Data Warehouse Performance: Optimization by Double Partitioning of Materialized Views

Mohamed El Emine Abdel Wedoud
CSSEL, Abdelmalek Essaadi University, Faculty of Science, Tetouan, Morocco

Mohamed Larbi Benmaati
CSSEL, Abdelmalek Essaadi University, Faculty of Science, Tetouan, Morocco

Emany Sidi
Higher Institute of Accounting and Business Administration, Nouakchott, Mauritania

ABSTRACT
Given the evolving volume of data storage in a data warehouse, partitioning is a very useful optimization option during the design phase of data warehouse. Materialized views also improve the execution time of data warehouse queries.

The primary concern of designers and end users of data warehouses is always the performance that ensures the production of the best results in the shortest time.

In this work, we will demonstrate that if we end up with the partitioning of materialized views, we can optimize the functioning of data warehouses.

Keywords
Data Warehouse, optimization, Partitioning, materialized views, performance.

1. INTRODUCTION
To help decision makers understand and improve the performance of their businesses, data warehouse is used which is large databases specifically for data analysis. It is designed for queries and analysis, not for processing day-to-day transactions.

The performance of this data warehouse is of great interest to designers and this is because design is the most important stage in their life cycle, and this is due to its permanent impact on its condition and functioning. .

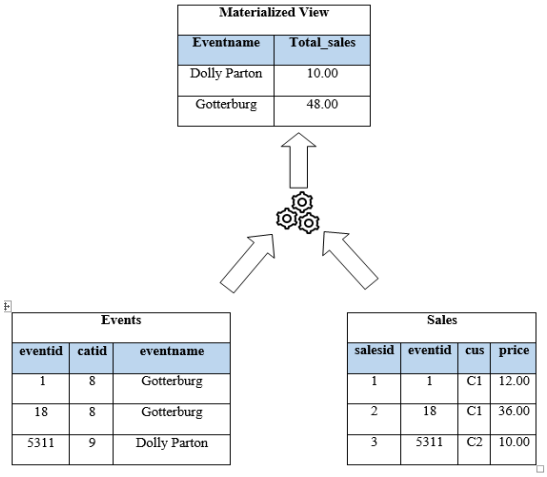
Designers are always looking to optimize the performance of data warehouses to be able to respond to so-called OLAP analytical queries. These data analysis queries are performed in an environment of high volume of data that keeps increasing day by day.

Given the evolving volume of data storage in a data warehouse, partitioning is a very useful optimization option during the design phase of these warehouses. Materialized views are used to improve the execution time of data warehouse queries.

To optimize the performance of data warehouses, we will combine two techniques: vertical and horizontal partitioning and materialized views.

2. MATERIALIZED VIEWS
We can define a view as a function from a set of base tables to a derived table and the function is recited each time the view is referenced; this is almost similar to the principle of the cache, it offers a quick access to a selected copy of data.

Materialized views are views, the data of which is materialized, that is, stored. Materialized views are quite useful objects, allowing a relatively large performance gain when used properly. In the data warehouse environment, materialized views can be called "summaries" because they are used to store aggregated and pre-calculated data. A materialized view helps avoid the overhead associated with expensive joins and aggregations for large analytical queries.



Materialized View

Eventname	Total_sales
Dolly Parton	10.00
Gotterburg	48.00

Events

eventid	catid	eventname
1	8	Gotterburg
18	8	Gotterburg
5311	9	Dolly Parton

Sales

salesid	eventid	cus	price
1	1	C1	12.00
2	18	C1	36.00
3	5311	C2	10.00

Figure 1: Materialized view

3. PARTITIONING
Partitioning a table is defined by dividing the table into several disjoint partitions. Recall that a partitioning scheme is the result of the process of fragmentation [1].

Two types of partitioning are possible: horizontal partitioning and vertical partitioning. In vertical partitioning, a relation is divided into sub relations called vertical fragments which are projections applied to the relation. Vertical partitioning naturally favors the processing of projection requests on the attributes used in the process of fragmentation, by limiting the number of fragments to access.

