SSRG International Journal of Computer Science and Engineering
ISSN: 2348 – 8387 /doi:10.14445/23488387/IJCSE-V8I9P102

Volume 8 Issue 9, 5-11, Sep 2021
©2021 Seventh Sense Research Group®

Original Article

A Novel Approach to Version XML Data Warehouse

Mitesh Athwani

Senior Director, Exusia Inc. Chicago, United States of America

Received Date: 01 August 2021
Revised Date: 02 September 2021
Accepted Date: 13 September 2021

Abstract *- Data warehouses are fast becoming a norm for a standard company. XML has emerged as a standard for communication and storage of data. A distributed data warehouse having data format as XML i.e., XML warehouses have gained much popularity. Data warehouses have emerged as an important aid in decision-making and in what-if-analysis. To achieve this purpose, the presence of historical data is essential. Versioning plays an important role particularly in maintaining historical data. Research and implementation of document versioning have been carried out on many fronts, but the issue of schema versioning is still in its early stage. In this paper, we have concentrated on both content versioning as well as schema versioning. Earlier work on schema versioning has largely taken the view of versioning on attributes. We have versioned the schema on a new approach taking the dimension level.*

To maneuver the challenging task of schema versioning we have also taken an example case study that has been used to describe the scenario in a better manner. Also, to portray our work, we have made a prototype using the .NET framework that depicts both schema versioning as well as content versioning. The approach is user-centric and therefore it allows the user to generate versions of schema according to their need specified in an XML document called" Version Specification Document" (appended to this paper). As an additional characteristic, it also allows validation of document files against the schema files.

Keywords - *Data Warehouse, XML, Versioning, Dimensions, Facts*

I. INTRODUCTION

A. Nature of the Problem

Data warehouses serve as an important tool for decision-making, what-if-analysis, future prediction, time series analysis and various other applications. It has also been highly used for Online Analytical Processing (OLAP). These applications require the maintenance of historical data along with the current data. Trend analysis has come up as one of the major applications. Mining tools extract interesting patterns from the data that aid in better decision making. Organizations cannot survive without future planning as they are highly dependent on the predictions based on the analysis and outcome of data warehouse.

In a data warehouse, data is collected from many sources for basis of decision making. It is the data that acts as source for analysis which is then used for studying the patterns and predicting the future trends. The accuracy of prediction is solely dependent on data being used. More accurate and updated is the data better is the analysis for prediction. Data warehouse cannot therefore allow data to be lost in any scenario. Thus, Versioning in warehouses tends to be an important issue to be investigated.

In our work we are dealing with an XML warehouse, where the data sources are XML documents. These XML sources are the backbone of data warehouse. But the problem arises when a source may evolve with time and need, which may lead to changes in structural definitions, hence causing mismatching of schemata being followed by the sources and the data warehouse. To this situation if there is need of change in data definitions. For example, it can be the change in the details, with few more properties being added, this would require a change to be made at the schema level as well. To make changes at the schema level, it would be a tedious and inefficient task to read the changes requested from the documents and then make the schema changes.

The possibility of changes in a data warehouse cannot be neglected or changes cannot be disallowed because the changes in the data are sensitive and can cause the entire trend pattern to vary. Thus, change negligence could lead to incorrect pattern trends being predicted. This would call for versioning in terms of content and schema. Thus, the platform is set up for versioning to be the solution of the times.

B. Previous Work

Compared to schema versioning most of the current work is focussed on providing content versioning for the data warehouse. Very less work has been done in the domain of schema versioning.

Few approaches have been carried out in the past in the case of schema versioning. One of them is to ignore the changes appearing in the data source. The problem with this approach is that sooner or later there would be considerable difference in schemata rules of the data source and the data warehouse, which could make the warehouse obsolete. There would be mismatching of current data with the old data, if the changes are not propagated into the warehouse. Such a situation would be a disaster as the basic purpose of data warehouse would be defeated i.e., aid in decision-making. Another approach could be to correctly reflect the changes occurring in the sources into the warehouse. This leads to correctly updating the content as well as schema of the data warehouse as per the data sources. The concept of versioning arises when one must deal with these mentioned issues [1, 2]. Another possibility is that the XML document is first changed into equivalent relational schema and then with the application of standard data warehouse methodology, one can incorporate it into a data warehouse [3].

