International Journal of Computer Theory and Engineering, Vol. 1, No. 3, August, 2009
1793-8201

From XML Schema to Cube

Parimala N. and Payal Pahwa

Abstract—**Data warehousing technology aims at providing support for decision making for operational data. Defining a data warehouse for data stored in XML (Extensible Markup Language) format should be addressed as various organizations use XML to facilitate the promotion of their businesses. In this paper we describe a semi-automated approach for designing multiple Cubes of multidimensional model from XML schema. In our approach an attribute tree is arrived at by parsing the XML schema. Pruning and grafting are used to remove unnecessary nodes in the attribute tree. Facts, dimensions and measures, which are used to describe the multidimensional cube, are identified. Single or multiple facts can be chosen giving rise to multiple cubes. The choice construct of the XML schema is shown to give rise to multiple Cubes in both the cases.**

Index Terms **- Data warehouse, XML schema, OLAP, Multidimensional Cubes, Attribute tree, Data Warehouse design**

I. INTRODUCTION

Data warehousing [1], [2] technology aims at providing support for decision making by integrating data from various heterogeneous systems in the data warehouse. In contrast to operational systems which support OLTP (on line transactional processing) applications, data warehousing technology aims at providing integrated, consolidated and historical data for OLAP (on line analytical processing) applications [3].The focus of OLAP tools [4], [5] is to provide multidimensional analysis to the underlying information. Towards this purpose, multidimensional models for the storage and presentation of data have been developed. Many efforts have been made to develop a multidimensional model whose main entities are facts and dimensions [6], [7], [8]. Data is organized as cubes that have several dimensions, which together define the multidimensional space. Each dimension comprises of a set of aggregation levels.

XML (Extensible Markup Language) is a meta markup language that provides a format for describing semi structured data. It is a method for putting structured data in a text file. XML documents are self describing so that both human and machine can understand it.

Due to its wide variety of features, today XML is used by various organizations as it facilitates the promotion of their businesses. Some of the key factors why XML has gained importance are:

Manuscript submitted April 21, 2009.

Prof. Parimala N. is Dean (School of computer and system sciences), Jawaharlal Nehru University, Delhi, India.

Prof. Payal Pahwa is professor in Guru Premsukh Memorial College of Engineering, I.P. university, Delhi.

- Different computers in an organization using XML can exchange data easily. This improves the data flow throughout the organization.
- XML data exchange is specifically designed to be used by small and medium sized organizations as it is less expensive than other forms of data exchange. Due to this, it has gained wide acceptance and can be used even for small businesses.
- XML provides a standard common format for multiple sources of information.
- XML is format independent and can generate multiple outputs in an application
- If an organization wants to store information for a long period of time then XML can be used for this.
- XML is flexible and thus used by number of organizations.
- XML is adaptable.

Thus more and more companies are inclined towards the use of XML for storing data for their business activities.

As more and more organizations and enterprises are using XML data for their day to day business activities so it has become necessary to integrate XML data into a data warehouse environment. It can provide the organizations with up-to-date information in their areas of business which in turn can help them in decision making

Attempts have been made to build data warehouse from XML data [9], [1], [11].The emphasis is on identifying the fact, the dimensions and their hierarchies. However, all these approaches do not consider the choice of sub elements that can be specified in a XML schema[12] while arriving at the multidimensional model. For example, the schema in Fig.1 the complexType promoterstructure represents the choice element. It is clear that the data which refers to organization will be totally disjoint from the data concerned with personal contact. The question is whether all the information should be in a single cube or should be separated out into two cubes. In all the approaches so far [6], [8] all the data is in one cube. In this paper we take the second approach. We propose that data concerned with the choice element be separated into multiple cubes.

The second aspect considered in this paper is whether a single XML schema can have more than one fact. We find that it is possible that a single XML schema may have more than one fact. As an example, consider the schema given in Fig. 6. Production and Order can be both facts, which can be analyzed along the product dimension. Therefore, choosing more than one fact is provided for in our system. The dimensions associated with each fact are arrived at. As a result we get multiple multidimensional cube definitions for a single XML schema. It turns out that the cubes may or may not have any common dimension.

The first step in our approach is to convert the XML schema into an attribute tree. While doing so the different grouping constructs-sequence, choice and all of an XML schema are considered. Of these different grouping elements that can be specified, we study the choice element in detail. The choice in an XML schema is clearly defined to mean that one of the sub elements will appear in a XML document corresponding to this schema. We carry this choice specification forward when we convert XML schema to an attribute tree. One or more than one fact is chosen in the attribute tree. Some nodes in the attribute tree may not carry additional information. Grafting and pruning is used to remove these nodes. Using some heuristics, the dimensions and the measures are identified for each fact.

