Hybrid Fragmentation of XML Data Warehouse Using K-Means Algorithm

Mohamed Kechar and Safia Nait Bahloul

University of Oran, LITIO Laboratory, BP 1524, El-M'Naouer, 31000 Oran, Algeria
{mkechar,nait1}@yahoo.fr

Abstract. The efficiency of the decision-making process in an XML data warehouse environment, is in a narrow relation with the performances of decision-support queries. Optimize these performances, automatically contribute in improving decision making. One of the important performances optimization techniques in XML data warehouse is fragmentation with its different variants (horizontal fragmentation and vertical fragmentation). In this paper, we develop a hybrid fragmentation algorithm combining a vertical fragmentation based on XPath expressions and a horizontal fragmentation based on selection predicates. To control the number of fragments, we use the K-Means algorithm. Finally, we validate our approach under Oracle Berkeley DB XML by several experiments done on XML data, derived from the XWB benchmark.

Keywords: XML Data Warehouse, Hybrid Fragmentation, XPath Expressions, Selection Predicates.

1 Introduction

With the emergence of XML, a large amount of heterogeneous XML data is manipulated by enterprises. Various works [11], [16], [26], and [27] have proposed to integrate and store the XML data to exploit them in decision-making (the birth of XML data warehouses). However, in a decision-making system, time is considered as a major constraint. The managers of the company should take appropriate decisions timely. Unfortunately, their decisions are based on analyzing done on the results of several quite complex queries, called decision-support queries. Characterized by join operations, selection operations and aggregation operations, the response times of these queries is generally quite high. Optimize the performances of such queries, contributes significantly to the improvement of decision-making. In this context, several performance optimization techniques have been proposed in the field of data warehouses, such as indexes, materialized views and data fragmentation. Among these techniques, fragmentation has received much interest by the researcher's community. Its efficiency has been proven in the relational databases [1], [13], [25], the object-oriented databases [5,6] and the relational data warehouses [3], [4], and [14]. However, few works on fragmentation have been proposed in the XML data warehouses. To fragment an XML data warehouse modeled by star schema [10], the authors in [22]

Y. Manolopoulos et al. (Eds.): ADBIS 2014, LNCS 8716, pp. 70–82, 2014.

use the primary horizontal fragmentation and derived horizontal fragmentation. They use the K-Means algorithm to group the selection predicates into disjoint classes defining the horizontal XML fragments. In [23], the authors propose two horizontal fragmentation techniques of an XML data warehouse. The first is based on the concept of minterms [25] and the second is based on predicates affinities[30]. The authors in [28], propose different models of partitioning of a multi-version XML data warehouses. They propose the partitioning model of XML documents, the partitioning model based on the XML schema of the XML data warehouse and the mixed model that combines the first two models. The approach proposed in [9], vertically fragment the XML data warehouse based on all frequently paths used by queries. The authors use the association rules to find the set of paths from which they derive the vertical fragmentation schema. To the best of our knowledge, no hybrid fragmentation approach, combining the vertical fragmentation and the horizontal fragmentation has been proposed to date in the context of XML data warehouse. Although its efficiency has been already proven in the relational databases [24], the Object Oriented databases [2], and the relational data warehouses [15]. For this fact, we present in this paper a hybrid fragmentation of an XML data warehouse. We partition vertically the structure of the data warehouse into vertical fragments by a classification of XPath expressions. Then we fragment horizontally the XML data of each vertical fragment by a classification of selection predicates. We use in our classification the K-Means algorithm[18] with the euclidean distance. In addition to its simplicity and its rapidity, it allows us to control the number of fragments.

The remainder of this paper is organized as follows. In Sect.2, we survey the different multidimensional models and we focus on the flat model that we use as a reference model. In Sect.3 we detail our hybrid fragmentation. Finally, we present some experimental results of our evaluations in Sect.4.

