International Journal of Computer Applications (0975 – 8887)
Volume 180 – No.44, May 2018

# Double Partitioning with global and local Indexing: Effect on Data Warehouse Performance

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
ISCAE* Research Group, Nouakwchott, Mauritania

## ABSTRACT
The design of data warehouses is the most important step in their life cycle and is due to its permanent impact on its condition and operation.

Strategic decision-making in a timely manner is an objective and a need for decision makers, especially in a production environment with a high frequency of updating.

Designers of data warehouses always try to minimize the execution time of the analysis requests and optimize the performance of the warehouse in order to present the reports in the best time and condition.

This article shows that during the design phase a double vertical and horizontal partitioning of the fact table and dimension tables with global and local indexing can optimize the logical and physical performance of the data warehouse.

## General Terms
Indexes, Pratitioning, Business Intelligence, Data Warehouse

## Keywords
Data Warehouse, optimization, BI, Indexes, performance

## 1. INTRODUCTION
Data Warehouse is a large database but different from traditional databases given the analytical aspect of its use. The performance of this data warehouse takes a big space of interest for designers and it is because design is the most important step in their life cycle and it is due to its permanent impact on its state and its operation.

OLAP queries for data analysis run all the data, which is time consuming, so you can target ranges of values using specific indexes.

Physical design has a direct influence on query execution time, especially partitioning and indexing.

There are two types of partitioning, vertical and horizontal, and there are also several types of indexes, we will use Global and Local indexes with double partitioning.

## 2. 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.

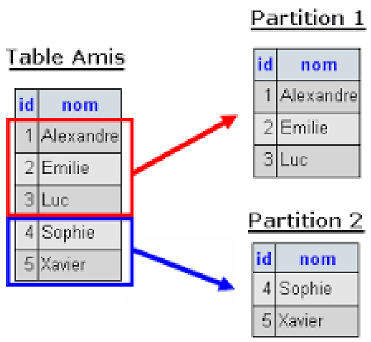



**Figure 1 : horizontal partitioning**

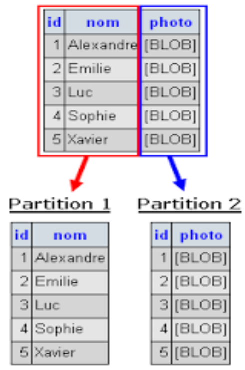
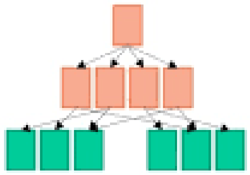



**Figure 2 : vertical partitioning**

## 3. INDEXES
An index is a structure added to the database to allow quick access to the data. It allows from an index key to find the physical location of the searched tuple.

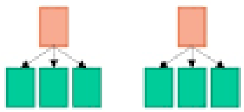
Among the indexing techniques proposed in the context of conventional databases, we can mention the B-Tree index, the hash index, the projection index, the join index, and so on. Most of these indexes are also used in relational warehouses. Some indexing techniques have emerged in the context of data warehouses such as binary indexes, binary join indexes, star join indexes. there are also local and global partitioned indexes, which we will use in our work.

A global index is a one-to-many relationship, allowing one index partition to map to many table partitions. A local index is a one-to-one mapping between an index partition and a table partition.



The global index applies to the table and all its partitions.

**Figure 3 : Global index**



Local index is partitioned just like the data.

**Figure 4 : Local index**

## 4. HYPOTHESIS

The Global and local indexing structures make it easier and faster to access data, but let's not forget that a large data table is always difficult to navigate even if you use indexing techniques.

To overcome this problem we propose the decomposition of fact tables and dimension tables with the partitioning mechanism. For the fact table, it will be partitioned vertically and the dimension tables will be partitioned horizontally.

In the fact table, we use local indexing and dimension tables we use global indexing.

So we demonstrated with this new design method that it is the most appropriate for optimizing the performance of data warehouses.

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.

