Abstract—Data Warehouse (DW) is a collection of subject-oriented data, integrated, nonvolatile and historized. It is modelled following different schema (star, snowflake, constellation). It is supplied from different data sources via transactional queries and provides analytical data to assist in decision making in Business Intelligence. The DW should be able to handle the diversity of data, query complexity and workload in order to meet the complex decision-support queries whose processing time can take several hours. To improve the performance of DWs, the administrator uses different optimization techniques: indexing, materialized views, fragmentation and parallelism.

The approach suggested in this paper is a query optimization technique in DWs using a Web service. This optimization is based on the horizontal fragmentation (primary and derived) of DW star schema. In order to facilitate the task of DW’s administrator, our approach will be accessible via a Web service that automatically both generates a horizontal fragmentation schema and scripts needed to implement it.

Keywords—Data Warehouse, Web service, Horizontal fragmentation, Primary & Derived fragmentation, Business Intelligence

I. INTRODUCTION

With the technological advances experienced by Internet, many "conventional" services now use this tool. It is mainly based on Web services: they allow you to share information and provide or obtain services.

To query a DW, the decision makers use expensive decision-support queries. To reduce the total query execution cost in DWs, administrators use different techniques to select an optimized schema of DW. The physical implementation of the selected schema requires a lot of time and expertise.

In this context, we propose a solution of fragmentation via the web service. It will be available on the web and intended for DW’s administrators. The schema of fragmentation recommended will be generated from a query load and physical parameters sent by the administrator.

In Section 2 we present the related work on the various optimization techniques, and then we expose the problem of selecting a schema of horizontal fragmentation. Section 3 presents the web services technology used in the field of DWs. We explain the k-means classification method and its implementation with regard to DWs in Section 4. In Section 5, we develop our strategy for selecting a fragmentation schema of DWs using the web services technology. The validation of our approach through tests will be shown in Section 6. Then we end up with a conclusion and perspectives.

II. RELATED WORK

A. Optimization Techniques

To reduce query execution cost and improve processing time in the DWs, administrators use different optimization techniques [1], namely indexing, materialized views and fragmentation. The selection of an optimization technique has been the research focus of several studies. There are two types of selection [2]: (i) isolated selection consists of implementing an optimization technique at a time. (ii) Combined selection consists of implementing either jointly or sequentially two or more techniques by exploiting the dependencies between them.

Selecting a schema of DWs is an NP-complete problem [3]. There are no algorithms that provide an optimal solution in a finite time, but we note the existence of several optimization techniques addressing this problem, among which we mention: the combinatorial optimization, metaheuristic, optimization by learning and evolutionary optimization guided by knowledge [4]. They use different algorithms and methods such as Ant Colony algorithms, Simulated Annealing, Tabu Search, Particle Swam Optimization [4], Genetic algorithm, K-Means [5], Greedy algorithm [6], Hill-Climbing [7].

1) Indexing: An index is a data structure allowing the direct and rapid access to tuples of voluminous relations. Indexing is an important option in the physical design phase of DWs [8]. Several types of indexes are used in the field of databases. The Binary Join Indexes (BJI) are most effective in the DWs [2]. They constitute a combination of the join index and the binary index. The latter has been proposed to pre-compute joins between one or more dimension tables and the fact table in the DWs star schema.
2) **Materialized views**: A materialized view is a relation that contains the result of executing a query. Although physically stored it improves query performance by pre-calculating and storing complex and costly operations. It introduces the problem of their maintenances [3].

3) **Fragmentation**: is to share out a set of DW data into several disjoint partitions. The combination of these partitions produces the full source data without loss or addition of information. There are three types of fragmentation [3]:

- **Horizontal Fragmentation (HF)** which consists of partitioning a table following a selection predicate.
- **Vertical Fragmentation (VF)** that allows partitioning a table according to a projection query.
- **Mixed Fragmentation (MF)** that allows partitioning a table by combining HF and VF.

HF does not duplicate data, this is an important means to improve the performance of DWs [7], since it reduces the time of storage and maintenance. In the data warehouse we talk about the primary horizontal fragmentation (PHF) and derived horizontal fragmentation (DHF) [7]. The first (PHF) consists of fragmenting the dimension tables and the second (DHF) of fragmenting the fact table according to the fragments of dimension tables.

