Web Service for Incremental and Automatic Data Warehouses Fragmentation

Ettaoufik Abdelaziz
RITM Lab., ESTC, CED Engineering Sciences
ENSEM, Hassan II University
Casablanca, Morocco

Mohammed Ouzzif
RITM Lab., ESTC
Hassan II University
Casablanca, Morocco

***Abstract*—The data warehouses (DW) are proposed to collect and store heterogeneous and bulky data. They represent a collection of thematic, integrated, non-volatile and histories data. They are fed from different data sources through transactional queries and offer analytical data through decisional queries. Generally, the decisional queries execution cost on large tables is very high. Reducing this cost becomes essential to enable decision-makers to interact in a reasonable time. In this context, DW administrators use different optimization techniques such as fragmentation, indexing, materialized views, and parallelism. On the other hand, the volume of data residing in the DW is constantly evolving. This can increase the complexity of frequent queries, which can degrade the performance of DW. The administrator always has to manually design a new fragmentation scheme from the new load of frequent queries. Having an automatic fragmentation tool of DW becomes important. The approach proposed in this paper aims at an incremental horizontal fragmentation technique of the DW through a web service. This technique is based on the updating of the queries load by adding the new frequent queries and eliminating the queries which do not remain frequent. The goal is to automate the implementation of the incremental fragmentation in order to optimize the new queries load. An experimental study on a real DW is carried out and comparative tests show the satisfaction of our approach.**

***Keywords*—Data warehouse; horizontal fragmentation; incremental fragmentation; frequent queries; web service**

I. Introduction

Business intelligence is a sector in full development. It is a management term that refers to the means, tools and methods that support the process of collecting, consolidating, modeling, analyzing and restoring information [1]. To store heterogeneous and voluminous data, data warehouse (DW) has been proposed [2]-[4]. It is very promising technology and is being accepted rapidly across all domains of the industries [1]-[5]. DW is an essential component of almost every modern enterprise information system [6]. It provides a new and wide idea of the company and gives better performance of database [7]. It can be defined as a model of a concrete business system representing a set of all of the states of that system during a given interval of time [8]. It is represented by multidimensional cubes supporting a large volume of data that can reach several thousand gigabytes (terabyte). Each dimension of the cube represents an axis of analysis and each element of the cube represents the fact analyzed. The DW is modeled according to one of the three models, star schema, snowflake schema and constellation schema. In the star modeling case, the measurements are represented by a fact table and each measurement dimension represented by a dimension table [9]. The fact table contains attributes representing quantitative data named measurements and foreign keys referencing the dimension tables, whereas the dimension table contains attributes representing qualitative data that can be used as the analysis axis.

The DW are often questioned by complex decisional queries. These queries invoke a very large fact table and include joins between the fact table and several dimension tables. In order to reduce the cost of this queries kind, the DW Administrator uses different optimization techniques. Horizontal fragmentation (HF) is one of the most used techniques; it consists of partitioning a dataset of DW to several disjoint partitions. This partitioning is madding from an attributes list and selection predicates extracted from the query load noting that the number of fragmentation sub-schemes of fact table can be very large. It is given by N=$\prod_{i=1}^{g} mi$ where, mi represents the number of fragments of dimension table and g represents the number of dimension tables participating in fragmentation process [10]. To avoid the explosion of this number, the problem of HF schema selection is handled under the maximum number of fragments constraint required by the administrator. The selection of a DW HF schema has been proven to be NP-complete problem [11]-[15]; there is not an exact solution to this kind of problem. For this purpose, several works have treated the HF schema selection problem using different algorithms and different methods in order to select a schema close to the most optimal schema. Gacem, *et al*. in [14] grouped these works into four categories: 1) predicate-guided works; 2) affinity-guided works; 3) cost-guided works; and 4) classification-guided works.

During the analysis of different works, we found that several works propose the HF schema selection approaches based on a static selection carried out manually by the DW's administrator. Few studies propose a static selection using the DW's administration tools [10]-[16]. In [17] the authors propose an approach for selecting an incremental HF scheme. This approach is based on the manual adaptation of the current HF schema to the new frequent queries. But it does not take into account the queries participating in the selection process of the current schema and which do not remain frequent. In this paper, we propose an approach for selecting an automatic and incremental HF schema using a web service. It is based on the adaptation of the current HF schema to changes appeared in the frequent queries load.

In Section 2 we give a brief overview of related work on the static selection problem of a HF schema and on the interaction between DW and web service. Then we present our approach of automatic and incremental selection of a HF schema in Section 3. Section 4 presents our experimental study on a physical DW. And we will conclude with a conclusion and perspectives.

II. Selection of Optimization Techniques

We want to offer a lightweight but powerful and online data warehouse performance optimization tool. The DW performance is considered the main concern for designers [9]. This performance is based on optimization of the queries execution time. The fragmentation, indexing and materialized views are a set of techniques used to improve the queries execution cost. The fragmentation is used in case of very huge fact and dimension tables, both of these tables can be physically partitioned. Whereas materialized views store aggregated data to yield faster access of data and reduced query response time [4]. It is a redundant structure that duplicates data in DW and occupies a supplement memory space. The space constraint forces to select an optimal subset of materialized views to attain the balance between query cost and space limits [4]. The indexing belongs also to the redundant structure; it represents data structures allowing direct and rapid access to the tuples of a voluminous relation. It optimizes queries by minimizing the amount of data to be used in calculations. Optimizing queries execution time consists in selecting and implementing one or several optimization techniques. This selection can be isolated, sequential or combined. The first case consists in implementing one optimization technique at a time. Whereas the second case consists of implementing two or more techniques sequentially and third case joint selection of two or more optimization techniques by exploiting the dependencies between them.

A. *Selection of a HF schema*

The selection of an HF schema consists in partitioning the DW schema into several disjoint sub-schemas in order to optimize the queries load executed on DW. The combination of these sub-schemas produces the full source data without loss or addition of information. There are three types of fragmentation [11]:

1) Horizontal fragmentation (HF) consists of partitioning a table following a selection predicate.