2 Multidimensional Modeling of XML Data

In the literature, different XML data warehouse models have been proposed. In [11] the XML data Warehouse is represented by a collection of homogeneous XML documents. Each XML document represents a fact with its measures and its dimensions. In [8], the authors propose the hierarchical model in which they use a single XML document containing all facts and all dimensions. Each fact is represented by an XML element containing its measures and the references to the XML elements containing its dimensions. In addition to the hierarchical model, they define the flat model represented by a single XML document. Each fact in this document is represented by a single XML element containing its measures and its dimensions in the form of XML sub-elements. The XCube model proposed by [17], uses an XML document named FaitsXCube to represent facts and another XML document named DimensionsXCube to represent dimensions. By analogy to the relational star model [19], the authors in [10] and [27], model the XML data warehouse by a central XML document containing all facts with their measures surrounded by several XML documents representing the dimensions. These XML documents are linked by primary keys and foreign keys.

Performance evaluations of these different models of XML data warehouses have been conducted in several works. For example in [8], the authors have conducted evaluations and comparisons of performances between the hierarchical model, the flat model, and XCube model. They noticed that the flat model provides better performance compared with the other two models, except that it introduces redundancy of the dimensions. A performance comparison between the star model, the flat model, and the model proposed in [11] has been carried out in [10]. The authors have shown that the star model provides improved performance for queries that use two joins. However, from three joins, the performances decrease in favour of the flat model. In order to improve the response time of XQuery queries, a join index has been proposed in [20]. By carefully inspecting this index, we found that his representation is in compliance with a flat model (a single XML document containing all the facts with their measures and dimensions). Based on these performance evaluations, we use the flat model depicted in Fig.1) as a reference model to represent the XML data warehouse.

In the following sections, we describe our hybrid fragmentation approach.

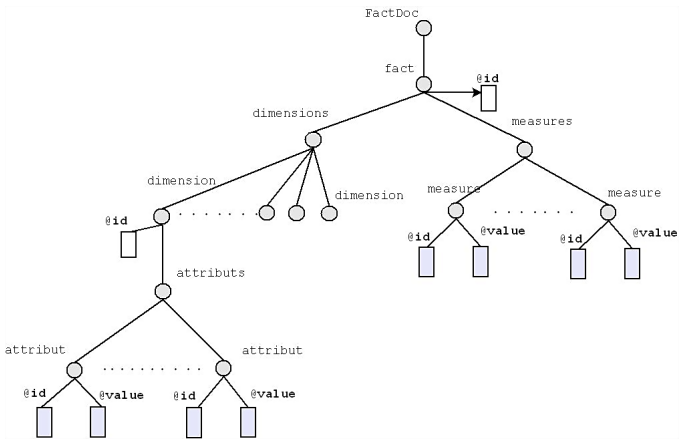

Fig. 1. Reference Model of the XML Data Warehouse

3 Hybrid Fragmentation of the XML Data Warehouse

In this section, we detail the three main phases of our hybrid fragmentation approach. For the remaining sections, letters E, T and D refer respectively to, the set of the names of distinct XML elements, the set of names of distinct XML attributes and the set of distinct data values. Δ represents the XML data warehouse modeled by the flat model and W is the workload executed on Δ. We use in our approach the two concepts of the XPath expression (Definition 1) and the selection predicate (Definition 2).

Definition 1. *A path expression EC is a sequence* $root/e_1/.../\{e_n|@ak\}$*, with* $\{e_1, ..., e_n\} \in E$ *and* $@a_k \in T$*. The expression EC may contain the symbol '*' which indicates an arbitrary element of E, the symbol '//' indicating a sequence of elements* $e_i/.../e_j$ *such as* $i < j$ *and the symbol '[i]' which indicates the position of the element* e_i *in the XML tree [7].*

Definition 2. *A selection predicate is defined by the expression* $Pred_j := P\ \theta\ value \mid \phi_v(P)\ \theta\ value \mid \phi_b(P) \mid Q$*, with P a terminal XPath expression,* $\theta \in \{=, <, >, \leq, \geq, \neq\}$*, value* $\in D$*,* ϕ_v *is an XPath function, that returns values in D,* ϕ_b *is a Boolean function and Q denotes an arbitrary XPath expression [7].*

3.1 Vertical Fragmentation Based on XPath Expressions

We define the vertical fragmentation of XML data warehouse Δ, by partitioning its structure into K vertical fragments $VF_1, ..., VF_K$. Each fragment is a projection of a set of XPath expressions frequently accessed by the workload. In this phase we proceed by:

Extraction of XPath Expressions. Each XQuery query belonging to W is in conformity with the basic syntax of the $FLWOR$ expression *(For, Let, Where, Order by, Return)* [29]. For each query, we perform a syntactic analysis by clause and we extract all its XPath expressions. Thus we identify the overall set of XPath expressions EC used by the workload W.