Two approaches are particularly applicable to schema versioning situation. One of them is called Schema evolution where schema is updated and data from old schema to new schema is transferred. Only the latest version is kept. Another approach is known as Schema versioning in which history of all the versions is maintained [4]. The versions are kept as different physical copies, or they are maintained logically. Another approach is to focus on meta-schema-based mapping. In this way, the mapping between the source and the internal schema does not need to be hard wired thereby providing flexibility. User is given the option of selecting the schema if the need arises and the mapping process is not hidden from them [5].

Some other approaches provide easy versioning schemes. New versions could be added by giving numbers to them in increasing order. It is also possible to add an attribute in the schema which designates the schema. One can also use the basic property of "target Namespace" for a different context of schema versioning. New versions can be physically kept in other locations. This would lead to an increment in the copies of schemas [6]. Creating intermediate versions of data warehouse and producing the complete version of the entire data warehouse gradually is another possible approach. This simplifies the mapping process between the old and the new versions of data warehouse since the updates are performed gradually [7].

C. Contribution of our Paper

The purpose of our paper is to propose a mechanism that would help in making the xml warehouse up to date with the evolution of data source. All the changes appearing in the data sources would be invited (if they are in accordance with the need of the data warehouse) with the creation of new version of schema. Also, the possibility of content versioning would help in better storage of evolving data. We have proposed a mechanism which would make it possible to automatically bring changes to the schema according to the changes in the documents. This would ease the way the changes have to be done. Apart from this, it would make the formatting rules of the documents from the sources (of distributed data warehouse) less stringent as more than one version of schema would be used for validation of the documents.

We have defined the schema mapping at the dimensional level of a data warehouse. A detail study of the possible changes in the data warehouse is required which could affect the schema definitions to change. This study is then used to formulate a "version specification document" i.e., change specification document, which consists of all the possible changes that would require a different schema version to be generated. These details are then used to generate a new version of the schema; a schema would be a representation of a dimension of data warehouse.

In our case the new version of schema that is generated would not invalidate the documents that follow the definition of previous version of schemas. This same approach has been used for versioning the content in a data warehouse. With the addition of content in XML documents there is a need for change in schema as well, this required change in schema is done by changing the cardinality property of schema document.

Thus, approach we have taken is to provide versioning to both the content as well as on the schema. Schema versioning is done on the dimensional level. We have designed a case study on ": Criminal Data Warehouse" based on which we have performed our versioning mechanism. A prototype in .NET framework is made to depict the versioning both on the content and schema level. The prototype supports Graphical User Interface (GUI), and the approach is hence user centric.

II. VERSIONING IN XML WAREHOUSE

A. A New Approach

a) Mechanism

The need of versioning in data warehouse has grown over the time but there has not yet been a provider who would have given the absolute solution. We have tried to divide the need as well as our solution logically. The logical separation of our thoughts could be mapped to a scenario of a real-life distributed data warehouse. To understand our division of ideas we should first look at the setup of a distributed data warehouse.

In a distributed data warehouse, the sources play the same role as they play in a conventional data warehouse, the difference being that the sources are not located at a single site. The sources in a distributed data warehouse are located at different sites. Each source is connected to each other and a data repository. We can classify the data repositories as a data mart in an organisational data warehouse scenario. These data repositories are all connected to form a data warehouse. The entire picture of distributed data warehouse seems to be divided into parts, but the key role of distributed architecture is to depict a single global data warehouse image on the front. The architecture remains the same in a warehouse which holds XML data with the only distinction being the format of content. Each source holds as well as provides data to the repository as XML data. Thus, the situation changes to a distributed XML warehouse.

With the above architecture described, the possibility of versioning can appear at different levels of the warehouse. First, we look at the schema versioning prospects. In a data warehouse, whenever a source provides data, it is important for the data warehouse but there are large numbers of data sources that act as data provider. If each provider uses a different format, although the representation of data is in XML, then the data warehouse would simply become a warehouse of digital files. It would be impossible for any kind of classification mechanism to be applied. To control this problem there is always a set of rules that are to be followed by each data provider or there is a standard pre-processing stage that converts data according to predefined structural rules. These rules act as a schema to the data. Each time when data must be entered into the warehouse, there is a validation of data according to the schema rules. With XML as the format being followed in our data warehouse, the significance of schema enriches as well. Not only would schema perform data warehouse format rules but also it would act as a schema of the XML documents which would be validated against it. The beneficial part of this dual role of schema is that the XML document files would not be declared as well defined unless they confirm to the schema, this implies that the documents that enter the data warehouse will all be according to the warehouse format requirements.

