XML Warehouse Modelling and Querying

Fatma Abdelhedi, Landry Ntsama, and Gilles Zurfiuh

IRIT SIG/ED - 118 route de Narbonne, 31062 Toulouse, France
{fatma.abdelhedi,landry.ntsama,gilles.z urfluh}@irit.fr
http://www.irit.fr/-Equipe-SIG

Abst ract. Integrating XML documents in data warehouse is a major issue for decisional data processing and business intelligence. Indeed this type of data is increasingly being used in organisations' information system. But the current warehousing systems do not manage documents as they do for extracted data from relational databases. We have therefore developed a multidimensional model based on the Unified Modeling Language (UML), to describe an XML Document Warehouse (XDW). The warehouse diagram obtained is a Star schema (StarCD) which fact represents the documents class to be analyzed, and the dimensions correspond to analysis criteria extracted from the structure of the documents. The standard XQuery language can express queries on XML documents, but it is not suitable for analyzing a warehouse as its syntax is too complex for a non IT specialist. This paper presents a new language aimed at decision-makers and allows applying OLAP queries on a XDW described by a StarCD.

Keywords: XML warehouse, OLAP, XQuery, Multidimensional Modelling.

1 Introduct ion

Due to the increasing volume of information contained in databases and web, decision-makers are facing problems concerning the extraction, the aggregation and the analysis of data containing heterogeneous formats. In such an available mass of information, only transactional data extracted from relational databases are mainly exploited. Complex data contained in XML documents [9] are on the other hand barely exploited, even not at all. Yet, all these data represent a significant source of information for the decision-making process.

Thus, in order to integrate XML documents into a warehouse, it is first of all advisable to have a multidimensional model. We have extended the classical star diagram model [2], proposing a model where facts and dimensions are extracted from XML-Schemas [11]) that describe the documents in the source. The hierarchical structure of the documents is then partially preserved, making the dimensional schema easy to understand for the decision-makers.

We consider our model completely defined. Therefore, this paper is mainly dedicated to an OLAP query language that allows decision-makers to analyse an XDW described by a StarCD. This language is indented for non IT specialists, and its relatively simple syntax is based on the principles developed for the complex object query language [1].

2 Related Works

Our work is developed within the context of warehousing of "documents-centric" XML documents [6]. We aim to define mechanisms that allow the decision-makers to simply manipulate complex objects organized according to a hierarchical structure (XML-Schema).

In this context, the authors of [3] propose to describe an XDW using the class diagram model of UML. In their warehouse schema, the documents are represented by a fact **(xFact)**linked to virtual dimensions. The authors emphasize the modelling aspect, without defining a concrete approach for the OLAP analysis. But they briefiy present an algorithm based on XQuery to analyze a warehouse. The proposed data model in their paper enables to describe an XDW using objects classes linked to each other, and organized and defined following the user requirements. So the documents structure does not appear in the schema, as it is represented in the document source.

The authors in [5] propose a data model identical to the one above to describe an XDW. They extend the query language MDX which is basically an OLAP query language for the classical data warehouse. This extended language (XML-MDX) integrates a set of operators defined to manipulate an XML cube, allowing particularly the analysis of textual contents. The XQuery is used during the creation of the XML cube, for the calculation of the measures and definition of the dimensions. As in [3], the document structure is not represented in the warehouse schema. Furthermore, the query language designed for the OLAP analysis turns out to be complex because it requires that the decision-maker has computing knowledge, yet he is not an IT specialist.

In [7], the authors propose a new multidimensional model for the OLAP analysis of XML documents. This model called "the galaxy model" allows to define a warehouse schema as a dimensions graph. The dimensions of a galaxy are linked to each other by one or several nodes that express the compatibility between them. The fact is chosen among dimensions during the warehouse querying process. So a dimension in a galaxy schema represents as well an axis and a subject of analysis. The authors extend the classic multidimensional algebra which they adapt to their model, joining new operators into it, particularly for the textual data analysis. In their paper, the proposed data model for the XDW does not keep the initial structure of the documents. The elements in the XML documents are unstructured in the warehouse schema, so we could reasonably think that it establishes an obstacle for the query expression by the decision-makers.

