March-April 2020
ISSN: 0193-4120 Page No. 5734 - 5742

Attribute-Based Fragmentation of XML Data Warehouse

Rihane Dkaich1, Ikram El Azami1 and Abdelaziz Mouloudi1
1MISC Laboratory, Faculty of sciences, Kenitra, Morocco
r.dkaich@gmail.com, akram_elazami@yahoo.fr, mouloudi_aziz@hotmail.com

Article Info
Volume 83
Page Number: 5734 - 5742
Publication Issue:
March - April 2020

Article History
Article Received*: 24 July 2019*
Revised*: 12 September 2019*
Accepted*: 15 February 2020*
Publication*: 29 March 2020*

Abstract:
Faced with the continuous increasing amount of data and the complexity of multidimensional queries, data warehouses must be split to be stored in distributed architectures and to enhance the performance of the queries. On the other hand, XML documents promise a considerable flexibility towards storing and manipulating different types of data warehouse hierarchies. In this article, we propose a method for the fragmentation of XML data warehouses by splitting the dimension documents via an attribute-based method, and splitting the fact document based on the fragments of the dimensions.

Keywords: *Data Warehouse, XML, Attribute, Predicate, Fragmentation, Dimension, Fact, Metadata, Decisional Queries.*

I. Introduction

Data warehouses have been playing a major role lately in the area of business intelligence, they are repositories that organize data in a way that allows the users to exploit it in the reporting operations, to analyze it via data-mining algorithms and to rely on it in the decision-making stage. However, the constantly increasing volume of data and the fact that data warehouses emphasize historized data, makes the matter of storing it and performing queries on it harder if not impossible. Therefore, many studies focused on fragmenting data warehouses in order to lighten the data storage and enhance the queries performance. The researches aimed also at finding the best structure of a data warehouse, to make the fragmentation and storing easier, and to ensure an optimal communication between data on different fragments. Some of these structures were XML documents as they are flexible for storing and interrogating multiple hierarchy types that the dimension instances might have.

The study that we carry out in this paper, is a method for splitting an XML data warehouse, organized in a similar structure to the star schema, using the dimension attributes that appear in a query workload of the most frequent queries that interrogate the data warehouse. Our method performs a syntactic analysis of the query workload to

March-April 2020
ISSN: 0193-4120 Page No. 5734 - 5742

define the attributes susceptible of participating in the fragmentation, then makes a size-based comparison to keep the highly effective attributes and eliminate the lesser effective. The dimensions are then fragmented based on the final set of attributes and the facts are fragmented based on the dimension fragments.

In the rest of this paper, we start by presenting multiple studies in the litterarure that focus on XML data warehousing and data warehouse fragmentation. Next, we introduce our method and explain our motives, the, we present and algorithm for generating the splitting attributes, and finally we propose an algorithm for dimensions fragmentation.

II. STATE OF THE ART AND RELATED WORK

The work conducted in the area of data warehouse fragmentation has been driven by the need to solve the problem of massive data warehouses and to enhance the performance of decisional queries. Most of the solutions were based on splitting the data warehouses on distributed architectures. Some of the researches focused on modifying the fragmentation methods designed for relational data such as the primary horizontal fragmentation and the derived horizontal fragmentation [9] and their adaptation to the multidimensional context while others aimed at conceiving new approaches adequate to the multidimensional structure and its different design schemas (star schema, snowflake schema or galaxy schema).

The work in [1] proposes a method for splitting data warehouses based on their logical data model. This fragmentation guarantees the conformity of the resulting models with the standards of the multidimensional design, and also the adequation of the fragmentation results with the needs and specifications of each site. This approach of fragments partitioning on multiple sites takes into consideration the frequency of use of each fragment, the constraint of communication between data and between sites and the constraint of data loading. It also integrates a control method that evaluates the performance of a given partitioning.

[12] proposes a set of approaches for the data warehouses optimization. These approaches are based on the utilization of three optimization techniques: the primary horizontal fragmentation, the derived horizontal fragmentation and the indexes of binary joins.

