPPKEY, S_NAME, S_ADDRESS, S_PHONE
  5  HAVING SUM(TOTAL_REVENUE) =
  6      (SELECT MAX(SUM(R.TOTAL_REVENUE))
  7      FROM REVENUE R
  8      GROUP BY R.S_SUPPKEY)
  9  ORDER BY S_SUPPKEY;

Explained.

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

PLAN_TABLE_OUTPUT
----------------------------------------------------------------------------------------------------
Plan hash value: 853867656

----------------------------------------------------------------------------------------------------
| Id  | Operation                       | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                |           |  859K|   104M|  1390   (2)| 00:00:01 |
|*  1 |  FILTER                         |           |      |       |            |          |
|   2 |   SORT GROUP BY                 |           |  859K|   104M|  1390   (2)| 00:00:01 |
|*  3 |    HASH JOIN                    |           |  859K|   104M|  1369   (1)| 00:00:01 |
|   4 |     INDEX FAST FULL SCAN        | TASK1IDX2 |  3109 |  239K|    18   (0)| 00:00:01 |
|*  5 |     TABLE ACCESS BY INDEX ROWID BATCHED| LINEITEM  |  859K|    39M|  1348   (0)| 00:00:01 |
|   6 |      BITMAP CONVERSION TO ROWIDS       |           |      |       |            |          |
|*  7 |       BITMAP INDEX RANGE SCAN          | TASK1IDX1 |      |       |            |          |
|   8 |   SORT AGGREGATE                      |           |     1 |    48 |  1370   (2)| 00:00:01 |
|   9 |    SORT GROUP BY                      |           |     1 |    48 |  1370   (2)| 00:00:01 |
|  10 |     TABLE ACCESS BY INDEX ROWID BATCHED| LINEITEM  |   859K|    39M|  1348   (0)| 00:00:01 |
|  11 |      BITMAP CONVERSION TO ROWIDS       |           |       |       |            |          |
|* 12 |       BITMAP INDEX RANGE SCAN          | TASK1IDX1 |       |       |            |          |
----------------------------------------------------------------------------------------------------

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
              "LINEITEM"."L_SHIPDATE"<=TO_DATE(' 2002-05-26 00:00:00', 'syyyy-mm-dd hh24:mi:ss') AND
              "LINEITEM"."L_SHIPDATE">=TO_DATE(' 1995-11-08 00:00:00', 'syyyy-mm-dd hh24:mi:ss')
GROUP BY
              "L_SUPPKEY"))
   3 - access("L_SUPPKEY"="S_SUPPKEY")
   5 - filter("L_SUPPKEY">=0)
   7 - access("LINEITEM"."L_SHIPDATE">=TO_DATE(' 1995-11-08 00:00:00', 'syyyy-mm-dd
              hh24:mi:ss') AND "LINEITEM"."L_SHIPDATE"<=TO_DATE(' 2002-05-26 00:00:00', 'syyyy-mm-dd
              hh24:mi:ss'))
  12 - access("LINEITEM"."L_SHIPDATE">=TO_DATE(' 1995-11-08 00:00:00', 'syyyy-mm-dd
              hh24:mi:ss') AND "LINEITEM"."L_SHIPDATE"<=TO_DATE(' 2002-05-26 00:00:00', 'syyyy-mm-dd
              hh24:mi:ss'))
```

(3) Costs of Improvement:

The cost of creating the index is 0.5MB 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='TASK1IDX2'


Index Size (MB)
---------------
             .5
```

(4) Report from improvement:

```
SQL> SET ECHO ON
SQL> SET FEEDBACK ON
SQL> SET LINESIZE 300
SQL> SET PAGESIZE 300
SQL>
SQL> CREATE INDEX TASK1IDX2 ON SUPPLIER(S_SUPPKEY, S_NAME, S_ADDRESS, S_PHONE);

Index TASK1IDX2 created.
SQL>
SQL> SELECT S_SUPPKEY, S_NAME, S_ADDRESS, S_PHONE, SUM(TOTAL_REVENUE) AS TOTAL_REVENUE
  2  FROM REVENUE
  3  GROUP BY S_SUPPKEY, S_NAME, S_ADDRESS, S_PHONE
  4  HAVING SUM(TOTAL_REVENUE) =
  5      (SELECT MAX(SUM(R.TOTAL_REVENUE))
  6      FROM REVENUE R
  7      GROUP BY R.S_SUPPKEY)
  8  ORDER BY S_SUPPKEY;



S_SUPPKEY S_NAME                    S_ADDRESS                   S_PHONE         TOTAL_REVENUE
---------- ------------------------- --------------------------- --------------- -------------
      2993 Supplier#000002993        CDRN7azuEWTawl9G            30-541-514-5637 12267625.5

1 row selected.
```

Optimization #4 – Allocation of Additional db_cache_size

Note that this optimization was completed after I had completed my optimizations for all 5 tasks. The other tasks were tested for read operation and query cost improvements but showed none so I have not included them in their respective reports.

