Fragmenting very large XML data warehouses via K-means clustering algorithm

Alfredo Cuzzocrea, Jérôme Darmont, Hadj Mahboubi

Fragmenting Very Large XML Data Warehouses via *K*-Means Clustering Algorithm

Alfredo Cuzzocrea[1], Jérôme Darmont[2], and Hadj Mahboubi[2]

[1] ICAR-CNR & University of Calabria
Via P. Bucci, 41C, Rende
87036 Cosenza, Italy
E-mail: cuzzocrea@si.deis.unical.it
[2] University of Lyon (ERIC Lyon 2)
5 avenue Pierre Mendès-France
69676 Bron Cedex, France
E-mail: {hadj.mahboubi, jerome.darmont}@eric.univ-lyon2.fr

Abstract: XML data sources are more and more gaining popularity in the context of a wide family of *Business Intelligence* (BI) and *On-Line Analytical Processing* (OLAP) applications, due to the amenities of XML in representing and managing semi-structured and complex multidimensional data. As a consequence, many XML data warehouse models have been proposed during past years in order to handle heterogeneity and complexity of multidimensional data in a way traditional relational data warehouse approaches fail to achieve. However, XML-native database systems currently suffer from limited performance, both in terms of volumes of manageable data and query response time. Therefore, recent research efforts are focusing the attention on *fragmentation techniques*, which are able to overcome the limitations above. Derived horizontal fragmentation is already used in relational data warehouses, and can definitely be adapted to the XML context. However, classical fragmentation algorithms are not suitable to control the number of originated fragments, which instead plays a critical role in data warehouses, and, with more emphasis, distributed data warehouse architectures. Inspired by this research challenge, in this paper we propose the use of K-means clustering algorithm for effectively and efficiently supporting the fragmentation of very large XML data warehouses, and, at the same time, completely controlling and determining the number of originated fragments via adequately setting the parameter K. We complete our analytical contribution by means of a comprehensive experimental assessment where we compare the efficiency of our proposed XML data warehouse fragmentation technique against those of classical derived horizontal fragmentation algorithms adapted to XML data warehouses.

1 Introduction

Nowadays, XML has become a standard for representing complex business data [19], so that decision support processes that make use of XML data sources

are now increasingly common. However, XML data sources bear specificities that would be intricate to handle in a relational environment. Among these specificities, we recall: heterogeneous number and order of dimensions, complex aggregation operations [69] and measures, ragged dimensional hierarchies [19]. Hence, many efforts towards the so-called *XML Data Warehousing* have been achieved during the past few years [26, 72, 85], as well as efforts focused to extend *XQuery* [20] with near *On-Line Analytical Processing* (OLAP) [31, 43] capabilities such as advanced grouping and aggregation features [19, 59, 77].

In this context, performance is a critical issue, as actual XML-native database systems (e.g., *eXist* [60], *TIMBER* [46], *X-Hive* [82], and *Sedna* [37]) suffer from limited performance, both in terms of volumes of manageable manageable data and response time to complex *analytical queries*. These issues are well-known to data warehouse researchers, and they can be addressed by means of the so-called *fragmentation techniques* [84]. Fragmentation consists in splitting a given data set into several *fragments* such that their combination yields the original data warehouse without information loss nor information addition. Fragmentation can subsequently support a meaningful *distribution* of the target data warehouse, e.g. on *Data Grids* [32] or across *Peer-To-Peer* (P2P) *Networks* [48]. In the relational context, *derived horizontal fragmentation* is acknowledged as the best-suited one to data warehouses [13]. Basically, this approach consists in fragmenting a given relation with respect to *query predicates* defined on another relation. Apart from the above-mentioned research efforts, other XML data fragmentation approaches have also been proposed recently [21, 22, 24, 38, 51], but they do not take into account multidimensional schemas explicitly (i.e., star, snowflake, or fact constellation schemas [49]).

In derived horizontal fragmentation, dimensional tables first undergo a *primary horizontal fragmentation*. Output fragments are then used to horizontally fragment the fact table into sub-tables that each refer to a primary dimensional fragment. This process is termed *derivation*. Primary horizontal fragmentation plays a critical role, as it heavily affects the performance of the whole fragmentation process. In the relational context, two major algorithms address this issue: *predicate construction* [66] and *affinity-based* [64] algorithms. However, these approaches suffer from an important limitation that makes them unsuitable to XML Data Warehousing. In fact, in both algorithms the number of fragments is not known in advance neither can be set as input parameter, while in XML Data Warehousing it is crucial to master this parameter, especially as distributing M fragments over N nodes, with $M > N$, can be a critical issue in itself. In order to become convinced of this aspect, it suffices to think of the fragmentation problem in *Distributed Data Warehousing* environments [32]. Here, due to load-balancing and scalability issues, node number can become very large, but massive-in-size data warehouses can still represent a problematic instance to be fragmented. Therefore, the need for completely controlling the number of output fragments makes perfect sense.

Starting from these considerations, in this paper we propose *the usage of K-means [54] clustering algorithm for supporting the efficient fragmentation of XML data warehouses while controlling the number of generated fragments through the parameter K*. The latter specific feature has immediate benefits towards efficiently supporting XML Data Warehousing in itself, as it will be clear throughout the paper. Our proposed approach is inspired from a proposal coming from the object-

oriented databases domain [33]. Summarizing, our proposal consists in clustering the predicates of a reference query-workload posed to the target XML data warehouse in order to produce primary horizontal fragments from dimensional tables (XML documents, respectively), with one fragment meaningfully corresponding to one *cluster of predicates*. Primary fragmentation is then derived on facts. Queries based on predicates of the target query-workload are then evaluated over the corresponding fragments only, instead of the whole data warehouse, thus introducing a faster response time. The number of fragments is directly related to the number of K-means-obtained clusters (it is actually equal to $K + 1$ – Section 4.4).

The remainder of this paper is organized as follows. In Section 2, we discuss state-of-the-art research in fragmentation techniques for relational data warehouses and XML databases, and also *Data-Mining-based fragmentation techniques* [33, 36, 42], which, briefly, propose applying Data Mining techniques in order to drive the fragmentation phase. The latter is the class of techniques where our research should be conceptually positioned. Section 3 focuses the attention on the XML data warehouse model we adopt as reference data model of our research. In Section 4, we introduce our K-means-based XML data warehouse fragmentation approach. Section 5 experimentally compares the efficiency of our proposed technique against those of classical derived horizontal fragmentation algorithms adapted to XML data warehouses, and shows its superiority in accomplishing the desired goal. Finally, Section 6 contains conclusions of our research, along with future research directions in fragmentation techniques for XML data warehouses.

2 Related Work

In this Section, we first provide a brief taxonomy of relevant fragmentation techniques, which have been originally proposed in the relational context mainly. Then, we focus the attention on three aspects that represent the conceptual/theoretical foundations of our research, i.e. relational data warehouse fragmentation techniques, XML database fragmentation techniques, and, finally, Data-Mining-based fragmentation techniques.

2.1 Taxonomy of Fragmentation Techniques

In the relational context, it is possible to identify three main fragmentation techniques: *vertical fragmentation*, *horizontal fragmentation*, and *hybrid fragmentation*.

Vertical fragmentation splits a given relation R into sub-relations that are *projections* of R with respect to a subset of attributes. It consists in grouping together attributes that are frequently accessed by queries. Vertical fragments are thus built by projection. The original relation is reconstructed by simply joining the fragments. Relevant examples for techniques belonging to this class are the following. Navathe *et al.* vertically partition a relation into fragments and propose two alternative fragmentation methods: *progressive binary partitioning* [63] and *graphical partitioning* [65]. The first method is based on three matrices (one capturing the *Usage*, one capturing the *Affinity* and another one capturing the *Coordinates* of queries) while the second one exploits an objective function. In [63], authors

present techniques for applying vertical fragmentation in the following specialized application contexts: databases stored on homogeneous devices, databases stored in different memory levels, and distributed databases.

Horizontal fragmentation divides a given relation R into sub-sets of tuples by exploiting query predicates. It reduces query processing costs by minimizing the number of irrelevant accessed instances. Horizontal fragments are thus built by selection. The original relation is reconstructed by fragment union. A variant, the so-called derived horizontal fragmentation [13], consists in partitioning a relation R with respect to predicates defined on another relation, said R'. Other significant horizontal fragmentation techniques are the following. Major algorithms that address horizontal fragmentation are *Predicate-Construction-Based* [29] and the *Affinity-Based* [65] methods (Section 2.2).

Finally, hybrid fragmentation consists of either horizontal fragments that are subsequently vertically fragmented, or, by contrary, vertical fragments that are subsequently horizontally fragmented. Noticeable samples of these approaches are: (*i*) *Grid Creation* [64], which proposes a mixed fragmentation methodology allowing us to obtain a sub-optimal partition of a given relation belonging to a distributed database, and (*ii*) *View-Based Fragmentation* [70], which exploits views to build database fragments.

