WEB SERVICE FOR INCREMENTAL DATA WAREHOUSES FRAGMENTATION ASSISTED WITH TEMPORARY MATERIALIZED VIEWS

[1]ABDELAZIZ ETTAOUFIK, [2]MOHAMMED OUZZIF

[1] RITM Lab., ESTC, CED Engineering Sciences, ENSEM, Hassan II University, Casablanca, Morocco.

[2] RITM Lab., ESTC Hassan II University, Casablanca, Morocco.

E-mail: [1]aettaoufik@gmail.com, [2]ouzzif@est-uh2c.ac.ma

ABSTRACT

The Data warehouse (DW) has become the essential component of almost every modern enterprise information system. It is proposed to collect and store heterogeneous and bulky data. DW represents a collection of thematic, integrated, non-volatile and histories data. It is feds from different data sources and it is highly required of the system to respond to complex online analytical queries. Generally, the analytical 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, different optimization techniques such as fragmentation, indexing, materialized views, and parallelism are used. 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 improve the DW performance from the new load of frequent queries. Distance Administration of DW through the web becomes important. The approach proposed in this paper aims at an incremental horizontal fragmentation technique assisted with temporary materialized view through a web service. This technique is based on managing the temporary materialized view in order to optimize the new frequent queries before proceeding on the implementation of the fragmentation. An experimental study on a real DW is carried out and comparative tests show the satisfaction of our approach.

Keywords: *Data Warehouse, Cloud Computing, materialized views, incremental fragmentation, frequent queries, web service.*

1. INTRODUCTION

DW has been proposed to store heterogeneous and voluminous data [13],[14],[15]. It is very promising technology and is being accepted rapidly across all domains of the industries [12],[16]. DW is an essential component of almost every modern enterprise information system [23]. It provides a new and wide idea of the company gives better performance of data base [17]. 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 [19]. 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 [20]. 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 can be used as the analysis axis.

The DW is often required by complex analytical 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 such fragmentation and materialized views. Horizontal fragmentation (HF) is one of the most used techniques, it consists in partitioning a dataset of DW to several disjoint partitions. Materialized view is a redundant structure, it is an efficient and effective OLAP query optimization technique to minimize query response time [23]. The selection of each optimization technique is madding from an attributes list and selection predicates extracted from the query load. Noting that the number of fragmentation sub-schemas 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 [1]. 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 horizontal fragmentation schema has been proven to be NP-complete problem [2],[3],[4],[5],[6],[7], there is not an exact solution to this kind of problem. For this purpose, several works use heuristics in order to select a schema close to the most optimal schema. Several works have dealt with the HF schema selection problem using different algorithms. Gacem et al. in [6] grouped these works into four categories: predicate-guided works, affinity-guided works, cost-guided works and 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 [1],[9]. In [10] the authors propose an approach for selecting an incremental HF schema. 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 scheme assisted with temporary materialized views using a web service. It is based on managing the temporary materialized view in order to optimize the new frequent queries before proceeding on the implementation of the fragmentation.

In section 2 we give a brief overview of related work on the selection problem of a HF scheme and materialized views. Then we present our approach of incremental optimization in section 3. Section 4 presents our experimental study on a physical DW. And we will conclude with a conclusion and perspectives.

2. DW 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 [20]. 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 [15]. 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 [15]. 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.

Optimizing queries execution time consists in selecting and implementing owe 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.

2.1. Selection of Fragmentation

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 implementation of the selected schema is performed either manually by the administrator of the DW or automatically by the DW's administration assist tools. Typically, DW' administrators use different optimization algorithms to select an optimal schema. Thus, the administrator always has to determine the frequent queries load. Therefore, Static selection remains less efficient since it does not adapt to the query load evolution. Several research studies deal with the fragmentation schema selecting problem. We will introduce in the following the works dealing with fragmentation and we will present some algorithms used in this context. We will focus on the HF of DW.