[13] focuses on reducing the number of predicates for data warehouses partitioning approaches, their method applies the optimization algorithm of the number of predicates via a semantic correlation, then reduces the resulting number via a geographic correlation.

The approach in [2] optimizes the problem of selection of the best vertical fragment schema, and a genetic algorithm is used for the data warehouse fragmentation. The approach distributes the relational schema of a data warehouse horizontally and then vertically in order to reduce the execution cost of the queries.

[11] suggest an algorithm that selects the best fragmentation schemas combining the mathematical cost model in [10] with the algorithm of simulated annealing.

The method in [14] simultaneously applies a vertical data fragmentation and a fragment allocation to the adequate sites across the network. To enhance the attributes clusters generation the Bond Energy Algorithm is used with better affinity measure, the algorithm then calculates the cost of allocating each cluster to each site and allocates each cluster to the most appropriate site.

[15] propose a big data warehouse design approach, and a partitioning method that aims to control the number of the resulting fragments via a predicate classification method based on the Bond Energy Algorithm. The method uses a complete and minimal list of predicates to begin with.

Our research in [3] focused on using cloud environments for XML documents data storage. We also proposed a method based on the MapReduce model for query processing and for the construction of OLAP cubes from distributed XML documents.

[4] propose two partitioning methods for XML data warehouses. The first one splits the dimensions based on the Algorithm of Predicate Construction [8], and splits the fact table based on the resulting fragments of the dimensions. The method defines the predicates that appear in a query workload and, using the COM-MIN algorithm in [6], it generates a complete and minimal set of predicates for each dimension, them splits the dimensions based on the appropriate set.

The second method utilizes the Affinity Based Strategies in [7] to define the dimension fragments. It creates a matrix of predicates use and then constructs the Affinity Matrix *(Aff)* from the first matrix and a vector of query frequencies, then uses a spanning tree to create cycles that contain each a set of predicates sharing values in *Aff*.

And last but not least, [5] propose a method whose purpose is to create XML data warehouse fragments that are included in queries with common characteristics. Their method uses the k-means classification technique in [16] to classify the predicates then builds a fragmentation schema and generates the XML warehouse fragments. The goal behind using k-means is to obtain an a priori known number ***k*** of classes (fragments) by positioning k centroids in the predicate space and minimizing the sum of the intra-class distance:

$$\sum_{i=1}^{k} \sum_{xj \in Ci} (x_j - \mu_i)^2$$

Where Ci, i = 1, ..., k are the k classes and μi is the centroid of the points xj ∈ Ci.

III. ATTRIBUTE-BASED FRAGMENTATION METHOD

Our approach consists of an attribute-based fragmentation of XML data warehouses. The choice of XML was driven by the fact that it allows to easily represent non or poorly structured data [17]. It is flexible to the storage and interrogation of the fact and dimension instances of the data warehouse despite the multiple hierarchy types they

March-April 2020
ISSN: 0193-4120 Page No. 5734 - 5742

might have (non-strict hierarchy, recursive hierarchy and incomplete hierarchy) and can deal with all the hierarchies at the same time [18]. However, many structures were proposed in the literature:

- A collection of XML documents, where each document stores one fact and the corresponding dimension instances (X-Warehousing);
- An XML documents that contains all the facts and another that stores all the dimension instances (XCube);
- A collection of XML documents where each fact and each dimension instance are stored in a separate XML document (XML-OLAP);
- An XML document that gathers all the facts and a separate XML document for each dimension (a similar organization to a star schema in the relational context).

Each of the structures above has its own pros and cons regarding the number of XML documents and the way the facts and dimensions are represented [18]. A performance evaluation of these different representations was conducted by Boukraa et al. [19], four data warehouses respecting each one of the four representations were generated, the XML documents were filled and a comparison of the processing time of a query workload of nine queries on each of the representations have shown that storing facts in an XML document and each dimension on a separate XML document guarantees the best performances as to the data warehouse interrogation.

Thus, we will be using this representation because it enhances the performance of the queries and because of its analogy with the classic star schema that we are familiar with. Figure 1 shows an example of the star schema.