This schema definition thus directs the inflow of data in a data warehouse. The importance of change in schema definitions can be understood by this fact that if there were a change in the schema this would cause problems to the integrity of data warehouse. Even if we try to handle the changes in schema there are two possibilities. Firstly, the documents that are present in the data warehouse would have to be upgraded according to the new schema definitions which is a tedious task (if at all it is done), otherwise it may logically cause problems if the XML format in presently stored data is manipulated. Secondly, the change of schema definition is not a favourable step as the schema plays a pivotal role in a data warehouse, even more important in an XML warehouse.

In our work, we have followed a methodology that simplifies the above discussed issues. We have viewed the dimension in a data warehouse as the schema. It means that each dimension acts as a schema for the data concerned with it i.e., the content is logically divided onto dimensions. The segregation of a single global schema into dimensional schemas helps in better working of data warehouse as the XML documents can be guided to their respective schemas for validation thus following a multidimensional model [8] for schema validations. Also, when there is a need of change in the schema definition of one dimension, it does not hinder working of entire data warehouse since other dimensional schemas can still be used for validations. To tackle the issue of change in schema definitions we followed the strategy of schema versioning. Before going into the details of schema versioning we should get familiar with the term "version specification document". A deep study is done to understand and compile the possible future changes in the XML document structure. This compilation leads to a set of possible needed changes in the present schema definition. We call this set as "version specification document".

When there is an evolution in the document files that requires a similar up hilling in the schema definitions, we are required to create a version of that dimensional schema. This need of versioning will be entertained only if it is of use to the organisation that is using the data warehouse, otherwise it is disregarded. This need of organisation has been converted into a possible set of changes in the "version specification document". Therefore, if the requested changes in the schema are according to the "version specification document", then a new version of the schema is created. The XSD schema definition of the "version specification document" is shown in Fig 1.

The new version of the schema consists of all the expected elements from the "version specification document". We add the entire set in the new version of the schema even if there was a request of only a single element to be added. The reason for this bulk set addition to schema is that the entire set has been compiled by the experts and is important to the data warehouse so even the "still not requested" elements might get included in the XML documents with evolution.