2.2 Data Warehouse Fragmentation

Several research studies address the issue of fragmenting relational data warehouses, either to efficiently evaluate analytical queries, or to efficiently distribute these data warehouses on settings like data grids and P2P networks.

In order to improve ad-hoc query evaluation performance, Datta *et al.* [35] propose exploiting a vertical fragmentation of facts to build the index *Cuio*, while Golfarelli *et al.* [40] propose applying the same fragmentation methodology on data warehouse views. Munneke *et al.* [61] instead propose an original fragmentation methodology targeted to multidimensional databases. In this case, fragmentation consists in deriving a global data cube from fragments containing a sub-set of data defined by meaningful slice and dice OLAP-like operations [31, 43]. In [61], authors also define an alternative fragmentation strategy, named *server*, which removes one or several dimensions from the target data cube in order to produce fragments having fewer dimensions than the original data cube.

Bellatreche and Boukhalfa [13] apply horizontal fragmentation to data warehouse star schemas. Their fragmentation strategy is based on a reference query-workload, and it exploits a genetic algorithm to select a suitable partitioning schema among all the possible ones. Overall, the proposed approach aims at selecting an *optimal fragmentation schema* that minimizes query cost. Wu and Buchmaan [78] recommend to combine horizontal and vertical fragmentation for query optimization purposes. In [78], a fact table can be horizontally partitioned with respect to one or more dimensions of the data warehouse. Moreover, the fact table can also be vertically partitioned according to its dimensions, i.e. all the foreign keys to the dimensional tables are partitioned as separate tables.

In order to distribute a data warehouse, Noaman *et al.* [66] exploit a top-down strategy making use of horizontal fragmentation. In [66], authors propose an algorithm for deriving horizontal fragments from the fact table based on input queries

defined on all the dimensional tables. Finally, Wehrle *et al.* [76] propose distributing and querying a data warehouse by meaningfully exploiting the capabilities offered by a *Computational Grid*. In [76], authors make use of derived horizontal fragmentation to split the target data warehouse and build the so-called *block of chunks*, which is a set of data portions derived from the data warehouse and used to query optimization purposes, being each portion computed as a fragment of the partition.

In summary, the above-outlined proposals generally exploit derived horizontal fragmentation to reduce irrelevant data accesses and efficiently process join operations across multiple relations [13, 66, 76]. From active literature [50], we also recognize that, in order to implement derived horizontal fragmentation of data warehouses, the outlined approaches prevalently make use of the following two main fragmentation methods:

- *Predicate-Construction-Based Fragmentation* [29] This method fragments a given relation by using a complete and minimal set of predicates [66]. Completeness means that two relation instances belonging to the same fragment have the same probability of being accessed by any arbitrary query. Minimality guarantees that there is no redundancy in predicates.

- *Affinity-Based Fragmentation* [65] This method is an adaptation of the vertical fragmentation approach [40] to the horizontal fragmentation one [64]. It is based on the *predicate affinity concept* [84] according to which affinity is defined in terms of query frequency. Specific predicate-usage and affinity matrices are exploited in order to cluster selection predicates. A cluster is here defined as a *selection predicate cycle*, and forms a fragment of a dimensional table itself.

2.3 XML Database Fragmentation

Recently, several fragmentation techniques for XML data have been proposed in literature. These techniques propose splitting an XML document into a new set of XML documents, with the main goal of either improving XML query performance [21, 38, 52], or distributing or exchanging XML data over a network [22, 24].

In order to fragment XML documents, Ma *et al.* [51, 52] define a new fragmentation notion, called *split*, which is inspired from the oriented-object databases context. This fragmentation technique splits elements of the input XML document, and assigns a reference to each so-obtained sub-element. References are then added to the *Document Type Definition* (DTD) defining the input XML document. This avoid redundancy and inconsistence problems that could occur due to fragmentation process. Bonifati *et al.* [21, 23] propose a fragmentation strategy for XML documents that is driven by the so-called *structural constraints*. These constraints refer to intrinsic properties of XML trees such as the depth and the width of trees. In order to efficiently fragment the input XML document by means of structural constraint, the proposed strategy exploits heuristics and statistics simultaneously.

Andrade *et al.* [7] propose applying fragmentation to an *homogeneous* collection of XML documents. In [7], authors adapt traditional fragmentation techniques to an XML document collection, and make use of the *Tree Logical Class* (TLC) algebra [68] to this goal. Authors also experimentally evaluate these techniques

and show that horizontal fragmentation provides the best performance. Gertz and Bremer [38] introduce a distribution approach for XML repositories. They propose a fragmentation method and outline an allocation model for distributed XML fragments in a centralized architecture. In [38], authors also define horizontal and vertical fragmentation for XML repositories. Here, fragments are defined on the basis of a *path expression language*, called *XF*, which is derived from *XPath* [30]. In more detail, fragments are obtained via applying an *XF* expression on a graph representing XML data, named *Repository Guide* (*RG*). Moreover, authors provide exclusion expressions that ensure fragment coherence and disjunction rigorously.

Bose and Fegaras [24], argue to use XML fragments for efficiently supporting data exchange in P2P networks. In this proposal, XML fragments are interrelated, and each fragment is univocally identified by an *ID*. Authors also propose a fragmentation schema, called *Tag Structure*, which allows us to define the structure of fragments across the network. In turn, the structure of fragments can be exploited for data exchange and query optimization purposes. Bonifati *et al.* [22] also define an XML fragmentation framework for P2P networks, called *XPath-To-Partition* (XP2P). In this proposal, XML fragments are obtained and identified via a single root-to-node path expression, and managed on a specific peer. In addition, to data management efficiency purposes, in [22] authors associate two XPath-modeled path expressions to each fragment, namely *super fragment* and *child fragment*, respectively. Given an XML fragment f, the first XPath expression identifies the root of the fragment f' from which f has been originated; the second XPath expression instead identifies the root of a f's child XML fragment. These path expressions ensure the easily identification of fragments and their networked relationships.

In summary, the above-outlined proposals adapt classical fragmentation methods, mainly investigated and developed in the context of relation data warehouses, in order to split a given XML database into a meaningfully collection of XML fragments. An XML fragment is defined and identified by a path expression [22, 38], or an XML algebra operator [7]. Fragmentation is performed on a single XML document [51, 52], or an homogeneous XML document collection [7]. Another secondary result deriving from this is represented by the claim stating that, to the best of our knowledge, XML data warehouse fragmentation has not been addressed at now by active literature. This further confirms the innovation carried out by our research.

2.4 Data-Mining-based Fragmentation

Although Data Mining has already proved to be extremely useful to select physical data structures that enhance performance, such as indexes or materialized views [1, 8, 9, 83], few fragmentation approaches that exploit Data Mining exist in literature. Therefore, it is reasonable to claim that the latter is a relatively-novel area of research, and a promising direction for future efforts in data warehouse and database fragmentation techniques.

Gorla and Betty [42] exploit *association rules* for vertical fragmentation of relational databases. Authors consider that association rules provide a natural way to represent relationships between attributes as implied by database queries. Basically, their solution consists in adapting the well-known algorithm Apriori [5] by selecting the non-overlapping item-sets having highest support and by grouping their respective attributes into one partition. Then, the algorithm exploits a cost

model to select an optimal fragmentation schema.Darabant and Campan [33] propose using K-means clustering for efficiently supporting horizontal fragmentation of object-oriented distributed databases. This research has inspired our work. In more detail, the method proposed in [33] clusters object instances into fragments via taking into account all complex relationships between classes of data objects (aggregation, associations and links induced by complex methods). Finally, Fiolet and Toursel [36] propose a parallel, progressive *clustering algorithm* to fragment a database and distribute it over a data grid. This approach is inspired by the sequential clustering algorithm *CLIQUE* [2] that consists in clustering data by means of projection operations.

Even though in limited number, these studies clearly demonstrate how Data Mining can be efficiently used to support horizontal and vertical fragmentation of both data warehouses and databases, throughout association rule mining and clustering, respectively.

3 A Reference XML Data Warehouse Model

Actual XML data warehouse models from the literature [41, 69, 72] share a lot of concepts, mostly originating from classical results developed in the relational context. Despite this common origin, actual XML data warehouse models are nonetheless all different. From this evidence, in [58] a unified, reference XML data warehouse model that synthesizes and enhances existing models is proposed. This proposal represents the fundamental data model of our proposed XML data warehouse fragmentation technique. Given this significant relationship between [58] and the the research we propose in this paper, before to detail our XML data warehouse fragmentation approach (Section 4), in this Section we review the XML data warehouse model [58].

