**Final Project - Deliverable #3**

Student Name : Aditi Bhujbal

* What indexes should you create to improve the performance of the queries you wrote for deliverable #2? Why?

**Answer:** To improve performance of queries I wrote in deliverable #2, I created following indexes on tables in dw\_olap DB :

* **Indexes on SALES\_FACT table :**

**create index salesfact\_seller\_i**

**on sales\_fact(SELLERUSERID);**

**create index salesfact\_buyer\_i**

**on sales\_fact(BUYERSUSERID);**

**create index salesfact\_item\_i**

**on sales\_fact(ITEMID);**

Graphical user interface, text, application

Description automatically generated

The reason behind creating indexes on columns SELLERUSERID, BUYERSUSERID & ITEMID is that they are foreign keys that are frequently used to join sales\_fact table to seller\_dim, buyers\_dim and items\_dim respectively. As there many distinct values for all these columns, I have created a B-tree index on all the three columns.

Graphical user interface, application

Description automatically generated

* **Indexes on SELLERS\_DIM table :**

**--bitmap index on column state as it has low selectivity or very few distinct values.**

**create bitmap index sellers\_state\_i**

**on sellers\_dim(state);**

**--composite index on fname and lname as they are used together in where or group by clause.**

**create index sellers\_fname\_lname\_i**

**on sellers\_dim(fname, lname);**

**Graphical user interface, text, application, email

Description automatically generated**

The reason behind creating bitmap index on column STATE of table sellers\_dim is that there are very few distinct values in it and is used in where clause of query to compare sellers’ state with buyers’ state to answer first question in deliverable #2.

As I had used fname and lname in result and in group by clause of query to find top seller, I created a composite index on fname and lname columns in sellers\_dim table.

Graphical user interface, application

Description automatically generated

* **Indexes on BUYERS\_DIM table :**

**--bitmap index on column state as it has low selectivity or very few distinct values.**

**create bitmap index buyers\_state\_i**

**on buyers\_dim(state);**

**--composite index on fname and lname as they are used together in where or group by clause.**

**create index buyers\_fname\_lname\_i**

**on buyers\_dim(fname, lname);**

Graphical user interface, text, application, email

Description automatically generated

The reason behind creating bitmap index on column STATE of table buyers\_dim is same as that of STATE column in sellers\_dim table. There are very few distinct values in STATE column of buyers\_dim and is used in where clause of query to compare buyers’ state with sellers’ state to answer first question in deliverable #2.

As I had used fname and lname in result and in group by clause of query to find top seller, I created a composite index on fname and lname columns in sellers\_dim table.

Graphical user interface, table

Description automatically generated

* **Indexes on ITEMS\_DIM table :**

Only itemid is used in where or group clause of queries in deliverable #2, and column itemid already has an index created on it as it is a primary key of column. Hence, I did not have to create any new index on this table.

Table, Word

Description automatically generated

* Can you rewrite the queries you wrote for deliverable #3 to improve their performance? Why or why not?

**Answer:** Some of the queries that I wrote for deliverable #2 can be rewritten to improve performance, all the queries are tested with “explain plan” option as below:

* Cost of a **queries to answer Q1 in Deliverable#2 –**

All the three queries provided for **Q1 in Deliverable#2** and the new query I wrote using non-correlated subquery have same COST = 3, hence, the query can be rewritten to achieve desired result, however they all have the same performance. Cost of each query is as below:

**select distinct a.selleruserid, b.state seller\_state, a.buyersuserid, c.state buyer\_state**

**from sales\_Fact a, sellers\_dim b, buyers\_dim c**

**where a.selleruserid = b.userid**

**and a.buyersuserid = c.userid**

**and b.state != c.state;**

Table

Description automatically generated

**--Alternate way with join**

**select distinct sf.buyersuserid, b.state buyer\_location,**

**sf.selleruserid, s.state seller\_location**

**from sales\_fact sf**

**join sellers\_dim s on (sf.selleruserid = s.userid)**

**join buyers\_dim b on (sf.buyersuserid = b.userid)**

**where b.state != s.state;**

A picture containing table

Description automatically generated

**--using correlated subquery**

**select distinct selleruserid, buyersuserid**

**from sales\_fact a**

**where not exists**

**(select 'X'**

**from sellers\_dim s, buyers\_dim b**

**where b.state = s.state**

**and s.userid = a.selleruserid**

**and b.userid = a.buyersuserid);**

Graphical user interface

Description automatically generated with medium confidence

new query written as part of deliverable #3:

**--using non-correlated subquery-**

**select distinct selleruserid, buyersuserid**

**from sales\_fact**

**where (selleruserid, buyersuserid) in**

**(select s.userid, b.userid**

**from sellers\_dim s, buyers\_dim b**

**where b.state != s.state);**

**Table

Description automatically generated**

* Cost of a **queries to answer Q2 in Deliverable#2 –**

**select distinct name ietm\_name, selleruserid as "sold by", buyersuserid as "bought by"**

**from sales\_fact**

**left join items\_dim using (itemid);**

**Table

Description automatically generated**

**select distinct name item\_name, buyersuserid, selleruserid**

**from sales\_fact sf, items\_dim i**

**where sf.itemid = i.itemid(+);**

**Graphical user interface, application, table

Description automatically generated**

**--new query written as part of deliverable #3**

**--using non-correlated subquery:**

**select (select name from items\_dim where itemid = sf.itemid) as "item name",**

**selleruserid, buyersuserid**

**from sales\_fact sf;**

**Graphical user interface, application

Description automatically generated**

As cost of new query written using subquery is 3 which is less than both the queries that used join to achieve the same result. Hence, a query rewritten with subquery to answer Q2 performs better.

* Cost of a **queries to answer Q3 in Deliverable#2 –**

Graphical user interface, text, application, table

Description automatically generated

Query rewritten without using items\_dim table –

Graphical user interface, application, table

Description automatically generated

No change in cost, hence both the queries are good to use to answer Q3 in deliverable #2., however, I would still prefer the first query, as it provides us more details at same cost.

* Cost of a **queries to answer Q4 in Deliverable#2 - to find top seller:**

Table

Description automatically generated

New query using max() function – to find top seller:

Table

Description automatically generated

The new query that uses **max()** function and **having** clause to find the highest amount sold has less cost than the query that uses “**rownum = 1**“. Hence, new query written performs better to answer Q4 in deliverable #2.

* Cost of a **queries to answer Q5 in Deliverable#2 – to find top buyer**

Table

Description automatically generated

New query using max() function to find top buyer:

Table

Description automatically generated

The new query that uses **max()** function and **having** clause to find the highest amount bought by buyer has less cost than the first query that uses “**rownum = 1**“. Hence, I would say new query written performs better to answer Q5 in deliverable #2.