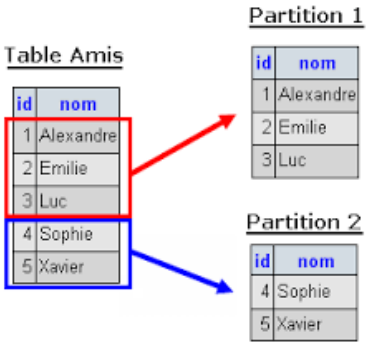


Figure 2: Horizontal Partitioning

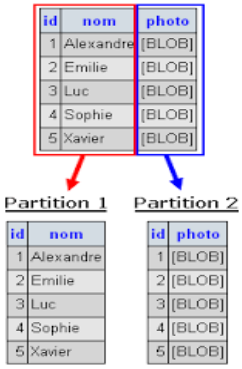


Figure 3: Vertical Partitioning

4. HYPOTHESIS

Materialized views speed up responses to analytical queries. We propose to create a materialized view from our model shown in figure 5. We will partition our materialized view horizontally by date, and vertically by store.

Our materialized view is used to store aggregated data relating to sales operations on dimensions: customers, stores and products.

The queries used in this experiment are star join queries, and the DBMS used is Oracle with its Enterprise edition.

5. ANALYSIS AND RESULTS

In this study, we used a data warehouse with:

- One fact table (Sales).

- Four dimension tables (Customers, Products, Time, Stores).

- Then we inserted 3 million rows of data to overload the fact table.

- For the dimension tables, the contents of the fact table have been respected for their loading.

Below is the design schema of the data warehouse used in our study.

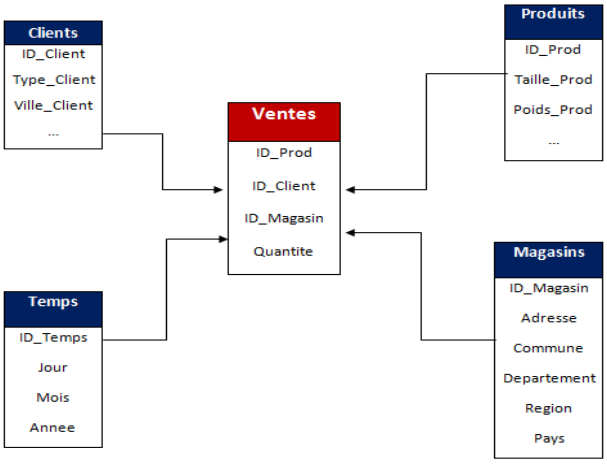


Figure 1 : data warehouse used in experimentation

5.1 Technical characteristics of the study environment

The following table presents the technical characteristics of the environment in which our study has been realized. We used great resources to find the best results.

Table 1. Technical characteristics of the study environment

Physical Memory	Storage Disk	Operation System	RDBMS
8 Gb	2.5 TB	Windows Server 2019	Oracle Entreprise Edition

5.2 Star join query used

This star join query is used to stress the data warehouse because of the aggregation it contains. Our goal is to compare the results found after running this query.

We are interested in the behavior of the data warehouse before and after the double partitioning that we will apply on materialized view.

This query will be used three times:

a - In the initial state of the warehouse, ie with a simple design without partitioning or materialized view.

b - Secondly, we will create a materialized view

c - In the third iteration, we partition the materialized view

After each step, the results are retrieved for comparison.

Table 2 : Star join query

```
SELECT Temps.Annee, Produit.Taille_Prod,
Temps.Mois, SUM(Ventes.quantite)

FROM Ventes INNER JOIN Date ON
Ventes.ID_Temps = Temps.ID_Temps INNER
JOIN Ventes.ID_Prod = Produit.ID_Prod

GROUP BY Temps.Annee
```

5.3 Results

Step 1: Simple design

We executed our request on the data warehouse, designed with without partitioning or materialized view. The results found are:

Memory Occupied by process	execution time	Size compression	processor cores
76%	60 s	1.53 TB	70%

Step 2: Materialized view

Memory Occupied by process	execution time	Size compression	processor cores
65 %	38 s	1.11 TB	77%