There are similarities between indexes and materialized views. Both belong to the redundant structure. They also share the same storage resource and require regular updates. The presence of an index on a materialized view can make it more attractive and vice versa [6].

There are also similarities between IIB and HF. Both allow data distribution in a fact table according to the distribution of dimension tables, in order to reduce the execution cost of star joins. In addition, both share the same selection attributes. They are selected in a combined manner [2].

**B. Problem of selecting a horizontal fragmentation schema**

Selecting a horizontal fragmentation scheme consists of: (1) determining a set of dimension tables to fragment, (2) distributing data tuples of a dimension table following its attributes, (3) fragmenting the fact table according to the schema of fragmentation of tables dimension [7].

Note that the number of N possible of fragmentation schema of a fact table can be very large. It is given by $N = \prod_{i=1}^{g} m_i$ where $m_i$ is the number of fragments of the dimension table and $g$ is the number of dimension tables that have participated in the process of fragmentation [7]. To avoid the explosion of this number, Boukhalfa et al. (2008) addressed the issue of selecting a schema of fragmentation as a problem of constrained optimization. They used heuristics such as Hill-Climbing and Simulated Annealing.

However, Barr M. and L. Bellatreche [9] discussed the different approaches used to select a horizontal fragmentation schema, namely (i) Selection predicates-based approaches. They are based on the identification of selection predicates to create subsets in order to minimize the cost of query execution. (ii) Affinity-based approaches use query frequencies to select a primary fragmentation schema. (iii) Data mining-based approaches use data mining algorithms to select a fragmentation schema.

**III. WEB SERVICES TECHNOLOGY**

Web service is described as being a set of features achieving specific tasks accessible by computer network [10]. The concept of Web service primarily indicates an application available on Internet by a service provider and accessible by customers through standard Internet protocols [11]. The Web service enables applications to remotely communicate via Internet or Intranet independently from the platforms and from the languages. It uses message exchange norms based on XML namely SOAP and WSDL [10]. The Web services are thus autonomous software components and self-descriptive; accordingly, they constitute new model for applications integration. A Web service draws a set of features exposed on Internet or on Intranet, either by applications or for applications, without human intervention, and in real time as well.

In previous works, some authors have treated the use of Data Warehouses with Web services. Zhong et al. [12] proposed a pattern of distributed DW based on Web service. This revival of data storage architecture is called Data WebHouse. Etienne et al. [10] suggested an architecture oriented Web service for the constitution of mini-cubic SOLAP for mobile customer. Mehedintu et al. [14] treated the construction stages of a DW which are Web compatible. Within the same vein, we focus on the pertinent use of the web service for DWs.

**IV. K-MEANS CLASSIFICATION ALGORITHM**

The K-means method is one of the regrouping techniques used by various applications [15]. K-means partitions a set of data in K disjoined classes. This result is achieved by positioning K centroid within the space of data sources (predicates) and by minimizing the sum of intra-class distances $\sum_{i=1}^{k} \sum_{x_i \in C} (x_i - \mu_i)^2$, $i = 1...k$ (k classes at output, and $\mu_i$ is centroid point of $x_i \in C_i$) [16]. The disadvantage of k-means is that its use requires the pre-specification of the various classes to be gathered, and may also be blocked in a local optimum [17]. The k-means classification technique was largely used for the physical design of databases. We refer to the work of Darabant and Campan [17] which proposed a horizontal fragmentation method of distributed and object-oriented database. In so doing, they have used the K-means classification technique. Mahboubi and Darmont [15] have also used this technique to fragment the DW of XML. Bouchakri et al. [2] have equally used the k-means to classify the attributes between the binary joint indices and horizontal fragmentation. We are suggested an architecture oriented Web service to select a schema of horizontal fragmentation using a k-means classification algorithm. The k-means method classifies the predicates of selection using the matrix query_predicate M (table 2). It generates fragmentation schema made up of K fragments.
V. PROPOSED APPROACH

The main goal is to improve the performance of Data Warehouse through Web services. The DW’s administrator connects to the Web server and invokes the service dedicated. After analyzing the query workload sent by the administrator, the web service proposes a horizontal fragmentation schema of DW (e.g. fig. 1). The administrator must determine the following data from DW with a fact $F$ and $i$ dimensions $D = \{D_1, D_2, ..., D_i\}$:

- A description file of the DW: DWXml (e.g. fig. 4);
- The query workload: $Q = \{Q_1, Q_2, ..., Q_j\}$;
- The constraint MNF (The Maximum Number of Fragments).