To our knowledge, there is currently no modelling standard defined for XML documents. Furthermore, the works presented above only answer partially the problem we address. In our approach, we assume that the decision-makers know the structure of the document they need to analyze. Thus they can retrieve this structure in XDW schema and easily express their analysis OLAP queries.

3 Contribution

3.1 Warehouse Model Formalization

We have developed a multidimensional model that describes a XDW, under the shape of a star schema named StarCD. This model is based on the UML formalism [4] to represent Facts and Dimensions. It is built from the analysis of an XML document source described by a unique XML-Schema (XSchema), and it supports the creation of the analytics queries by decision-makers.

A StarCD is defined as follows:
$StarCD = (F, D)$ where:

- F corresponds to the fact of the schema, and it represents also the root element of the XSchema that define the analysed document class.
- $D = \{D_1, \ldots, Dn\}$ is a set of dimensions associated to the fact.

The fact is characterized by a set of measures, being textual or numeric type. The measures are described either by attributes or classes linked to the fact by aggregation links. The hierarchical structure is then identical to the one represented in the XSchema of the source. The fact is defined as follows:
$F = (M, Agg)$ where:

$M = \{M_1, \ldots, Mp\}$ is a set of measures.
$Agg = \{ag_1, \ldots, agp\}$ is the set of aggregation fonctions associated to each measure, with ag included in $\{SUM, COUNT, AVG, MAX, MIN\}$.

The dimensions are characterized by parameters, organized into hierarchies. Those hierarchies indicate the granularity level going from the top level parameter (All) to the bottom level one, corresponding to the parameter linked to the fact into the star schema. Dimensions are defined as follows:
$D = (P, H)$ where:

$P = \{P1, \ldots, ps\}$ is the set of parameters of the dimension D.
$H = \{h_1, \ldots, hs\}$ is the set of hierarchies in which parameters are organized, having h defined as follows:

- $h = \{p_1, p_2, \ldots, All\}$ where pl is the parameter linked to the fact (corresponding to the bottom level parameter).

In the StarCD, dimensions are described as a set of classes, each class representing a parameter that enables to aggregate some fact instances. Each dimension is a tree structure whose root is linked to the fact by an association link "By". The weak attributes correspond to the attributes of classes in the schema.

The presented model introduces the *reverse hierarchy* concept. In that type of hierarchy, the top level parameter is actually linked to the fact in order to keep the XSchema structure in the star schema presented to the decision-maker.