XPath Expressions-Queries Usage Matrix (XPQUM). Defines the use of each XPath expression by the set of queries. We create in $XPQUM$, a line i for each XPath expression $EC_i \in EC$ and a column j for each query $Qj \in W$. If the query Q_j use EC_i, then $XPQUM(i, j) = 1$, else $XPQUM(i, j) = 0$.

Vertical Fragmentation. In this step, we use the K-Means classification algorithm [18] (the choice of K-Means is justified by its simplicity and rapidity) to partition the set of XPath expressions into subsets (classes) that present a usage similarity by queries. With the $XPQUM$ matrix as classification context and an integer K indicating the number of vertical fragments, the K-Means algorithm generates K disjoint classes of XPath expressions. The XPath expressions of the class C_i describe the structure of the vertical fragment VF_i and the set of fragments VF_i $(i = 1...K)$ defines our vertical fragmentation schema noted VFS. After this partitioning (fragmentation), we assign every query to the vertical fragments needed to its processing. We formalize this assignment as following: Let:

- $C_1, C_2, ..., C_k$ the sets of XPath expressions defining respectively the vertical fragments $VF_1, VF_2, ..., VF_k$,
- SQ_i is the set of query assigned to VF_i,
- d is the number of queries requiring join operations in VFS schema,
- A the set of XPath expressions used by the query Q_j,

Then

1. If $A \subseteq C_i$ then $SQ_i \leftarrow SQ_i \cup \{Q_j\}$.
2. If $A \subseteq (C_x \cup ... \cup C_y)$ then $SQ_x \leftarrow SQ_x \cup \{Q_j\}$,...,$SQ_y \leftarrow SQ_y \cup \{Q_j\}$ and $d \leftarrow d + 1$.

In case (2), the processing of the query Q_j, requires a join operations between the vertical fragments $VF_x, ..., VF_y$. These join operations are among the causes of performance deterioration. For this fact, we minimize the number of join queries (the d number) appearing in the vertical fragmentation schema VFS. We vary N times the value of the number K of vertical fragments (N is random integer) and for each value, we generate a vertical fragmentation schema. Among these N schemas, we select the optimal according to the following rule:

Rule.1. A vertical fragmentation schema is optimal if and only if it contains a minimum of queries requiring join operations between the vertical fragments. Formally:

$$VFS_i \ \textit{is optimal} \equiv \forall j \in [1..N], \exists i \in [1..N] \ / \ (d_i < d_j) \ \textit{with} \ i \neq j \ . \quad (1)$$

d_i is the number of join queries in the fragmentation schema VFS_i.

Then for each vertical fragment $VF_i \in VFS_i$, we create a vertical script VS_i represented by a XQuery query. The execution context (the clause for) of this query is the XML data warehouse Δ and its clause *return* represents the projection of all XPath expressions belonging to C_i. The selected vertical fragmentation schema is the final result of this first phase as represented by the Fig.2.

In the next section we detail the horizontal fragmentation of each vertical fragment belonging to this schema.

3.2 Horizontal Fragmentation Based on Selection Predicates

In the second phase of our hybrid fragmentation, we fragment horizontally the XML data of each vertical fragment VF_i into L horizontal fragments FH_{i1},...,FH_{iL}. The following steps are executed for each vertical fragment as represented by the Fig.3.

Extraction of Selection Predicates. We perform a syntactical parsing of the *where* clause of each query belonging to the set SQ_i (the set of queries assigned to the vertical fragment VF_i). This parsing allows us to extract the set of selection predicates noted PS_i.

Selection Predicates-Queries Usage Matrix (SPQUM). It defines the use of selection predicates of PS_i by the queries of SQ_i. The SPQUM lines correspond to the selection predicates and its columns represent queries. if the predicate p_x exists in the *where* clause of the query Q_y then $SPQUM(x, y) = 1$, else $SPQUM(x, y) = 0$.