Our approach was inspired by the work of [18] and [1]. It distributes the dimensions based on the attributes and the facts based on the fragments of the dimensions. We chose the attributes as a fragmentation criterion over the predicates in order to minimize the data redundancy by chosing to keep the attribute with the higher priority and eliminate the ones with lower priority when two or more attributes split the same dimension differently, this way we also minimize the number of fragments and won't have to impose this number to the partitioning algorithm as the results depend strongly of the choice of this number, and picking it arbitrarily doesn't necessarily produce an optimal fragmentation.

March-April 2020
ISSN: 0193-4120 Page No. 5734 - 5742

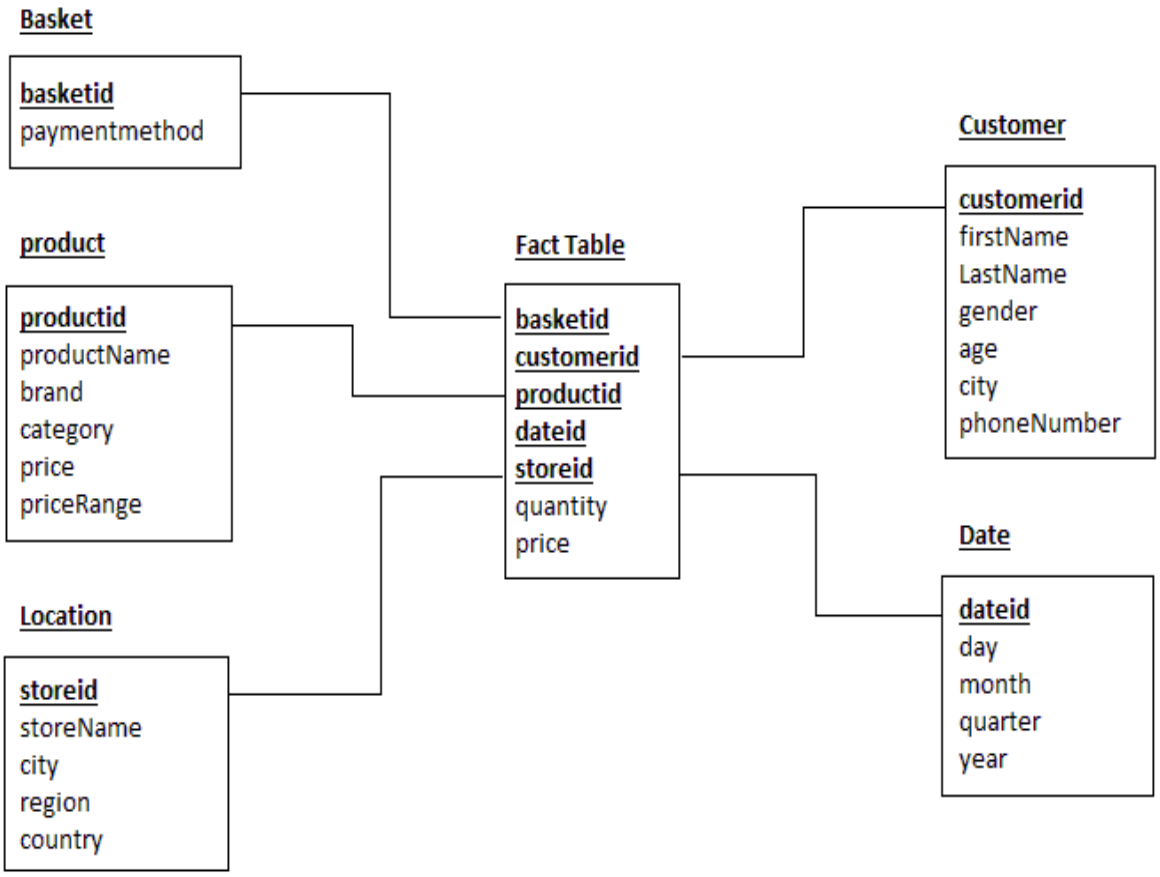

Figure 1: Star Schema Example

Analysis of the query workload

The first step of our method consists of defining the query workload ***W*** of the most frequent queries. ***W*** consists of a set of the queries that are frequently performed on the data warehouse on an everyday basis. Figure 2 gives an example of a decisional query expressed in XQuery.

```
for $comm in doc("factTable.xml")//commande,
    $prod in doc("product.xml")//product,
    $cust in doc("customer.xml")//customer

where $comm/product=$prod/@id
and $comm/customer=$cust/@id
and $prod/price > 1000
and $cust/age < 30

return
<achat>
  <productName>{data($prod/name)}</productName>
  <productPrice>{data($prod/price)}</productPrice>
  <customerAge>{data($cust/age)}</customerAge>
  <quantity>{data($comm/quantity)}</quantity>
</achat>
```

