Heterogeneous Data Exchanged Over the Internet and Intranets Using XML Formats

Meruva Srinadh, Dr. Syed Umar, M.Raj Gopal Reddy

Abstract: Data warehouse is a database that collects and integrates data from heterogeneous sources in order to support a decision making process. Data exchanged over the Internet and intranets has recently become an important data source, having XML as a standard format for exchange. The possibility of integrating available XML data into data warehouses plays an important role in providing enterprise managers with up-to-date and relevant information about their business domain. We have developed a methodology for data warehouse design from the source XML Schemas and conforming XML documents. As XML data is semi-structured, data warehouse design from XML brings many particular challenges. In this paper the final steps of deriving a conceptual multidimensional scheme are described, followed by the logical design, where a set of tables is created according to the derived conceptual scheme. A prototype tool has been developed to test and verify the proposed methodology.

Keywords: e-Commerce & e-Government, Data Warehousing, Software Engineering, XML

I. INTRODUCTION

Data warehousing systems support the enterprises in the process of extracting useful, concise and handy information for decision-making out of the huge quantity of data stored in their information systems. Since conventional design techniques cannot be successfully applied to build data warehouses, a substantial effort has been made to devise ad hoc methodologies for seamlessly integrating data from heterogeneous sources and putting them into multidimensional form in order to feed them into the warehouse and make them accessible to OLAP (On-Line Analytical Processing) and reporting tools. Recently, as the Internet has evolved into a global platform for information exchange, and e-commerce has emerged as a strongly competing reality, a large number of organizations view the web as an integral part of their communication and business. In this process, the possibility of integrating data extracted from the web into data warehouses (which in this case will be more properly called web warehouses [2]) is playing a key role in providing the enterprise managers with up-to-date and comprehensive information about their business domain. On the other hand, the Extensible Markup Language (XML) has become a standard for the exchange of semi-structured data [1], and large volumes of XML data already exist. Therefore, integrating XML data into web warehouses is a hot topic. XML documents can be associated with or validated against either a Document Type Definition (DTD) [13] or an XML Schema [14]. Although XML data are self-describing, important information about their structure, that is necessary for directly building a warehouse, cannot be obtained without seeing their DTD or XML Schema. XML Schemas considerably extend the capabilities of DTDs, especially from the point of view of data typing and constraining. In particular, the cardinality can be specified in more detail. Furthermore, XML Schemas introduce more powerful and flexible mechanisms for defining keys and their references in the way that is similar to key and foreign key mechanism in relational databases. Because of all its advantages comparing to the DTD, XML Schema is becoming more used than DTD. In this paper we propose a semi-automated methodology for conceptual design of web warehouses from XML sources modeled by XML Schemas. Several conceptual models for data/web warehouses were devised in the literature [3]: in this paper we will adopt the Dimensional Fact Model (DFM) described in [6]. We believe that conceptual design has a key role in determining the quality of the warehouse in terms of documentation, user satisfaction, and reusability; once a conceptual scheme has been obtained, the logical and physical schemes for the warehouse are mainly determined by the target platform for implementation. In general, conceptual design of data/web warehouses entails transforming a schema that describes source operational data into a multidimensional schema for modeling the information that will be analyzed and queried by business users. For instance, [5] discusses how this can be achieved by navigating many-to-one relationships when the source operational data are described by Entity/Relationship schemas. When the sources are modeled by XML Schemas, two main issues arise: firstly, since XML models semi-structured data, not all the information needed for design can be safely derived; secondly, two different ways of representing relationships in XML Schemas are possible, each achieving different expressive power. In our methodology, conceptual design is carried out by first creating a Schema graph, then navigating the functional
dependencies expressed by its arcs in order to derive a correct multidimensional representation. The problem of correctly inferring the needed information is solved by querying the source XML documents and, if necessary, by asking the designer’s help. The approach is implemented in a prototype which reads an XML Schema and produces in output the conceptual scheme for the web warehouse. The paper is structured as follows. After briefly discussing some related approaches in Section II and explaining multidimensional modeling in Section III, in Section IV we show how relationships are modeled in XML Schemas. In Section V we propose our methodology for conceptual design and show how an XML Schema can be converted into a multidimensional schema that conceptually models a web warehouse. Finally, in Section VI the conclusions are drawn.