2) Vertical fragmentation (VH) allows partitioning a table according to a projection query.

3) Mixed fragmentation (MF) allows partitioning a table by combining HF and VF.

In data warehouses field, we talk about the primary horizontal fragmentation (PHF) and derived horizontal fragmentation (DHF). The PHF consists on partitioning the dimension tables whereas the DHF consists on partitioning the fact table according to the fragmentation schema of dimension tables.

Typically, DW's administrators use different optimization algorithms to select an optimal schema. Thus, the administrator always has to determine the frequent queries load. The selection of an optimal schema of HF can be static or incremental.

1) Static selection

The process of static selection has following parameters in entrance:

- A data warehouse schema
- A set of frequent queries
- A constraint W describes the maximum number of fragments.

The static selection of HF schema consists of selecting a set of tables to fragment and determinate the sub-partitions for each table selected in order to minimize the total queries execution cost under the maximum number of fragments. The static selection of HF schema remains less efficient since it does not adapt to the queries load evolution. In this case, the process of static selection mast be triggered and a new HF schema should be generated.

Several research studies deal with the queries optimization problem [2]-[6], [12]-[15], [18]-[22]. We will introduce below some works dealing with static selection of fragmentation schema and will also present some algorithms used in this context. We will focus on the HF of DW.

In [15], the authors treat the HF of DW using a method based on the algorithm of ant colonies. They proposed a cost based approach. This approach differs from existing approaches since it does not start with the direct application of an HF schema selecting algorithm, but it adds a new brick to map the HF selection problem. Then they solve the mapped problem by exploiting all the research conducted to solve this problem kind. On the other hand, the authors propose in [18] an XML data warehouses performance optimization technique by fragmentation and distribution on a grid. To do this, they used a method to adapt the most widely used fragmentation techniques in the relational domain to XML DW. The final fragmentation schema is generated by an original fragmentation method based on the k-means classification technique to control the number of fragments. Finally, they proposed a distribution approach of a XML data warehouse on a grid whereas in [14], the authors propose a scalable approach based on classification and election for a fragmentation supporting bulky loads. To this effect, they proposed a method for reducing the input queries load in order to minimize the selection problem complexity and the queries execution time. The proposed approach is based on two principles steps: 1) the queries classification to reduce the load size; and 2) the election to produce a new load that substitutes the initial load. This new load will be used as a main load to split the DW. From their part, the authors in [12] are interested to implementing a DW horizontal fragmentation approach. This approach represents the following characteristics: 1) cost model based approach; 2) it allows control of the generated number of fragments; and 3) DW fragmentation is performed from a maximum number of fragments set by the administrator according to the following scenario: PHF of the dimension tables according to which the DHF from the fact table is performed. The authors have demonstrated that this scenario is most appropriate for data warehouses, as it improves selections operations on dimension tables and the joint operations between facts and dimensions whereas in [19], the authors propose a method based on a classification algorithm to reduce the number of predicates in the data warehouses before fragmentation. The proposed method encompasses four phases: 1) a preliminary phase for determinate the set of predicates selection; 2) a coding phase for coding the predicates as binary matrices; 3) a classification phase of these predicates using the k-means algorithm; and a final phase 4) to reduce the number of predicates. All works that we have cited in this section are based on static selection of HF schema. They do not evolve as the query load changes.

2) Incremental selection

The works deal with the selection of HF schema which is based on static selection. This selected schema does not remain validate when a new frequent costly query coming integrated the frequent queries load. The incremental selection of an HF schema represents a solution of static selection limits. The process of incremental selection has following parameters in entrance:

- The current HF schema.
- A set of frequent queries.
- A set of new frequent queries.
- A set of queries that does not remain frequent.
- A constraint W describes the maximum number of fragments.

The incremental selection of HF schema consists on adapting the current HF schema to new frequent queries load. This adaptation is realized by execution of splitting operation when new frequent query coming integrates the frequent queries load, or by merging when an old frequent query does not remain frequent.

Very little works have dialed with dynamic and an incremental optimization of DW performance [17]-[23]. In [17], the authors have opted for the incremental selection of a fragmentation schema. They have presented an incremental HF approach. This approach is based on splitting of partitions in order to adapt the current fragmentation schema with the newly queries load. The authors have proved that this approach is very important than the selection of new fragmentation schema because this last requires the merging of all fragments followed by several splitting operations.

B. Methods of implementation

After selection of one or several optimization techniques, an implementation of the generated schema represents the next step. This implementation can be carried out manually by DW's administrators or automatically by DW administration tools and tuning tools.

1) Manual Implementation

In order to implement the selected DW fragmentation the administrator must seizes the necessary scripts then execute them in DW. These scripts are represented by scripts of fragmentation and scripts for rewriting queries. This task is difficult and requires an expertise of the administrator and an additional time.

In the most works that we have quoted, the administrators implement manually the selected schemas. This implementation is realized in the following steps:

- Generating HF schema by using one or several optimization algorithms or using other method.
- Seizing the scripts of fragmentation.
- Seizing the scripts of rewriting queries.
- Executing the some scripts in DW.

2) Automatic implementation

The automatic implementation of selected HF schema represents an approach to reducing the manual efforts and to minimizing the implementation time.

Very little works propose an automatic fragmentation approach. In [24], the authors propose a method based on exploitation of recent statistical data access for dynamic fragmentation in DW. This fragmentation is represented by the automatic selection and automatic implementation of selected schema whereas in [17], the authors propose the ParAdmin tool of administration and tuning of DW. Among other things, this tool assists the administrator in HF task and indexing task. To do this, it implements various algorithms for selecting an HF schema and different algorithms for selecting a binary join index (BJI) configuration. Similarly, the proposed tool supports isolated selection of HF technique, and multiple selections of HF and BJI. On their part, in [16] the authors present a tool named AdminFIC. This tool allows a combined selection of a fragmentation schema and BJI based on selection attributes classification. The selection of each technique is carried out by the genetic algorithm guided by a cost model.

