An Approach for Generating an XML Data Warehouse Schema using Model Transformation Language

Zoubir Ouaret, Rachid Chalal[1], Omar Boussaid[2],
[1]High National School of Computer Science, Algeria
[2]ERIC, Univ. of Lyon 2, France
z_ouaret@esi.dz, omar.boussaid@univ-lyon2.fr, r_chalal,@esi.dz

JDIM
Journal of Digital Information Management

ABSTRACT: *Traditionally, the multidimensional schema of the data warehouse is derived from data sources that are mainly the company's internal data, well-known and structured, by identifying facts, dimensions and numeric measurements through a manual analysis of the operational schemas. With the proliferation of new platforms of communication in today's information societies, there has been growing numbers of web-based applications such as online social networks that generate huge amounts of XML data on the web. Therefore, it is increasingly important to develop an appropriate warehousing approach for such ever-growing XML data sources. However, XML documents have a complex hierarchical structure. Moreover, designing and building DWs is tedious, timeconsuming, error-prone and expensive process. In this paper, we describe an approach for automatically generating and building the star schema for data warehouse from XML schema. This approach is extensively based on standards (UML, XML, QVT, and XSLT). First, we model the structure of XML (XML Schema) using the Unified Modeling Language, which is the standard language for object oriented analysis and design. Then, we provide an algorithm that automatically selects the multidimensional concepts. After that, a representation in an XML schema language as the XML data warehouse schema description is derived automatically from UML star schema. Furthermore, we choose Query/Views/ Transformation (QVT), which is also OMG standard transformation language for defining and formalizing transformations between models. Finally, a prototype tool is implemented for testing and evaluating our approach and its transformations.*

Subject Categories and Descriptors
H.2.7 [Database Administration]: Data warehouse and repository; **D.3.3 [Language Constructs and Features]** Data types and structures

General Terms : Data warehouse Design, XML, XML Schema

Keywords: XML Data Warehouses, Web Data Sources, UML; QVT, Transformation Language, Automation.

Received: 16 June, 2014, Revised 29 July 2014, Accepted 8 August 2014

1. Introduction

In the past decade, with the proliferation of new platforms of communication in today's information societies, the world of computing has been changed; there has been growing numbers of web-based applications such as online social networks that generate huge amount of XML data in the web. Companies therefore have an increased need to explore an effective means to manage and integrate such ever-growing XML data as data sources in data warehouses for further analysis and decision-making. From a historical point of view, Bill Inmon is known as the "*Father of Data Warehousing*", introducing in 1990 the accepted definition of a data warehouse which was defined as 'subject oriented', integrated, time-varying, non-volatile collections of data. As XML data sources are the new

data formats, classical data warehouse approaches have to be revisited in the new context of XML data warehousing which provide flexibility interesting in the context of web and semi-structured data sources. Thus, many design methods have been proposed in this context.

Several works such as [4], [28], [16], [2], [22] introduced semiautomated approaches for designing XML data warehouses. However, these approaches present some limitations and drawbacks: they do not provide mechanisms for automatically discovering a fact and its own measures; they only focus on how to automatically identify dimensions. Moreover, they do not express transformations with standard language among DW design steps (data sources model to multidimensional (MD) model and (MD) model to implementation model, UML model to XML model …), and therefore, there is a need for models transformations language as QVT language and to standardize and unify methodologies related to design XML DWs, which has been endorsed by the OMG. Automating Data Warehouse Conceptual and logical Schema Design using standard transformation language can provide real value to BI development projects, and increase their chance of success with DW schema quality while reducing cost, risk and manual effort.

Obviously, the development of DWs is particularly complex and requires an appropriate life-cycle (Conceptual design: requirement analysis and data source identification; Logical design and Physical design), that has many phases to be accomplished: accessing and collecting XML data sources, extracting the XML structure and schema if it not provided, preprocessing or cleaning step (duplicates removal, merge/purge, name & address matching, field value standardization), modeling schemas using UML models or graph models, extracting multidimensional elements such as fact, measure and dimension, creating a logical multidimensional model (star/snowflake schema), and translating the logical representation into an XML Schema, which viewed as an XML data warehouse schema description, supported by a native XML database, which allows to store XML data sources in native form according to this description. Most researches focus on the intermediate logical design phases which are the most important step in the DW design process, while few provide approaches including both "*conceptual design*" and "*logical design*" levels. However, transformations and rules from these approaches are not expressed by standard transformation languages. The first motivation for this paper stems from this context. In addition, building a data warehouse is often a complex combination of process and technology that requires a manual effort and significant user intervention. To overcome these complexities, usually, amplified in quasi-manual approach, researchers have proposed multiple approaches for semiautomated design of XML data warehouses. So far, no fully automatic approaches have emerged, unfortunately. The second motivation stems from this context to increase the level of automation and make easier the job of DW/BI designers.