II. XML SCHEMA

An XML schema [13] describes the structure of an XML document. It has a nested structure starting with a root element. The schema defines the elements and attributes that can appear in a document. These can be either simple or complex type. Order indicators define the order of elements in the XML schema. The order indicator may be a choice, a sequence or all. The choice indicator specifies that either one child element or another can occur. The sequence indicator specifies that the child elements must appear in a specific order and the all indicators specifies that the child elements can appear in any order and that each child element must occur only once. The elements in the XML schema can contain text, other elements, a mixture of text and elements or nothing at all. The cardinality of the sub elements can be expressed using minOccurs and maxOccurs attributes. The maximum number of times an element may appear is determined by the value of a maxOccurs attribute in its declaration. This value may be a positive integer or the term unbounded to indicate there is no maximum number of occurrences. Similarly, the minimum number of times an element may appear is determined by the value of minOccurs attribute in its declaration.

```
<?xml version="1.0" encoding="utf-8"?>
  <xsd:schema
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:complexType name="RoadEvent">
<xsd:sequence>
<xsd:element name="promoter" type="promoterstructure"/>
<xsd:element name="location" type="locationstructure"/>
<xsd:element name="time" type="timestructure"/>
<xsd:attribute name="number" type="xsd:positiveInteger"/>
<xsd:attribute name="status" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="promoterstructure">
<xsd:choice>
<xsd:element name="organization" type="orgstructure"/>
<xsd:element name="personal contact"
type="personalstructure"/>
</xsd:choice>
</xsd:complexType>
<xsd:complexType name="orgstructure">
<xsd:sequence>
<xsd:attribute name="name" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>

<xsd:complexType name="personalstructure">
<xsd:sequence>
<xsd:attribute name="name" type="xsd:string"/>
<xsd:attribute name="address" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="locationstructure">
<xsd:sequence>
<xs:complexContent>
<xs:extension base="locationaddressstructure">
<xsd:attribute name="city" type="xsd:string"/>
</xs:extension>
 </xs:complexContent>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="locaddressstructure">
<xsd:sequence>
<xsd:attribute name="longitude" type="xsd:string"/>
<xsd:attribute name="lattitude" type="xsd:string"/>
/xsd:sequence>
</xsd:complexType
<xsd:complexType name="timestructure">
<xsd:sequence>
<xsd:element name="day" type="monthstructure"/>
</xsd:sequence>
</xsd:complexType
<xsd:complexType name="monthstructure">
<xsd:sequence>
<xsd:attribute name="month" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType
```

Fig. 1 XML Schema

III. THE DATA MODEL DM

In [14] we have defined a multidimensional data model DM whose main components are dimensions and a fact scheme. We include the definition here so that the mapping can be defined.

A Cube has the following components

- N dimensions
- The fact scheme

A Dimension is composed of

- a set of dimensional attributes V. Each attribute has a set of instances associated with it.
- a set of non dimensional attributes N.
- a connected, directed graph D(V, E). Every vertex in the graph corresponds to an aggregation Level, and an edge (ai, aj) reflects that ai can be rolled up to aj. An instance of aj decomposes into a collection of instances of ai. Each Level corresponds to granularity in the Dimension.

A Fact scheme is an expression of the form f [D1: A1, D2: A2 … Dn: An] → [M1, M2 ….Mk] where Ai is a

IACSIT International Association of Computer Science and Information Technology WWW.IACSIT.ORG

dimensional attribute of dimension Di. M1 ….Mk are distinct measures.

IV. Rev Defining cubes

In this section we describe a semi automatic approach for building multiple Cubes from XML sources. Starting with the XML schema the following steps are performed:

1. Map XML schema to an attribute tree
2. Choose fact/facts
3. Define dimensions and measures for each fact.

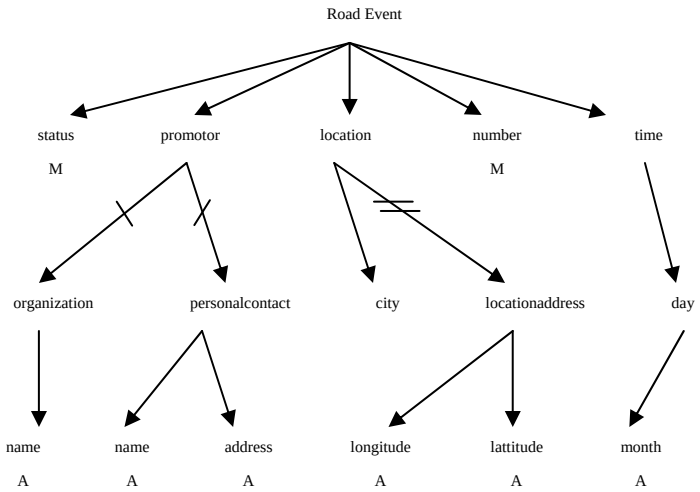

Fig. 2 Attribute tree of XML Schema of Fig. 1

V. Building an attribute tree

As a first step the XML schema is converted to an attribute tree. Essentially a node in the tree represents a construct of the XML schema. The edges are drawn so that the attribute tree represents the hierarchical relationships between the different constructs of the XML schema. Thus, the tree is a directed tree. The construction process is explained below. While doing so the different constructs of XML schema are considered. XML schema can have simple and complex types, which have grouping constructs - ***sequence***, ***choice*** and ***all***. In addition a simple type or a complex type can be extended or restricted. In the case of restriction, only the properties can be restricted. On the other hand, a type can be extended by adding attributes and elements. We consider each of these.