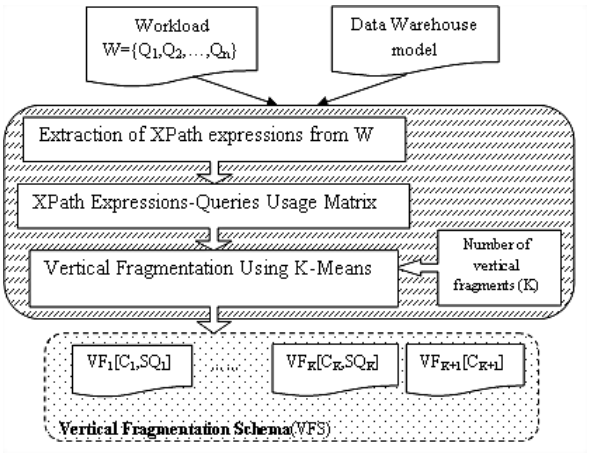

Fig. 2. Vertical fragmentation of the XML data warehouse

Horizontal Fragmentation. Using the K-Means algorithm, we group into classes the selection predicates that present a usage similarity by queries. Specifying the number L of the horizontal fragments, the algorithm partitions the set of the selection predicates of the $MUPSR$ matrix in L disjoint classes representing the horizontal fragmentation schema noted HFS_i. The selection predicates of each class C_{ij} (i the index of the vertical fragment and $j = 1..L$) define the XML data of the horizontal fragment FH_{ij}. According to this partitioning, we assign each query belonging to SQ_i to the horizontal fragments needed to its processing as follows:
Let:

- $Q_h \in SQ_i$,
- $C_{i1}, C_{i2}, ..., C_{iL}$ are the sets of selection predicates corresponding to the horizontal fragments $FH_{i1}, FH_{i2}, ..., FH_{iL}$.
- PSQ_h the set of the selection predicates used by the query Q_h,
- $d^{'}$ the number of queries requiring union operations between FH_{ij},
- SQ_{ij} the set of queries assigned to the fragment FH_{ij}

Then

1. If $PSQ_h \subseteq C_{ij}$ then $SQ_{ij} \leftarrow SQ_{ij} \cup Q_h$.
2. if $PSQ_h \subseteq (C_{ix} \cup ... \cup C_{iy})$ then $SQ_{ix} \leftarrow SQ_{ij} \cup \{Q_h\}$,...,$SQ_{iy}$. $\leftarrow SQ_{iy} \cup \{Q_h\}$ and $d^{'} \leftarrow d^{'} + 1$.

In the case (2), the processing of the query Q_h, requires the union of the horizontal fragments $FH_{ix}, ..., FH_{iy}$. In order to reduce these union operations, we vary $N^{'}$ times the value of the number of horizontal fragments L and we generate a horizontal fragmentation schema for each value. Among these $N^{'}$ fragmentation schemas, we select the best according to the following rule:

Rule.2. An horizontal fragmentation schema noted HFS is optimal if and only if it contains a minimum of queries requiring union operations between horizontal fragments

$$HFS_i \ \ is \ \ optimal \equiv \forall j \in \left[1..N^{'}\right], \exists i \in \left[1..N^{'}\right] \ \ / \ \ \left(d_i^{'} < d_j^{'}\right) \ \ with \ \ i \neq j. \quad (2)$$

$d_i^{'}$ is the number of union queries in the fragmentation schema HFS_i.

For each horizontal fragment $HF_{ij} \in HFS_i$, we create a horizontal script HS_{ij} represented by a XQuery query. The execution context (the clause for) of this query is the vertical fragment VF_i and its $where$ clause is the disjunction between the selection predicates belonging to C_{ij}.

At the end of these two phases, we generate an XML document containing the hybrid fragmentation schema noted $HDFS$. For this, we merge each vertical fragment $VF_i \in VFS$ with its horizontal fragments belonging to HFS_i.