The two proposed tools allow implementing a new HF schema. Even if this implementation will improve the DW performance, it is very expensive from the implementation view point on an already fragmented DW. The fragmentation of this latter requires several mergers of old partitions followed by several partitioning operations.

C. Data warehouses and web services

A web service can be represented by set of features exposed on internet or on intranet, either by applications or for applications in real time and without human intervention. Previous works have treated the combine between DW and web services. In [25], the authors presented a new distribution of DW called data web house (DWH). This pattern of DW distribution is based on Web service. For their part, the authors in [26] proposed an architecture of DW oriented web service. This architecture is based on construction of mini-cubic SOLAP for mobile customer whereas in [27], the authors has presented a prototype named Data warehouse fed with Web Services (DaWeS), and they have explored how ETL using the mediation approach benefits this trade-off for enterprises with complex data warehousing requirements. The pricipal gol of this approach is to reduce the manual effort. For their part the authors in [28], discussed the combinaison between web service and DW. This discution is based on opportunities for using DW at real-time through a web service in terms of data modeling, acquisition and analysis, analyses and designs the real-time data warehouse architecture.

In our approach, we use a web service for monitoring and improving automatically the DW performance and reduce both the manual efforts and the implementation cost.

III. Proposed Approach

In order to control and improve DW performance we present a new approach based on automatic and incremental fragmentation using a web service. However, the web service selects and implements an HF schema, and then it monitors the DW performance to avoid any degradation kind. This approach offers a remote administration and an automatic implementation of an HF schema. The web service allows the administrator to:

- View the DW state.
- View the frequent queries load.
- Be aware of new queries that integrate the load of frequent queries and queries that remain more frequent.
- Set the used optimization algorithm (the genetic algorithm in our case).
- Set the value of maximum number of fragments (W).
- Set the value of storage space reserved for redundant structures (indexes and materialized views).

We define our approach of automatic and incremental HF by taking inspiration from the works of Bouchakri, *et al.* [10] who presented a manual incremental HF approach and did not deal with the case of queries that are no longer frequents.

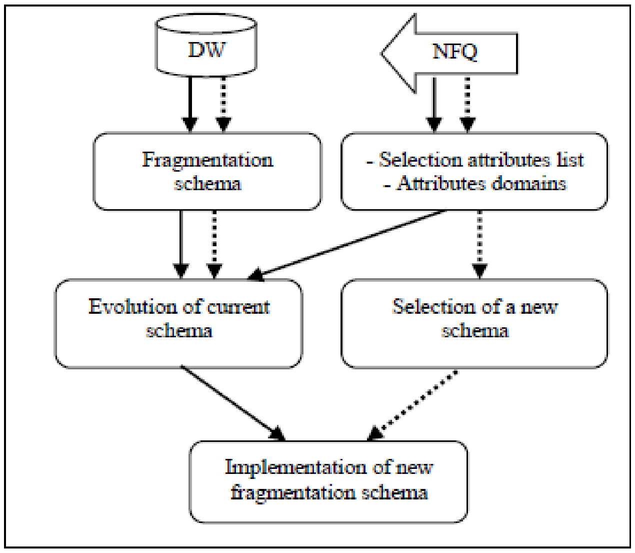

DW
NFQ
Fragmentation schema
- Selection attributes list
- Attributes domains
Evolution of current schema
Selection of a new schema
Implementation of new fragmentation schema

Fig. 1. General description of proposed approach.

Our approach follows the process of Fig. 1: First, the set of new queries are detected then the new frequent queries (NFQ) are determined. The NFQ determines a new fragmentation attributes or adds new domain extensions to the old attributes.

A. *Queries treatment*

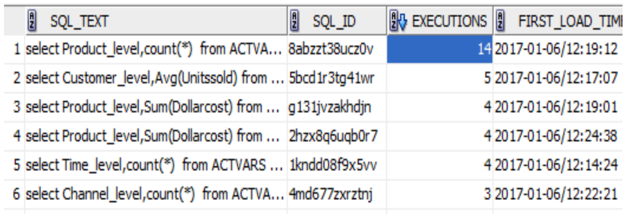
The web service uses a module for queries management. This module is responsible for processing all requests querying the DW. This treatment takes place in the following steps:

- Establish the frequent queries list.
- Compare the new list with the frequent queries list saved.
- Determine the list of NFQ.
- Determine the queries list that is no longer frequents.
- For each NFQ, determine the fragmentation attributes list and the selection predicates list.

1) Establish the frequent queries list: In order to retrieve the queries list that is querying it, Oracle stores all queries information in a view named: V$SQLAREA. The execution of a projection on this view makes it possible to construct a most frequent queries list that interrogates the DW. Fig. 2 presents an example of projection result on the V$SQLAREA view:

SELECT sql_text, sql_id, executions, first_load_time
FROM V$SQLAREA
ORDER BY executions DESC;

The execution of this query returns the result illustrated in Fig. 2.



	SQL_TEXT	SQL_ID	EXECUTIONS	FIRST_LOAD_TIM
1	select Product_level,count(*) from ACTVA...	8abzzt38ucz0v	14	2017-01-06/12:19:12
2	select Customer_level,Avg(Unitssold) from ...	5bcd1r3tg41wr	5	2017-01-06/12:17:07
3	select Product_level,Sum(Dollarcost) from ...	g131jvzakhdjn	4	2017-01-06/12:19:01
4	select Product_level,Sum(Dollarcost) from ...	2hzx8q6uqb0r7	4	2017-01-06/12:24:38
5	select Time_level,count(*) from ACTVARS ...	1kndd08f9x5vv	4	2017-01-06/12:14:24
6	select Channel_level,count(*) from ACTVA...	4md677zxrztnj	3	2017-01-06/12:22:21

Fig. 2. Example of most frequent queries list.

The "SQL_TEXT" field represents the query body, the "SQL_ID" field represents the query identifier, the "EXECUTIONS" field represents the executions count of query, and the "FIRST_LOAD_TIME" field represents the hour of query first execution. This last allow us to build the frequent queries list from a given date.