Hence, at that time there would not be another version of schema required to handle the changes in the document. Also, it is important to note that we do not invalidate the previous XML documents i.e., even the latest version of schema can be used to validate the earliest documents in the data warehouse. This helps in the maintenance of the versions as well, that is if an old version of the schema has become obsolete then it can be removed without any issue because if there is an XML document confirming to the older version of the dimensional schema it can be validated using the new version of the schema as well. We use the "minOccurs" property of XML schema for not invalidating the older XML documents. This aspect of support for older XML documents i.e., backward compatibility makes the schema definitions more flexible with respect to the data providers hence more user (in this case sources) centric.

```
<?xml version="1.0" encoding="utf-8" ?>
<xsd:schema
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="rs" type="nodetype" />
<xsd:complexType name="nodetype">
<xsd:sequence>
<xsd:element name="node" type="innode"
minOccurs="1" maxOccurs="unbounded" />
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="innode">
<xsd:sequence>
<xsd:element name="parent" type="xsd:string" />
<xsd:element name="root_element"
type="xsd:string" />
<xsd:element name="nodename" type="xsd:string" />
<xsd:element name="nodeatt" type="xsd:string" />
<xsd:element name="nodetype" type="xsd:string" />
</xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

Fig 1: XSD schema definition of "Requirement Specification Document"

The idea of schema versioning has taken birth because of the ever-growing data and its increasing importance in a data warehouse. This data itself can have more than single value even for the same element in an XML data warehouse. The multi valued content needs proper handling for better and precise analytical studies. The possible method than has been used earlier in data warehouses for such multi valued data has been to create new copies of document files holding new values. In an XML warehouse, we can follow a better methodology for handling such multi valued elements. Here again we have followed the same policy of "version specification document". This document contains set of all the elements that could possibly require content versioning. If the requested element that carries new content (for an already existing element in an XML document file of data warehouse) is present in the "version specification document" then content versioning is carried out. The XML properties can be of good use for doing the versioning of content in such situations. We can simply add another instance of the same element in the XML document of the warehouse carrying the new content. With the addition of content in the XML document file, we should make the corresponding required changes in the schema. These changes would be in the cardinality of occurrence of element in the documents i.e., by increasing the value of "maxOccurs" value of the element in the schema definition.

B. Case Study

***a) Introduction*:** We now discuss a case study where we can sense the need of the discussed mechanisms of schema versioning and content versioning. The architecture of the data warehouse in the case would be the same distributed XML warehouse. Also, the mechanism of schema versioning will be carried out at the dimensional level, or we can say we will have the concept of dimensional schemas. The warehouse we are addressing in our case study is an XML warehouse having data of criminals of a country i.e., criminal warehouse.

***b) Structural View*:** The criminal warehouse in our case study has the XML data sources and the data repositories in distributed architecture. The databases of criminals at the city level acts as the XML data sources while the state view of these sources together represents the data repository for each state. These repositories have been referred to as data marts in an organisational view.

***c) Versioning*:** We have discussed versioning of schema as well as content at the dimensional level. We, therefore, now take an example of a dimension of the criminal data warehouse. The example dimension with its properties is:

***Personal Details of convict*:**

- ID
- First Name
- Middle Name
- Last Name
- Date of birth
- Father's Name
- Mother's Name
- Gender
- Marital Status
- Occupation
- Number of siblings
- Fingerprints
- Height
- Weight
- Distinguishable mark on body
- Complexion
- Religion
- Literacy

The XSD schema definition for the above dimensional schema is shown in Fig 2.

The possibilities of schema versioning and content versioning in the dimension mentioned above are:

***Schema Versioning*:**

- Alias
- Number of children
- Picture of convict

Content Versioning:

- Height
- Weight

These mentioned possibilities should also be present in the "version specification document".

The XSD schema definition of the discussed dimension would have the added elements as mentioned under the schema versioning possibilities. The newly generated XSD schema definition of the dimension has been shown in the prototype implementation section of our paper in Fig 3. The newly added elements have been marked as highlighted.