In [7], 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 scheme 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 other hand, the authors propose in [8] 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 scheme is generated by an original fragmentation method based on the k-means classification technique to control the number of fragments. Finally, they proposed an distribution approach of a XML data warehouse on a grid. Whereas in [6], 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 representative of the initial load. This new load will be used as a main load to split the DW. From their part, the authors in [4] are interested to implementing a DW horizontal fragmentation approach. This approach represents the following characteristics: (i) cost model based approach, (ii) it allows control of the generated number of fragments, and (iii) DW fragmentation is performed from a maximum number of fragments set by the administrator according to the following scenario : primary HF of the dimension tables according to which the derived HF 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 [22] 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 for phases: a preliminary phase for determinate the set of predicates selection, a coding phase for coding the predicates as binary matrices, a classification phase of these predicates using the k-means algorithm and a final phase to reduce the number of predicates.

2.2. Selection of Materialized Views

The problem of materialized views selection consists in determining the queries that must be materialized in order to optimize the cost of executing a given queries load. This optimization can be realized under certain constraints such as the storage space and the maintenance cost. The problem of materialized views selection in DW is an NP-complete problem, Many approaches have been presented in the literature to achieve good solutions to this problem [31]. We present in this part a few works dealing with the problem of materialized views selection. In [13], the authors propose an approach dealing with the materialized views selection problem. This approach is based on extraction of frequent motives. In this approach the set of queries is transformed to a transaction database where a transaction corresponds to a query and the items in a transaction are the predicates of the original query. Whereas in [23] the authors propose an approach for materialized view selection using artificial bee colony optimization. In this regard, they use an artificial bee colony (ABC) based view selection algorithm which has been adapted by incorporating N-point and GBFS based N-point random insertion operations, to select Top-K views from a multidimensional lattice. From their part, in [15] the authors present an approach for selecting a set for materialized views in order to optimize the queries execution time. To do this, the authors have implemented particle swarm optimization (PSO) algorithm on lattice framework to select an optimal set of views for materialization in data warehouse by minimizing query processing cost. Besides, the authors propose in [32] an approach for generating a set of sub materialized views from an existing materialized view. This newly views allow to answer user query without consulting the parent materialized view.

2.3. Selection of Fragmentation and Materialized Views

This two techniques share the some selection predicates and they can be in interaction because we can partitioning a materialized view and we can materializing a partition. The partitioning is controlled by the number maximum of fragments. Whereas the materialized view is controlled by the physical space reserved to redundant structures. Few studies has opted for selecting of fragmentation and materialized views. In [30], the authors proposed an approach for selection of partitioning, materialized views and indexes. They have proved that the combination of these three techniques reduces the query processing cost and the maintenance overhead significantly.

2.4. Methods of Implementation

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

2.4.1. Manual implementation

In order to implement the selected schema of optimization, the administrator must seizes the necessary scripts then execute them in DW. These scripts are represented by scripts of fragmentation, script for creating materialized views, scripts for rewriting queries and maintenance scripts. 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:

1) For fragmentation

- ✓ 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) For materialized views

- ✓ Generating a set of materialized views
- ✓ Seizing the scripts of creating the set of materialized views
- ✓ Executing the some scripts in DW

2.4.2. Automatic implementation

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

Very little works propose a dynamic implementation approach. In [18], the authors propose a method based on exploitation of recent statistical data access for dynamic fragmentation in DW. Whereas in [1], 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 scheme and different algorithms for selecting a binary join index (BJI) configuration. Similarly, the proposed tool supports isolated selection of HF technique, and multiple selection of HF and BJI. On their part, in [9] the authors present a tool named AdminFIC. This tool allows a combined selection of a fragmentation scheme and BJI based on selection attributes classification. The selection of each technique is carried out by the genetic algorithm guided by a cost model. On other hand, the authors propose a tool named DynaMat. This system allows creating and implementing the materialized views dynamically. It saves the queries evolutions and generates a set of materialized views optimizing the new queries load.

The first 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.

2.5. 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 [21], the authors presented a new distribution of DW called data webhouse (DWH). This pattern of DW distribution is based on Web service. For their part, the authors in [26] proposed architecture of DW oriented web service. This architecture is based on construction of mini-cubic SOLAP for mobile customer. Whereas in [24] the authors has presented a prototype named Data warehouse fed with Web Services (DaWeS), and they has explored how ETL using the mediation approach benefits this trade-off for enterprises with complex data warehousing requirements. The principal goal of this approach is to reduce the manual effort. For their part the authors in [25] discussed the combination between web service and DW. This discussion 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.