State-of-the-art XML data warehouse models assume that the target data warehouse is composed by XML documents representing both facts and dimensions. All these studies mostly differ in the way dimensions are handled, and the number of XML documents that are used to store facts and dimensions. A performance evaluation study of these different representations has shown that representing facts in one singleton XML document and each dimension in one singleton XML document allows the best performance [25]. Moreover, the above representation model also allows us to model fact constellation schemas without the need of duplicating dimension information, thus achieving the so-called *shared dimensions* [49]. This has several benefits for what concerns with the scalability of the model, which is an extremely critical factor in Data Warehousing. According to this representation model, several fact documents can indeed share the same dimensions. Hence, we adopt this architecture model. In more detail, our reference XML data warehouse model is composed by the following XML documents:

- $dw - model.xml$, which stores warehouse metadata;
- a set of documents $facts_f.xml$, such that each document stores information related to a set of facts f;
- a set of documents $dimension_d.xml$, such that each documents stores the member values of the dimension d.

Document $dw - model.xml$ (Figure 1) defines the multidimensional structure of the target data warehouse. The root node, named as `DW-model`, is composed by two kinds of nodes: `dimension` and `FactDoc`, respectively.

A `dimension` node models a dimension of the data warehouse. In a `dimension` node, the following elements are contained: (*i*) element `@id` that models the absolute identifier of the dimension d; (*ii*) element `@path` that models the path to the corresponding document $dimension_d.xml$ storing the related dimension information; (*iii*) a set of `Level` elements, such that each element models a level L of the (possible) hierarchical levels of the dimension d. Under a `Level` element, we have: (*i*) element `@id` that models the absolute identifier of the level L; (*ii*) a set of `attribute` elements, such that each element models an attribute a of the level L. Under an `attribute` element, we have: (*i*) element `@name` that models the name of the attribute a; (*ii*) element `@type` that models the type of the attribute a.

A `FactDoc` node models a fact of the data warehouse. In a `FactDoc` node, the following elements are contained: (*i*) element `@id` that models the absolute identifier of the fact f; (*ii*) element `@path` that models the path to the corresponding document $facts_f.xml$ storing the related fact information; (*iii*) a set of elements `measure`, such that each element models a measure m of the fact f; (*iv*) a set of `dimension` elements, such that each element references a dimension d of the XML data warehouse schema. Under a `measure` element, we have: (*i*) element `@id` that models the absolute identifier of the measure m; (*ii*) element `@type` that models the type of the measure m. Under a `dimension` element, we have the element `@idref` that models the reference to the corresponding dimension d of the XML data warehouse schema.

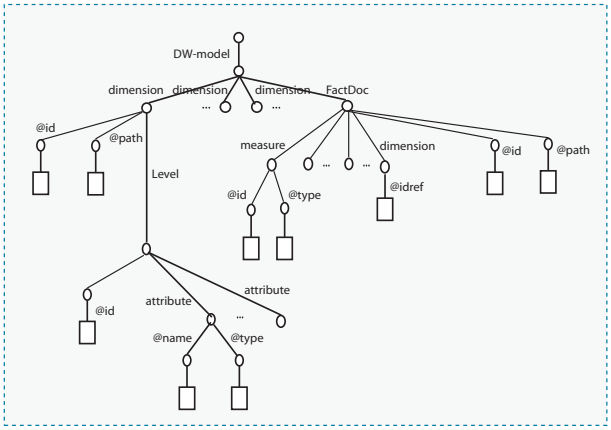

Figure 1 The XML Document $dw - model.xml$

Figure 2 shows the structure of a document $facts_f.xml$ (Figure 2(a)) and a document $dimension_d.xml$ (Figure 2(b)), respectively. The Figure also details the relationship between facts and dimensions, and how this relationship is captured in our reference XML data warehouse model. A $facts_f.xml$ document (Figure 2(a)) stores facts. It is structured in terms of the document root node, `FactDoc`, which contains an element `@id` that models the absolute identifier of the fact, and a set of elements `fact`, such that each element instantiates a fact of the XML data warehouse schema in terms of measure values and dimension references. Here, measures and dimensions are modeled in a similar way to what provided for the

document $dw-model.xml$ storing the warehouse metadata. The fact-to-dimension relationship is captured by means of conventional XML identifiers. Finally, a $dimension_d.xml$ document (Figure 2(b)) stores a dimension, including its possible hierarchical levels. The document root node, `dimension`, contain the following nodes: (i) element `@dim-id` that models the absolute identifier of the dimension; (ii) a set of elements `Levels`, such that each element models a level L of the dimension d, and contains a collection of elements `instance` that defines member attribute values v of the level L. Overall, this allows us to model an OLAP hierarchical level in all its characteristics and values. Here, attributes are modeled in a similar way to what provided for the document $dw-model.xml$ storing the warehouse metadata. In addition, an `instance` element also contains the elements `@Roll-Up` and `@Drill-Down`, respectively, which both define the hierarchical relationship of the actual level within the modeled dimension, and support classical OLAP data cube exploration operations.

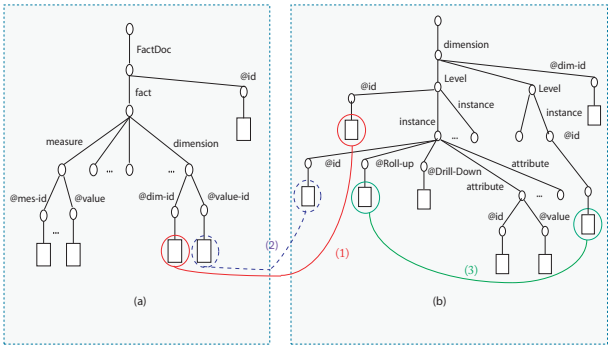

Figure 2 The XML Documents $facts_f.xml$ (a) and $dimension_d.xml$ (b)

3.1 Example

In this Section, we provide a sample four-dimensional XML data warehouse represented by means of our reference data model. Consider the *Dimensional Fact Model* (DFM) [39] depicted in Figure 3, which models the data warehouse *Sales* one can find in a typical retail application. In this schema, *Quantity* and *Amount* play the roles of measure, whereas *Customer*, *Supplier*, *Part* and *Date* play the roles of dimension. Figure 4 provides instead an overview of the set of XML documents that, according to our reference model, describes the data warehouse *Sales*.

4 *K*-Means-based Fragmentation of XML Data Warehouses

In this Section, we present and discuss our K-means-based fragmentation approach for XML data warehouses. In this respect, we first provide an overview on the proposed technique, by highlighting the fundamental tasks it is composed, and then we focus the attention on each of these tasks in a greater detail.

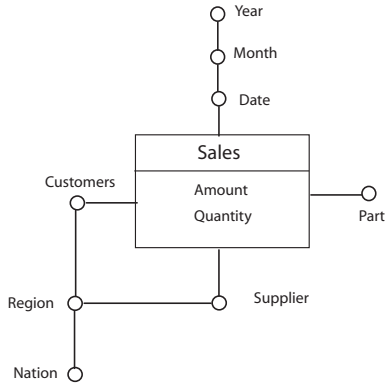

Figure 3 The Sample Data Warehouse *Sales*

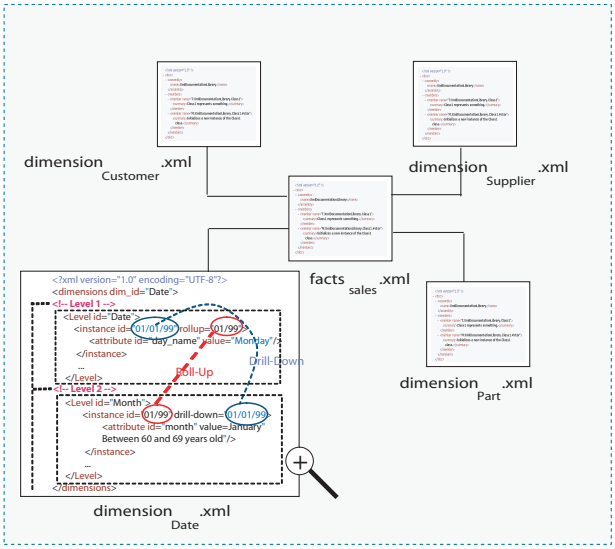

Figure 4 XML Documents associated to the Data Warehouse *Sales* of Figure 3

4.1 Overview

Since the aim of fragmentation is that of optimizing query response time, the prevalent fragmentation strategies are workload-driven [13, 21, 38, 64, 66], i.e. they assume a reference query-workload and try to optimize queries belonging to this query-workload rather than any arbitrary query than can be posed to the target data warehouse. We highlight that, fixing a reference query-workload QW, does not mean to efficiently answer queries in QW solely and discard the other (still possible) queries, but rather that queries in QW represent a set of queries that (i) are *probabilistically* likely to be posed to the data warehouse, and (ii) any other arbitrary query to the data warehouse is *probabilistically* likely to be "similar" to queries in QW. Therefore, the final goal is that of exploiting query-workload information to improve query evaluation. For what regards practical issues, it should be noted that any conventional *Data Warehouse Server* embeds monitoring

tools that are able to gather statistics on the query flow posed to the server. These statistics, which are originally meant for data warehouse maintenance and tuning (e.g., index tuning), represent an invaluable source of information to define and model query-workloads, even complex in nature (e.g., analytical queries).