The algorithm to construct an attribute tree from the XML schema is as follows:

Create a node to represent the root element of the XML schema.

a For an attribute a node is created corresponding to it and marked as A. An edge is drawn from the node of which it is a sub part and this newly created node.

b An element with no sub elements in the XML schema belongs to the category of simple type and a node is created corresponding to it. An edge is drawn from the node of which it is a sub part and this newly created node.

c An element in the XML schema that is defined as a complex type is mapped to a new node and an edge is drawn from the parent node to this node. Further, the grouping constructs are mapped as follows:

i) A node is created for each sub element that is a sequence and connected to the node corresponding to the complex type.

ii) A node is created for each alternative element of the choice construct of an XML schema and connected to the node of the complex type. A choice construct is graphically represented by drawing a parallel line across the edge, as shown in fig. 2.

iii) If the grouping element is all then it is treated as sequence as discussed above.

d In a XML schema the cardinality is 1:N, in the direction from the root to the element. The minOccurs and maxOccurs options specify the cardinality N of the element concerned. In the absence of any specification, the default cardinality for element is minOccurs=0 and maxOccurs=1 and for attribute it is minOccurs=1 and maxOccurs=1. These are used to unambiguously specify the cardinalities in the direction of the edge in the attribute tree. When the cardinality is 1:N then the user has to decide whether to include the element or not.

e A simple or a complex type can be used as a base and a new type can be specified as an extension of the base type. Since the base type is already defined, a node corresponding to it already exists. A new node is created for the extended type with an edge between them. This edge is marked with a pair of parallel lines across the edge as shown in Fig. 2.

Applying the above steps, the XML schema of Fig. 1 is converted to an attribute tree of Fig. 2.

A. Pruning and Grafting

The attribute tree can be pruned and unnecessary nodes can be removed. If a node that is a leaf is pruned then only that node is deleted. If a node other than a leaf is pruned then the sub tree rooted at this node is deleted. For example in Fig. 2 if name is pruned then only it is deleted from the tree and if personalcontact is pruned then name and address are automatically deleted. However this needs input from the user.

Grafting one node to another is clubbing two nodes and removing the edge between them. More formally, let n1 and n2 are two nodes and e(n1, n2) the edge between them. Grafting n2 to n1 implies

a) delete e
b) connect all edges connected to n2 to n1
c) delete n2.

There are multiple situations when grafting can be useful. The situations and the action taken is explained below.

1) When the cardinality between two nodes is 1:1: In this case grafting one onto the other is permitted. The cardinality 1:1 can mean that one of the nodes is superfluous. However, the grafting is performed only after confirmation from the user as most of the constructs in the XML schema display 1:1 relationship.

2) When choice construct is represented: The grafting is done for every alternate element of the choice construct i.e. for every edge that has a parallel line. However, it is deferred and performed later as explained in rule 2 of section VI.

3) When extension of types is represented: In this case the base type becomes superfluous if all the elements of the base type are available with the extended type. Both the nodes are not necessary. The extended type is retained with all its

elements and the elements of the base type. That is, if n and n1 are two nodes such that the edge e1 (n, n1) has a double parallel line drawn across it as shown in fig. 2, then the edges other than e1 which are connected to n in the direction of the edge to n1 are attached to n1. Node n1 is grafted to n.
At the end of pruning and grafting we have an attribute tree with unnecessary nodes removed. From the attribute tree of Fig. 2 we get the attribute tree shown in Fig. 3.

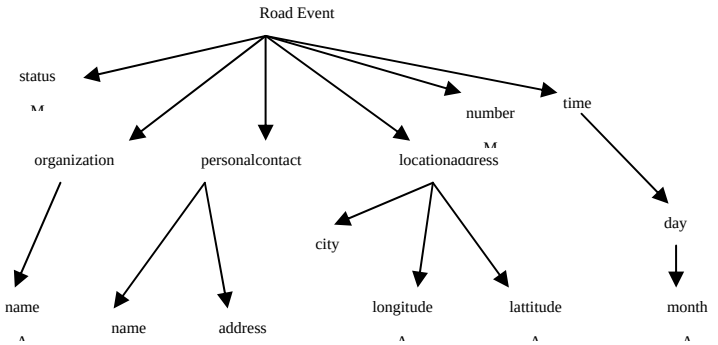

Fig. 3 Attribute tree after pruning and grafting

VI. CHOOSING FACT

After the XML schema is converted to an attribute tree, fact F is chosen in the attribute tree. The fact is chosen by the user. The fact can be a single node or more than one fact can be identified by the user. Once the fact is chosen by the user, dimensions and measures are identified

A. Choosing A Single Fact

Firstly let us consider that a user chooses a single fact. Let F be the fact chosen by the user. This corresponds to the f of the fact scheme. Then the nodes, which are connected to F, can be either measures or dimensions. We define the following heuristics to choose measures and dimensions in the attribute tree.