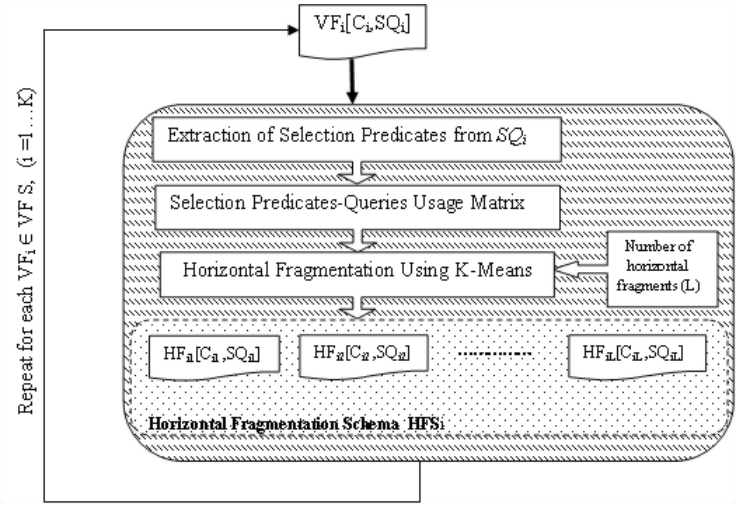

Fig. 3. Horizontal fragmentation of each vertical fragment

3.3 Query Processing on the Fragmented Data Warehouse

The access to the XML data, after fragmentation, should be transparent to the users of the warehouse. To ensure transparency, query processing must be performed on fragmented XML data warehouse. For this, we rewrite the queries according to their assignments carried out during the previous two phases. For each query of the workload:

1. We run through the hybrid fragmentation schema (the XML document) and we identify all fragments needed to its processing.
2. In its execution context, we replace the unfragmented data warehouse by the already identified fragments.

3. If it requires join operations between fragments, we adjust its *where* clause by adding a join qualifications.
4. If it requires union operations between hybrid fragments, we add to its clause *for* the XQuery function *distinct-deep* which removes the duplicate XML data from its result.

In order to prove the effectiveness of the hybrid fragmentation detailed in the previous sections, we have conducted various evaluations that we present in the following section.

4 Experimental Studies

4.1 Experimental Conditions

We have conducted our evaluations under Oracle Berkeley DB XML[12] (an XML native database allowing the storage of voluminous XML documents and implements the XQuery1.0 queries execution engine). We have used the XML dataset from the XML Data Warehouse Benchmark (XWB) proposed in [21]. Modeled with a star schema, the XML data warehouse of the XWB contains the sales facts characterized by the measures: quantity of purchased product and amount of purchased product. These facts are analyzed by the dimensions: products, consumers, suppliers and time. While respecting the definition of flat model (Sect.2), we have merged the facts and dimensions into a single XML document representing our data warehouse. As a programming language, we have used the Java language to implement our hybrid fragmentation algorithm in which we have used the *K-Means*[1] library. The machine used for our experiments is equipped with a Intel Pentium processor and 02 GB of main memory.

4.2 Experimental Assessment and Analysis

In order to prove the effectiveness of our hybrid fragmentation algorithm, we have performed various experiments. In the first, we have used a XML data warehouse composed of 2000 facts and we have (i) calculated the global response time of 19 queries executed on the original XML data warehouse, (ii) fragmented this data warehouse into 02 vertical fragments VF_1 and VF_2, (iii) calculated the global response time of the same queries on the vertically fragmented data warehouse. In the second experiment, we have (i) fragmented respectively VF_1 and VF_2 into 04 and 06 horizontal fragments (ii) calculated the global response time of the 19 queries on the new hybrid fragments. Figure 4, summarizes the results of this two experiments, and the Fig.5 shows the details of the queries response time before fragmentation, after the vertical fragmentation, and after the hybrid fragmentation.

According to the results shown in Fig.4, and compared to the unfragmented XML data warehouse, we observe that the vertical fragmentation improves the

[1] `https://www.hepforge.org/downloads/jminhep/`

global response time of the workload to 30%. As against, the global response time of the same workload is improved to 82% after applying the hybrid fragmentation on the XML data warehouse. The detailed results shown by the Fig.5, allows us to see clearly the effect of the hybrid fragmentation on queries response times. Indeed, after a vertical fragmentation of the data warehouse, the processing of the queries Q_3, Q_5, Q_7, Q_9, Q_{11},Q_{14}, and Q_{15}, requires a join operation between the two vertical fragments VF_1 and VF_2. Their response time have not been improved, on the contrary we notice a significant deterioration in the performances of the queries Q_7, Q_{14}, and Q_{15}. However, only the response times of the queries requiring a single vertical fragment VF_1 or VF_2, have benefited from some improvement. But after the hybrid fragmentation, we observe a meaningfully enhancement in the response time of each query, in particularly join queries, that which proves the effectiveness of our hybrid fragmentation algorithm.