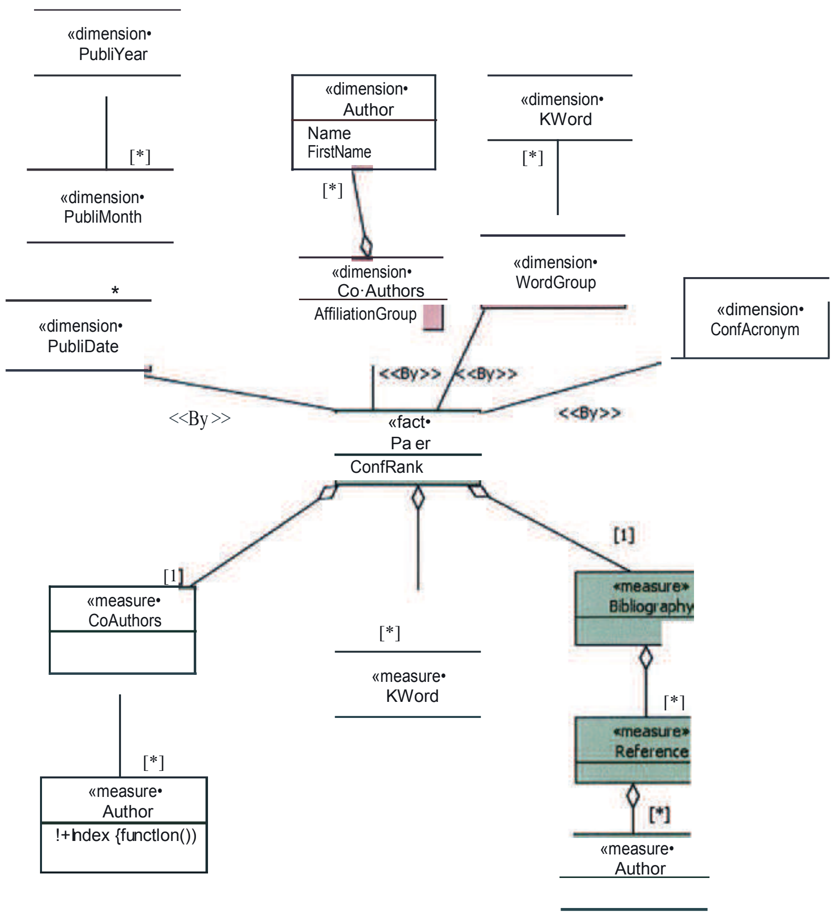


Fig. 1. XML Warehouse schema(StarCD)

Example 1: The StarCD presented in Fig. 1 represents a warehouse schema created from a collection of scientific papers; fact and measures are in the bottom of the schema, and dimensions are in the top.

This StarCD respects the hierarchical tree structure defined in the XSchema, from which it has been created. Thus, we can see an example of reverse hierarchy in the dimension Co-Authors. The class Co-Authors is directly linked to the fact because it is a first level parameter in the XSchema that describes the document source. Thus in the schema, the parameters in this dimension are reversed.

3.2 Multidimensional Expression Language

The query language presented in this section is inspired by the object query language OQL [1]. It allows decision-makers to express complex OLAP queries, and also allows the manipulation of the objects presented in the multidimensional schema StarCD. Its structure is built to make the navigation in the tree structure of an XML file easy by means of simple queries, and by using the following syntax:

```
Analyse <measure>
From <variables that define the warehouse classes>
By <dimensions>
```

The clause *Anal yse* contains the measures and their applied aggregation fonctions. The clause *From* indicates the elements in the StarCD which will be used either as measures, or dimensions. Each element is associated to a variable ("alias") that helps to avoid ambiguity during the hierarchies' course. The clause *By* specifies the variable of the dimensions to use. The BNF structure of a query is as follows:

```
<Query_spec>  ::= <Analyse_clause><From_cla use><By_clause>
<Analyse_clause> ::= Analyse <Operator> (<Meseure_name>)
<From_clause>  ::= From <alias> in <XML_Element_Name>
 (. <XML_Element_Name>) * (, <alias> in <XML_Element_Name>
 (. <XML_Element_Name>) *)*
<By_clause> ::= By <alias>( , <alias>) *
<Operator> ::= COUNT  ı  SUM  ı  AVG  ı  MAX  ı  MIN
```

In [8], the authors defined a "closed core" of OLAP operators. We picked some of those operators dedicated to complex analysis, to express multidimensional operations from the model above presented. Those operators are:

Granularity-related operators: Drill-down, Roll-up
Structural operators: DRotate, Switch, Nest

In order to illustrate the impact of those operators, we will consider for example the query Rl expressed using the StarCD in Fig. 1. That query calculates the number of papers published by publication's date (parameter PubliDate):

```
Ri:     Analyse count(p)
        From p in Paper, d in p.PubliDate
        By d
```

Granularity-related Operators. Granularity-related operators allow analyzing data by aggregating or desegregating them according to the level of detail required. They change the granularity level of the parameter used for data calculation. This change of level can be done upwards (Roll-up) or downwards (Drill-down).

Example 2
Roll-up: This operator aggregates measures by changing the parameter of a dimension. Hence, the query R2 will count the number of papers ordered by publication date, increasing the date level to month:

```
R2:     Analyse count(p)
        From p in Paper, d in p.PubliDate, m in d.PubliMonth
        By m
```

Drill-down: This operator increases the level of detail of measures, that is, it represents the element used in the calculation at a lower level of granularity in the clause From. So the query R3 will count the number of papers by publication date, but decreasing the aggregation level on date to be measured in days. The queries R3 and Rl are equivalent then:

```
R3:     Analyse count(p)
        From p in Paper, d in p.PubliDate
        By d
```

On reverse hierarchies, these two operators above are also reversed.