1. If a node connected to F is a leaf then it is a measure Mi (marked as M in Fig.2). If a node connected to F is not a leaf node then it defines a dimensional hierarchy. The tree with n as root forms the collection of dimensional and non dimensional attributes. Within the tree (Fig.2) all nodes marked as 'A' are non-dimensional attributes. Remaining nodes are dimensional attributes. The directed tree with root n is treated as the graph of the dimension. For example; in Fig.2 status and number are measures. The node Time defines a dimensional hierarchy which has time and day as dimensional attributes and month as non dimensional attribute.
2. Let multiple edges connected to a node n have a parallel line across them (which will be the case for a choice construct). These nodes are mapped to dimensions. If they are to be translated as dimensions, we observe that the fact instances will relate to one of the dimensions at a time and never all of them. Therefore, it is preferable to have these nodes as dimensions in different cubes. In such a situation, we create multiple Cubes. Each Cube will have only one of the nodes connected to n. Only one choice element construct appears as a dimension in one cube. Effectively, there are as many cubes as edges with a single parallel line. For each edge e1(n, n1), graft n1 to n and create a Cube with all other dimensions and measures as derived in 1.

The above heuristics give an initial definition of the Cube. Let us consider Fig. 2. As clearly seen in the figure there is a choice construct in the edges connected to promoter marked as a parallel line across them. The choice is between organization and personalcontact. We create multiple cubes as explained above. Certain adjustments are made. For example, since day can be rolled up to month, month is moved from a non-dimension attribute to a dimensional attribute. Two Cubes for the XML schema of Fig.1 are defined below:

Cube OrgRoadEvent

Dimension Org with V = {organization} and
N= {Name}
Dimension Place V = {locationaddress, city } and
N = {longitude, latitude } Graph = (V, E) where
E = {(locationaddress, city) }
Dmension TimeOfEvent with V = { time, day, month } and N = { } Graph = (V, E) where E = {(time, day), (day, month) }
Fact Scheme
OrgRE [Org: organization, Place: locationaddress, TimeOfEvent: time] → [Number, status]
Cube PerRoadEvent
Dimension Promotor with V= {personalcontact } and
N = {name, address }
Dimension Place V = { locationaddress, city } and
N = {longitude, latitude } Graph = (V, E) where
E = {(locationaddress, city) }
Dimension TimeOfEvent with V = { time, day, month } and N = { } Graph = (V, E) where E = {(time, day), (day, month) }

Fact scheme
PerRE [Place: locationaddress, TimeOfEvent: time, Promotor: personalcontact] → [Number, status]

B. *Choosing Multiple Facts*

Now let us consider the case where the user identifies more than one fact. Let F and F' be the two facts identified by the user. The following cases arise regarding the relationship between those elements in the XML schema, which correspond to the nodes identified as facts in the attribute tree
a) the nodes are part of choice construct
b) there are intervening nodes between the two fact nodes. That is, the nodes are connected indirectly through other nodes.
c) there is no intervening node between the two fact nodes. That is, there is an edge between them.
We will discuss these cases individually

Case (a) Nodes are part of choice construct

Let us consider the case when the nodes are part of choice construct. If the nodes are part of a choice construct then they are elements of the same parent element P. Let F and F' be two nodes as identified by the user. If there is a choice construct among these nodes then they are considered to

IACSIT International Association of Computer Science and Information Technology WWW.IACSIT.ORG

exhibit ‘is-a’ relationship with P. At any given point in time, only one of the fact i.e. either F or F’ is considered and the other is ignored. If the node F is considered then the parent P is grafted to F and a cube is arrived at. Similarly if F’ is considered then the parent P is grafted to F’ and a cube is obtained. The parent P is grafted because its choice construct is considered and thus the parent node now becomes superfluous and can be grafted. As an example, consider the following XML schema and its corresponding attribute tree.

```
< xs: complexType name =”Item” >
<xs: choice>
<xs: element name = “Library book” type =  “booktype”/>
<xs: element name = “Moviecd” type = “movietype”/>
</xs: choice>
<xs: attribute name = “Title” type = “xs: string”/>
</xs: complexType>
<xs: complexType name = “booktype”>
<xs: sequence>
<xs: attribute name = “Pages” type = “xs: string”/>
<xs:element name = “Author” type = “ authortype”/>
<\xs: sequence>
<\xs: complexType>
<xs: complexType name = “authortype”>
<xs: sequence>
<xs: attribute name = “Name” type = “xs: string”/>
<xs: attribute name = “add” type = “xs: string”/>
<\xs: sequence>
<\xs: complex type>
<xs: complex Type name = “Movie type”>
<xs: element name = “Actor” type = “actor type”>
<\xs: complex type>
<xs: complex Type name = “Actor type”>
<xs: sequence>
<xs: attribute name = “ First name” type = “xs: string”/>
<xs: attribute name = “Last name” type = “xs: string”/>
<xs: sequence>
<xs: complex type>
```