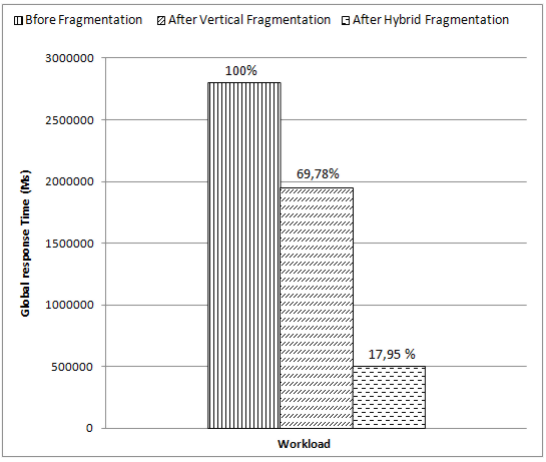

Fig. 4. Global response time of the workload on 2000 facts

In the Third experiment, we have applied our hybrid fragmentation algorithm on three XML data warehouses of different sizes: 2000, 4000, and 8000 facts. We have fragmented each data warehouse according to the same previous hybrid fragmentation schema and we have calculated the global response time of 19 queries before and after fragmentation on each data warehouse. The obtained results shown by the Fig.6, confirm that our hybrid fragmentation always guarantee an improvement of the performances even after the increase of the size of the XML data warehouse.

Indeed, fragmenting XML data warehouse by our algorithm allows us to:

1. Group in hybrid fragments (XML documents) the XPath expressions (vertical fragmentation) and the XML data (horizontal fragmentation) needed in processing queries.
2. Generate fragments of small sizes compared to the size of the unfragmented data warehouse.

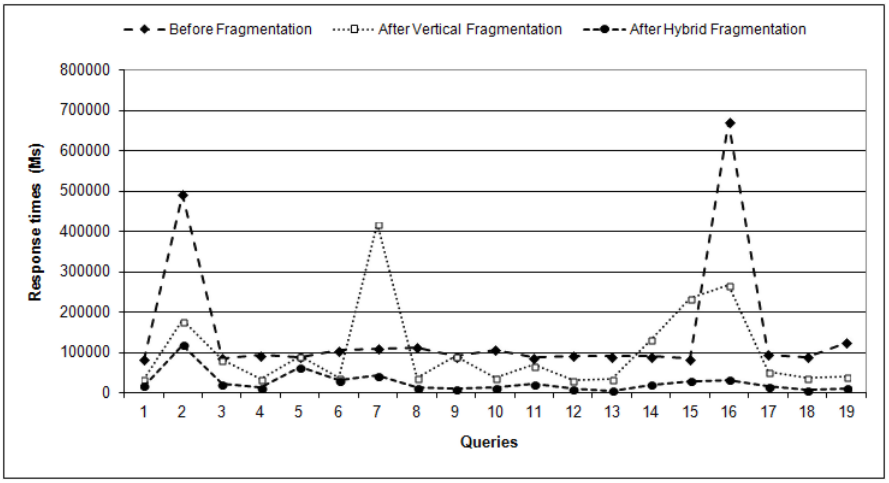

Fig. 5. Response time by query

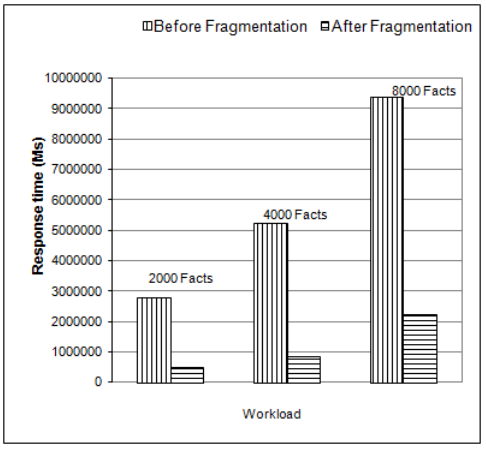

Fig. 6. Response time of the workload on different sizes of data warehouses

The first point, allows us to improve the search time of the XML data to satisfy a query. On the other side, the second point, allows us to improve the time needed to browse the XML structure of the unfragmented data warehouse to search data. According to these two points, we justify the performances improvement provided by our hybrid fragmentation algorithm.