2) Compare the new list with the frequent queries list saved: The web service uses a table dedicated to saving the information of the frequent queries load. This information will be used later to update the queries load by comparing this queries list to the current list of frequent queries. In our approach, two queries are considered identical if they use the same selection attributes with the same predicates.

Consider the following three queries Q1, Q2 and Q3:

Request Q1

```
SELECT MAX(Prix)
FROM Product P, Purchase A, Suppliers S
WHERE P.IdP = A.IdP  AND P.IdF = S.IdF
        AND P.NomProduct ='P5'
        AND  F.Ville  = 'Casablanca';
```

Request Q2

```
SELECT MAX(Prix)
FROM Product P, Purchase A, Suppliers S
WHERE P.IdP = A.IdP  AND P.IdF = S.IdF
        AND P.NomProduit ='P4'
        AND  F.Ville  = 'Rabat';
```

Request Q3

```
SELECT AVG(Prix)
FROM Product P, Purchase A, Suppliers S
WHERE P.IdP = A.IdP  AND P.IdF = S.IdF
        AND P.NomProduit='P5'
        AND  F.Ville  = 'Casablanca';
```

The two queries Q1 and Q3 use the same selection attributes with the same predicates. They are considered identical even if the two queries use two different aggregation functions. On the other hand, the query Q2 uses other selection predicates; it is different from the two queries Q1 and Q3. The Table 1 shows an example of saving data for the three queries Q1, Q2, and Q3.

TABLE. I. QUERIES DATA SAVING TABLE

Request	ibute	predicate
Q1	NomProduit	P5
Q1	Ville	Casablanca
Q2	NomProduit	P4
Q2	Ville	Rabat
Q3	NomProduit	P5
Q3	Ville	Casablanca
Q3	Ville	Casablanca

From Table 1 data, the web service generates the domain of each selection attribute. AN example of attributes domains presentation is illustrated in Table 2.

TABLE. II. ATTRIBUTES DOMAINS

Ville	NomProduit
Casablanca	P4
Rabat	P5

When the web service detects a NFQ, it updates the queries information table. This update allows adding a new selection attributes or define a new extension of the old attributes in order to trigger the optimization task.

3) Determine the list of NFQ: An NFQ is a detected request and does not belong to the current list of frequent queries. An NFQ determines either new selection attributes or domain extensions of the old attributes.

4) Determine the queries list that are no longer frequents: A query that does not remain frequent is a query belonging to the old load of frequent queries. For this purpose, this query must be present in the query table filled in by the web service since it participated in the selection process of the fragmentation current schema. On the other hand, this query will not be selected from the frequent queries determined from the V$SQLAREA view.

5) For each NFR, determine the fragmentation attributes list and the selection predicates list: Each query is processed as a character string. Any selection attribute “AtS” of any NFQ must be presented in one of the following forms: “WHERE AtS”, “HAVING AtS”, “AND AtS”, and “OR AtS”. The selection attributes and the join attributes are distinguished by the fact that the join attributes are compared to other join attributes, while the selection attributes are compared by values. These values represent the selection predicates. The selection criterion is written as follows: “AtS opr PrS”, where AtS represents a selection attribute, “PrS” represents a selection predicate and “opr” designates one of the following comparison operators {=, <,>, <>, ≤, ≥,IN, NOT IN}.

Consider the following query Q4:

Request Q4

```
SELECT AVG(Prix)
FROM Product P, Purchase A, Fournisseurs F
WHERE P.IdP = A.IdP  AND P.IdF = A.IdF
        AND P.NomProduit ='P5'
        AND  (F.Ville  = 'Casablanca' OR F.Ville='Rabat');
```

From the query Q4, the module extracts the attributes list {NomProduit, Ville} and the selection predicates list {P5, Casablanca, Rabat}.

In order to consider the NFQ in the DW optimization process, two optimization methods can be defined: 1) implementation of a new fragmentation schema; and 2) the fragmentation current schema adaptation.

B. Implementation of a new fragmentation scheme (INFS)

Several research works treat the implementation of a new HF schema through the use of different optimization algorithms. The major problem encountered is that the INFS does not take into account the fragmentation previous scheme.

The INFS improves the DW performance, but it is very expensive from the implementation view point on an already fragmented DW. It requires several merge operations of the old partitions for reconstruct the no fragmented DW. Then use several partitioning operations for implement the new fragmentation schema.

C. Adaptation of current schema of fragmentation

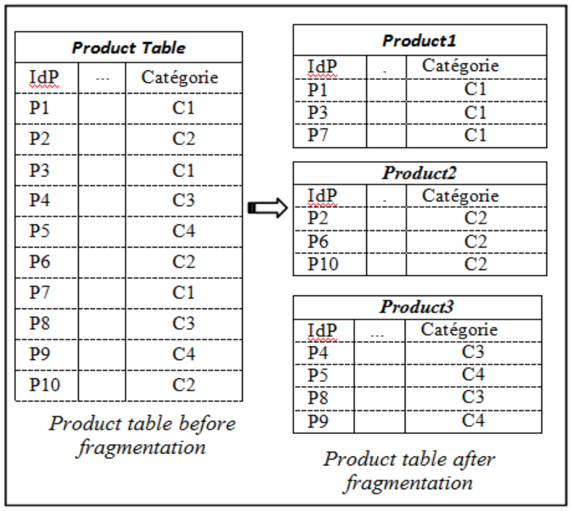
In order to take into account the execution of a NFQ, an adaptation of the fragmentation current schema must be performed. This adaptation can be achieved by bursting of the DW fragments in the appearance case of the new attributes or the new selection predicates, and by fusions in the disappearance case of the former attributes or the fragmentation predicates. Under Oracle, is used the SPLIT PARTITION function to split a fragment into two, and the MERGE PARTITION function to combine two fragments.

The SPLIT function increases the number of fragments, whereas the MERGE function decreases the number of fragments.

The adaptation of the fragmentation schema can be implemented in four different ways:

1) Splitting fragments

Consider a DW containing a facts table named Purchase and a dimension table named Product. The Product table data before and after fragmentation is shown in Fig. 3. The Products table is partitioned according to the model shown in Fig. 4.