Fig. 4 XML Schema

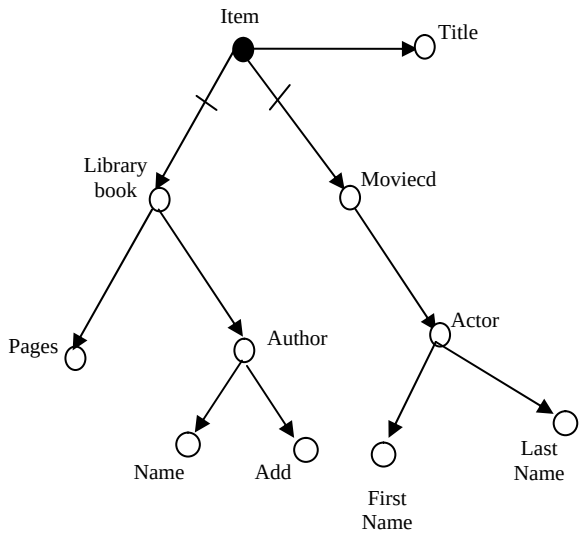

Fig. 5 Attribute tree of XML schema of fig. 4

There is a choice construct in the XML schema and is represented by parallel lines in the above attribute tree. The item can either be a Librarybook or Moviecd. Thus there are two facts Librarybook and Moviecd. First the subtree under Library book is ignored. Item is grafted to Movie cd. Next the subtree under Moviecd is ignored and Item is grafted to Library book. After making adjustments regarding dimension and measures, the following cubes are arrived at:

Cube Lbook

Dimension Ath with V = {Author} and N = {name, address}
Fact scheme
Lib book [Ath: Author] → [Pages, Title]
Cube Mcd
Dimension Act with V = {Actor} and N = {first name, last name}
Fact scheme
Movie cd [Act: Actor] →[Title]

Case(b) There are intervening nodes between the two fact nodes.

When there are intervening nodes between the two fact nodes then related cubes are created. If F and F’ are the two facts identified by the user then F is mapped to the fact scheme of one cube and F’ to the other. In order to create the dimensions of the two cubes we consider the intervening nodes. There may be one or more nodes, which connect these facts. If there is a single node connecting the two facts then it is usually treated as a common dimension and this dimension is defined in both the cubes. If there is more than one node connecting the two fact nodes then these intervening nodes are treated as dimensions in either of the cubes. However this requires an input from the user. The following example considers the case when there is a single intervening node between the two fact nodes.