5 Conclusion

The processing time of the decision-support queries on an XML data warehouse is quite high especially on a large volume of XML data. However, minimizing this

processing time significantly contributes to the improvement of decision-making process. In this context, we proposed a new fragmentation approach of XML data warehouse called hybrid fragmentation. Firstly, we introduced the different multidimensional models of XML data. Based on several evaluations conducted between these models, we have chosen the flat model as a reference model to represent the XML data warehouse. Then, we detailed our hybrid fragmentation algorithm in which we combined a vertical fragmentation based on XPath expressions with a horizontal fragmentation based on the selection predicates. In our approach we used the K-Means algorithm to control the number of fragments and generate a fragmentation schema offering more improvement of performance. Finally, we conducted various experiments to prove the validity of our algorithm. The results obtained allowed us to confirm the effectiveness of our proposed hybrid fragmentation. In future work, we plan to conduct an experimental comparison between the fragmentation algorithms proposed in [9] and [22], and our hybrid fragmentation algorithm.

References

1. Agrawal, S., Narasayya, V., Yang, B.: Integrating vertical and horizontal partitioning into automated physical database design. In: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, SIGMOD 2004, pp. 359–370. ACM, New York (2004), http://doi.acm.org/10.1145/1007568.1007609
2. Baio, F., Mattoso, M.: A mixed fragmentation algorithm for distributed object oriented databases. In: Proc. of the 9th Int. Conf. on Computing Information, pp. 141–148 (1998)
3. Bellatreche, L., Bouchakri, R., Cuzzocrea, A., Maabout, S.: Horizontal partitioning of very-large data warehouses under dynamically-changing query workloads via incremental algorithms. In: Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC 2013, pp. 208–210. ACM, New York (2013), http://doi.acm.org/10.1145/2480362.2480406
4. Bellatreche, L., Boukhalfa, K., Richard, P.: Data partitioning in data warehouses: Hardness study, heuristics and ORACLE validation. In: Song, I.-Y., Eder, J., Nguyen, T.M. (eds.) DaWaK 2008. LNCS, vol. 5182, pp. 87–96. Springer, Heidelberg (2008)
5. Bellatreche, L., Karlapalem, K., Simonet, A.: Horizontal class partitioning in object-oriented databases. In: Tjoa, A.M. (ed.) DEXA 1997. LNCS, vol. 1308, pp. 58–67. Springer, Heidelberg (1997), http://dl.acm.org/citation.cfm?id=648310.754717
6. Bellatreche, L., Karlapalem, K., Simonet, A.: Algorithms and support for horizontal class partitioning in object-oriented databases. Distrib. Parallel Databases 8(2), 155–179 (2000), http://dx.doi.org/10.1023/A:1008745624048
7. Berglund, A., Boag, S., Chamberlin, D.: andez, M.F.F.: Xml path language (xpath) 2.0, 2nd edn. (December 2010)
8. Boucher, S., Verhaegen, B., Zimányi, E.: XML Multidimensional Modelling and Querying. CoRR abs/0912.1110 (2009)