Struct ural Operators. The rotation operator DRotate changes the dimension used for the calculation of a query. It means, in our case, a change in the parameter we need to use in the clause From.

Example 3: The query R4 calculates the number of papers by group of authors (Co-Authors) . A rotation has been performed on the query R3 to change the dimension PubliDate by Co-Authors:

```
R4:     Analyse count(p)
        From p in Paper, co in p.Co-Authors
        By co
```

The operator Switch exchanges the positions of two or more variables of a parameter in a dimension. It only implies a visual effect since the measures values do not change. The operator Nest imbricates a parameter into another.

Example 4: Let's consider the following query that calculates the number of papers by authors, publication dates and key-word (parameters Author, PubliDate and KWord) . The query is expressed as follows:

```
Analyse count(p)
From p in Paper,
     k in p.WordGroup.KWord,
     a, in p.CoAuthors.Author,
     d in p.PubliDate
By a.FirstName, a.LastName, k, d;
```

Switch will exchange the positions of variables a.FirstName and a.LastName, the clause By then becomes:

```
By a.LastName, a.FirstName, k, d
```

Nest will imbricate the variable d (for PubliDate) into the variable k (for KWord). The clause By is again transformed and becomes:

```
By a.LastName, a.FirstName, k.d
```

Likewise the Switch operator, Nest also has a merely visual effect ; therefore, the previously calculated values do not change.

We consider that an XML structure can be defined as a tree structure graph; thus, the presented language allows navigating in that tree, and helps decision-makers to identify the depth of the elements during the query processing.

Query Expression. We propose below two queries examples defined from the StarCD.

Query 1: Calculate the number of papers by conference
Measure: count (Paper)
Parameter: ConfAcronym

```
Analyse count (p)
From p in Paper, c in p.ConfAcronym
By c
```

This query calculates for each distinct value of ConfAcronym, the associated number of papers.

Query 2: Calculates the number of papers by key-word and by year of publication
Measure: count (Paper)
Parameter: KWord (dimension WordGroup) and PubliYear (dimension PubliDate)

```
Analyse count (p)
From p in Paper,
     k in p.WordGroup.KWord,
     d in p.PubliDate.PubliYear
By k, d
```

The query groups the papers by key-words, and after by year of publication. For each couple, it then counts the number of papers in each corresponding group.

3.3 Experimentation

The materialized warehouse is supplied from a scientific document collection, sharing the same XML structure (XSchema). It contains as much facts as the source has documents. The dimensions are stored in distinct documents, and are linked to the fact by references (see Fig. 3). Indeed, we do believe that saving facts and dimensions in separated xml files insures the singularity of the values of the dimensions, and also helps to suppress the redundancy in the warehouse.

The physical XDW have been created into the database eXist-DB, which is a native XML database that enables to easily manipulate XML collections with the query language XQuery [10]. In that context, the queries expressed by a decision-maker are then translated into XQuery in order to be applied to the warehouse. Hence, a translator has been created to realise that operation. It performs an automatic translation of any analytic query into an XQuery query, with transparency for the user. The architecture of the translator is shown in Fig. 2. This translator takes the OLAP query written by the decision-maker, and uses the warehouse schema to build the translated query.

Once the query is seized by the user, the translator proceeds to perform both a lexical and a syntactic analysis. After that, the query is sent to the code generator, which performs a semantic analysis (type verification and optimisation based on the source language) before finally translating the query into XQuery. We have defined the following translation method:

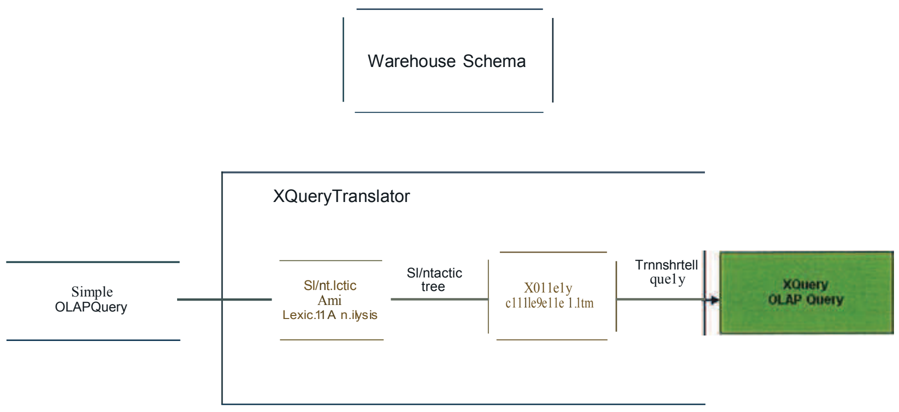
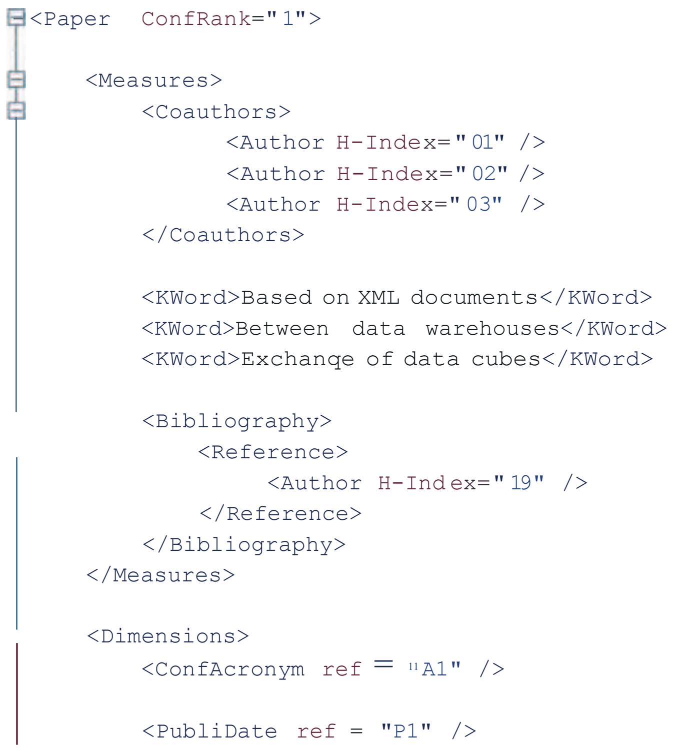


Fig. 2. Translator Architecture


```
<Paper  ConfRank="1">

    <Measures>
        <Coauthors>
              <Author H-Index="01" />
              <Author H-Index="02" />
              <Author H-Index="03" />
        </Coauthors>

        <KWord>Based on XML documents</KWord>
        <KWord>Between  data  warehouses</KWord>
        <KWord>Exchanqe of data cubes</KWord>

        <Bibliography>
            <Reference>
                 <Author H-Index="19"  />
            </Reference>
        </Bibliography>
    </Measures>

    <Dimensions>
        <ConfAcronym ref = "A1"  />

        <PubliDate ref = "P1"  />
```

Fig. 3. XML Fact Structure

The clause Analyse: the measures in this clause correspond to the results to be returned. Those values will appear in the clause *return* of the XQuery language.
The clause From: a clause *for* will be defined in XQuery for each variable in the clause From. Only the variables defined as dimensions (variables in the clause By of the analytic Query) will have an equivalent clause *for*.

The clause **By:** The variables in this clause have several roles:

- They allow to define the dimensions
- They allow to define the display order (the clause ***order by*** in XQuery)
- They allow to apply a restriction to the facts that are linked to the chosen dimensions (The clause ***let*** in XQuery)

Example 5: We present in Tab. 1 and Tab. 2 two analysis queries examples, showing their translations into XQuery, as realized by the translator.