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Production" type="ProductionType" />
  <xs:complexType name="ProductionType">
   <xs:sequence>
<xs:element name=”Product” type=”ProductType”
 minOccurs="1" maxOccurs="unbounded" />
   </xs:sequence>
<xs:attribute name=”Time” type=”xs:time” />
<xs:attribute name=”Factory” type=”xs:string” />
 </xs:complexType>
 <xs:complexType name=”ProductType”>
   <xs:sequence>
  <xs:element name=”Order” type=”OrderType”
minOccurs=”1” maxOccurs=”unbounded” />
</xs:sequen</xs:complexType>
   <xs:complexType name=”OrderType”>
<xs:attribute name=”Time” type=”xs:time” />
<xs:attribute name=”Retailer” type=”xs:string” />
<xs:attribute name=”Client” type=”xs:string” />
   </xs:complexType>
```

</xs:schema>

Fig. 6 XML schema

The attribute tree for the above schema using the method explained is shown below:

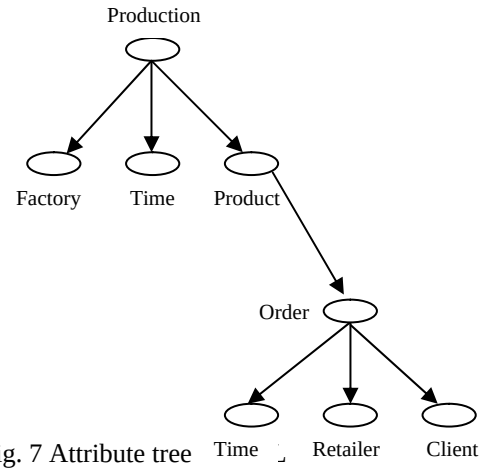

Fig. 7 Attribute tree

In this case if user identifies Production and Order as two facts then Product is an intervening node between the two. There is no relationship between them. So as explained earlier, two cubes with Product as the common dimension are formed. The first cube has Production as the fact scheme and Product as a dimension. As per the heuristics given above both Factory and Time become measures. But Time is deleted as a measure and introduced as a dimension. In the second cube, Order is the fact scheme and Product is a dimension. As before adjustments are made to measures and dimensions and Time is again treated as a dimension. The two cubes are shown below. Note that Time will become a common dimension if the semantics of Time in the first schema is the same as the semantics in the second cube. Else they will stay as separate dimensions.

Cube Production
Dimension Pro with V={Product} and N={Order}
Dimension Timeofproduction with V = {Time} and
N = { }
Production[Pro: product, Timeofproduct: time]
→ [Factory]
Cube Order
Dimension Pro1 with V = {Product} and N = { }
Dimension Timeoforder with V = {time} and N={ }
Order[Pro1: product, Timeoforder: time] → [Retailer, Client]

Case (c) there is no intervening node between the two fact nodes

If F and F' are the two facts identified by the user and there is no intervening node between the two fact nodes then their connecting edge is removed and two separate cubes with F and F' as fact scheme are created. The dimensions and the measures are identified for each cube ce>
separately as given in heuristics defined above. The cubes created are unrelated to each other i.e. they do not share a common dimension. For example consider the following XML schema and its corresponding attribute tree:

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="Order" type="OrderType" />
<xs:complexType name="OrderType">
 <xs:sequence>
<xs:element name=”Product” type=”ProductType”
 minOccurs="1" maxOccurs="unbounded" />
 </xs:sequence>
 <xs:attribute name=”Client” type=”xs:string” />
 <xs:attribute name=”Address” type=”address type” />
</xs:complexType>
<xs:complexType name=”ProductType”>
 <xs:sequence>
<xs:element name=”Brand” type=”BrandType” />
 <xs:element name=:Warehouse” type=”WarehouseType”
 minOccurs=”1” maxOccurs=”unbounded” />
 </xs:sequence>
 <xs:attribute name=”ProductName” type=”xs:string” />
 <xs:attribute name=”ProductQty”
type=”xs:positiveInteger” />
</xs:complexType>
<xs:complexType name=”BrandType” >
<xs:attribute name=”BrandName” type=”xs:string” />
<xs:attribute name=”Category” type=”xs:string” />
</xs:complexType>
<xs: complexType name=”address type”>
<xs: attribute name=”city” type=”xs: string”/>
<xs:attribute name=”Longitude” type=”xs:string”/>
</xs:complexType>
<xs:complexType name=”WarehouseType” >
 <xs:attribute name=”WarehouseAddress”type=”xs:string”
/>
</xs:complexType>
</xs:schema>
```

Fig. 8 XML schema

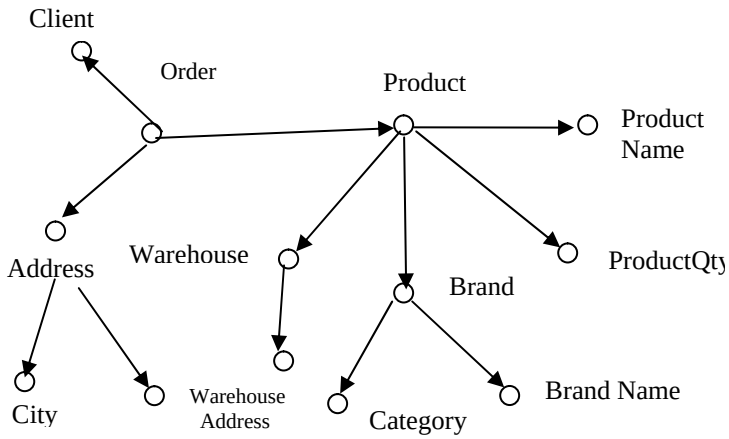

Fig. 9 XML schema of Fig. 8

If Order and Product are the two facts identified by the user, then, the connecting edge between them is removed and the following cubes are arrived at. These cubes share no common dimension.

Cube Or
Dimension add with
V = {address, city} and N = {Longitude}
Graph = (V, E) where E= { address, city}
Fact scheme
Order [add: address} ⟶ [Client]
Cube Pro
Dimension brd with

IACSIT International Association of Computer Science and Information Technology WWW.IACSIT.ORG

V = {brand} and N = {category, name}
Graph = (V, E) where E= {brand, category}
Dimension wh with
V={warehouse} and N={address}
Fact scheme
Product [brd: brand, wh:warehouse] [name, quantity]
The first cube will have Order as fact scheme with Client as measure and Address as its dimension. The second will have Product as fact scheme with Name and Quality as measures. It will also have Brand and Warehouse as dimensions.

VII. CONCLUSION

In this paper we have proposed a semi-automated approach for arriving at multiple cubes of the data model DM of a data warehouse from XML schema. The approach adapted by us converts the XML schema to an attribute tree. We have also identified grouping constructs, attributes, 1:1 cardinality and extension to types in the XML schema and carried it over to the attribute tree. Pruning to remove unnecessary nodes in the attribute tree is undertaken. Grafting removes nodes, which are superfluous. It has been shown that whenever there is a choice construct or multiple facts, then more than one cube of the data model DM is arrived at.