The properties that we have mentioned under the headings of schema versioning and content versioning could arise as a need or may also be due to upgrading the data warehouse to hold more data specific to each convict.

The other dimensions of the criminal data warehouse will similarly have their need for schema versioning and content versioning. The other possible dimensions are:

- Type Of crime under which the convict is being convicted
- Court details
- Punishment details
- Related cases
- Previous convictions

C. Prototype Implementation

To depict the versioning mechanisms that we have discussed in the paper, we have developed a prototype that shows the same on XML documents. The prototype has been developed for depicting the example of the case study that we have mentioned above.

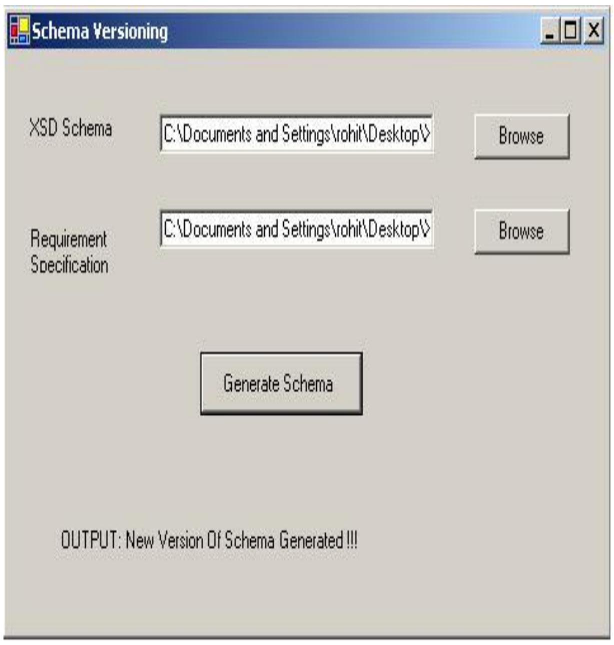
There are three functionalities that have been included in the prototype that carry out the required tasks of versioning. The three components of the prototype are:

XML Validator: The Validator is used to validate an XML document against an XSD schema. The aim of the Validator is to help in knowing whether the XML document is confirming to a particular version of the schema or not.

Schema Versioning: The most important component of the prototype is schema versioning component. It requests the user for an XSD schema and a "version specification document" in XML format. The XSD schema here can be mapped to as a dimensional schema in a real-life data warehouse scenario. A new version of the given XSD schema is created based on the inputs of the "version specification document". This schema can then be validated against the latest documents using the XML Validator of the prototype. The snapshot from the prototype after a schema version has been generated has been shown in Fig 4.

Content Versioning: The other important component of the prototype is of content versioning. The user is required to provide the XML document where the content is to be added, the XSD schema file to which the XML document confirms and a data file in XML format which contains the elements to be added. The content from data file is extracted and then converted to XML format before adding it to the XML document. The corresponding changes in the XSD schema definition file are also made.

We have show Version specification document in the end as appendix.


Schema Versioning
XSD Schema
C:\Documents and Settings\rohit\Desktop\
Browse
Requirement Specification
C:\Documents and Settings\rohit\Desktop\
Browse
Generate Schema

OUTPUT: New Version Of Schema Generated !!!

Fig 2: Schema Versioning in our prototype

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xsd:schema
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="details" type="detailstype" />

<xsd:complexType name="detailstype">
<xsd:sequence>
<xsd:element name="cname" type="nametype"/>
<xsd:element name="dob" type="xsd:date"/>
<xsd:element name="fathers_name"
type="xsd:string"/>
<xsd:element name="mothers_name"
type="xsd:string"/>
<xsd:element name="gender" type="xsd:string"/>
<xsd:element name="marital_status"
type="xsd:string"/>
<xsd:element name="occupation" type="xsd:string"/>
<xsd:element name="no_of_siblings"
type="xsd:string"/>
<xsd:element name="finger_prints"
type="xsd:string"/>
<xsd:element name="height"
type="xsd:positiveInteger"/>
<xsd:element name="weight"
type="xsd:positiveInteger"/>
<xsd:element name="distinguishible_mark"
type="xsd:string"/>
<xsd:element name="complexion"
type="xsd:string"/>
<xsd:element name="religion" type="xsd:string"/>
<xsd:element name="literacy" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>

<xsd:complexType name="nametype">
<xsd:sequence>
<xsd:element name="first" type="xsd:string"/>
<xsd:element name="middle" type="xsd:string"/>
<xsd:element name="last" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

Fig 3: XSD Schema Definition of example dimension

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="details" type="detailstype" />
<xsd:complexType name="detailstype">
<xsd:sequence>
<xsd:element name="id" type="xsd:positiveInteger"/>
<xsd:element name="cname" type="nametype" />
<xsd:element name="dob" type="xsd:date" />
<xsd:element name="fathers_name" type="xsd:string"
/>
<xsd:element name="mothers_name"
type="xsd:string" />
<xsd:element name="gender" type="xsd:string" />
<xsd:element name="marital_status"
type="xsd:string" />
<xsd:element name="occupation" type="xsd:string"
/>
<xsd:element name="no_of_siblings"
type="xsd:string" />
<xsd:element name="finger_prints" type="xsd:string"
/>
<xsd:element name="height"
type="xsd:positiveInteger" />
<xsd:element name="weight"
type="xsd:positiveInteger" />
<xsd:element name="distinguishible_mark"
type="xsd:string" />
<xsd:element name="complexion" type="xsd:string"
/>
<xsd:element name="religion" type="xsd:string" />
<xsd:element name="literacy" type="xsd:string" />
<xsd:element minOccurs="0" maxOccurs="1"
name="number_of_children"
type="xsd:positiveInteger" />
<xsd:element minOccurs="0" maxOccurs="1"
name="picture_of_convict"
type="xsd:positiveInteger" />
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="nametype">
<xsd:sequence>
<xsd:element name="first" type="xsd:string" />
<xsd:element name="middle" type="xsd:string" />
<xsd:element name="last" type="xsd:string" />
<xsd:element minOccurs="0" maxOccurs="1"