Figure 2: An XQuery decisional query

The query performs a join operation between the documents "factTable.xml", "product.xml" and

March-April 2020
ISSN: 0193-4120 Page No. 5734 - 5742

"customer.xml", selection operations and return data in an XML structure.

The second step is to perform a syntactic analysis of each query from the query workload. This operation consists of extracting useful information from the queries that will allow us to perform the fragmentation of the dimensions. The needed data is retrieved in the where clause of the queries, where we can determine the attributes and the information related to each one of them. Figure 3 gives an example of the predicates extracted from the where clause of a query, where **$prod/price** is an attribute and **$prod/price > 1000** is the predicate exploiting it.

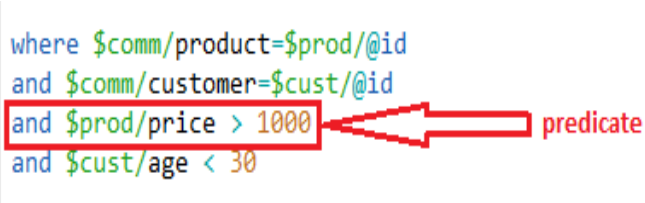

Figure 3: example of a predicate extracted from the where clause of a query

The attribute Meta-Data document

Once we define the attributes that appear in the query workload, we proceed to the extraction to all the data that is related to the attributes and that will participate in the fragmentation, the data in question is the dimension that the attribute belongs to, and the queries exploiting it, for each query we look for the predicate in which the attribute appears, the size of the result fragment (in Mo) if a fragmentation based on the predicate should take place and the frequency of the query. The extracted data will then be stored in an attribute meta-data XML document, figure 4 is an example of this document.

```
<attribute id="a1">
  <dimension>d3</dimension>
  <query id="q1">
    <frequency>50</frequency>
    <predicate>p5</predicate>
    <fragmentSize>0.251</fragmentSize>
  </query>
</attribute>

<attribute id="a2">
  <dimension>d1</dimension>
  <query id="q3">
    <frequency>20</frequency>
    <predicate>p6</predicate>
    <fragmentSize>0.153</fragmentSize>
  </query>
  <query id="q2">
    <frequency>30</frequency>
    <predicate>p3</predicate>
    <fragmentSize>0.095</fragmentSize>
  </query>
</attribute>

<attribute id="a3">
  <dimension>d3</dimension>
  <query id="q4">
    <frequency>10</frequency>
    <predicate>p1</predicate>
    <fragmentSize>0.312</fragmentSize>
  </query>
  <query id="q3">
    <frequency>20</frequency>
    <predicate>p2</predicate>
    <fragmentSize>0.224</fragmentSize>
  </query>
</attribute>
```

Figure 4: The Attribute-Metadata Document

The splitter attributes

In this step we generate a binary attribute-dimension matrix in where an element A_{ij} equals 1 if the attribute j is in the dimension i and equals 0 otherwise. Figure 5 is an example of an attribute-dimension matrix.

March-April 2020
ISSN: 0193-4120 Page No. 5734 - 5742

	a1	a2	a3	a4	a5	a6	a7
d1	0	1	0	0	0	0	0
d2	0	0	0	0	0	1	0
d3	1	0	1	0	0	0	0
d4	0	0	0	1	1	0	0
d5	0	0	0	0	0	0	1

Figure 5: the attribute-dimension matrix

From the attribute-dimension matrix we can spot the conflicted attributes (a1 and a3 from the dimension d3, and a4 and a5 from the dimension d4), those attributes share the same dimension and we will have to choose among them the appropriate attribute for the fragmentation.