3. PROPOSED APPROACH

In order to control and improve DW performance we present a new approach based on automatic and incremental fragmentation assisted with temporary materialized views 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 and a selection incremental of materialized views. The service allows the administrator to:

- ✓ View the data warehouse 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 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 assisted with temporary materialized views by taking inspiration from the works of Bouchakri et al. [10] who presented a manual incremental horizontal fragmentation approach and did not deal with the case of queries that are no longer frequents. Our approach follows the process of Figure 1: First, the set of new queries are detected, and then the new frequent queries (NFQ) are determined. The NFQ determines a new fragmentation attributes or adds new domain extensions to the old attributes.

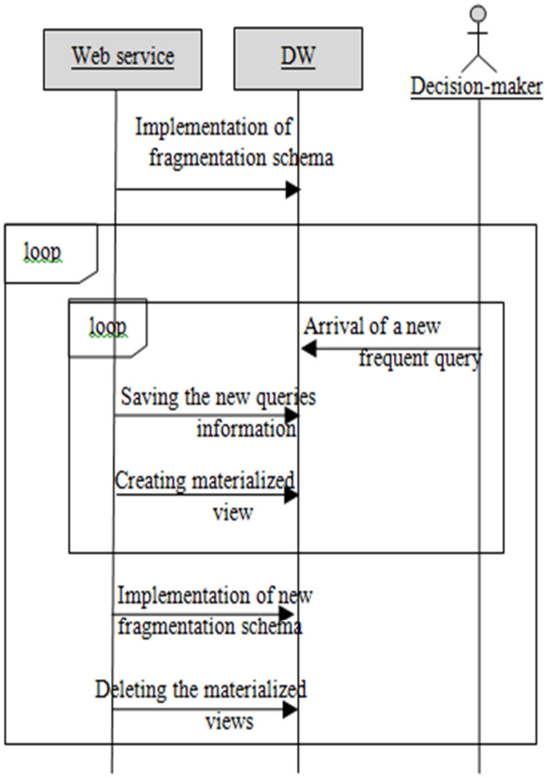

Web service
DW
Decision-maker
Implementation of fragmentation schema
loop
loop
Arrival of a new frequent query
Saving the new queries information
Creating materialized view
Implementation of new fragmentation schema
Deleting the materialized views

Figure 1: General Description Of Proposed Approach

3.1. 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:

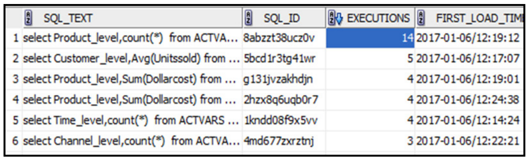
- ✓ Detect all NFQ
- ✓ For each NFQ, determine the selection attributes list and the selection predicates list.
- ✓ Create a new materialized view
- ✓ Establish the new frequent queries list;
- ✓ Compare the new list with the frequent queries list saved;
- ✓ Determine the queries list that are no longer frequents (NonFQ);
- ✓ Delete the materialized view associate to NonFQ

✓ Adapt the fragmentation schema
✓ Delete all materialized views

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

Figure 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 latter 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 1: Queries Data Saving Table

Request	Attribute	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. The attributes domains presentation is illustrated in Table 2.

Table 2: 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 defining 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 NonFQ : is a query that does not remain frequent and it is 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 scheme. 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 selection 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 scheme and (2) the fragmentation current scheme evolution.

3.2. Add the Temporary Materialized Views

Generally, access to a materialized view is less costly than accessing the tables since the materialized views generate few tuples than tables. We have chosen to automatically select temporary materialized views in order to assist the implementation of an incremental HF schema. For this purpose, upon arrival of an NFQ, the web service checks the availability of the memory space reserved to redundant structures. It then creates a temporary materialized view optimizing the NFQ in question.