The approach used to effectively exploit the information embedded into the query-workload can be exploited in different ways, depending on the particular application scenario considered (e.g., relational databases, peer-to-peer databases, object-oriented databases, and so forth). In the particular context represented by the fragmentation of data warehouses, state-of-the-art approaches exploit *selection predicates* of workload queries in order to derive *suitable fragments*. Our proposed approach still belongs to this family. Figure 5 sketches our K-means-based XML data warehouse fragmentation technique. The proposed technique takes as input the XML data warehouse (both including schema and instance) and the reference query-workload. It returns as output the fragmented XML warehouse and the so-called *fragmentation schema*, which are ad-hoc meta-data describing how the data warehouse has been fragmented and schemas of fragments. These schemas are definitively useful to query optimization purposes. As intermediate steps, the following ones arise: (*i*) extraction of selection predicates from the workload queries; (*ii*) predicate clustering by means of algorithm K-means; (*iii*) fragment construction with respect to predicate clusters generated at the previous step.

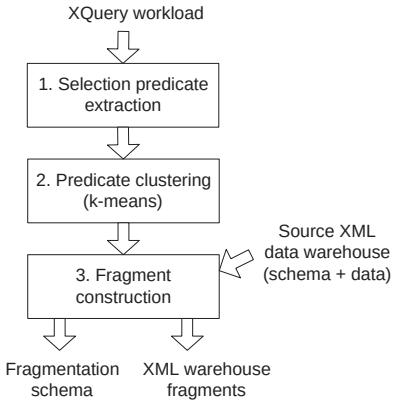

Figure 5 K-Means-Based XML Data Warehouse Fragmentation Overview

4.2 Extraction of Selection Predicates

Given a query-workload QW, the output selection predicate set SP is obtained by simply parsing queries in QW and extracting the predicates of such queries. For instance, consider Figure 6, where a sample XQuery workload $QW = \{q_1, q_2, ..., q_{10}\}$ is depicted. Figure 7 shows instead a portion of the output selection predicate set $SP = \{p_1, p_2, p_3, p_4, ...\}$, which has been geenrated according to our proposed approach. Here, for instance, p_2 and p_3 are selection predicates obtained from query $q_2 \in QW$. It should be noted how actually a large number of XML parsing tools such as *Java DOM* [47] are available in order to adequately fulfill the application requirement determined by the selection predicate extraction phase of our proposed XML data warehouse fragmentation technique.

```
q1    for $x in //FactDoc/Fact,
      $y in //dimension[@dim-id="Customer"]/Level/instance
      where $y/attribute[@id="c_nation_key"]/@value>"15"
      and $x/dimension[@dim-id="Customer"]/@value-id=$y/@id
      return $x

q2    for $x in //FactDoc/Fact,
      $y in //dimension[@dim-id="Customer"]/Level/instance,
      $z in //dimension[@dim-id="Part"]/Level/instance
      where $y/attribute[@id="c_nation_key"]/@value="13"
      and $y/attribute[@id="p_type"]/@value="PBC"
      and $x/dimension[@dim-id="Customer"]/@value-id=$y/@id
      and $x/dimension[@dim-id="Part"]/@value-id=$z/@id
      return $x
...
q10   for $x in //FactDoc/Fact,
      $y in //dimension[@dim-id="Customer"]/Level/instance,
      $z in //dimension[@dim-id="Date"]/Level/instance
      where $y/attribute[@id="c_nation_key"]/@value="13"
      and $y/attribute[@id="d_date_name"]/@value="Sat"
      and $x/dimension[@dim-id="Customer"]/@value-id=$y/@id
      and $x/dimension[@dim-id="Part"]/@value-id=$z/@id
      return $x
```

Figure 6 A Sample XQuery Workload

```
p1 = $y/attribute[@id="c_nation_key"]/@value>"15"
p2 = $y/attribute[@id="c_nation_key"]/@value="13"
p3 = $y/attribute[@id="p_type"]/@value="PBC"
p4 = $y/attribute[@id="d_date_name"]/@value="Sat"
```

Figure 7 Some Selection Predicates Extracted from the Sample XQuery Workload of Figure 6

Parsed predicates are then coded into a *Query-Predicate Matrix* QP, whose general term qp_{ij} is equal to 1 if the predicate $p_j \in SP$ appears in the query $q_i \in QW$, otherwise it is equal to 0. For instance, the matrix QP derived from the query-workload QW of Figure 6 and the selection predicate set SP of Figure 7 is featured in Table 1.

	p_1	p_2	p_3	p_4	...
q_1	1	0	0	0	
q_2	0	1	1	0	
...					
q_{10}	0	0	1	1	

Table 1 The Query-Predicate Matrix QP derived from the Query-Workload of Figure 6 and the Selection Predicate Set of Figure 7

It should be noted that, being matrix-based, the proposed approach could expose scalability issues. In particular, these problem could occur in the presence of query-workloads characterized by a high cardinality, and too "dense" queries, i.e. queries defined on top of a significant number of predicates. In turn, this originates a large number of rows and a large number of columns in the Query-Predicate matrix, respectively. In more detail, the number of columns of the Query-Predicate matrix also depends on the degree of similarity/dissimilarity between selection predicates embedded in the target query-workload. Similarly, the opposite problem could be experienced. When query-workloads characterized by a low cardinality and too "sparse" queries, i.e. queries defined on top of a small number of predicates, are handled, the extracted information (i.e., the selection predicate set) could not be

enough to fulfill the goal of building a "reliable" input for algorithm K-means. Contrary to the previous case, in this special case the derived Query-Predicate matrix is sparse. While both topics are very interesting and should merit a proper research effort, they are outside the scope of this paper, and we will hereafter assume of dealing with query-workloads that do not expose "problematic" characteristics whose some instances have been mentioned above.

4.3 Predicate Clustering

The main goal of our XML data warehouse fragmentation technique consists in obtaining fragments able to optimize data accesses for queries of the target query-workload. In turn, this allows us to take advantages in the query evaluation phase, as the overall response time of typical queries posed to the data warehouse (e.g., OLAP queries) can be lowered. Since horizontal fragments (Section 2.1) are built from selection predicates, clustering predicates with respect to queries achieves the goal above. Predicates that are syntactically similar are indeed grouped in a same cluster, which helps building an horizontal fragment. Intuitively enough, we ideally aim at building rectangles of 1s in the Query-Predicate matrix QP that correspond to clusters of predicates, as 1 denotes the occurrence of a predicate p_j in a certain query q_i. To this end, in our proposal we adopt the widely-used clustering algorithm K-means in order to effectively accomplish this task.

Given a data set D, algorithm K-means takes as input a vector of object attributes of D (i.e., predicates as columns of the Query-Predicate matrix QP, in our case), and returns as output a set of K clusters $C = \{C_1, C_2, ..., C_K\}$ by finding the *centers* of so-called "natural" clusters [28] in D via minimizing the total *intra-cluster variance* of C, which is defined as follows:

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

where x_j denotes a data item in D belonging to a certain cluster $C_i \in C$, and μ_i denotes the *centroid* (i.e., the mean point) of data items $x_j \in C_i$.

Usually, having K as an input parameter is viewed as a drawback for clustering algorithms, as this limits the quality of the final cluster set obtained. Contrary to this, in our proposed XML data warehouse fragmentation technique this peculiarity turns to be an advantage, since we aim at controlling and limiting the number of clusters/fragments generated by the fragmentation approach. As a baseline guideline, K could be set as equal to the number of nodes the XML data warehouse will be distributed on.

In our fragmentation framework, in order to exploit a reliable already-available implementation of K-means, we make use of *Weka* [44], a collection of Machine Learning algorithms for Data Mining tasks. In more detail, we exploit the Weka's SimpleKMeans implementation of K-means. Rather than more complex ones, SimpleKMeans makes use the *Euclidean* distance for computing distances between data items and clusters. Looking at our specific case, SimpleKMeans takes as input the matrix QP (actually, the vector of predicates $p_j \in SP$) and the parameter K, and returns as output the set of predicate clusters C. For instance, consider the Query-Predicate matrix QP of Table 1. By setting $K = 2$, SimpleKMeans produces the following output:

$$C = \{\{p_1\}, \{p_2, p_3, p_4\}\}$$

Finally, it should be noted how our proposed XML data warehouse fragmentation framework is indeed open to be customized for any other clustering algorithm beyond K-means. This nice feature, which makes our framework orthogonal to the particular clustering algorithm chosen, is indeed due to the independence ensured by the Query-Predicate matrix, on which any clustering algorithm can run.