XML file size has no limit, but it takes memory (RAM) as file size of the XML file, so long XML size file parsing considerably impacts the performance, hence, we take the size of parsed data during queries execution into consideration when we split our XML documents.

Finally, with the algorithm 1 we generate a set DS of dimension splitters, this set contains the attributes participating in the fragmentation, DS is initialised with all the attributes with no conflicts and each high priority attribute that we choose from the conflicted attributes is appended to DS.

Algorithm 1: Splitter Attributes Algorithm

Input: ***A***: a set of all the attributes, *"metadata.xml"*
Output: $DS \subset A$
1: $DS = \emptyset$
2: for each attribute $ai \in A$
3: If no attribute in *"metadata.xml*" has the same dimension as *ai*
4: $DS=DS \cup \{ai\}$
5: else
6: $a'=ai$
7: **for each** $aj \in A$ where *ai* and *aj* are in the Same dimension
8: **If** $Sa(ai)>Sa(aj)$
9: $a'=ai$
10: $DS = DS \cup \{a'\}$
11: return *DS*

The choice is based on an approximation of the size of data that needs to be charged by the query for every fragmentation based on one of these attributes.

$$Sa = \sum_{i=1}^{n} fi * si + (\sum_{j=1}^{m} fj) * S$$

Where **Sa** is the approximate size of loaded data if we split the corresponding dimension based on the attribute **a, n** is the number of the queries that exploit the attribute **a**, ***fi*** is the frequency of the query **i** that use the attribute **a**, ***si*** is the size of the fragment of the dimension based on the predicate where **a** appears in the query **i**, **m** is the number of the queries that exploit other attributes in the same dimension as **a**, ***fj*** is the frequency of the query **j** and ***S*** is the size of the dimension that contains the attribute **a.**

The attribute with the lower size of loaded data is the attribute with the higher priority.

Data fragmentation

Once we define the attributes that will participate in the fragmentation, algorithm 2 splits the dimensions according to the predicates corresponding to the attributes in DS that it extracts from the metadata document, and creates a fragmentation metadata document *"fragMD.xml"* that it updates with the queries that use each fragment. This document will serve later in creating the fact fragments.

The fragmentation of the fact document is similar to the fragmentation of the dimensions it looks in the document "fragMD.xml" for the dimension fragments whose predicates appear in the same queries and creates a fact predicates set **FP** that is used in the fragmentation.

March-April 2020
ISSN: 0193-4120 Page No. 5734 - 5742

Algorithm 2: Dimension Fragmentation Algorithm

Input: *D*: the dimensions, *"metadata.xml", DS*
Output: F: the fragmented dimensions, "fragMD.xml"

1: $F = \emptyset$
2: Create("fragMD.xml")
3: for each $ai \in DS$
4: find the suitable dimension $d \in D$ and the predicates *P* exploiting *ai* from *"metadata.xml"*
5: *Fd=**split_dimension**(d,P)*
6 : $F = F \cup Fd$
7: **Update**("fragMD.xml")
8: return *F*

Algorithm 3: Fact Fragmentation Algorithm

Input: *F*: the dimension fragments, *"fragMD.xml"*
Output: *FF*: the fragmented fact

1: $FP = \emptyset$
2: for each $fi \in F$
3: *P'= **Find_Predicate**(fi)*
4: **for each** *fj* in *F-{fi}*
5 : **if** *fi* and *fj* are parsed by the same query //in "fragMD.xml"
6 : *p = **Find_Predicate**(fj)*
7: *P' = P'∪{p}*
8 : *F = F-{fi}*
9 : $FP = FP \cup P'$
10: *FF = **Split_Fact**(FP)*
11: return *FF*

IV. CONCLUSION

Partitioning a data warehouse is an overriding operation given the amount of data it stores. The fragmentation method is a crucial choice to make as it impacts the communication between data, the performance of the queries and the number of fragments.

In this article, we proposed a fragmentation method for XML data warehouses organized in a similar structure to a star schema. It splits the dimension documents based on the attributes extracted from a query workload of the most frequent queries, then splits the fact document in adequation with the dimension fragments. The improvement of the method via classifying the attributes will be the purpose of our next study.