The creation of materialized views is a transient step, it allows to optimize a query quickly before proceeding with the fragmentation task which is expensive in term of selection schema cost and implementation cost. To this end, the web service creates a materialized view optimizing the detected NFQ. The created materialized view will be declared in queries optimizer to be considered when this latter selects the most optimal execution plan.

When the web service does not find enough physical space for view creation and if does not detect a materialized view associate to a NonFQ, it triggers the fragmentation process, namely the adaptation of the current schema or the INFS and then it removes all temporary materialized views created. The following algorithm represents the management code for temporary materialized views.

Algorithm 1: Temporary materialized views processing algorithm

```
input :
        Q :  NFQ (New Frequent Query)
        FQL : Frequent Queries Load
        AS: Available space
        Vs: set of temporary materialized views
output :
        Object obj
     Begin
        V: new materialized view
        If size of V <AS Then
        Create  V= {Creating materialized view}
        Declare V={ declaring in queries optimizer }
        Add V to set of materialized views
        obj = V
        Else
        Fragment (scenario 1 / scenario 2)
        obj=Fragmented DW
        For each Vi in Vs Do
        delete Vi
        End For
        End If
End
```

3.3. Implementation of a New Fragmentation Schema (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 schema.

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.

3.4. The Adaptation of Current Schema of Fragmentation

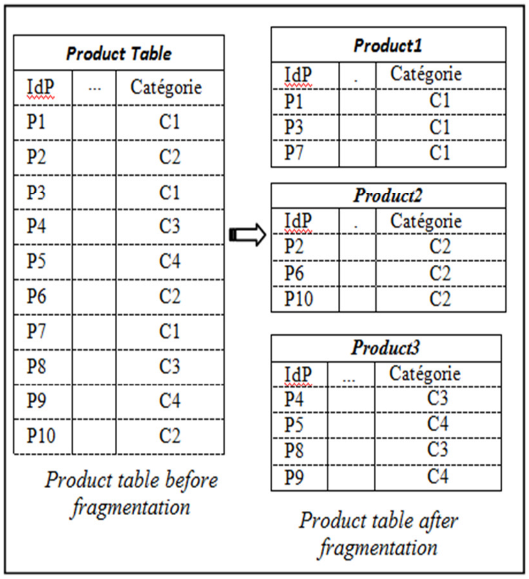
When the set of temporary materialized views occupying all space memory reserved to redundant structures, the web service triggers the processes of adaptation of the fragmentation current schema in order to take into account the execution of a NFQ. This adaptation can be achieved by bursting of the DW 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 evolution of the fragmentation scheme can be implemented in four different ways:

3.4.1.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 are shown in Figure 3. The Products table is partitioned according to the model shown in Figure 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

Figure 3: Product table before and after fragmentation

Catégorie	C1	C2	C3	C4
Catégorie	1	2	3	3

Figure 4: Fragmentation schema of Product table

Consider the following query Q5:

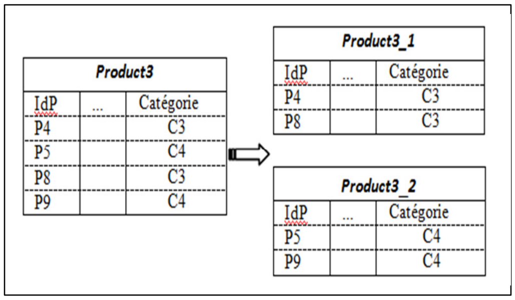
Request Q5

```
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 scheme illustrated in Figure 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 Figure. 6

Catégorie	C1	C2	C3	C4
Catégorie	1		3	4

Figure 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

Figure 6: Products table fragmented according to the new fragmentation scheme after application of the SPLIT function

3.4.2.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 scheme that respects the W constraint. The following code represents the algorithm used to adapt the HF schema to the new frequent queries load.

Algorithm 2 : 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
```

3.4.3.Merging fragments

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

Request Q6

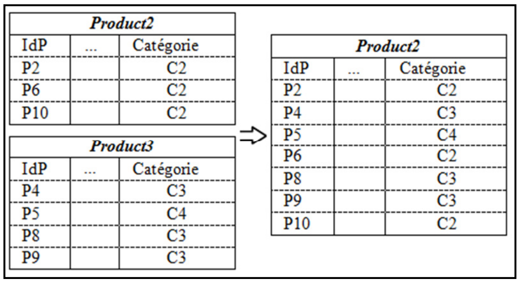
```
SELECT AVG(Prix)
FROM Purchase A, Product P
WHERE A.Idp = P.IdP
      AND P.Categorie = 'C2'
```

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 scheme illustrated in Figure 7. The adaptation of the fragmentation current scheme 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 Figure 8.