4.4 Fragment Construction

The fragmentation construction step of our XML data warehouse fragmentation technique is composed by two sub-steps (Figure 8), the fragment schema construction and the proper fragment construction, respectively. In the first step, predicate cluster set C is joined to the warehouse schema stored in the document $dw - model.xml$ in order to produce a new XML document named as $frag - schema.xml$ that models the fragmentation schema (Figure 9). The root node of $frag - schema.xml$, called `Schema`, is composed by a set of `fragment` elements. Each `fragment` element models a fragment f generated by the fragmentation process. A `fragment` element contains the element `@id`, which models the absolute identifier of the fragment f, and a set of elements `dimension`, which model the warehouse dimensions. A `dimension` element contains the element `@name`, which models the name of the dimension d, and the element `predicate`, which stores the predicate p used for the fragmentation process. Finally, a `predicate` element contains the element `@name`, which models the name of the predicate p. To give an example, consider Figure 10, where the fragmentation schema corresponding to the cluster set C of the running example (Section 4.3) is shown.

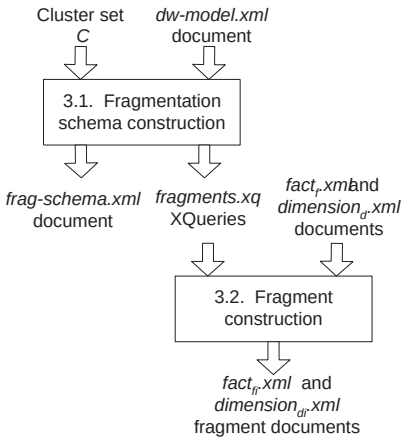

Figure 8 Fragment Construction Sub-Steps

The fragment schema construction sub-step also outputs a set of XQuery queries, which are stored in the script $fragments.xq$. Applied to the set of documents $facts_f.xml$ and $dimension_d.xml$ modeling the target XML data warehouse, these queries finally produce in output the actual set of fragments, which are stored in a set of documents $facts_{f_i}.xml$ and $dimension_{d_i}.xml$, with $i = 1, ..., K + 1$. These

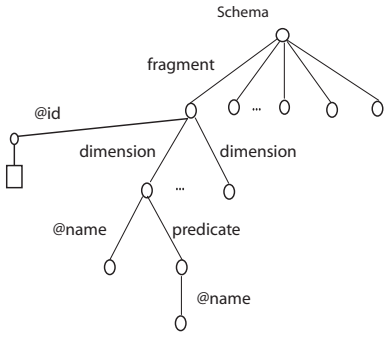

Figure 9 The XML Document $frag - schema.xml$

```
<Schema>
        <fragment id="f1">
                <dimension name="Customer">
                        <predicate name="p1"/>
                </dimension>
        </fragment>
        <fragment id="f2">
                <dimension name="Customer">
                        <predicate name="p2"/>
                </dimension>
                <dimension name="Part">
                        <predicate name="p3"/>
                </dimension>
                <dimension name="Date">
                        <predicate name="p4"/>
                </dimension>
        </fragment>
</Schema>
```

Figure 10 The Output XML Document $frag - schema.xml$ corresponding to the Running Example Fragmentation Process

documents represent the final result of the overall fragmentation process. As fragments, these documents indeed bear the same schema than the original data warehouse. In particular, the $(K+1)^{th}$ fragment/document is based on an additional predicate, named as $ELSE$, which is defined as the negation of the conjunction of all predicates in SP and it is necessary to ensure completeness of the fragmentation (Section 2.2). In our running example, $ELSE = \neg(p_1 \wedge p_2 \wedge p_3 \wedge p_4)$.

Figure 11 provides an excerpt from the script *fragments.xq* that generates fragment $f2$ of Figure 10. As shown in Figure 11, dimension fragments are generated first, one by one, through selections exploiting the predicate(s) associated to the current dimension (i.e., the first three queries from Figure 11). Then, fragmentation is derived on facts by joining the original fact document to the newly-created dimension fragments (i.e., the last query from Figure 11).

5 Experimental Assessment

It has been already demonstrated that derived horizontal fragmentation is an NP-hard problem [14]. It follows that devising a theoretical evaluation of our XML data warehouse fragmentation technique, even highly significant, would be particularly hard, although some asymptotic analysis for very simple cases could be

```
element dimension{ attribute dim-id{Customer}, element Level{
attribute id {Customers},
for $x in document("dimension_Customer.xml")//Level
where $x//attribute[@id="c_nation_key"]/@value="13"]
return $x }
}
element dimension{ attribute dim-id{Part}, element Level{
attribute id {Part},
for $x in document("dimension_Part.xml")//Level
where $x//attribute[@id="p_type"]/@value="PBC"]
return $x }
}
element dimension{ attribute dim-id{Date}, element Level{
attribute id {Date},
for $x in document("dimension_Date.xml")//Level
where $x//attribute[@id="d_date_name"]/@value="Sat"]
return $x }
}
element FactDoc {
for $x in //FactDoc/Fact,
$y in document("dimension_Customer_f2.xml")//instance,
$z in document("dimension_Part_f2.xml")//instance,
$t in document("dimension_Date_f2.xml")//instance
where $x/dimension[@dim-id="Customer"]/@value-id=$y/@id
and $x/dimension[@dim-id="Part"]/@value-id=$z/@id
and $x/dimension[@dim-id="Date"]/@value-id=$t/@id
return $x
}
```

Figure 11 Excerpt from the Script $fragments.xq$ Generating the Fragment $f2$ of Figure 10

still investigated. Therefore, in this Section we provide the experimental assessment of our proposed technique, which gives us a reliable case towards the validation of the effectiveness and efficiency of the technique.

5.1 Experimental Settings

In our experimental assessment, we use *XML Data Warehouse Benchmark* (XWeB) [56] as test platform. XWeB is a benchmark XML data warehouse based on the reference model presented in Section 3. XWeB also provides an XQuery-modeled decision-support query-workload that is exploited to stress the query performance of XML data warehouse query and processing algorithms running on the benchmark.

XWeB warehouse stores facts related to *Sales* of a typical retail application scenario, on top of which the following SUM-based measures are defined: *Amount* and *Quantity* (of purchased products). Four dimensions complete the XWeB multidimensional model: (*i*) *Customer*, which models customers purchasing products; (*ii*) *Supplier*, which models the suppliers furnishing products; (*iii*) *Date*, which models the temporal dimension of the XWeB warehouse; (*iv*) *Part*, which models the products. Facts are stored in the document $facts_{Sales}.xml$, whereas dimensions are stored in the documents $dimension_{Customer}.xml$, $dimension_{Supplier}.xml$, $dimension_{Date}.xml$ and $dimension_{Part}.xml$, respectively. XWeB warehouse characteristics are summarized in Table 2.

XWeB query-workload is composed by queries that exploit the warehouse through join and selection operations. In order to obtain a significant fragmentation, in our experimental assessment we extend the XWeB workload by adding selection pred-

Facts	**Maximum Number of Facts**
Sales	$7,000$
Dimensions	**Number of Instances**
Customer	$1,000$
Supplier	$1,000$
Date	500
Part	$1,000$
Documents	**Size (MB)**
$facts_{Sales}.xml$	2.14
$dimension_{Customer}.xml$	0.431
$dimension_{Supplier}.xml$	0.485
$dimension_{Date}.xml$	0.104
$dimension_{Part}.xml$	0.388

Table 2 XWeB Warehouse Characteristics

icates. The so-obtained workload is available at [55].

As regards XML data management aspects, in our experimental assessment we use the *X-Hive* XML native DBMS [82] to store and query the data warehouse. As regards the hardware infrastructure of our experimental framework, we use a *Pentium Core 2* host at 2 GHz equipped with 1 GB RAM and running *Windows XP*. Finally, our experimental software platform is written in *Java* and interacts with X-Hive and Weka through their respective APIs.

5.2 Comparison Fragmentation Techniques

In our experimental assessment, we compare our proposed K-means-based fragmentation technique (denoted as KM) with classical derived horizontal fragmentation techniques, namely predicate construction (denoted as PC) and affinity-based (denoted as AB) primary fragmentation techniques (Section 2.2), which we adapt and specialize to XML data warehouses [57]. In order to compare even with the baseline instance, we also consider the case in which no fragmentation is applied (denoted as NF).

5.3 First Experiment: Query Response Time

In the first experiment of our experimental campaign, we measure the query response time needed to evaluate all the queries of the target query-workload. For what regards KM, we arbitrarily fix $K = 8$, which could correspond to the number of hosts of a conventional cluster of computers. The fragments we obtain are stored in distinct collections, in order to simulate a reliable fragment distribution. This well simulates a setting in which each collection can be considered as stored on a distinct node of the network on which the data warehouse is distributed, and, moreover, each collection can be identified, targeted and queried separately. Overall, this realizes a distributed data warehouse environment finely. In order to measure the query execution time of the whole query-workload over the fragmented data warehouse,

we first identify fragments involved by queries thanks to the document $frag-schema.xml$, and then we execute queries over fragments and save execution times. To simulate parallel execution, like in a cluster computer scenario, we consider the maximum execution time. This provides us with a reliable estimation of the query response time needed to execute all the queries of the target query-workload due to a parallel execution.