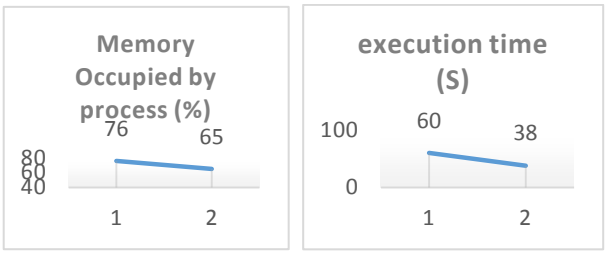


Figure 6: Memory and time curve (STEP 2)

Step 3: **Partition the materialized view**

Memory Occupied by process	execution time	Size compression	processor cores
32%	**26s**	**1.02 TB**	**92%**

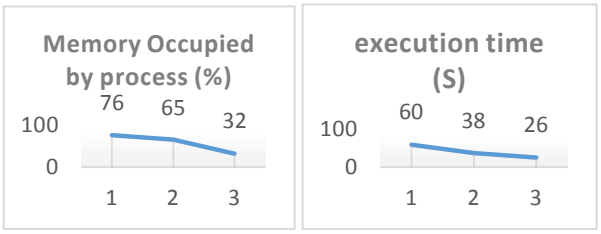


Figure 7: Memory and time curve (STEP 3)

5.4 Results analysis

After double partitioning the materialized views vertically and horizontally based on time and stores, we saw a remarkable optimization of the performance of our data warehouse.

Our method has given results on a logical and physical level such as reducing memory occupancy by 44%, minimizing query execution time by more than 50% and increasing processor cores allocated to this operation.

The comparison curves between the second and third step clearly show the effects of partitioning the materialized views.

6. CONCLUSION AND PERSPECTIVES

To conclude, we have approved by our experimental approach that the double partitioning of materialized views is a suitable solution for optimizing the performance of data warehouses.

Partitioning vertically based on stores and the horizontal based on time allows materialized views of data to perform sufficiently large for queries over large volumes of data.

As perspectives, we will work on several diversified techniques based on double partitioning and materialized views to have further results of optimizing the performance of data warehouses.

7. REFERENCES

[1] Double Partitioning with global and local Indexing: Effect on Data Warehouse Performance" (Volume 180/Number 44 (ISBN: 973-93-80898-72-9) Authors: Mohamed El Emine Abdel Wedoud, Mohamed Larbi Benmaati, Emany Sidi.

[2] R. Kimball, L. Reeves, M. Ross, The Data Warehouse Toolkit. John Wiley Sons, NEW YORK, 2nd edition, 2002.

[3] Selection Of Materialized View Using Query Optimization In Database Management : An Efficient Methodology., International Journal of Database Management Systems 2(4). 2010 . Authors : V. M. Thakare, KArde.

[4] W. Inmon, Building the Data Warehouse., John Wiley Sons, fourth edition, 2005.

[5] Patrick and Quass, Dallan O'Neil, "Improved Query Performance with Variant Indexes," vol. 26, no. 2, 1997.

[6] Impact of using Snowflake Schema and Bitmap Index on Data Warehouse Querying (Volume 180/Number 15 (ISBN: 973-93-80898-08-9)) Authors: Mohammed Benjelloun, Mohamed El Merouani, El Amin A. Abdelouarit.

[7] Using Snowflake Schema and Bitmap Index for Big Data Warehouse Volume (Volume 180/Number 8 (ISBN: 973-93-80897-91-9)) Authors: Mohammed Benjelloun, Mohamed El Merouani, El Amin A. Abdelouarit.

[8] The Impact of Partitioned Fact Tables and Bitmap Index on Data Warehouse Performance (Volume 135/Number 13 (ISBN: 973-93-80891-16-1)) Authors: Emany Sidi, Mohamed El Merouani, El Amin A. Abdelouarit.

[9] Star Schema Advantages on Data Warehouse: Using Bitmap Index and Partitioned Fact Tables (Volume 134/Number 13 (ISBN: 973-93-80890-95-3)) Authors: Emany Sidi, Mohamed El Merouani, El Amin A. Abdelouarit.