Catégorie				
Catégorie	1	2	2	2

Figure 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

Figure 8: Product table fragmented according the new fragmentation schema after application of MERGE function

3.4.4.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 frequency of the sub-domains by the queries. Then, it proceeds to the bursting the most used sub-domains until obtaining a new fragmentation scheme. This new scheme increases the number of fragments that does not exceed the W value.

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

Algorithm 3 : 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
```

4. TESTS AND RESULTS

We performed tests on APB1 benchmark [11] 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 and the memory space reserved to redundant structures fixed to 100MB, 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, it triggers by creating a materialized view optimizing the NFQ detected. 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, addition of materialized views, and implementation of a new HF schema and adaptation of the current schema. In second step we fixed the memory space reserved to redundant structures to 0MB, 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 Oracle optimizer.

Table 3: 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 4: 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

4.1. Queries Execution Cost

1. Variation of query

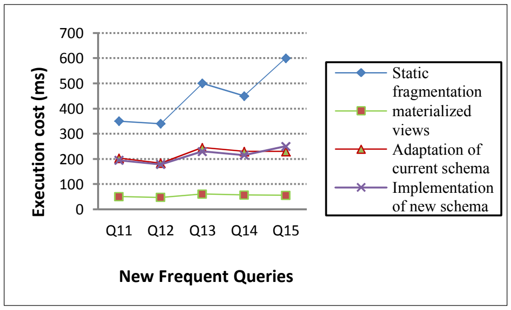


Figure 9: Execution Cost of new Queries

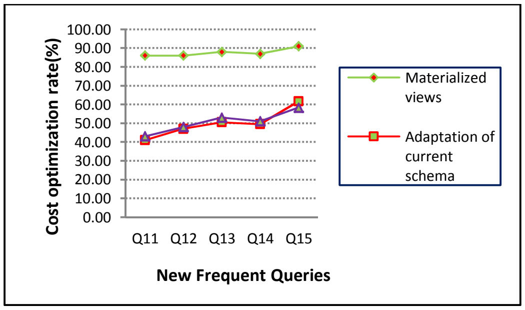


Figure 10: Reduction rate of new queries execution cost

2. Variation of number of new frequent queries

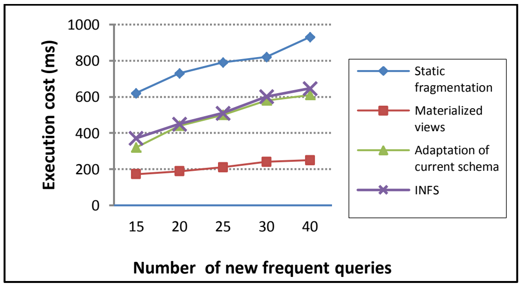


Figure 11: Execution cost for different number of new frequent queries

The results illustrated in figures 9, 10 and 11 show the impact of incremental fragmentation assisted with materialized views on DW performance. Figure 9 shows the execution cost of NFQ without incremental optimization, after adding materialized views and after incremental fragmentation. While figure 10 shows the reduction rate of the frequent new queries execution cost in the three cases. Thus, Figure 11 shows the execution cost of different number of queries in different cases.

We notice that the materialized views offer the best optimization compared to both adaptation of current schema and INFS. This is caused by the fact that the web service generates a materialized view for each NFQ by respecting the memory space reserved for redundant structures. Thus, the INFS generates a new optimal schema optimizing the total execution cost of the queries load. This results show the impact of incremental fragmentation assisted by materialized views on DW performance.

4.2. Implementation Cost

In second test, we recovered the implementation cost of temporary materialized views and incremental fragmentation implementation cost through the web service. We compared the implementation cost of temporary materialized views with the implementation cost of a new HF schema and the adaptation cost of the current schema. For this purpose we have executed the set of NFQ illustrate in table 4 successively several times. The results obtained are summarized in the following illustration.

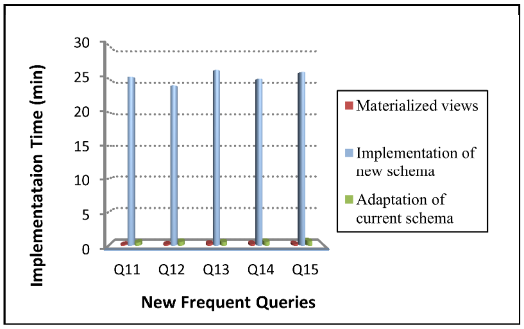


Figure 12:Implementation cost of materialized views an incremental HF schema in the case of new frequent queries

Figure 12 shows the implementation cost of incremental HF schema assisted with temporary materialized views for different new queries detected by the web service. The results obtained show that the implementation total cost of a new HF schema is very expensive compared to the adaptation total cost of the first HF schema and compared to implementation cost of temporary materialized views. This latter presents the best way to improve the DW performance. This is proved by the fact that the INFS does not take into account the fragmentation current schema, but the implementation of materialized views requires the generation an execution of scripts for creating materialized views optimizing the set of NFQ. The total cost for implementing a new schema can be written as follows: $CT_{imp} = C_{sel} + C_{imp}$.