Figure 12 shows the query response time for the target query-workload with respect to the data warehouse size expressed in number of facts. The Figure clearly demonstrates that fragmentation significantly improves query response time, and that *KM* fragmentation allows us to achieve a better performance than *PC* and *AB* fragmentation when the warehouse size scales up. Obviously, *KM* also outperforms *NF*. More precisely, workload execution time is, on the average, 86.5% faster with *KM* fragmentation than *PC* fragmentation, and 36.7% faster with *KM* fragmentation than *AB* fragmentation. Our approach performs better than classical derived horizontal fragmentation techniques also because the latter techniques originate much more fragments when compared with ours, i.e. 159 with *PC* fragmentation, 119 with *AB* fragmentation and 9 with *KM* fragmentation. Hence, when classical fragmentation techniques are applied, at workload execution time queries must access a large number of fragments (up to 50 from our observations of the actual experiment), which significantly multiplies both query distribution and result reconstruction costs. Contrary to this, when the *KM* fragmentation technique is applied, the number of accessed fragments is much lower (typically 2 fragments in the actual experiment).

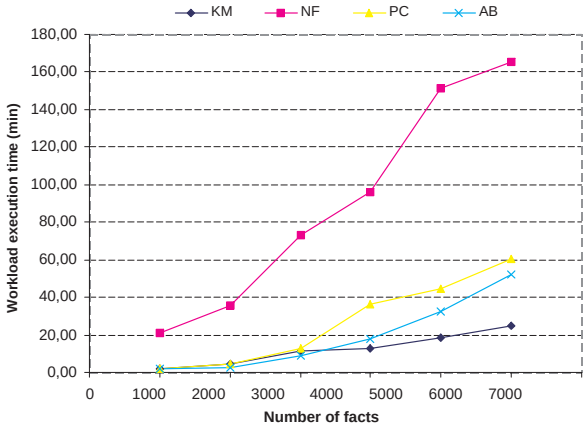

Figure 12 Query Response Time of Comparison Fragmentation Techniques

5.4 Second Experiment: Fragmentation Cost

In the second experiment of our experimental campaign, we compare the *PC*, *AB* and *KM* ($K = 8$) fragmentation strategies in terms of fragmentation costs, i.e. we investigate the execution time of proper fragmentation algorithms. Before going into details, we focus the attention on the complexity of fragmentation algorithms. Let $|SP|$ denote the cardinality of the selection predicate set SP. It follows than

the algorithm complexities for the comparison fragmentation techniques are the following: $O(2^{|SP|})$ for *PC*, $O(|SP|^2)$ for *AB*, and $O(|SP|)$ for *KM* fragmentation technique. Therefore, on a pure theoretical plane, our proposed XML data warehouse fragmentation technique exposes a complexity lower that those of comparison approaches.

Indeed, despite theoretical issues, when algorithms' performance is evaluated, it is necessary to find a fair trade-off between effective gain and computational overheads. Therefore, it is mandatory to develop a reliable experimental evaluation. In this respect, Table 3 summarizes the results we obtain for an arbitrarily-fixed data warehouse size equal to $3,000$ facts. Obtained results clearly show that *KM* fragmentation technique outperforms both *PC* and *AB* fragmentation techniques.

It should be noted that our results are not fully-in-line with above-introduced algorithms' complexities, as in our experimental assessment we include the time required by constructing fragments in the overall evaluation of computational overheads of algorithms. Hence, since *PC* and *AB* fragmentation techniques originate a large number of fragments, building such fragments requires a large number of costly join operations accordingly, thus leading to long running times. An immediate conclusion coming from this experimental evidence states that, while *PC* and *AB* fragmentation techniques are likely to run in an offline manner, *KM* fragmentation technique could on the other hand be envisaged to run in an online manner, thus turning to be perfectly suitable to OLAP applications.

	PC	***AB***	***KM***
Execution Time (h)	16.8	11.9	0.25

Table 3 Fragmentation Cost of Comparison Fragmentation Techniques

5.5 Third Experiment: Influence of the Number of Clusters

In the third experiment of our experimental campaign, we fix the data warehouse size to $4,000$ and $5,000$ facts, respectively, and vary the parameter K of the *KM* fragmentation technique in order to observe the influence of the number of clusters on the workload response time. Figure 13 confirms that performance improves quickly when fragmentation is applied, but it tends to degrade when the number of fragments increases, according to the discussion provided in Section 5.3. Furthermore, results depicted in Figure 13 suggest to us that the optimal number of clusters for our benchmark data warehouse and related query-workload lies between 4 and 6, which allows us to conclude that over-fragmentation (i.e., generating an excessive number of fragments) must be detected and avoided in distributed data warehouses (note that, in Figure 13, $K = 1$ corresponds to the *NF* experimental setting, i.e. one fragment corresponding to the original data warehouse).

6 Conclusions and Future Work

In this paper, we have introduced an approach for fragmenting XML data warehouses that is based on Data Mining, and, more precisely, on K-means clustering algorithm. Classical derived horizontal fragmentation strategies run automatically,

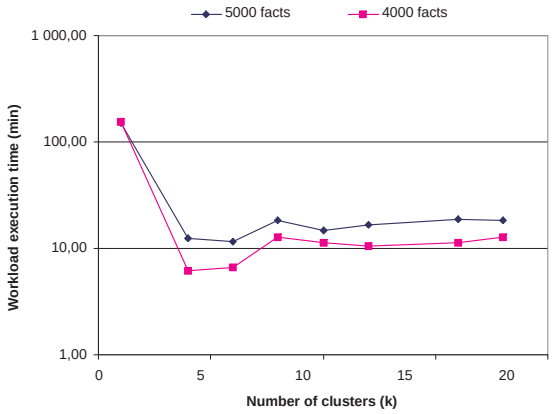

Figure 13 Influence of the Number of Clusters for the *KM* Fragmentation Technique

and output an unpredictable number of fragments, which is indeed nonetheless crucial to keep under control in realistic distributed data warehouses. By contrary, our proposed fragmentation approach allows us to fully master the number of fragments through the parameter K of K-means algorithm.

In order to validate the effectiveness and the efficiency of our proposal, we have compared our fragmentation strategy to meaningful adaptations of the two prevalent fragmentation methods for relational data warehouses, i.e. the *PC* and *AB* fragmentation techniques, to the specialized context of XML data warehouses. Obtained experimental results show that our approach significantly outperforms both comparison techniques (along with the baseline case in which no fragmentation is applied) under several perspective of experimental analysis.

Upon the fragmentation results above, future work is focused to the problem of effectively and efficiently distributing XML data warehouses on data grids. This issue raises several challenges that include decomposing a global query posed to the *grid-enabled XML data warehouse* into a set of sub-queries to be sent to the correct grid nodes, and meaningifully reconstructing the global result from intermediate sub-query results. In this direction, properly indexing the distributed data warehouse in order to guarantee good performance seems to be a critical aspect.

Finally, in a continuous effort towards minimizing data warehouse administration functions and aiming at auto-administrative systems [8, 9], we plan to make *dynamic* our Data-Mining-based fragmentation approach. Here, the main idea consists in performing *incremental fragmentation* as long as the target data warehouse is refreshed (e.g., during maintenance operations). This could be achieved by exploiting an *incremental* variant of K-means clustering algorithm [74].

References and Notes

1 Agrawal, S., Chaudhuri, S. and Narasayya, V.R. (2000) "Automated Selection of Materialized Views and Indexes in SQL Databases", *Proceedings of the 26th International Conference on Very Large Data Bases*, pp. 496–505.

2 Agrawal, R., Gehrke, J., Gunopulos, D., and Raghavan, P. (1998) "Automatic subspace

clustering of high dimensional data for data mining application", *Proceedings of 1998 ACM-SIGMOD International Conference on Management of Data*, pp. 94-105.

3 Agrawal, R., Imielinski T., Swami A. (1993) "Mining Association Rules between Sets of Items in Large Databases ", *in Proceedings of 1993 ACM-SIGMOD International Conference on Management of Data*, pp. 207-216.

4 Agrawal, S., Narasayya, V.R., and Yang, B. (2004) "Integrating Vertical and Horizontal Partitioning into Automated Physical Database Design", *Proceedings of the 2004 SIGMOD International Conference on Management of Data*, pp. 359–370.

5 Agrawal, R., and Srikant, R. (1994) "Fast Algorithms for Mining Association Rules in Large Databases", *Proceedings of the 20th International Conference on Very Large Data Bases*, pp. 487–499.

6 Andrade, A., Ruberg, G., Baião, F.A., Braganholo, V.P., and Mattoso, M. (2005) "PartiX : Processing XQuery Queries over Fragmented XML Repositories", *TR ES-691/05*, Computer Science Department, Federal University of Rio de Janeiro, Brazil.