Table 1. Query 1 Calculate for each conference the average number of authors)

Analysis Query	Translated Query

`Analyse avg(count(a))`
`From p in Paper,`
`    a in p.CoAuthors.Author,`
`    c in p.ConfAcronym`
`By c`
	`for $a in //DConfAcronym/ConfAcronym`
	`let $doc := //Paper[Dimmensions`
	`      /ConfAcronym/©ref = $a/©id]`
	`      /Measures/MCoauthors/MAuthor`
	`return`
	`<group>`
	`   <Acronym>{$a/text()}</Acronym>`
	`   <avgAuthor>{avg(count($doc))}</avgAuthor>`
	`</group>`

Table 2. Query 2 Calculates the number of papers by key-word and by year of publication

Analysis Query	Translated Query

`Analyse count(p)`
`From p in Paper,`
`    k in p.WordGroup.KWord,`
`    d in p.PubliDate.PubliYear`
`By k, d`
	`for $a in //DWordGroup/WordGroup,`
	`    $b in distinct-values($a/KWord),`
	`    $c in //DPubliDate/PubliDate,`
	`    $d in $c/PubliYear/©year`
	`   let $doc := //Paper[Dimensions`
	`            /WordGroup/©ref = $a/©id`
	`            and Dimensions/PubliDate`
	`            /©ref = $c/©id]`
	`   return`
	`   if (exists($doc))`
	`   then <group>`
	`         <Kword>{$b}</Kword>`
	`         <Year>{data($d)}</Year>`
	`         <nbPaper>{count($doc)}</nbPaper>`
	`       </group>`
	`   else()`

The queries presented above are illustrating the simplicity to express an OLAP query using our language, compared to the same expression written in XQuery. Then the decision-maker has the benefits of the power of XQuery, being preserved from its complexity due to the transparency given by the translator.

4 Conclusions

In this paper, we proposed a new OLAP query language for the XML document warehouses. This language is applied on a multidimensional model defined from the object modelling formalism UML, and helps the decision-maker to be preserved from the complexity of the existing query languages or algebra. This language is translated into XQuery in order to be applied on a XDW, so we developed a translator that translate automatically any simple OLAP query into XQuery and apply it on a materialized XDW. We used the JavaCC technology (Java Compiler Compiler) to build the translator, because it allows combining the advantages of Java and the Language Theory.

We intend to integrate some new operators (such as Push, Pull or HRotate) to manage more complex queries. Furthermore, to validate our approach we are developing a prototype that will enable a semi-automatic modelling process for a document warehouse, as well as its querying through its StarCD.

References

1. Barry, D.K., Cattell, R.G.G.: The Obkect Database Standard. Morgan Kaufmann publisher (1997)
2. Golfarelli, M., Maio, D., Rizzi, S.: The dimensional fact model: a conceptual model for data warehouses. International Journal of Cooperative Information Systems 07 (1998)
3. Nassis, V., Rajagopalapillai, R., Dillon, T.S., Rahayu, W.: Conceptual and systematic design approach for xml document warehouses (2005)
4. Object Management Group (OMG): Unified Modelling Language (UML), http://www.uml.org/
5. Park, B.-K., Han, H., Song, I.-Y.: XML-OLAP: A multidimensional analysis framework for XML warehouses. In: Tjoa, A.M., Trujillo, J. (eds.) DaWaK 2005. LNCS, vol. 3589, pp. 32-42. Springer, Heidelberg (2005)
6. Pérez, J.M., Llavori, R.B., Aramburu, M.J., Pedersen, T.B.: Integrating data warehouses with web data: A survey. IEEE Trans. Knowl. Data Eng. 20(7), 940-955 (2008)
7. Ravat , F., Teste, O., Tournier: R., Zurfluh, G.: A conceptual model for multidimensional analysis of documents (2007)
8. Ravat, F., Teste, O., Zufluh, G.: Algèbre OLAP et langage graphique (2006)
9. W3C-Consortium: Extensible Markup Language (XML), http://www.w3.org/XML/
10. W3C-Consortium: W3C XML Query (XQuery), http://www.w3.org/XML/Query/
11. W3C-Consortium: XML Schema Part 0: Structures, 2nd edn., http://www.w3.org/TR/2004/REC-xmlschema-1-2 0041028/