Product Table

IdP	...	Catégorie
P1		C1
P2		C2
P3		C1
P4		C3
P5		C4
P6		C2
P7		C1
P8		C3
P9		C4
P10		C2

Product table before fragmentation

Product1

IdP	.	Catégorie
P1		C1
P3		C1
P7		C1

Product2

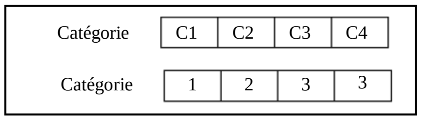
IdP	.	Catégorie
P2		C2
P6		C2
P10		C2

Product3

IdP	...	Catégorie
P4		C3
P5		C4
P8		C3
P9		C4

Product table after fragmentation

Fig. 3. Product table before and after fragmentation.



Catégorie	C1	C2	C3	C4
Catégorie	1	2	3	3

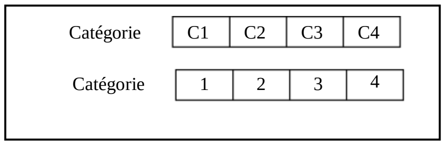
Fig. 4. Fragmentation schema of Product table.

Consider the following query Q5:

Request Q5

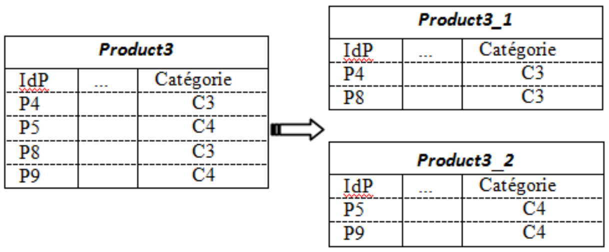
```
SELECT AVG(Prix)
FROM Purchase A,  Product P
WHERE A.Idp = P.IdP
   AND P.Categorie =  'C3'
```

The query Q1 is an NFQ, the response time optimization of this query leads to the application of the fragmentation schema illustrated in Fig. 5. The adaptation of the current fragmentation scheme must be carried out by the application of the function SPLIT on the third fragment of the Product table (Product3) and will produce a new fragment. The result of this adaptation is presented in Fig. 6.



Catégorie	C1	C2	C3	C4
Catégorie	1	2	3	4

Fig. 5. New fragmentation schema.



Product3

IdP	...	Catégorie
P4		C3
P5		C4
P8		C3
P9		C4

Product3_1

IdP	...	Catégorie
P4		C3
P8		C3

Product3_2

IdP	...	Catégorie
P5		C4
P9		C4

Fig. 6. Products table fragmented according to the new fragmentation scheme after application of the SPLIT function.

2) Splitting followed by fusion

If the number of fragments generated exceeds the maximum number of fragments (W), and if the web service does not detect other requests that do not remain frequent, it proceeds to classify the selection attributes according to their use frequencies by the queries. Then, it merges the sub-domains the months used until obtaining an HF schema that respects the W constraint. The following code represents the algorithm used to adapt the HF schema to the new frequent queries load.

Algorithm1: Splitting followed by fusion algorithm

```
input :
   Q : NFQ (New Frequent Query)
   FQL : Frequent Queries Load
   A: Set of fragmentation attributes
   P : Set of fragmentation predicates
   W : maximum number of fragments
output :
   Adapted fragmentation scheme
Begin
   Extract = { Predicates_Extract(Q)}
   For each predicate Pi in Extract Do
      Split = {add new fragments}
   End For
   FQL =  FQL + Q // updating the queries load
Calculate(FN) // Fragments number
   If FN>W Then
      Sort = {Scheduling of sub-domains}
      Do
         Merge = { Grouping of fragments }
      Until FN<=W
   End If
End
```

3) Merging fragments

Consider the DW shown in Fig. 3 which is fragmented according to the fragmentation schema shown in Fig. 4. Assume that the Q6 query that participated in the fragmentation process, no longer frequent, and also the other frequent queries using the same selection predicates do not exist.

Request Q6

```
SELECT AVG(Prix)
FROM Purchase A,  Product P
WHERE A.Idp = P.IdP
   AND P.Categorie =  'C2'
```

The fragment dedicated to the selection predicate “C2” is no longer useful and can deteriorate the response time of other requests. The grouping of this fragment with other fragments reduces the number of fragments loaded when executing the most frequent queries, which reduces the joins number to be performed for respond to queries. For example, if a request queries the DW with the clause “WHERE P.Categorie<> ‘C1’ ”. The response to this query requires access to two fragments (Product2 and Product3) instead of a single fragment, which consumes extra time.

The response time optimization of other requests leads to the implementation of the HF schema illustrated in Fig. 7. The adaptation of the fragmentation current schema must be carried out by applying the MERGE function to group the two fragments (Product2 and Product3). The result of this adaptation is presented in Fig. 8.

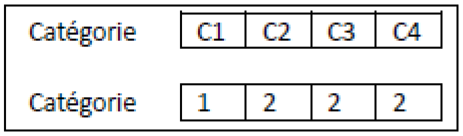


Fig. 7. New fragmentation schema for merging.

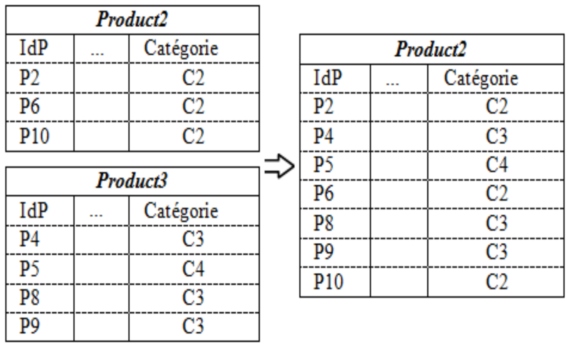

Product2
IdP
...
Catégorie
P2
C2
P6
C2
P10
C2
Product3
IdP
...
Catégorie
P4
C3
P5
C4
P8
C3
P9
C3
Product2
IdP
...
Catégorie
P2
C2
P4
C3
P5
C4
P6
C2
P8
C3
P9
C3
P10
C2

Fig. 8. Product table fragmented according the new fragmentation schema after application of MERGE function.

