On Building XML Data Warehouses

Laura Irina Rusu[1], Wenny Rahayu[1], and David Taniar[2]

[1] LaTrobe University, Department of Computer Science and Computer Engineering
Bundoora, VIC 3086, Australia
{lrusu,wenny}@cs.latrobe.edu.au
[2] Monash University, School of Business Systems, Clayton, VIC 3800, Australia
David.Taniar@infotech.monash.edu.au

Abstract. Developing a data warehouse for XML documents involves two major processes: one of creating it, by processing XML raw documents into a specified data warehouse repository; and the other of querying it, by applying techniques to better answer user's queries. The proposed methodology in our paper on building XML data warehouses covers processes such as data cleaning and integration, summarization, intermediate XML documents, and updating/linking existing documents and creating fact tables.

1 Introduction

In the last few years, building a data warehouse for XML documents has become a very important issue, when considering continual growing of representing different kind of data as XML documents [1]. Many papers have analysed how to design a better data warehouse for XML data, from different points of view (e.g. [1, 2, 5, 6]) and many other papers have focused on querying XML data warehouse or XML documents (e.g. [7, 8]), but almost all of them considered only the design and representations issues of XML data warehouse or how to query them and very few considered optimisation of data quality in their research.

In this paper, we propose a practical methodology for building XML documents data warehouse, by ensuring that the number of occurrences of dirty data, errors, duplications or inconsistencies is minimized as much as possible and a good summarisation exists. The main purpose of this paper is to show systematic steps to build an XML data warehouse and it is important to note that our proposed steps for building an XML data warehouse is generic enough to be applied on different XML data warehouse models.

2 Related Work

Many researchers have studied how to construct a data warehouse, first for relational databases [2, 3, 4] but in the last years, for XML documents [5, 6], considering the spread of use for this kind of documents in a vast range of activities.

Z.R. Yang et al. (Eds.): IDEAL 2004, LNCS 3177, pp. 293–299, 2004.

A concrete methodology on how to construct an XML data warehouse analysing frequent patterns in user historical queries is provided in [6], starting from determining which data sources are more frequently accessed by the users, transforming those queries in *Query Path Transactions* and calculating the *Frequent Query Paths* which stay at the base of building data warehouse schema. In [5], an XML data warehouse is designed from XML schemas, proposing a semi-automated process. Authors choose facts for data warehouse and, for each fact, follow few steps in order to obtain star-schema: building the dependency graph from schema graph, rearranging the dependency graph, defining dimensions and measures and creating logical schema. [3] considers the aspect of data correctness and propose a solution where data-cleaning application is modelled as a directed acyclic flow of transformations, applied to the source data.

Our proposed method focuses on practical aspects of building XML data warehouses through several practical steps, including data cleaning and integration, summarization, intermediate XML documents, and updating/linking existing documents and creating fact tables. We have developed generic methods whereby the proposed method is able to be applied to any collection of XML documents to be stored in an XML data warehouse.

3 Proposed Method on Building XML Data Warehouses

Our paper proposes a systematic approach on building a data warehouse from initial XML documents, developing necessary fact and dimensions and linking them.

Each of the steps involved is described in the next sections of our paper. Generalisation of the proposed methodology is extremely important; therefore for each of these steps we propose general rules or techniques, with examples on how to apply them.

3.1 Data Cleaning and Integration

Cleaning data is a very important step in our process, so it should be analysed very carefully, as it can save a lot of future workload and time during the following steps. In both cases, whether an XML schema exists or not, applying rules provided below will give positive results. The main difference is that without an XML Schema the rules below will have to be applied iteratively throughout the XML document.

Rule1. If a schema exists, we should verify ***correctness of all schema stipulations***:

- verify if a correct use of *name of elements and attributes* in the entire document;
- observe if *data type & natural logic* is respected;
- verify if all elements and attributes are entered in their *schema-specified hierarchy*;
- verify if *order indicators* are respected (“all”, “choice”, “sequence” etc);
- verify if number of occurrences of elements/attributed is respected;
- verify any other schema related specification / restriction;

Rule2. Eliminating duplicate records, for example a name can be entered two or more times, in a different manner (surname&firstname, firstname&surname etc).

Rule3. Eliminate inconsistencies from elements & attributes values, for example existence of two different dates of birth for the same person [3]. As they cannot be two, the right one only should be kept.

Rule4. Eliminating errors in data, determined by entering process, for example mistyping.

Some of the data-cleaning processes have to be done manually, because they require user intervention and occasionally domain expert understanding of the area.