9. Boukraâ, D., Boussaïd, O., Bentayeb, F.: Vertical fragmentation of XML data warehouses using frequent path sets. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2011. LNCS, vol. 6862, pp. 196–207. Springer, Heidelberg (2011), http://dblp.uni-trier.de/db/conf/dawak/dawak2011.html#BoukraaBB11
10. Boukraa, D., Riadh Ben, M., Omar, B.: Proposition d'un modèle physique pour les entrepôts XML. In: Premier Atelier des Systèmes Décisionnels (ASD 2006), Agadir, Maroc (2006)
11. Boussaid, O., BenMessaoud, R., Choquet, R., Anthoard, S.: Conception et construction d'entrepôts XML. In: 2ème journée francophone sur les Entrepôts de Données et l'Analyse en ligne (EDA 2006), Versailles. RNTI, vol. B-2, pp. 3–22. Cépaduès, Toulouse (Juin 2006)
12. Brian, D.: The Definitive Guide to Berkeley DB XML (Definitive Guide). Apress, Berkely (2006)
13. Ceri, S., Negri, M., Pelagatti, G.: Horizontal data partitioning in database design. In: Proceedings of the 1982 ACM SIGMOD International Conference on Management of Data, SIGMOD 1982, pp. 128–136. ACM, New York (1982), http://doi.acm.org/10.1145/582353.582376
14. Dimovski, A., Velinov, G., Sahpaski, D.: Horizontal partitioning by predicate abstraction and its application to data warehouse design. In: Catania, B., Ivanović, M., Thalheim, B. (eds.) ADBIS 2010. LNCS, vol. 6295, pp. 164–175. Springer, Heidelberg (2010), http://dl.acm.org/citation.cfm?id=1885872.1885888
15. Elhoussaine, Z., Aboutajdine, D., Abderrahim, E.Q.: Algorithms for data warehouse design to enhance decision-making. WSEAS Trans. Comp. Res. 3(3), 111–120 (2008), http://dl.acm.org/citation.cfm?id=1466884.1466885
16. Golfarelli, M., Rizzi, S., Vrdoljak, B.: Data warehouse design from XML sources. In: Proceedings of the 4th ACM international workshop on Data warehousing and OLAP, DOLAP 2001, pp. 40–47. ACM, New York (2001), http://doi.acm.org/10.1145/512236.512242
17. Hümmer, W., 0004, A.B., Harde, G.: XCube: XML for Data Warehouses. In: DOLAP, pp. 33–40 (2003)
18. MacQueen, J.: Some Methods for Classifcation and Analysis of Multivariate Observations. In: Proceeding of Fifth Berkley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–296 (1967)
19. Kimball, R.: A dimensional modeling manifesto. DBMS 10, 58–70 (1997), http://portal.acm.org/citation.cfm?id=261018.261025
20. Mahboubi, H., Aouiche, K., Darmont, J.: Un index de jointure pour les entrepôts de données xml. In: 6émes Journées Francophones Extraction et Gestion des Connaissances (EGC 2006), Lille. Revue des Nouvelles Technologies de l'Information, vol. E-6, pp. 89–94. Cépadués, Toulouse (2006)
21. Mahboubi, H., Darmont, J.: Benchmarking xml data warehouses. In: Atelier Syst emes Décisionnels (ASD 2006), 9th Maghrebian Conference on Information Technologies (MCSEAI 2006), Agadir, Maroc (December 2006)
22. Mahboubi, H., Darmont, J.: Data mining-based fragmentation of xml data warehouses. In: DOLAP, pp. 9–16 (2008)
23. Mahboubi, H., Darmont, J.: Enhancing xml data warehouse query performance by fragmentation. In: Proceedings of the 2009 ACM Symposium on Applied Computing, SAC 2009, pp. 1555–1562. ACM, New York (2009), http://doi.acm.org/10.1145/1529282.1529630
24. Navathe, S.B., Karlapalem, K., Ra, M.: A mixed fragmentation methodology for initial distributed database design. Journal of Computer and Software Engineering 3(4), 395–426 (1995)

25. Ozsu, M.T.: Principles of Distributed Database Systems, 3rd edn. Prentice Hall Press, Upper Saddle River (2007)
26. Pokorný, J.: XML Data Warehouse: Modelling and Querying. In: Proceedings of the Baltic Conference, BalticDB&IS 2002, vol. 1, pp. 267–280. Institute of Cybernetics at Tallin Technical University (2002), http://portal.acm.org/citation.cfm?id=648170.750672
27. Rusu, L.I., Rahayu, J.W., Taniar, D.: A methodology for building xml data warehouses. IJDWM 1(2), 23–48 (2005)
28. Rusu, L.I., Rahayu, W., Taniar, D.: Partitioning methods for multi-version xml data warehouses. Distrib. Parallel Databases 25(1-2), 47–69 (2009), http://dx.doi.org/10.1007/s10619-009-7034-y
29. Walmsley, P.: XQuery. O'Reilly Media, Inc. (2007)
30. Zhang, Y., Orlowska, M.E.: On fragmentation approaches for distributed database design. Information Sciences - Applications 1(3), 117–132 (1994), http://www.sciencedirect.com/science/article/pii/1069011594900051