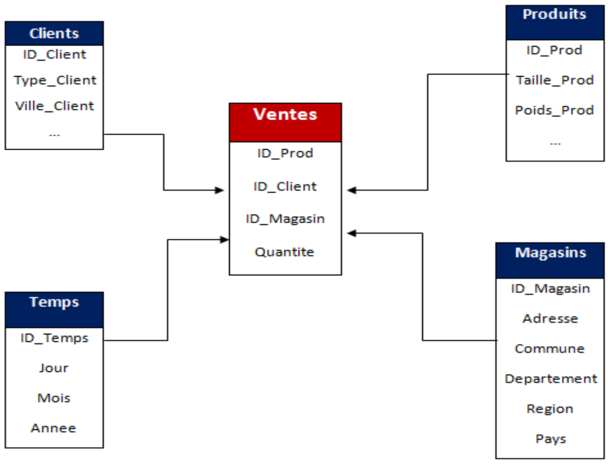



**Figure 5 : data warehouse used in experimentation**

### 5.1 Technical Characteristics Of The Study Environment

The following table presents the technical characteristics of the environment in which our study has been realised. 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 TB** | **Windows Server 2016** | **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 and also the global and local indexing.

This query will be used three times:

a - In the initial state of the warehouse, ie with a simple design with a star schema but without partitioning or indexing.

b - Secondly, we will partition the fact table vertically with the local indexing.

c - In the third iteration, we partition the dimension tables horizontally and we apply Global indexing.

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)<br>FROM Ventes INNER JOIN Date ON Ventes.ID_Temps = Temps.ID_Temps INNER JOIN Ventes.ID_Prod = Produit.ID_Prod<br>GROUP BY Temps.Annee |

## 5.3 Results

**<u>Step 1</u> : Simple design without partitioning or indexing**

We executed our request on the data warehouse, designed with the star model, but without partitioning or indexing. The results found are:

| Memory Occupied by process | execution time | Size compression | processor cores |
|---|---|---|---|
| **81%** | **65 s** | **1.23 TB** | **65%** |

**<u>Step 2</u> : Partition the fact table vertically with the local indexing**

| Memory Occupied by process | execution time | Size compression | processor cores |
|---|---|---|---|
| **55 %** | **42 s** | **1.01 TB** | **87%** |

**<u>Step 3</u> : Partition the dimension tables horizontally and we apply Global indexing**

| Memory Occupied by process | execution time | Size compression | processor cores |
|---|---|---|---|
| **37%** | **30s** | **1.09 TB** | **90%** |

## 5.4 Results Analysis

The results found show that double partitioning of the fact table and the dimension tables, and the use of the global and local indexes, made it possible to optimize the logical and physical performance of the data warehouse.

When we practiced our custom design method we found that:

- The memory occupied by the process has decreased by 40%
- Query execution time has become faster by twice
- The memory units can now store more values than in the case before which made it possible to compress the size of the data warehouse.
- The processor cores are used more and more which gives a better result, because if the processor uses a larger capacity we will have a more relevant and faster result.

# 6. CONCLUSION AND PERSPECTIVES

The use of dual partitioning of the fact and dimension tables with global and local indexing plays an important rôle in optimizing the performance of data warehouses. Horizontal partitioning of the fact table makes it possible to select recording ranges according to attributes (columns) and the horizontal partitioning of the dimension table (Time) makes it possible to analyze between two dates or in a specific date interval with global indexing. This custom design maximizes the physical and logical resources of the data warehouse.

Data warehousing is used for the most part in areas known for dynamism and increasing scalability such as banking, commerce, industry and finance. These areas need efficient warehouses and with a design that can support their analysis needs. So many improvements are required by the decision makers to enable them to make decisions in the best working environment.

In future work, we will work on the caching axis of star join queries used in this study and the uses of materialized views. Our goal in this perspective is to speed up query response time and provide a more appropriate work environment for data warehouse decision makers and analysts.

# 7. REFERENCES

[1] Surajit and Narasayya, Vivek Chaudhuri, "Partitioning strategies in distributed object-oriented database systems," 1997.

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

[3] S. Chaudhuri, U. Dayal,An Overview of Data Warehousing and OLAP Technology., ACM SIGMOD RECORD. 1997

[4] E. Omiecinski, and S. Navathe. M. Frank, *Adaptive and automated index selection in rdbms*. Vienna, Austria: International Conference onExtending Database Technology, 1992.

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

[6] K. Wu and P. Yu, Range-based bitmap Indexing for high cardinality attributes with skew. Washington, DC, USA: the 22nd International Computer Software and Applications Conference. IEEE Computer Society, 1998.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] Data Warehouse Tuning: The Supremacy of Bitmap Index (Volume 79/Number 7 (ISBN: 973-93-80878-07-5)) 2013 Authors: El Amin Aoulad Abdelouarit, Mohamed El Merouani, Abdelatif.