4) *Fusion followed by splitting*

Executing the fragment merging operation reduces the number of fragments. In this case, the web service triggers the optimization process of the other frequent queries. This process begins with the scheduling of the sub-domains which are constituted by several selection predicates. This classification is carried out following the use of frequency of the sub-domains by the queries. Then, it proceeds to the bursting the most used sub-domains until obtaining a new fragmentation schema. This new schema increases the number of fragments that does not exceed the W value.

The following code illustrates the algorithm used to adapt the HF schema to the new load of frequent queries:

Algorithm 2 : Fusion followed by splitting algorithm

```
Input :
    NFQ : Non Frequent Query
    FQL : Frequent Queries Load
    A: Set of fragmentation attributes
    P : Set of fragmentation predicates
W : maximum number of fragments
Output :
    Adapted fragmentation scheme
Begin
    Q = FQL – NFQ
Extract = { Predicates_ Extract(NFQ)}
    For each predicate Pi in Extract DO
        For each Qi in Q DO
            If Qi does not use Pi Then
             Merge = { Grouping fragments of predicate Pi}
            End If
        End For
    End For
    FQL = Q // updating the frequent queries load
Calculate (FN) //fragments number
    If FN<W Then
   Sort = { Scheduling of sub-domains}
        DO
            Split = {add new fragments}
        Until FN=W
    End If
End
```

IV. Tests and Results

We performed tests on APB1 benchmark generated under Oracle 11g. We use an already fragmented DW (static fragmentation) according to an HF schema generated from load of ten queries presented in Table 3.

In first step, we fixed the maximum number of fragments (W) to 10 then we started by executing new queries set illustrated in Table 4. First, we executed query Q11 several times. The web service detects the presence of an NFQ and triggers the selecting process of optimization. We have successively executed the new queries set presented in Table 4. Then we retrieved the execution cost of each request in different cases: static fragmentation, implementation of a new HF schema and adaptation of the current schema. In second step we varied the maximum number of fragments (W) and we executed simultaneously a set of new queries illustrate in Table 4 then we retrieved the execution cost in the three cases.

To calculate the queries execution cost, we used the method EXPLAIN PLAN offered by the Oracle optimizer.

TABLE. III. Frequent Queries Load List

Request	Attribute
Q1	class_level
Q2	family_level
Q3	line_level 3
Q4	division_level
Q5	year_LEVEL
Q6	class_LEVEL, retailer_level
Q7	year_LEVEL, class_level
Q8	month_level, retailer_level
Q9	year_level, retailer_level, line_level
Q10	month_level, division_level, all_level

TABLE. IV. NEW FREQUENT QUERIES LIST

Request	Attribute
Q11	group_level
Q12	year_level, month_level, class_level, group_level
Q13	month_level, group_level, retailer_level, all_level
Q14	month_level, division_level
Q15	customer_level

A. *Queries execution cost*

1) *Variation of Query*

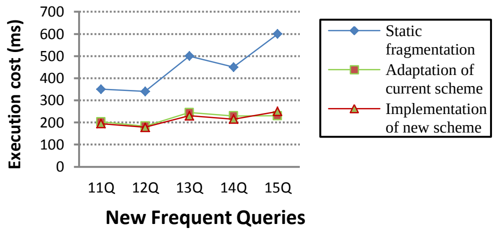


Fig. 9. Execution cost of new queries.

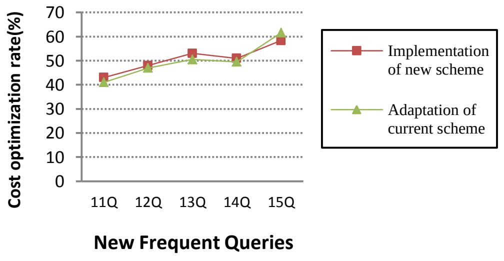


Fig. 10. Reduction rate of new queries execution cost.

Fig. 9 shows the execution time of new frequent queries in three cases: 1) static fragmentation; 2) implementation of new fragmentation schema; and 3) adaptation of the current fragmentation schema whereas Fig. 10 shows the reduction rate of the execution cost of new frequent queries.

We notice that the adaptation of the current schema and the INFS have the best cost compared to the static fragmentation. This is caused by the fact that the INFS generates a new optimal schema optimizing the total execution cost of the queries load. Thus the adaptation of the current schema generates, generally, a fragment for each frequent request. This results show the incremental fragmentation impact on DW performance.

2) *Variation of W*

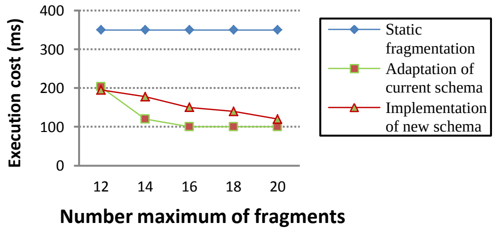


Fig. 11. Execution cost by varying W.

In order to see the influence of the W constraint on the optimization in the incremental fragmentation case, we have varied it and for each value we have executed both the list of new frequent queries illustrated in Table 4. We have noted the execution cost. The results illustrated in Fig. 11 show that the best cost is obtained by the approach of adapting the current fragmentation scheme. This is demonstrated by the fact that when executing a list of new frequent queries, the INFS approach generates a fragmentation scheme optimizing the cost of the whole load of frequent queries whereas the approach of current schema adaptation optimizes the total cost of the list of frequent new queries by generating new fragments under the maximum number of fragments constraint.

B. *Incremental fragmentation implementation time*

In second test, we recovered the incremental fragmentation implementation time through the web service. We compared the implementation time of a new HF scheme with the adaptation time of the current schema. For this purpose we have executed the new queries successively several times. In the other hand, we recovered the incremental fragmentation implementation time through the web service for different values of W. We compared the implementation time of a new HF schema with the adaptation time of the current schema. For this purpose we have executed the simultaneously a set of new frequent queries several times for each value of W.