- 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.
- 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 case of adding of temporary materialized views approach, the C_{sel} is null because in this case the web service does not selects any schema, it proceeds directly to generate and execute the creation script of materialized views. The same, the selection cost (C_{sel}) in case of adaptation of current schema is negligible because the current schema is always validate, and 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.

5. CONCLUSION

In this work, we adopted an approach of incremental HF assisted with by temporary materialized views using a web service. The materialized views allow optimizing the DW before proceeding to the fragmentation process. When the web service does find enough memory space it removes all materialized views and triggers the fragmentation task. Two incremental selection scenarios were proposed: (i) selection of a new HF schema and (ii) adaptation of the current schema to evolution of the frequent queries load. Both scenarios optimize queries response time. 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 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 DW performance in reasonable time, it increases the availability of DW 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 indexing and materialized views.

REFERENCES

[1] K. Boukhalfa, L. Bellatreche, P. Richard, F*ragmentation Primaire et Dérivée: Étude de Complexité, Algorithmes de Sélection et Validation sous ORACLE10g,* LISI(Rapport de Recherche, N° 01 -2008), Mars, 2008.

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

[3] 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

[4] 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.

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

[6] 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)

[7] 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.

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

[9] 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.

[10] 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).

[11] APB1 OLAP Council, "APB-1 olap benchmark, release II", http://www.olapcouncil.org/research/bmarkly.htm

[12] 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.

[13] 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.

[14] 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

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

[16] 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

[17] 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

[18] 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

[19] 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

[20] 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

[21] 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

[22] 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

[23] 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

[24] 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

[25] 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

[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] S. Aissi, M. Gouider, T. Sboui, L. Ben Said, Enhancing spatial data warehouse 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

[28] 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

[29] 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

[30] L. Bellatreche, M. Schneider, H. Lorinquer, M. Mohania, Bringing Together Partitioning, Materialized Views and Indexes to Optimize Performance of Relational Data Warehouses, *International Conference on Data Warehousing and Knowledge Discovery* DaWaK, pp. 15-25, 2004

[31] R. Goswami, D. K Bhattacharyya, *Approaches and issues in view selection for materializing in data warehouse*, International Journal of Business Information Systems, Vol 21, No 1, January 2016

[32] S. Ullah Khan, Data warehouse enhancement manipulating materialized view hierarchy, *International Conference on Digital Information Management (ICDIM),* 2013 Eighth, 10-12 Sept. 2013, Date Added to IEEE Xplore: 06 January 2014