7 Andrade, A., Ruberg, G., Baião, F.A., Braganholo, V.P., and Mattoso, M. (2006) "Efficiently Processing XML Queries over Fragmented Repositories with PartiX", *Proceedings of EDBT 2006 Workshops - Current Trends in Database Technology*, LNCS Vol. 4254, pp. 150–163.

8 Aouiche, K., Jouve, P.-E., and Darmont, J. (2006) "Clustering-Based Materialized View Selection in Data Warehouses", *Proceedings of the 10th East-European Conference on Advances in Databases and Information Systems*, LNCS Vol. 4152, pp. 81–95.

9 Azefack, S., Aouiche, K., and Darmont, J. (2007) "Dynamic Index Selection in Data Warehouses", *Proceedings of the 4th International Conference on Innovations in Information Technology*, `http://arxiv.org/pdf/0809.1965`.

10 Baril, X., and Bellahsène, Z. (2003) "Designing and Managing an XML Warehouse", *A.B. Chaudhri, A. Rashid, and R. Zicari (eds.), XML Data Management: Native XML and XML-enabled Database Systems*, Addison Wesley, pp. 455–473.

11 Bellatreche, L. (2000) "Logical and Physical Design in Data Warehousing Environments", *Proceedings of the EDBT 2000 PhD Workshop*, `://www.edbt2000.uni-konstanz.de/phd-workshop/papers/Belletreche.ps`.

12 Bellatreche, L. (2003) "Techniques d'Optimisation des Requtes dans les Data Warehouses", *Proceedings of the 6th International Symposium on Programming and Systems*, pp. 81–98.

13 Bellatreche, L., and Boukhalfa, K. (2005) "An Evolutionary Approach to Schema Partitioning Selection in a Data Warehouse", *Proceedings of the 7th International Conference on Data Warehousing and Knowledge Discovery*, LNCS Vol. 3589, pp. 115–125.

14 Bellatreche, L., Boukhalfa, K., and Richard, P. (2008) "Horizontal Partitioning in Data Warehouse: Hardness Study, Selection Algorithms and Validation on ORACLE10G", *Proceedings of the 10th International Conference on Data Warehousing and Knowledge Discovery*, LNCS Vol. 5182, pp. 87–96.

15 Bellatreche, L., Karlapalem, K., and Li, Q. (1998) "Derived Horizontal Class Partitioning in OODBs: Design Strategies, Analytical Model and Evaluation", *Proceedings of the 17th International Conference on Conceptual Modeling*, LNCS Vol. 1507, pp. 465–479.

16 Bellatreche, L., Karlapalem, K., Mohania M.K., and Schneider, M. (2000) "What Can Partitioning Do for Your Data Warehouses and Data Marts?", *Proceedings of the 4th International Database Engineering and Applications Symposium*, pp. 437–446.

17 Bellatreche, L., Karlapalem, K., and Mohania M.K. (2001) "Some Issues in Design of Data Warehousing Systems", *S.A. Becker (ed.), Developing Quality Complex Data Bases Systems: Practices, Techniques, and Technologies*, Idea Group Publishing, pp. 125–172.

18 Bellatreche, L., Schneider, M., Mohania, M.K., and Bhargava, B.K. (2002) "PartJoin: An Efficient Storage and Query Execution for Data Warehouses", *Proceedings of the 4th International Conference on Data Warehousing and Knowledge Discovery*, LNCS Vol. 2454, pp. 296–306.

19 Beyer, K.S., Chamberlin, D.D., Colby, L.S., Ozcan, F., Pirahesh, H. and Xu, J. (2005) "Extending XQuery for Analytics", *Proceedings of the 2005 SIGMOD International Conference on Management of Data*, pp. 503–514.

20 Boag, S., Chamberlin, D., Fernndez, M.F., Florescu, D., Robie, J., and Simon, J. (2007) *XQuery 1.0: An XML Query Language – W3C Recommendation*, `http://www.w3.org/TR/xquery/`.

21 Bonifati, A., and Cuzzocrea, A. (2007) "Efficient Fragmentation of Large XML Documents", *Proceedings of the 18th International Conference on Database and Expert Systems Applications*, LNCS Vol. 4653, pp. 539–550.

22 Bonifati, A., Cuzzocrea, A., Matrangolo, U., and Jain, M. (2004) "XPath Lookup Queries in P2P Networks", *Proceedings of the 6th International Workshop on Web Information and Data Management*, pp. 48–55.

23 Bonifati, A., Cuzzocrea, A., and Zinno, B. (2006) "Fragmenting XML Documents via Structural Constraints", *Proceedings of the 10th East European Conference on Advances in Databases and Information Systems*, pp. 17–29.

24 Bose, S., and Fegaras, L. (2005) "XFrag: A Query Processing Framework for Fragmented XML Data", *Proceedings of the 8th International Workshop on Web and Databases*, pp. 97–102.

25 Boukraa, D., Ben Messaoud, R., and Boussaïd, O. (2006) "Physical Design of XML Data Warehouse", *Proceedings of the 2006 Decision-Support Systems Workshop, 9th Maghrebian Conference on Information Technologies*.

26 Boussaïd, O., Ben Messaoud, R., Choquet, R., and Anthoard, S. (2006) "X-Warehousing: An XML-Based Approach for Warehousing Complex Data", *Proceedings of the 10th East-European Conference on Advances in Databases and Information Systems*, LNCS Vol. 4152, pp. 39–54.

27 Bremer, J.-M., and Gertz, M. (2003) "On Distributing XML Repositories", *Proceedings of the 6th International Workshop on Web and Databases*, pp. 73–78.

28 Carmichael, J. W., George, J. A., and Julius R. S. (1968) "Finding natural clusters" *In Journal of Systematic Zoology*, vol. 17, pp. 144-150.

29 Ceri, S., and Pelagatti, G. (1984) *Distributed Databases: Principles and Systems*, McGraw-Hill.

30 Clark, J., and DeRose, S. (1999) *XML Path Language (XPath) Version 1.0 – W3C Recommendation*, `http://www.w3.org/TR/xpath`.

31 Codd, E.F., Codd, S.B., and Salley, C.T. (1993) "Providing OLAP to User-Analysts: An IT Mandate", `http://dev.hyperion.com/resource_library/white_papers/providing_olap_to_user_analysts.pdf`.

32 Costa, C.L., and Furtado, P. (2006) "Data Warehouses in Grids with High QoS", *Proceedings of the 10th International Conference on Data Warehousing and Knowledge Discovery*, LNCS Vol. 4081, pp. 207–217.

33 Darabant, A.S., and Campan A. (2004) "Semi-Supervised Learning Techniques: K-Means Clustering in OODB Fragmentation", *Proceedings of the 2nd International Conference on Computational Cybernetics*, pp. 333–338.

34 Darmont, J., Boussaïd, O., Ralaivao, J.C., and Aouiche, K. (2005) "An Architecture Framework for Complex Data Warehouses", *Proceedings of the 7th International Conference on Enterprise Information Systems*, pp. 370–373.

35 Datta, A., Ramamritham, K., and Thomas, H.M. (1999) "Curio: A Novel Solution for Efficient Storage and Indexing in Data Warehouses", *Proceedings of the 25th International Conference on Very Large Data Bases*, pp. 730–733.

36 Fiolet, V., and Toursel T. (2005) "Progressive Clustering for Database Distribution on a Grid", *Proceedings of the 4th International Symposium on Parallel and Distributed Computing*, pp. 282–289.

37 Fomichev, A., Grinev, M., and Kuznetsov, S. (2006) "Sedna: A Native XML DBMS", *Proceedings of the 32nd International Conference on Current Trends in Theory and Practice of Computer Science*, pp. 272-281.

38 Gertz, M., and Bremer, J.-M. (2003) "Distributed XML Repositories: Top-down Design and Transparent Query Processing", *TR CSE-2003-20*, Departement of Computer Science, University of California, Davis, CA, USA.

39 Golfarelli, M., Maio, D., and Rizzi, S. (1998) "The Dimensional Fact Model: a Conceptual Model for Data Warehouses", *International Journal of Cooperative Information Systems*, Vol. 7, No. 2-3, pp. 215–247.

40 Golfarelli, M., Maio, D., and Rizzi, S. (1999) "Vertical Fragmentation of Views in Relational Data Warehouses", *Proceedings of the 7th Italian Symposium on Advanced Database Systems*, pp. 19–33.

41 Golfarelli, M., and Rizzi, S. (2001) "Data Warehouse Design from XML Sources", *Proceedings of the 4th International Workshop on Data Warehousing and OLAP*, pp. 40–47.

42 Gorla, N., and Betty, P.W.Y. (2008) "Vertical Fragmentation in Databases Using Data-Mining Techniques", *International Journal of Data Warehousing and Mining*, Vol. 4, No. 3, pp. 35–53.

43 Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F., and Pirahesh, H. (1997) "Data Cube: A Relational Aggregation Operator Generalizing Group-by, Cross-Tab, and Sub Totals", *Data Mining and Knowledge Discovery*, Vol. 1, No. 1, pp. 29–53.