3.2 Data Summarisation

Because not the entire volume of existing data in the underlying database will be absorbed into the XML data warehouse, during *data summarisation* we must extract only useful and valuable information, so we will create another XML document(s) which will be, at the end, part of the data warehouse. Depending of how many levels of summarisation we will have for a specific dimension, we will either (i) create or populate new documents that contain *extraction* from initial data, or (ii) create *special-constructed* values (see Figure 1 as example).

a) we do not possess required data for the dimension, so we will need to create and populate the dimension as a new document – *a "constructed" dimension*;
b) we can find this necessary information by querying directly the primary document and searching for distinct values of a specific element– thus *an "extracted dimension"*.

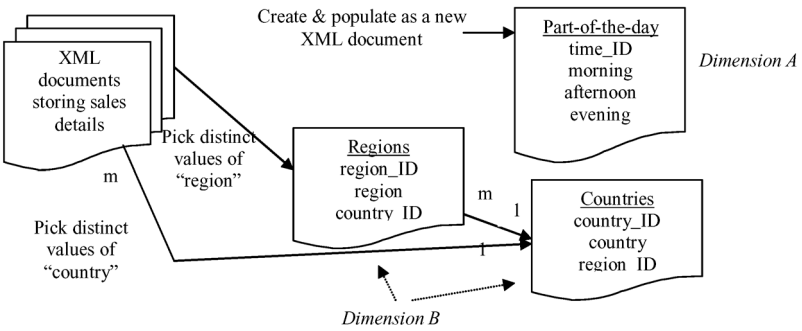

Fig. 1. Dimensions created and populated as new XML documents.

Techniques to Create "Constructed" Dimensions:
If necessary data do not exist in our document, steps are to identify which elements need to be created and which one will be the unique key for linking the dimension with the fact document in the data warehouse. A general way to construct it is:

```
for $a in (1,2,…n)
document {
  <new_element>
       <element_ID>{$a}</element_ID>
       <element_name1>{value}</element_name1>
       <element_name2>{value}</element_name2>
       ……..
  </new_element>
  ……..
}
```

where <new_element> is a tag representing a new node in the new created dimension, <element_namei>,i=1,2..are names of node's children, taking specific *values,* and <element_ID> will be the unique identifier of <new_element>.

Techniques to Create "Extracted" Dimensions:
If necessary data already exist in the document, we will extract distinct values of elements involved, in a newly created document, creating in the same time keys (e.g. <element_ID>).

```
let $a:=0
document {
for $t in distinct-values(doc("doc_name.xml")//element)
let $a:=$a+1
return
  <new_element>
       <elementID>{$a}</elementID>
       <element_name>{$t}</element_name>
       ……..
  </new_element>
}
```

where <new_element> is a tag representing a new created element in the dimension. It contains a key (<elementID>, which takes predetermined values), actual value which is the value of interest (that is <element_name>, e.g. values of "country") and any other elements which can be helpful in the dimension.

There can be situations where the desired data do not exist in the initial document, but they can be extracted from other existing elements, using specific XQuery functions. A general way to construct such a *partial-extracted* dimension is described below:

```
let $a:=0
document {
for $b in distinct-svalues(doc(doc_name.xml")/element)
  let $a:=$a+1
  return
       <new_element>
          <element_ID>{$a}</element_ID)
          <element1>{function1($b)}</element1>
          <element2>{function1($b)}</element2>
          <element3>{function2($b)}</element3>
          ……..
       </new_element>
}
```

where elements in are similar with the above example.

3.3 Creating Intermediate XML Documents

In the process of creating a data warehouse from collection of documents, creating intermediate documents is a common way to *extract valuable & necessary information*.

During this step, we are interested only in main activity data (data involved in queries, calculations etc.), from our initial document. At the same time, we will bring in the intermediate document elements from our initial document, which are keys to dimensions, if already exist. Actual fact document in data warehouse will be this intermediate document, but linked to the dimensions.

```
document {
for $t in (doc("doc_name.xml"))
return
       <temp_fact>
              <elem1_name>{$t//elem1_content}</elem1_name>
              <elem2_name>{$t//elem2_content}</elem2_name>
              ……
       </temp_fact>
}
```

where `<temp_fact>` is a tag representing a new element, containing `<elem1_name>` (name of the element) and `<elem1_content>` (value of element which is valuable for our fact document), etc.

3.4 Updating/Linking Existing Documents and Creating Fact Document

At this step all intermediate XML documents created in the earlier steps should be linked, in such a way that relationships between keys are established (see Fig 2).