These data will be sent to Web service.

![Diagram](Image)

**Fig. 1: Process invoking a Web service to create a DW fragmentation schema**

From a load of query $Q$, the Web service provides this following:

- The selection predicates $P = \{P_1, P_2, ..., P_k\}$;
- The matrix query-predicates $M$;
- The horizontal fragmentation schema.

Using the fragmentation schema and the DW’s description file (DWXml in fig. 6), the Web service generates the scripts for the primary and derived horizontal fragmentation.
Thus, the Web service offers the fragmentation schema and the list of codes generated to the administrator. The schema below summarizes the overall process of implementing a horizontal fragmentation schema according to the proposed approach.

![Fig. 2: General architecture of the proposed approach.](image)

Figure 3 shows an example code used to generate the dimension fragmentation script from the fragmentation schema and the description file DWXml. This code is written in Java. We define two classes, the class table(name, list of attributes) describing a relation by object model, and the class attribute(name, type, list of predicate) defining the attribute structure. This code consists of the following main methods:

- **createTable(Table t)**: This method generates SQL code to create table t. The input parameter is an object describing this table.
- **scriptFragmentation(Table t)**: writes SQL code partitioning the table t from its fragmentation schema.
- **generateScript (Table [ ] t)**: generates the primary horizontal fragmentation script of a dimensions list. This list represents all the dimensions proposed by SimpleKMeans class in the fragmentation schema.

```java
static void createTable(Table t) {
    try{
        File file;
        PrintWriter r=new PrintWriter(new FileWriter(file,true));
        String sql="create table "+t.name+_fg";
        r.println(sql);
        r.println("(");
        for(int i=0;i<t.attribute.size();i++)
        {
            sql=t.attribute.elementAt(i).name+" "+t.attribute.elementAt(i).type+"," ;
            r.println(sql);
        } 
        r.println(t.name.substring(0,4).toUpperCase()+"_COL NUMBER(15,5) NULL,"
        r.println("PRIMARY KEY ("+t.attribute.elementAt(0).name+"))
        r.close();
    }catch(Exception e){e.printStackTrace();}
}
```

```java
static void scriptFramentation(Table t){
    t.attribute=schemaTable(t);
    int nbrPartition=1;
    for(int i=0;i<t.attribute.size();i++)
```
Fig. 3: Example of code that generates the fragmentation script