It can be argued that whenever there is only one fact, then there should be only one cube. However, we split the XML schema into multiple cubes when there is a choice construct because if the choice construct is not taken into account then while arriving at cubes, a fact instance will have null values for the non-relevant dimension. Firstly, this will not be a quite a correct relationship between the fact instance and the dimension instances. Secondly, the processing time while querying will significantly increase. Due to creation of multiple cubes both the difficulties have been taken care of.

In an earlier proposal [15] we converted an XML schema to Canonical Conceptual Model [16] which was further, converted to an attribute tree. In the current proposal we have done away with the CCM. The proposal did not consider extension to types and the cardinalities in depth. The emphasis there was using CCM as an intermediary model.

We have implemented the approach using Java TM 2 Platform Standard Edition 5.0 Development Kit (JDK 5.0) as the front end and Oracle 10g as the back end. The Java Database Connectivity (JDBC) has been used for providing connectivity between Java programming

language and Oracle 10g.The implementation takes an XML schema as input and produces the cube definition as output.

DOM has been used to read XML data. Transfer of data from XML document to tables in Oracle is done using table based mapping described at http://www.xml.com. This type of mapping views an XML document as a serialized table.

During implementation the following points were considered:

1. As XML data types are not identical with the data types of Oracle, a table was created which contained the association of types in XML with a type in Oracle. For example the XML data type integer was translated to number in Oracle and similarly XML data type string was translated to varchar in Oracle.

2.An XML schema is mapped to multiple cubes giving rise to tables with the same name appearing in multiple cubes. If all the tables corresponding to a single XML schema are stored in a single database, then there is a name conflict for the tables representing common dimensions. A table each is created for the common dimension but every time one table is renamed. This information is stored in the data dictionary.

3. The attributes of the tables were defined to be in the same order as in XML schema so that it is easy to load XML data into the tables corresponding to the cubes.

We have tested our approach on the XML schemas taken from the following websites:

http://www.w3.org/XML/Schema
http://wwwxml.com/pub
/a/2000/11/29/schemas/part1.html
http://www.brics.dk/~amoeller/XML/schemas/xmlschema
-recipes.html

We show the results of our approach on the schema taken from http://www.w3.org/XML/Schema. For the sake of compactness we have given the reference of the XML schema and have not included the XML schema.

The attribute tree obtained from this XML schema is as follows:

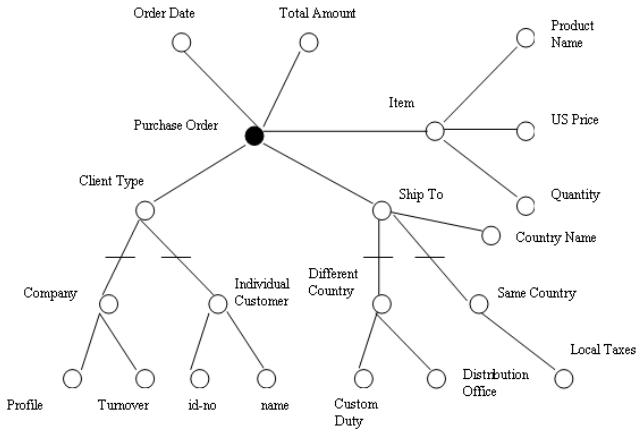

Fig. 10 Attribute tree

The above attribute tree is pruned and grafted as described above. The attribute tree obtained after pruning and grafting is shown below:

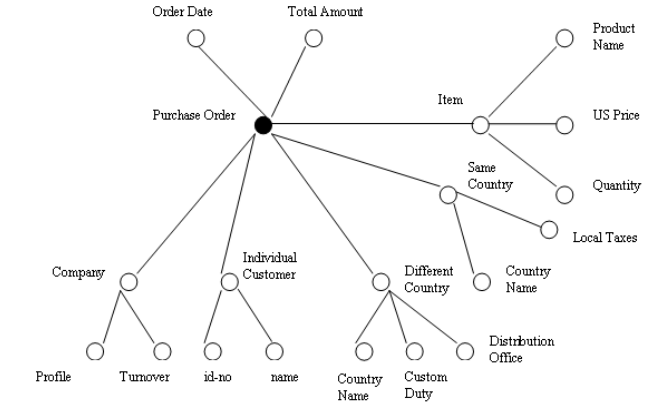

Fig. 11 Attribute tree after pruning and grafting

It can be clearly seen in the Fig.10 above that there is a choice construct in the edges connected to Client type marked as a parallel line across them. The choice is between Company and Individual customer. And there is also a choice construct between Different Country and Same Country. We

create multiple cubes as explained earlier. Cubes for the above XML schema are defined below:

Cube POCompany1
Dimension Comp with V = {Company} and N= {Profile, Turnover}
Dimension DC with V = {Different Country} and
N = {Country Name, Custom Duty, Distribution Office}
Dimension Itm with V = {Item} and
N = {Product Name, US Price, Quantity}
Fact Scheme
POC1 [Comp: Company, DC: Different Country, Itm:Item] → [Order Date, Total Amount]
Cube POCompany2
Dimension Comp with V = {Company} and N= {Profile, Turnover}
Dimension SC with V = {Same Country} and
N = {Country Name, Local Taxes}
Dimension Itm with V = {Item} and
N = {Product Name, US Price, Quantity}
POC2 [Comp: Company, DC: Different Country, Itm:Item] →[Order Date, Total Amount]
Cube POIndCustomer1
Dimension ID with V= {Individual Customer } and
N = {id_no, Name}
Dimension DC with V = {Different Country} and
N = {Country Name, Custom Duty, Distribution Office}
Dimension Itm with V = {Item} and
N = {Product Name, US Price, Quantity}
Fact scheme
POIC1 [ID: Individual Customer, DC: Different Country, Itm:Item] → [Order Date, Total Amount]
Cube POIndCustomer2
Dimension ID with V= {Individual Customer } and
N = {id_no, Name}
Dimension SC with V = {Same Country} and
N = {Country Name, Local Taxes}
Dimension Itm with V = {Item} and
N = {Product Name, US Price, Quantity}
POIC2 [ID: Individual Customer, SC: Same Country, Itm:Item] → [Order Date, Total Amount]

REFERENCES

[1] W. H. Inmon, "Building the Data warehouse", 1996, Second Edition John Wiley & Sons.
[2] R. Kimball, "The Data warehouse toolkit", 1996, John Wiley & Sons.
[3] S. Chaudhari and U. Dayal, "An overview of data warehousing and OLAP technology", SIGMOD Record, 26 (1), pp. 65-74, 1997.
[4] C. Li and X.S. Wang, "A data model for supporting on–line analytical processing", Proceedings of the 5th International Conference on Information and Knowledge Management (C1K M'96), pp. 81 – 88, 1996.
[5] OLAP Council. The APB – 1 Benchmark (1997) Available at http://www.olapcouncil.org/research/bmarkly.htm
[6] R. Agrawal, A. Gupta and S. Sarawagi, "Modeling multidimensional databases", Proceedings of 13th International Conference on Data Engineering (ICDE'97), IEEE computer society, pp.232-243, 1997.
[7] L. Cabibbo and R. Torlone, "Querying multidimensional databases", Proceedings of the 6th International Workshop on Database Programming Languages (DBPL6), Springer, pp. 319-335, 1997.
[8] P. Vassiliadis, "Modeling multidimensional databases, cubes and cube operations", Proceedings of the 10th International Conference on Scientific and Statistical DatabaseManagement (SSDBM), IEEE computer society, pages 53-62, 1998.
[9] M. Golfarelli, S. Rizzi and B. Vrdoljak, "Data warehouse design from XML sources", Proceedings DOLAP'01, Atlanta, pp. 40-47, 2001.
[10] J. Pokorny, "XML Data Warehouse: Modeling and Querying", In: Databases and Information Systems II, Selected Papers from the Fifth International Baltic Conference, Baltic DB & IS 2002 Kluwer Academic Publishers.
[11] B. Vrdoljak, , M. Banek and S. Rizzi, "Designing web warehouse from XML schemas", LNCS vol 2737/2003, pp 89-98.
[12] W3C XML schema. Available by WWW in: http://www.w3.org/XML/schema
[13] W3C. Extensible Markup Language (XML) 1.0 http://www.w3c.org/TR/REC-XML
[14] N. Parimala and P.Pahwa, "Coalescing Data Marts", Proceedings of XVI international conference on computer and information science and engineering (CISE). pp.280-285, 2006.
[15] P. Pahwa and N. Parimala, " Conceptual Design of Data warehouses from XML Schemata", 2nd International Conference on Intellectual Capital, Knowledge Management and Organizational Learning, ICICKM., Dubai. pp. 387-394, 2005.
[16] R.D.S. Mello and C.A. Heuser, "A bottom-up approach for integration of XML Sources", International Workshop on Information Integration on the Web. (WIIW'2001), UNIRIO, Riode Janeiro, Brazil, pp. 118-124, 2001.

Prof. Parimala N. is presently Dean (School of Computer and System Sciences) Jawaharlal Nehru University, Delhi. Her areas of interest include Data Warehousing and Software Engineering. She has various research publications in different international conferences and journals.
Prof. Payal Pahwa is professor in Guru Premsukh Memorial College of Engineering, I.P. university, Delhi. Database management systems and Data Warehousing are her areas of interest.

IACSIT International Association of Computer Science and Information Technology WWW.IACSIT.ORG