(1) Description of Improvement:

After I had completed my optimizations for all 5 tasks, I ran the following query:
Source: [https://docs.oracle.com/en/database/oracle/oracle-database/21/tgdba/tuning-database-buffer-cache.html#GUID-83733109-5119-4DDB-8A81-5302CE956BE2]

```
SQL> SELECT size_for_estimate, buffers_for_estimate, estd_physical_read_factor,
  2         estd_physical_reads
  3    FROM V$DB_CACHE_ADVICE
  4   WHERE name = 'DEFAULT'
  5     AND block_size = (SELECT value FROM V$PARAMETER WHERE name = 'db_block_size')
  6     AND advice_status = 'ON';


                                                                    Estd Phys
 Cache Size (MB)      Buffers ESTD_PHYSICAL_READ_FACTOR                 Reads
---------------- ------------ ------------------------- ----------------
              16        1,952                    6.3196        1,953,987
              32        3,904                    5.2904        1,635,756
              48        5,856                    4.3002        1,329,581
              64        7,808                    3.4472        1,065,867
              80        9,760                    2.8281          874,444
              96       11,712                     2.291          708,361
             112       13,664                    1.8551          573,597
             128       15,616                    1.5287          472,669
             144       17,568                    1.2939          400,061
             160       19,520                    1.1366          351,432
             176       21,472                    1.0226          316,181
             180       21,960                         1          309,194
             192       23,424                     .9285          287,097
             208       25,376                     .8397          259,615
             224       27,328                         .7491          231,630

             240       29,280                         .6593          203,850

             256       31,232                         .5707          176,461

             272       33,184                         .4939          152,725

             288       35,136                         .4755          147,023

             304       37,088                         .4682          144,769

             320       39,040                         .4575          141,471
```

```
SQL> SHOW PARAMETER db_cache_size;

NAME          TYPE        VALUE

------------- ----------- -----

db_cache_size big integer 184M
```

By default I have 184M already in the cache, meaning I can have up to a total of 184MB + 36MB = 222MB which is leftover from the given 100MB of transient storage space. The closest I can afford is 208MB which is estimated to reduce physical reads by just over 26%.

```
SQL> ALTER SYSTEM SET db_cache_size = 208M SCOPE=SPFILE;


System SET altered.


SQL> SHOW PARAMETER db_cache_size;

NAME          TYPE        VALUE

------------- ----------- -----

db_cache_size big integer 208M
```

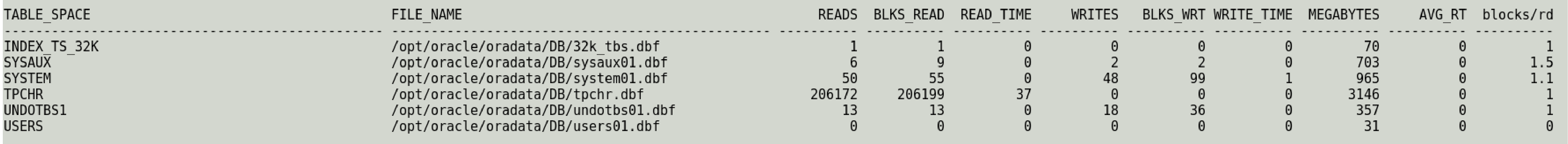
(2) Benefits of Improvement:

Allocating more transient storage to the data buffer cache significantly reduced the read operations as can be seen below

CSCI317 – Report1 - Samuel Ian Black – SIB979 - 6025821

utlbstat/utlestat (Screenshots used because of spacing issues)

Before improvement:

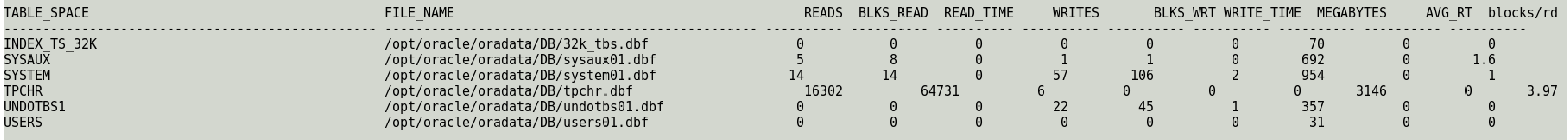


```
TABLE_SPACE                                        FILE_NAME                                                READS  BLKS_READ  READ_TIME     WRITES   BLKS_WRT WRITE_TIME  MEGABYTES     AVG_RT  blocks/rd
-------------------------------------------------- -------------------------------------------------- ---------- ---------- ---------- ---------- ---------- ---------- ---------- ---------- ----------
INDEX_TS_32K                                       /opt/oracle/oradata/DB/32k_tbs.dbf                        1          1          0          0          0          0         70          0          1
SYSAUX                                             /opt/oracle/oradata/DB/sysaux01.dbf                       6          9          0          2          2          0        703          0        1.5
SYSTEM                                             /opt/oracle/oradata/DB/system01.dbf                      50         55          0         48         99          1        965          0        1.1
TPCHR                                              /opt/oracle/oradata/DB/tpchr.dbf                     206172     206199         37          0          0          0       3146          0          1
UNDOTBS1                                           /opt/oracle/oradata/DB/undotbs01.dbf                     13         13          0         18         36          0        357          0          1
USERS                                              /opt/oracle/oradata/DB/users01.dbf                        0          0          0          0          0          0         31          0          0
```

TPCHR READS = **206,172**

TPCHR BLKS_READ = **206,199**

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                  0           0          0          0          0          0         70          0          0
SYSAUX                                             /opt/oracle/oradata/DB/sysaux01.dbf                 5           8          0          1          1          0        692          0        1.6
SYSTEM                                             /opt/oracle/oradata/DB/system01.dbf                14          14          0         57        106          2        954          0          1
TPCHR                                              /opt/oracle/oradata/DB/tpchr.dbf                     16302          64731          6          0          0          0       3146          0       3.97
UNDOTBS1                                           /opt/oracle/oradata/DB/undotbs01.dbf                0           0          0         22         45          1        357          0          0
USERS                                              /opt/oracle/oradata/DB/users01.dbf                  0           0          0          0          0          0         31          0          0
```

TPCHR READS = **16,302**

TPCHR BLKS_READ = **64,731**

Total READS improvement = 206,172 - 16,302 = **189,870**

Total BLKS_READ improvement = 206,199 - 64,731 = **141,468**

(3) Costs of Improvement:

The cost of allocating memory to the data buffer cache is 24MB of transient memory

```
SQL> SHOW PARAMETER db_cache_size;

NAME          TYPE        VALUE

------------- ----------- -----

db_cache_size big integer 208M
```

(4) Report from improvement:

```
SQL> CONNECT SYSTEM/oracle
Connected.
SQL> ALTER SYSTEM SET db_cache_size = 208M SCOPE=SPFILE;


System SET altered.


SQL> SHOW PARAMETER db_cache_size;
NAME          TYPE        VALUE
------------- ----------- -----
db_cache_size big integer 208M
SQL>
SQL> CONNECT TPCHR/oracle
Connected.
SQL> SELECT S_SUPPKEY, S_NAME, S_ADDRESS, S_PHONE, SUM(TOTAL_REVENUE) AS TOTAL_REVENUE
  2  FROM REVENUE
  3  GROUP BY S_SUPPKEY, S_NAME, S_ADDRESS, S_PHONE
  4  HAVING SUM(TOTAL_REVENUE) =
  5      (SELECT MAX(SUM(R.TOTAL_REVENUE))
  6      FROM REVENUE R
  7      GROUP BY R.S_SUPPKEY)
  8  ORDER BY S_SUPPKEY;

 S_SUPPKEY S_NAME                    S_ADDRESS                                S_PHONE         TOTAL_REVENUE
---------- ------------------------- ---------------------------------------- --------------- -------------
      2993 Supplier#000002993        CDRN7azuEWTawl9G                         30-541-514-5637    12267625.5

1 row selected.
```

CSCI317 – Report1 - Samuel Ian Black – SIB979 - 6025821

Total Costs

Persistent Storage: 7MB of 300MB

1. TASK1IDX1 = 6.5MB
2. TASK1IDX2 = 0.5MB

Transient Memory: 24MB of 100MB

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