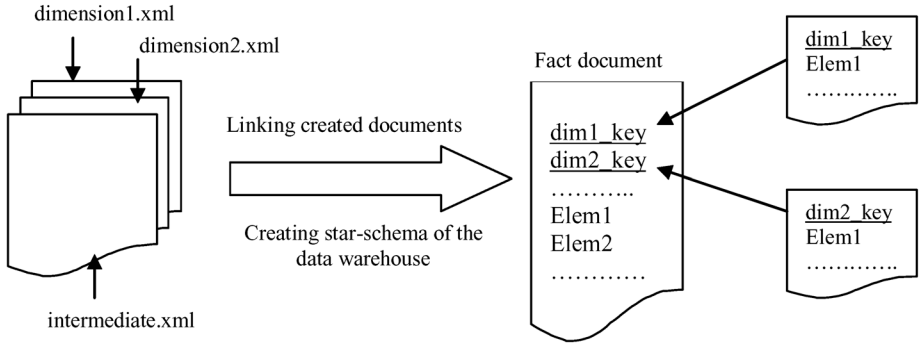

Fig. 2. Linking documents and creating star-schema of data warehouse.

If linking dimensions to intermediate document and obtaining fact are processed all together, the number of iterations through our initial document will be lower, so it subsequently reduces the processing time. A general way to do it can be:

```
let $a:=doc("dimension1.xml")          →e.g. time dimension
let $b:=doc("dimension2.xml")          →e.g. customer dimension
document
{
for $t in (doc("intermediate.xml")/node)
return
  <dim1_key>{for $p in $a where $p//element=$t//element
             return $p//dim1_key}
  </dim1_key>
  <dim2_key>{for $p in $b where $p//element=$t//element
             return $p//dim2_key}
  </dim2_key>
  ------------- ( for all dimensions ) ---------
  <elem1>{$t//elem1_name}</elem1>
  <elem2>{$t//elem2_name}</elem2>
  <elem3>{$t//elem1 * $t//elem2}</elem3>
  --- (for all extracted & calculated elements ) ----
}
```

In the example above, we just obtained the fact, where `<dim1_key>`, `<dim2_key>` etc represent the new created keys elements which will link the fact to dimensions and `<elem1>`, `<elem2>`, `<elem3>` etc are elements of the fact, extracted from intermediate document. A large range of operators can be applied, in order to obtain desired values for analysis (e.g. price * quantity=income).

4 Conclusions

Our paper has presented a systematic approach on how to build a data warehouse for XML documents, in a generic way, so that the rules and techniques can be applied to a wide-range of XML data warehouse models and implementations. After covering all steps involved, we obtain not only an efficient processing of creating a data warehouse, but also high quality data and a low level of redundancy.

Another strong accomplishment is that the steps of work and examples are presented in a very clear and easy manner, so that people who don not have too much knowledge of XQuery can iterate them, with adequate and proper modifications, in order to obtain a data warehouse corresponding to their necessities.

References

1. Widom J., Data Management for XML: Research Directions*, IEEE Data Engineering Bulletin*, 22, 44-52, Sept.1999.
2. Goffarelli, M., Maio, D., Rizzi, S., Conceptual design of data warehouses from E/R schemes Proc. *HICSS-31*, Kona, Hawaii, 1998.
3. Galhardas, H., Florescu, D., Shasha, D., Simon, E., An Extensible framework for data cleaning, Proc. of the *International Conference on Data Engineering*, 2000.

4. Roddick, J.F., Mohania, M.K., Madria, S.K., Methods and Interpretation of Database Summarisation, Database and Expert Systems Application, Florence, Italy, *Lecture Notes in Computer Science*, 1677, 604-615, Springer-Verlag, 1999.
5. Vrdoljak, B., Banek M. and Rizzi S., Designing Web Warehouses from XML Schema, Data Warehousing and Knowledge Discovery, *5th International Conference DaWak 2003*, Prague, Czech Republic, Sept.3-5, 2003.
6. Zhang J., Ling T.W., Bruckner R.M. and Tjoa A.M., Building XML Data Warehouse Based on Frequent Patterns in User Queries, Data Warehousing and Knowledge Discovery, *5th International Conference DaWak 2003*, Prague, Czech Republic, 2003.
7. Fernandez M., Simeon J. and Wadler P., XML Query Languages: Experiences and Exemplars, Draft manuscris, September 1999.
   http://homepages.inf.ed.ac.uk/wadler/topics/xml.html
8. Deutch A., Fernandez M., Florescu D., Levy A. and Suciu D., A Query Language for XML, Computer Networks, 31,1155-1169, Amsterdam, Netherlands, 1999.