The following example (e.g. fig. 4) represents the fragmentation script generated by the code as shown in Fig.3. This script creates dimensions partitioned according to the values of the field "*_COL" (* is the first four letters of the table’s name to fragment). The values of this field are updated following the fragmentation schema, using a stored procedure generated by the Web service. These values are Integers from 0 to N (fragment’s number of dimension). The derived fragmentation script is generated according to the primary fragmentation schema.

```
create table timelevel_fg
(
tid varchar(12),
year_level number(4),
quarter_level varchar(6),
month_level number(2),
week_level number(2),
day_level number(2),
TIME_COL NUMBER(15,5) NULL,
PRIMARY KEY (tid)
)
PARTITION BY LIST(TIME_COL)
(  PARTITION PT1 VALUES('0'),
  PARTITION PT2 VALUES('1'),
  PARTITION PT3 VALUES('2')
);
```

```
create table chanlevel_fg
(
base_level char(12),
all_level char(12),
PROD_COL NUMBER(15,5) NULL,
PRIMARY KEY (code_level)
)
PARTITION BY LIST(PROD_COL)
(  PARTITION PT1 VALUES('0'),
  PARTITION PT2 VALUES('1'),
  PARTITION PT3 VALUES('2'),
  PARTITION PT4 VALUES('3'),
  PARTITION PT4 VALUES('4')
);
```

```
create table prodlevel_fg  
(  
code_level char(12),
class_level char(12),
group_level char(12),
family_level char(12),
line_level char(12),
division_level char(12),
CHAN_COL NUMBER(15,5) NULL,
PRIMARY KEY (base_level)  
)  
PARTITION BY LIST(CHAN_COL)
(  
PARTITION PT1 VALUES('0'),
PARTITION PT2 VALUES('1')  
);  

Fig. 4: Example of fragmentation’s script generated using the web service

The DWXml file (e.g. fig. 6) is used to describe the DW structure in XML. Some authors have already treated the link between DW and XML. Mahboubi and Darmont [15] considered the DW as an XML document. This document contains the metadata defining information about data storage, data source’s originally (URL) and view’s specifications. Boussaid et al. [18] proposed an approach to the storage of complex data contained in XML documents called X-Warehousing. This approach defines a methodology to design complex DW using XML formalism. Boukraa et al. [19] proposed a physical schema of an XML Warehouse and provided a solution for optimizing these performances.

In order to generate the horizontal fragmentation scripts of a Data Warehouse star schema, the Web service must be informed about the XML file. The Web service extracts the tables’ name, the list of its attributes with their type and information about link fact-dimensions.

Each dimension $D_i$ has $PK_i$ primary key and $n_i$ attributes. A fact contains a set of composite keys that aim at referencing the primary key of dimensions. The others attributes in the fact table are called "facts" or "measures". In the DWXml file, fact and dimensions are defined by elements. Fields are represented by sub-elements. These elements have a list of properties namely "name", "type" and "ref". The "ref" property references foreign keys in the fact. Sub-element does not contain "ref" property, is called "measure".

The fig. 5 describe the benchmark APB1 whose XML representation as shown in fig. 6.
VI. EXPERIMENTAL PART AND RESULTS:

C. Platform and Tools:
In order to endorse and validate our approach, we have used various requests in the APB1 benchmark under ORACLE 11g platform. Our application has been developed under the Eclipse IDE. In our project, we have implemented the JAVA API "SimpleKMeans", the "JDOM" API for processing XML documents and the Java framework "Axis" (this is an implementation of SAOP Java Open Source).

We have programmed different modules such as the determination of selection predicates module; the query_predicate matrix module; the primary and derived horizontal fragmentation coding scripts module. To select a schema of primary and derived horizontal fragmentation, we have adopted the SimpleKMeans. To deploy the Web service, we have also used Tomcat (Servlet engine and JSP) and Axis.

D. Tests and results:
To implement our approach, we have conducted several tests. We have varied the query workload Q and the number of fragments MNF. We have also executed the scripts proposed by the Web service on APB1 benchmark.

The tests gave satisfactory results. The implementation of a fragmentation schema had been performed in short time. The improvement of DW performance had been observed by running the query rewritten in the fragmented DW. Example, for Q = 10 queries, the determination of selection predicates module returned the following list of predicates:

Table 1 shows the list of predicates for dimension table. As a sample in point, the dimension table "timelevel" contains two selection predicates; the first attribute "month_level" having the value "01" is coded as "P_1". The second attribute "year_level" having the value "1996" is coded "P_2".

The creation matrix query_predicate module returns the following matrix:

The query_predicate matrix represents the participation of selection predicates {P_1, P_2,..., P_9}in each application load, \{Q_1, Q_2,..., Q_10\}. This representation is binary-coded.

For a number of classes k = 4 and from the query-predicate matrix illustrated in Table 2, the k-means algorithm produced the following C fragmentation schema:

The fragmentation schema includes attributes selection by class:

Table 3
Figure 7 shows the machine’s interface described by a Web page that allows DW’s administrators to send input files after authentication. Thus, following specific treatment, the administrator will receive scripts to download.

VII. CONCLUSION AND PERSPECTIVES:

The DW’s fragmentation has been widely studied by industry and academic researchers. The selection and implementation of a fragmentation schema requires a lot of expertise, a task that is not often simple to carry out by DW’s administrators. In this context, we have proposed an approach to select a schema of horizontal fragmentation and generate necessary scripts automatically. Finally, to show the effectiveness of our approach, two parameters have been taken into consideration namely the time to implement a DW’s fragmentation schema, and the response time of queries against fragmented DW.

The steps undertaken to implement the approach have been widely explained, tested and validated through a case study. A number of varied queries have been sent by the administrator via the Web and provided with some DW’s information (DWXml). A program generates predicates that will be used to provide query-predicates matrix. It will be operated by the k-means method to provide an optimized horizontal fragmentation schema. Finally, the scripts used to implement this fragmentation schema will be available for direct download.

After studying the optimization approaches in detail, we will propose an indexing approach and another approach for creation of materialized views, both using Web service. In our future works, we will propose a Web service that allows sequential and combined selection of different optimization techniques. The idea is to turn the DW’s performance to operate automatically and make it easier for administrators.

REFERENCES