The results obtained are summarized in the following illustrations:

1) *Variation of Query*

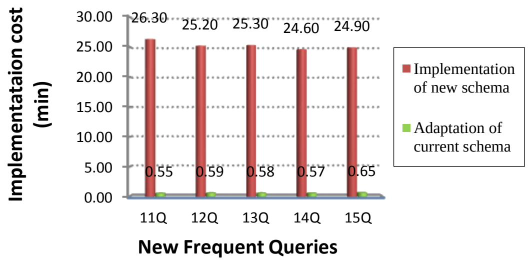


Fig. 12. Implementation cost of an incremental HF schema in the case of new frequent queries.

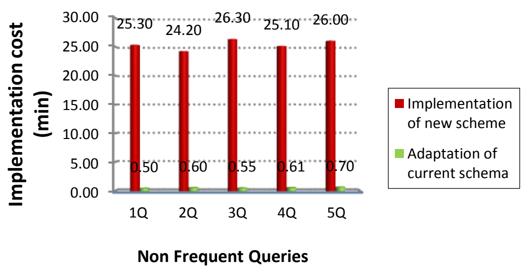


Fig. 13. Implementation cost of incremental HF schema in the case of queries that are no longer frequent.

Fig. 12 shows the implementation cost of incremental HF schema for different new queries detected by the web service. Whereas Fig. 13 illustrates the implementation time of the HF schema in the case of queries which participated in the first fragmentation process and which do not remain frequent. The two figures illustrate the implementation time of a new schema and the adaptation time of the current schema. The results obtained show that the implementation total cost of a new incremental HF schema is very expensive compared to the adaptation total cost of the current HF schema. This is proved by the fact that the INFS requires several merge operations of the old partitions for reconstruct the no fragmented DW. Then trigger an optimization algorithm or other method for generating a new optimal schema. And finally, execute several partitioning operations for implementing the new fragmentation schema in DW. The INFS cannot benefit from the old fragmentation schema. But, the adaptation of the current schema requires just the execution of some operations of partitioning, fusion or both. These operations are not very expensive.

2) Variation of W

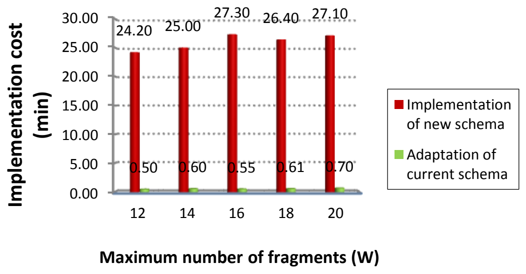


Fig. 14. Implementation cost of incremental HF schema for different values of W.

3) Variation of Number of New Frequent Queries

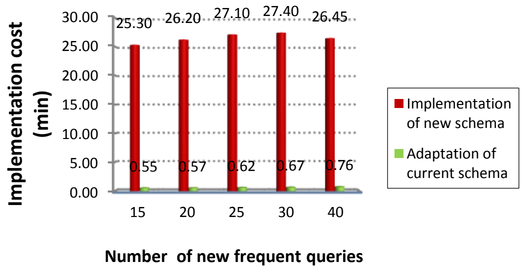


Fig. 15. Implementation cost of incremental HF schema for different number of new frequent queries.

Fig. 14 shows the implementation cost of incremental HF schema by the web service for different values of W, and Fig. 15 shows the implementation cost of incremental HF schema by the web service for different number of new frequent queries.

It is noted that the implementation cost of adaptation of the current schema does not vary much when we change the maximum number of fragments or the number of new frequent queries executed simultaneously. But the cost of implementing a new schema is very high compared to the cost of adapting the current schema. The cost of implementing a new schema can be written as follows: $CT_{imp} = C_{sel} + C_{imp}$.

4) C_{sel} : Represents the cost of selecting the new schema, it depends on the optimization algorithm and/or any other method used to select a schema close to the optimal schema. This selection generally requires a high cost compared to the implementation cost.

5) C_{imp}: Represents the implementation cost of the selected schema, it is the time needed to run the partitioning and/or merge scripts in the adaptation of the current schema case, and the time needed to execute the partitioning script in the INFS case.

In the adaptation approach, the C_{sel} is null because in this case the current schema is always validate. Thus, the web service product other fragments optimizing the new frequents queries. But in the INFS approach, the web service does not takes in the account the current schema, it proceeds by merging all fragments then splitting them until generating the schema already selected.

V. Conclusion

The goal of this work is to improve the DW performance. However, the proposed approach is based on incremental selection and automatic implementation of HF schema using a web service. Two incremental selection scenarios were proposed: 1) selection of a new HF schema; and 2) adaptation of the current schema to evolution of the frequent queries load. Both scenarios optimize queries execution cost. The implementation of the selected schema is completed automatically by the web service. The implementation cost of the selected schema according to second scenario is very low, which favors this scenario. Our approach differs from existing approaches since it makes it possible to monitor and improve automatically the DW performance. In the other works, the DW's administrators, generally, have to implement fragmentation manually, which requires a lot of time and an expertise. Very few works offer locally installed administration tools. On the other hand, although our approach allows improving the DW performance in reasonable time, it increases the availability of the latter since it will be manageable through the web.

We plan to study the possibility to automate the optimization of a varied queries load by combining fragmentation with other optimization techniques.

References

[1] P. Muley, Exploring the Scope of Data Warehouse and Business Intelligence Applications in Indian Higher Education Sector, IOSR Journal of Business and Management (IOSR-JBM) e-ISSN: 2278-487X, p-ISSN: 2319-7668. Volume 18, Issue 7 .Ver. I, PP 59-63, July 2016.

[2] M. K. Sohrabi*, V. Ghods, *Materialized View Selection for a Data Warehouse Using Frequent Itemset Mining*, Journal of Computers, Volume 11, Number 2, pp 140-148, March 2016.