REFERENCES

[1]. Tekaya K (2011) Fragmentation et allocation dynamiques des entrepôts de données. Thèse de doctorat, Faculté des sciences de Tunis.

[2]. Ziyati E (2010) Optimisation de requêtes OLAP en Entrepôts de Données : Approche basée sur la fragmentation génétique. Thèse de doctorat, Faculté des Sciences Rabat.

[3]. Dkaich R, El Azami I, Mouloudi A (2017) XML OLAP Cube in the Cloud Towards the DWaaS. International Journal of Cloud Applications and Computing (IJCAC), 7(1), 47-56.

[4]. Mahboubi H, Darmont J (2009) Enhancing XML Data Warehouse Query Performance by Fragmentation. 24th Annual ACM Symposium on Applied Computing (SAC 09), Hawaii, USA. ACM Press.

[5]. Mahboubi H, Darmont J (2008) Data Mining-based Fragmentation of XML Data Warehouses. ACM 11th International Workshop on Data Warehousing and OLAP (CIKM/DOLAP 08), Napa Valley, USA (pp. 9–11). ACM Press.

[6]. Ozsu M T,Valduriez P (1999) Principles of Distributed Database Systems, Second Edition. Prentice-Hall.

[7]. Navathe S B, Karlapalem K, Ra M (1995) A Mixed Fragmentation Methodology for Initial Distributed Database Design. Journal of Computer and Software Engineering, 3(4)

[8]. Noaman A Y, Barker K (1999) A Horizontal Fragmentation Algorithm for the Fact Relation in a Distributed Data Warehouse. ACM International Conference on Information and Knowledge Management (CIKM 99), Kansas City, USA, pages 154–161. ACM, 1999.

[9]. Bellatreche L (2000) Utilisation des Vues Matérialisées, des Index et de la Fragmentation dans la Conception logique et Physique d'un Entrepôt de Données. PhD

thesis, Université de Clermont-Ferrand II.
[10]. Bellatreche L (2008) Bitmap join indexes and data partitioning. Encyclopedia of Data Warehousing and Mining 2nd Edition, 5,37,38.
[11]. Bellatreche L, Boukhalfa K, Richard P (2011) Primary and referential horizontal partitioning selection problems. Concepts, Algorithms and Advisor Tool
[12]. Boukhalfa K (2009) De la conception physique aux outils d'administration et de tuning des entrepôts de données. Thèse de doctorat, Université de Poitiers
[13]. Ghorbel M, Tekaya K, Abdellatif A (2014) Réduction du nombre des prédicats pour les approches de répartition des entrepôts de données. ASD 2014, 29-31 Mai 2014, Hammamet, Tunisie.
[14]. Rahimi H, Parand F, Riahi D (2015) Hierarchical simultaneous vertical fragmentation and allocation using modified Bond Energy Algorithm in distributed database. Saudi Computer Society, King Saud University. Applied Computing and Informatics
[15]. Ghorbel M, Tekaya K, Abdellatif A (2018) Modélisation et répartition d'un Big Data Warehouse. Big data & Applications 12th edition of the Conference on Advances of Decisional Systems ASD 2018
[16]. Pham D, Dimov S, Nguyen C (2004) An Incremental K-means algorithm. Journal of Mechanical Engineering Science, 218(7), 783–795.
[17]. Darmont J, Boussaïd O, Ralaivao J C, Aouiche K (2005) An Architecture Framework for Complex Data Warehouses. 7th International Conference on Enterprise Information Systems (ICEIS 05), Miami, USA (pp. 370–373).
[18]. Mahboubi H (2008) Optimisation de la performance des entrepôts de données xml par fragmentation et répartition. Thèse de doctorat, Université Lumière Lyon 2.
[19]. Boukraa D, BenMessaoud R, Boussaïd O (2006) Proposition d'un Modèle physique pour les entrepôts XML. Atelier Systèmes Décisionnels (ASD 06), 9th Maghrebian Conference on Information Technologies (MCSEAI 06), Agadir, Morocco.