II. RELATED LITERATURE

The approach described in [10] is focused on populating multidimensional cubes by collecting XML data, but assumes that the multidimensional schema is known in advance (i.e., that conceptual design has been already carried out). In [11], the author shows how to use XML to directly model multidimensional data, without addressing the problem of how to derive the multidimensional schema. In [7] a technique for conceptual design starting from DTDs is outlined. That approach is now partially outdated due to the increasing popularity of XML Schema; besides, some complex modeling situations were not specifically addressed in the paper. In [8], DTDs are used as a source for designing multidimensional schemas (modeled in UML). Though that approach bears some resemblance to ours, the unknown cardinalities of relationships are not verified against actual XML data, but are always assumed to be -to-one. Besides, the id/idref mechanism used in DTDs is less expressive than key/keyref in XML Schema. One alternative approach to design from XML sources consists in first translating them into an equivalent relational schema, then starting from the latter to design the warehouse. Some approaches for translating XML documents into a relational database are proposed in the literature, both leaning on the DTD [9][12] or not [4], but insufficient emphasis is given to the problem of determining the cardinality of relationships, which instead has a primary role in multidimensional design.

III. MULTIDIMENSIONAL MODELING

Data from heterogeneous sources are collected and integrated into the data/web warehouse, which is aimed to support complex data analysis and decision making process. In order to make the data accessible to OLAP and reporting tools and enable efficient analysis of a large amount of data, a multidimensional data model is used in the warehouse. The Dimensional Fact Model [6] is a conceptual model, in which a data/web warehouse is represented by means of a set of fact schemes. A fact scheme is structured as a rooted graph whose root is a fact. The components of fact schemes are facts, measures, dimensions and hierarchies. A fact is a focus of interest for the decision-making process. It typically corresponds to events occurring dynamically in the enterprise world (such as sales or orders, for example). Measures are continuously valued (typically numerical) attributes that describe the fact. Figure 1 presents a fact scheme describing purchase orders as a fact, with unit Price, quantity and income as measures. Dimensions are discrete attributes which determine the minimum granularity adopted to represent facts. The dimensions in the purchase order example are product, customer and date. Hierarchies are made up of discrete dimension attributes linked by -to-one relationship, and determine how facts may be aggregated. In our example, there are hierarchies: customerID → city → country, productID → brand, and date → month. In other words, each hierarchy includes a set of attributes linked by functional dependences; for instance, city functionally determines country and productID determines brand. When building the fact scheme starting from an E/R scheme, the fact scheme is constructed by navigating the functional dependences starting from the chosen fact and by defining dimensions, measures and hierarchies. A fact may be represented either by an entity or by an n-ary relationship.

The fact scheme, as a conceptual schema, can be implemented either in a relational database or in a proprietary structure called multidimensional database. End users of OLAP tools should never be concerned about the storage of data, and should be able to treat the resulting database as a conceptually coherent multidimensional
structure. In the case of multidimensional database storage, data are stored in an array structure similar to the programming language array. On the other hand, when implementing the fact scheme in a relational database, the star schema is typically used. It is composed of one table with a multi-part key, called the fact table, and a set of tables with a single-part key, called dimensional tables. Figure 2 shows the star schema for the purchase order example. Every element of the multipart key in the fact table is a foreign key to a single dimension table. The fact scheme, as a conceptual scheme, can be implemented either in a relational database or in a proprietary structure called multidimensional database. End users of OLAP tools should never be concerned about the storage of data, and should be able to treat the resulting database as a conceptually coherent multidimensional structure. In the case of multidimensional database storage, data are stored in an array structure similar to the programming language array. On the other hand, when implementing the fact scheme in a relational database, the star schema is typically used. It is composed of one table with a multi part key, called the fact table, and a set of tables with a single-part key, called dimensional tables. Figure 2 shows the star schema for the purchase order example. Every element of the multipart key in the fact table is a foreign key to a single dimension table. In this paper we focus on using XML Schema and XML data as a source for designing web warehouses. To be able to navigate the functional dependencies (i.e. to-one relationships) and derive a correct multidimensional representation of the XML data, different ways of expressing relationships in XML Schema should firstly be examined.