[3] R. Nath, K. Hose, T. Pedersen, O Romero, A Programmable Semantic Extract-Transform-Load Framework for Semantic Data Warehouses, Journal of Information Systems March 3, 2017

[4] A Gosaina, Heena, *Materialized Cube Selection using Particle Swarm Optimization algorithm*, 7th International Conference on Communication, Computing and Virtualization 2016. s. Published by Elsevier.

[5] M. Shahid, U. Sheikh, B. Raza, M. Shah, A. Kamran, A. Anjum, Q. Javaid, *Application of Data Warehouse in Real Life: State-ofthe-art Survey from User Preferences' Perspective*, (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 7, No. 4, pp. 415-426, 2016

[6] BiriArun, T.V. V. Kumar, Materialized View Selection using Artificial Bee Colony Optimization, International Journal of Intelligent Information Technologies (IJIIT), vol. 13, issue 1, pp. 26-49, 2017

[7] A. Mateen, L. Chaudhary, *Reduce The Wastage of Data During Movement in Data Warehouse*, International Journal of Computer Applications (0975 – 8887) Volume 152 – No.8, pp. 20-24, October 2016

[8] I. Bojicic, Z. Marjanovic, *Domain/Mapping Model: A Novel Data Warehouse Data Model*, International Journal of Computers, Communications & Control (IJCCC) ·pp. 166-182, February 2017

[9] E. Sidi, M. El Merouani, E. A. Abdelouarit, *Star Schema Advantages on Data Warehouse: Using Bitmap Index and Partitioned Fact Tables*, International Journal of Computer Applications (0975 – 8887), Volume 134 – No.13, January 2016

[10] K. BOUKHALFA, L. BELLATRECHE, P. RICHARD, *Fragmentation Primaire et Dérivée: Étude de Complexité, Algorithmes de Sélection et Validation sous ORACLE10g*, LISI(Rapport de Recherche, N° 01 - 2008), Mars, 2008.

[11] L. BELLATRECHE, Techniques d'optimisation des requêtes dans les data warehouses, *Sixth International Symposium on Programming and Systems* (PS 2003), 2003, pp. 81-98.

[12] K. BOUKHALFA, L. BELLATRECHE, S. CAFFIAU, De l'optimisation de requêtes aux outils d'administration des entrepôts de données, RSTI - ISI (Ingénierie des Systèmes d'Information) (ISI 2009), vol. 13, n. 6/2008 , 2009.

[13] P. Kling, M. T.Ozsu, and D K. audjee, (2010). Distributed xml query processing: Fragmentation, localization and pruning. Technical report, University of Waterloo.

[14] A. Gacem, K. Boukhalfa, Nouvelle Approche Scalable Dédiée au Charges Volumineuses pour la fragmentation des Entrepôts de Données, *Proceedings of Maghreb Conference on Advances in Decision-making Systems* (ASD2013, pp. 61-73, 2013)

[15] M. BARR, L. BELLATRECHE, Approche dirigée par les fourmis pour la fragmentation horizontale dans les entrepôts de données relationnels, Revue : Nature & Technologie . n° 06, pp. 16- 24, Janvier 2012.

[16] R. BOUCHAKRI, *Une approche dirigée par la classification des attributs pour fragmenter et indexer des entrepôts de données*, Ph.D. Thesis, Ecole nationale Supérieure d'Informatique (ESI), Oued Semar, Alger, 2009.

[17] R. BOUCHAKRI, L. BELLATRECHE, Z. FAGET, Algebra-Based Approach for Incremental Data Warehouse Partitioning, *Proceedings of the 25th International Conference on Database and Expert Systems Applications* (DEXA 2014, pp. 441-448, 2014).

[18] Mahboubi H., *Optimisation de la performance des entrepôts de données XML par fragmentation et répartition*, Ph.D. Thesis, EDIIS, Universite Lumière Lyon 2, 2009.

[19] M. GHORBEL, K. TEKAYA, A. ABDELLATIF, Reducing the number of predicates for approaches to distribution of data warehouses, Revue des sciences et technologies de l'information, Volume 21 n°1 - pp.81-102, 2016

[20] S. Aissi, M. Gouider, T. Sboui, L. Ben Said, Enhancing spatial data arehouse exploitation: A SOLAP recommendation approach, *Proceeding of 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, 30 May-1 June 2016, Shanghai, China

[21] S. Gawali1, M. Vaidya, Selection and Maintenance of Materialized View Using Genetic Algorithm, International Journal Of Engineering And Computer Science ISSN:2319-7242 Volume 5 Issue 8 pp. 17715-17717, 2016

[22] V. Bhatnagar, N. Dahiya, M. Singh, Efficient Materialized View Selection for Multi-Dimensional Data Cube Models, International Journal of Information Retrieval Research, Volume 6 Issue 3, pp.52-74 July 2016

[23] M. Hamad, Y. Turky, *A Dynamic Warehouse Design Based on Simulated Annealing Algorithm*, Journal of Advanced Computer Science and Technology Research, Vol.6 No.1, pp1-8, March 2016

[24] H. Derrar, M. Ahmed-Nacer, O. Boussaid, *Exploiting data access for dynamic fragmentation in data warehouse*, International Journal of Intelligent Information and Database Systems 7(1):34 - 52, January 2013

[25] Z. Luo, Z. Kai-song, J Qiong, X. Hong-xia 1,Z. Kai-peng, Application of the Web Service Technology on the Data Warehouse System, Journal of Wuhan University of Technology, 2004

[26] E. Dubé, T. Badard, Y. Bédard, *A web service oriented architecture for the delivery of SOLAP mini-cubes to mobile clients*, International Journal of Geomatics and Spatial Analysis, Volume 19/2, pp.211-230, 2009

[27] J. Samuel, Towards a Data Warehouse fed with Web Services, Proceeding of *European Semantic Web Conference ESWC: The Semantic Web*: *Trends and Challenges* pp 874-884, 2014

[28] L. Jun, H. ChaoJu, Y. HeJin, Application of Web services on the real-time data warehouse technology, *Proceeding of International Conference on Advances in Energy Engineering* (ICAEE), 19-20 June 2010.