name="alias" type="xsd:string" />
</xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

Fig 4: XSD Schema Definition of example dimension after Schema Versioning

III. CONCLUSION

Versioning is an important mechanism for managing data, work, or development strategically. The growth and expansion of data should not appear as a hindrance to a data warehouse, instead more data leads the path to preciseness in analysis of trends and corresponding predictions. With the escalation of data there could also be evolution of details and the specifications of content in the sources, which in turn would affect the data warehouse. Versioning can play an important role in such scenario where the data evolution is an obvious aspect. In a distributed data warehouse, this possibility of content format evolution becomes more of certainty.

The advantages of versioning can be exemplified by the decision to segregate the global schema into dimensional schemas. Our approach has the advantage of creating versions of schema according to the specifications laid by the data warehouse experts through "version specification document". The creation of new version of schema also acts as a favorable step towards schema evolution. The backward compatibility of newly generated schemas also helps in the maintenance of the versions of schemas. With content versioning being possible we can have more than single version of content. The opportunity of content versioning in XML warehouse is easily possible because of the nature of schema definitions. It can therefore be very easily implemented in the scenario of XML warehouse.

If we do not have an XML warehouse, then our approach of validating documents against a schema and then generating new versions of schema is applicable to any data warehouse picture. The limitation would come when we try the backward compatibility of the newly generated schema. Also, the prototype implementation of the idea has a limitation that while doing schema versioning we cannot add an element of user-defined data type.

The actual tool of our prototype could have our proposal as functionality. The tool should have connectivity over network to the repositories of the data warehouse. Also, there could be a view feature that represents the evolution of schema versions through concept maps.

APPENDIX A

Format of Version Specification Document

The scope of versioning in the XML warehouse is identified and classified into schema versioning needs and content versioning needs. The following tables depict the need of versioning in the case study taken.

Scope of versioning (**schema**) in the proposed case study:

S.No	**Dimension Name where change is done**	**Measure/Attribute added in new version of schema**	**Reason for addition of attribute**
1	Personal details of convict/ criminal	**Alias**	*Convict having other known names that may be used for identifying him/her.*
2	Personal details of convict/ criminal	**Picture of convict**	*Picture identification is introduced in the system.*
3	Type of crime under which the criminal is being convicted	**Name of operation**	*It will be an important addition if the convict is arrested under a special operation because then the rules and judgments would be affected according to the severity of the operations importance.*

Scope of versioning (**content**) in the proposed case study:

S.No	**Dimension Name where change is done**	**Measure/Attribute added in new version of schema**	**Reason for addition of attribute**
1	Personal details of convict/criminal	**Height**	*It can change with the passage of time if the convict has been held earlier at a young age.*
2	Personal details of convict/criminal	**Weight**	*It can change with passage of time.*
3	Court Details	**District**	*Transfer of case from one district to another.*
4	Court Details	**Level of court**	*It could have more than one value with time if a case is petitioned at a new higher court.*
5	Court Details	**Jury**	*Jury team can change even during the case.*
6	Court Details	**Lawyer for**	*Change of lawyer.*
7	Court Details	**Lawyer against**	*Change of lawyer.*
8	Punishment details	**Convict no.**	*If a convict is transferred/moved from one prison to some other.*
9	Punishment details	Prison name	If a convict is transferred/moved from one prison to some other.
10	Punishment details	Length of sentence	If a convict is given recommendation base on good behavior.
11	Punishment details	Type of prison cell	Changes in prison cell due to behavior of convict (or punishment).

REFERENCES

[1] Bogdan Czejdo, Kenneth Messa, Tadeusz Morzy, Mikolaj Morzy, Janusz Czejdo Data Warehouses with Dynamically changing Schemas and Data Sources.

[2] Bartosz Bebel, Tadeusz Morzy, Robert Wrembel, Johann Eder, Christian Koncilla, Creation and Management of Versions in Multiversion Data Warehouse.

[3] Matteo Golfarelli, Stefano Rizzi, Boris Vrdoljak, Data Warehouse Design from XML Sources.

[4] Matteo Golfarelli, Stefano Rizzi, Jens Lechtenborger Gottifried Vossen, Schema Versioning in Data Warehouse.

[5] Li Yang , Naphtali Rishe, A Flexible & Effective XML Storage & Retrieval System.

[6] XML Schema Versioning.

[7] John Finianos, Jugdish Mistry, Versioning in Existing Data Warehouse.

[8] Marko Banek, Zoran Skoeir and Boris Vrdoljak, Logical Design of Data Warehouses from XML.