Figure 2. Star Schema

IV. RELATIONSHIPS IN XML SCHEMA

An XML Schema consists of type definitions, which can be derived from each other, and element declarations. The possibility of separating an element declaration from the definition of its type enables sharing and reusing of simple and composite types. The structure of XML data can be visualized by a Schema graph derived from a Schema describing the XML data source; the vertices of a Schema graph either correspond to elements/attributes or describe cardinalities of the relationships between them. The graph contains only data that are relevant for conceptual design of a web warehouse. Relationships precisely described in a Schema conform to only four relationship types; attributes and elements are not distinguished. The method has been adopted from [12], where DTD has still been used as a grammar. The basic principles for representing an XML Schema by a Schema graph will be discussed with reference to the purchase order example, taken from the W3C’s document [15]. A portion of an XML document describing a purchase order is presented in Figure 3.

```xml
<?xml version="1.0"?>
<purchaseOrder
  orderDate="1999-10-20">
  <shipTo country="US">
    <name>Alice Smith</name>
    ...
  </shipTo>
  <billTo country="US">
    <name>Robert Smith</name>
    ...
  </billTo>
  <items>
    ...
  </items>
</purchaseOrder>
```

Figure 3. XML data describing a purchase order

The purchase order document consists of a main element, purchase Order, and the sub-elements ship To, bill To, and items. These sub-elements in turn contain other sub-elements. order Date is an attribute of the purchase
Order element. Elements that contain sub-elements or carry attributes have complex types. On the other hand, simple type elements contain numbers, strings, dates, etc. and are neither allowed to have sub-elements nor attributes. Attributes always have simple types. The document conforms to the XML Schema presented in Figure 4. The purchase Order element is defined as a complex type PurchaseOrderType. In defining PurchaseOrderType, two of the element declarations, for ship To and bill To, associate different element names with the same complex type, namely US Address. by sub-elements Relationships in XML Schema can be specified by sub-elements with different cardinalities. An element is required to appear in the document when the value of the manicures attribute in its declaration is 1 or more. The maximum number of times an element may appear is determined by the value of a maxOccurs attribute. The default value for both the minOccurs and the maxOccurs attributes is 1. On the other hand, attributes may appear once or not at all. The occurrence of an attribute can be declared by setting the value of the use attribute in the Schema to required or optional. Since our methodology for conceptual design is based on detecting many-to-one relationships, in the following we will focus on the way those relationships can be expressed in the XML Schema. Two different ways of specifying relationships exist: by sub-elements and by using key and keyref elements. A. Modeling relationships

```xml
<xsd:element name="purchase Order" type="PurchaseOrderType"/>....
<xsd:complexType name="PurchaseOrderType">
  <xsd:sequence>
    <xsd:element name="shipTo" type="USAddress"/>
    <xsd:element name="billTo" type="USAddress"/>
    <xsd:element ref="comment" minOccurs="0"/>
    <xsd:element name="items" type="Items"/>
  </xsd:sequence>
  <xsd:attribute name="orderDate" type="xsd:date"/>
</xsd:complexType>.
```

Figure 4. Purchase order schema

In the Schema graph, we use the operators from the DTD element type declarations because of their simplicity. Concerning the greatest number of times the same sub element may appear within an element, we distinguish between two general types of relationships: -to-one relationship and -to-many relationship. On the other hand, if a sub-element is optional, it might not appear at all. Consequently, four general types of relationships are distinguished:

- **To-one** (the sub-element or attribute appears exactly once within its parent element),
- **Optional** -to-one (marked ?; the sub-element or attribute may appear once or not at all),
- **To-many** (marked +, the sub-element appears once or more)
- **optional** -to-many (marked *; the sub-element may appear zero or more times).

The Schema graph for the Schema describing a purchase order is shown in Figure 5. The default cardinality is exactly one and in that case no operator is shown. Element item is defined in the Schema as a sub-element of the element items with the values of its minOccurs and maxOccurs attributes set to 0 and “unbounded”, respectively. Therefore, there is a “*” operator assigned to the connection between items and item in Figure 5. If the minOccurs attribute of an element is, for instance, set to “2” and maxOccurs to “10”, the useful information we get from these values is that the element must occur and it can occur more than once, so there is a non-optional -to-many relationship that will be represented by a “*” operator in the Schema graph. The comment element is optional within PurchaseOrderType because the value of the minOccurs attribute in its declaration is 0. Therefore, it does not have to appear in the XML document in Figure 3.

Figure 5. Schema graph for a purchase order
To derive a fact scheme and enable multidimensional analysis of data, it is necessary to find -to-one relationships. The presented classification with only four types of relationships preserves the information about those relationships and eliminates unnecessary details. When creating a Schema graph from the Schema, only the operators indicating the relationship in the direction from the parent element to its child element can be marked. The cardinality in the opposite direction cannot be found out by exploring the Schema. Only by exploring the data that conforms to the Schema or by having some knowledge about the domain described by the Schema, it can be concluded about the cardinality in the direction from a child element to its parent element.