44 Holmes, G., Donkin, A., and Witten I.H. (1994) "Weka: A Machine Learning Workbench", *Proceedings of the 2nd Australia and New Zealand Conference on Intelligent Information Systems*, pp. 357–361.

45 Hümmer, W., Bauer, A., and Harde G. (2003) "XCube: XML for Data Warehouses", *Proceedings of the 6th International Workshop on Data Warehousing and OLAP*, pp. 33–40.

46 Jagadish, H.V., Al-Khalifa, S., Chapman, A., Lakshmanan, L.V.S., Nierman, A., Paparizos, S., Patel, J.M., Srivastava, D., Wiwatwattana, N., Wu, Y., and Yu, C. (2002) "TIMBER: A native XML database", *VLDB Journal*, Vol. 11; No. 4, pp. 274–291.

47 *Java Document Object Model*, `http://www.jdom.org/`.

48 Kalnis, P., Ng, W.S., Ooi, B.C., Papadias, D., and Tan, K.L. (2002) "An adaptive peer-to-peer network for distributed caching of OLAP results", *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pp. 25–36.

49 Kimball, R., and Ross, M. (2002) *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling* (2nd Edition), Wiley.

50 Koreichi, A., and Le Cun, B. (1997) "On Data Fragmentation and Allocation in Distributed Object Oriented Databases", *TR 1997-11*, PRiSM Laboratory, Versailles University, France.

51 Ma, H., and Schewe, K.-D. (2003) "Fragmentation of XML Documents", *Proceedings of the 18th Brazilian Symposium on Databases*, pp. 200–204.

52 Ma, H., and Schewe, K.-D., and Hartmann, S., and Kirchberg, M. (2003) "Distribution Design for XML Documents", *Proceedings of the 3rd International Conference on Electronic Commerce Engineering*, pp. 1007–1012.

53 Ma, H., Schewe, K.-D., and Wang, Q. (2006) "A Heuristic Approach to Cost-Efficient Fragmentation and Allocation of Complex Value Databases", *Proceedings of the 17th Australasian Conference on Database Technologies*, pp. 183–192.

54 MacQueen, J.B. (1967) "Some Methods for classification and Analysis of Multivariate Observations", *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281–297.

55 Mahboubi, H. (2008) *XWeB Query-Workload*, `http://eric.univ-lyon2.fr/~hmahboubi/Workload/workload.xq`.

56 Mahboubi, H., Darmont, J. (2006) "Benchmarking XML data warehouses", *Proceedings of the 2006 Decision-Support Systems Workshop, 9th Maghrebian Conference on Information Technologies*.

57 Mahboubi, H., and Darmont, J. (2009) "Enhancing XML Data Warehouse Query Performance by Fragmentation", *Proceedings of the 24th Annual ACM Symposium on Applied Computing*, pp. 1555–1562.

58 Mahboubi, H., Marouane, H., and Darmont, J. (2008) "XML Warehousing and OLAP", *J. Wang (ed.), Encyclopedia of Data Warehousing and Mining, Second Edition*, IGI Global, pp. 2109–2116.

59 Marouane, H., Mahboubi, H., and Darmont, J. (2008) "Expressing OLAP Operators with the TAX XML Algebra", *Proceedings of the 3rd International Workshop on Database Technologies for Handling XML Information on the Web*, pp. 61–66.

60 Meier V. (2002) "eXist: An Open Source Native XML Database", *A. Chaudhri, M. Jeckle, E. Rahm, and R. Unland (eds.), Web, Web-Services, and Database Systems, NODe 2002 Web and Database-Related Workshops*, LNCS Vol. 2593, Springer, pp. 169–183.

61 Munneke, D., Wahlstrom, K., and Mohania, M.K. (1999) "Fragmentation of Multidimensional Databases", *Proceedings of the 10th Australasian Database Conference*, pp. 153–164.

62 Nassis, V., Rajugan, R., Dillon, T.S., and Rahayu, J.W. (2005) "Conceptual and Systematic Design Approach for XML Document Warehouses", *International Journal of Data Warehousing and Mining*, Vol. 1, No. 3, pp. 63–86.

63 Navathe, S., Ceri, S., and Wiederhold, G., and Dou, J. (1984) "Vertical Partitioning Algorithms for Database Design", *Transactions on Database Systems*, Vol. 9, No. 4, pp. 680–710

64 Navathe, S.B., Karlapalem, K., and Minyoung, R. (1995) "A Mixed Fragmentation Methodology for Initial Distributed Database Design", *Journal of Computer and Software Engineering*, Vol. 3, No. 4, pp. 395–426.

65 Navathe, S., and Ra, N. (1989) "Vertical Partitionning for Database Design: A Graphical Algorithm", *Proceedings of the 1989 SIGMOD International Conference on Management of Data*, pp. 440–450.

66 Noaman, A.Y., and Barker, K. (1999) "A Horizontal Fragmentation Algorithm for the Fact Relation in a Distributed Data Warehouse", *Proceedings of the 8th International Conference on Information and Knowledge Management*, pp. 154–161.

67 Özsu, M.T., and Valduriez, P. (1999) *Principles of Distributed Database Systems*, Prentice-Hall.

68 Paparizos, P., Wu, Y., Lakshmanan, L.V.S., and Jagadish, H.V. (2004) "Tree Logical Classes for Efficient Evaluation of XQuery", *Proceedings of the 2004 SIGMOD International Conference on Management of Data*, pp. 71–82.

69 Park, B.K., Han, H., and Song, I.Y. (2005) "XML-OLAP: A Multidimensional Analysis Framework for XML Warehouses", *Proceedings of the 7th International Conference on Data Warehousing and Knowledge Discovery*, LNCS Vol. 3589, pp. 32–42.

70 Pernul, G., Karlapalem, K., and Navathe, B. (1991) "Relational Database Organization Based on Views and Fragments", *Proceedings of 2nd International Conference Database and Expert Systems Applications*, pp. 380–386

71 Pham, D.T., Dimov, S.S., and Nguyen, C.D. (2004) "An Incremental K-Means Algorithm", *Journal of Mechanical Engineering Science*, Vol. 218, No. 7, pp. 783–795.

72 Pokorný, J. (2002) "XML Data Warehouse: Modelling and Querying", *Proceedings of the 5th Baltic Conference on Databases & Information Systems*, pp. 267–280.

73 Rajugan, R., Chang, E., and Dillon, T.S. (2005) "Conceptual Design of an XML Fact Repository for Dispersed XML Document Warehouses and XML Marts", *Proceedings of the 20th International Conference on Computer and Information Technology*, pp. 141–149.

74 Rusu, L.I., Rahayu, J.W., and Taniar, D. (2005) "A Methodology for Building XML Data Warehouses", *International Journal of Data Warehousing and Mining*, Vol. 1, No. 23, pp. 67–92.

75 Vrdoljak, B., Banek, M., and Rizzi, S. (2003) "Designing Web Warehouses from XML Schemas", *Proceedings of the 5th International Conference on Data Warehousing and Knowledge Discovery*, LNCS Vol. 2737, pp. 89–98.

76 Wehrle, W., Miquel, M., and Tchounikine, A. (2005) "A Model for Distributing and Querying a Data Warehouse on a Computing Grid", *Proceedings of the 11th International Conference on Parallel and Distributed Systems*, pp. 203–209.

77 Wiwatwattana, N., Jagadish, H.V., Lakshmanan, L.V.S., and Srivastava, D. (2007) "X^3: A Cube Operator for XML OLAP", *Proceedings of the 23rd International Conference on Data Engineering*, pp. 916–925.

78 Wu, M.-C., and Buchmann, A.P. (1997) "Research Issues in Data Warehousing", *Proceedings of the 7th German Conference on Database Systems in Office, Engineering, and Scientific Applications*, pp. 61–82.

79 W3Schools (2009) *DTD Tutorial*, `http://www.w3schools.com/dtd/`.

80 *XQuery 1.0 and XPath 2.0 Functions and Operators – W3C Recommendation*, `http://www.w3.org/TR/xpath-functions/`.

81 Xyleme, L. (2001) "Xyleme: A Dynamic Warehouse for XML Data of the Web, *Proceedings of the 5th International Database Engineering and Applications Symposium*, pp. 3–7.

82 *X-Hive XML Database*, `http://www.x-hive.com/products/db/`.

83 Zaman, M., Surabattula, J., and Gruenwald, L. "An Auto-Indexing Technique for Databases Based on Clustering", *Proceedings of the 15th International Workshop on Database and Expert Systems Applications*, pp. 776–780.

84 Zhang, Y., and Orlowska, M.E. (1994) "On Fragmentation Approaches for Distributed Database Design", *Information Sciences*, Vol. 1, No. 3, pp. 117–132.

85 Zhang, J., Wang, W., Han, L., and Zhang, S. (2005) "X-Warehouse: Building Query Pattern-driven Data", *Proceedings of the 14th International Conference on World Wide Web*, pp. 896–897.