Modeling relationships by key and keyref elements
In XML Schema the key and keyref elements are used for defining keys and their references. The key element indicates that every attribute or element value must be unique within a certain scope and not null. If the key is an element, it has to be of a simple type. By using keyref elements, keys can be referenced. The advantage of this mechanism is that not just attribute values, but also element content and their combinations can be declared to be keys. Further, key and keyref elements are specified to hold within the scope of particular elements. Figure 6 presents a part of a Schema graph where the number attribute is defined as a key for the part element and, for each value of the partNum attribute, there must exist a number attribute with the same value. part and partNum attributes must be of the same simple type.

![Schema graph with key and keyref](image)

Figure 5. Schema graph with key and keyref

```xml
<xsd:element name="item" minOccurs="0" maxOccurs="unbounded">
  <xsd:complexType>
    ...
    <xsd:attribute name="partNum" type="SKU" use="required"/>
  </xsd:complexType>
  <xsd:keyref name="part_fKey" refer="partKey">
    <xsd:selector xpath="."/>
    <xsd:field xpath="@partNum"/>
  </xsd:keyref>
</xsd:element>
```

Figure 7. The keyref element
In conclusion, using key and keyref elements not only enables referencing other elements and attributes, but it also provides functional dependencies. The key/keyRef mechanism may be applied to any element and attribute content, as well as their combinations, and the scope of the constraint can be precisely specified.

V. FROM XML SCHEMA TO MULTIDIMENSIONAL SCHEMA
In this section we propose a semi-automatic approach for building the conceptual schema of a web warehouse starting from an XML Schema. The methodology consists of the following steps:

Preprocessing the XML Schema.
Creating and transforming a Schema graph.
Choosing facts.
For each fact:
Building an attribute tree from the Schema graph.
Rearranging the attribute tree.
Defining dimensions and measures.
The attribute tree is an intermediate structure used to move towards a multidimensional representation of data. After the attribute tree has been built from the Schema graph, it can be rearranged, and dimensions and measures are defined. However, this phase of conceptual design necessarily depends on the user requirements.
and cannot be carried out automatically. The goal of this paper is to describe only the steps of conceptual design that can be performed automatically or semi-automatically.

Preprocessing the Schema

The relationships in the Schema can be specified in a complicated and redundant way. Therefore, we transform some structures to simplify the Schema, similarly as DTD was simplified in [7]. There are also many Schema structures that are neither relevant in detecting relationships nor carry any data content, so they have no impact on the later steps of our algorithm and can be excluded from the Schema. The transformations for simplifying a Schema include converting a nested definition into a flat representation. For instance, if there is a choice element in an element declaration, exactly one of the sub-elements declared inside the choice element must appear in a document conforming to that Schema. An example is shown in Figure 8. Using the choice element, it is defined that the price of an ordered item can be expressed either in US dollars or in Euros. From our point of view, the important information here is only that both elements are optional, and they cannot appear more than once. The fact that exactly one of them must occur as sub-element of item has no significance, as it is unknown which one. The resulting simplified structure, although not being equal with the choice expression, preserves all the needed information about the cardinalities of relationships.

```
<element name="item" ...
<choice>
  <element name="priceUSD" type="priceType"/>
  <element name="priceEUR" type="priceType"/>
</choice>
```

**Figure 8. The choice element of the Schema**

Figure 9 shows the same part of the schema after its simplification. The choice element is removed from the schema and a minOccurs attribute is added to each of the two price elements: priceUSD and priceEUR, always with value "0".

```
<element name="item" ...
<choice>
  <element name="priceUSD" type="priceType" minOccurs="0"/>
  <element name="priceEUR" type="priceType" minOccurs="0"/>
</choice>
```

**Figure 9. Schema preprocessing**

As another example of Schema preprocessing, when an element contains many identical sub-elements on the same level, they are all merged into one sub-element with the maxOccurs value “unbounded”.

VI. CONCLUSIONS

In this paper we have presented a semi-automated approach for conceptual design of web warehouses from XML Schemas. After transforming the XML Schema into a Schema graph, this graph is navigated starting from a vertex/arc in order to detect the functional dependencies to be modeled within the conceptual schema for the warehouse. The algorithm proposed also takes into account the existence of attributes shared between two or more hierarchies and the presence of attributes where two or more paths of functional dependencies converge. The algorithm has been implemented within a prototype which thus acts as a valuable support for conceptual design of web warehouses.

REFERENCES