Based on this twofold observation, in this paper, we propose a QVTbased approach for the automatic generation of XML DW schema from the XML schema describing XML sources using UML since we consider it well supported and familiar to large user groups. This approach firstly proposes the transformation of the W3C XML Schema describing a global schema of data sources such as DPLB. xsd into a UML class diagram, and secondly shows how to use a set of rules to extract important multidimensional elements. Our focus is not only on automating XML Data Warehouses design, but also on providing a set of rules based on the QVT transformation language to specify the transformation from XSD to UML, UML to UML Star and UML Star to XSD Star. Our approach is open since transformation algorithm is adaptable by changing transformation rules. In addition our approach provides a high degree of automation that enables designers to quickly and easily design and building conceptual and logical schema of XML DW also and to reduce deployment costs.

Finally, a prototype tool is implemented for testing and evaluating our approach and its transformations. An open, flexible tool that facilitates the design of XML data warehouse called *"XUML Star Tool"*. The proposed approach is fully automatic and does not require any technical expertise in multi-dimensional data modeling.
The rest of the paper is organized as follows. Section 2 introduces related work on XML data warehousing systems and comparative analysis of approaches according the design and generation process. Section 3, presents a rule-based approach for generating an XML Data Warehouse Schema using *QVT* transformation language. In Section 4 describes an automatic tool that we have developed to support the proposed approach, and Section 5 provides conclusions and discusses future work.

2. State of the Research

In the last few years, several techniques and approaches for the conceptual, logical and physical design of DW systems from XML data sources have been proposed with various degrees of automation. In this section, we present a brief discussion about some of the most well-known approaches.

Conceptual semi automatic XML data warehouse design was first introduced by [4]. Starting from a DTD, the design process is carried out as follows: After simplifying DTD, creating DTD graph, then building the attribute tree for each selected fact, finally defining measures and dimensions. However, this approach starts with a single DTD. Starting from multiple DTDs, [16] present a step-by-step approach for building an XML data warehouse. The main idea of this approach is the use of the existing star model to XML data sources context. A new semi-automatic approach is presented by [28] in the context of the web data, they use XML schema for designing a logical schema of XML DW. In this approach, a design process is carried out by pre-processing the XML schema to remove complex

and redundant specifications of relationship and creating a schema graph, then navigating its arcs to generate an appropriate MD schema. Another proposal to semi-automatically build XML data warehouses is presented by [26]. It describes a Graph-semantic - based approach for the creation of DW logical schema.

Furthermore, based on the UML at the conceptual level, an interesting approach is presented by [6] that present a framework for integrating XML data and relational data sources. The construction of a UML snowflake diagram presupposes that the designer has detail knowledge about the field. In addition, they focus on finding the multidimensional structures directly in the XML data sources. Further, a conceptual approach named *"xFACT"* for the construction of XML repository is presented in [12], in which every fact is described in a single XML document and XML virtual views for representing dimensions. Another interesting approach is presented in [25] that introduces a global document warehouses *(GxFact)* using the xFACT approach [12].

Proposals for XML Data Warehouses at physical level, which consider storages and implementations of multidimensional schemas and the optimization techniques (such as indexing, MV) also exist, but they are quasi-manual approaches. The most interesting proposal is [24] in which the authors define a methodology that consists in four well-defined steps: cleaning, summarization, intermediating XML documents, updating/ linking existing documents and creating fact tables.. Another interesting approach on physical level is also presented by [17] that build XML warehouses, in which each fact and dimensions are stored as XML documents, they produce XML warehouse from a single repository of XML documents for facts and one repository of XML documents for each dimension. Materialized views-based approach to create an XML data warehouse by frequent query patterns discovered from user historical queries is presented by Zhang in [30]. Thus, the design process can be applied in a completely automated fashion by analyzing the user's query logs. Moreover, based on materialized views, an approach named *"DAWAX"* for designing and managing the XML documents warehouse at the physical levels is presented in by [1], in which XML documents are filtered according to the user requirements before storing them in a repository "XML warehouse".

Obviously, the final users' requirements are very important in the DW design. One of the most relevant proposals combining user analysis objectives and XML data sources was presented by [2], it is mixed approach for warehousing complex data at a logical level, named "X-warehousing". Another approach presented by [9] has extended the "X-warehousing" approach to a physical, they present an XML-based data warehousing and online analysis approach named "*X-WACoDa*". Another semi automatic approach is introduced in [22] that take into account both user requirements and data sources for implementing OLAP Systems. However, a primary focus of this approach is for the analysis of document-centric, whereas an "*Xwarehousing*" approach [2] that focus on data-centric. A similar approach that focuses on analyzing XML text-rich documents was presented by [19] that introduce an approach to semi-automatically build a conceptual MD schema from document-centric XML document.

Recently, other semi automatic approaches [21] and [3] have been developed. For proposal in [21] aimed at creating MD logical schemas using attribute tree. The author in [3] has adopted ROLAP techniques for generating a DW logical schema from XML schema through semi-automated process. With regard to the Models transformation techniques, there are only a few approaches [8], [10], [31] and [15] on the development of DWs. Unfortunately, these approaches focus only on RDBMS as data sources and didn't consider the XML data sources.

Moreover, for more detailed information related to XML data warehousing, including data integration and multidimensional modeling. The interested reader can refer to [20], [23] and our recent previous work Ouaret et al in [14]. Perez et al in [20] present a classification of approaches on combining Data Warehouse (DW) and Web data. They focus on particularly in the building DWs for XML data, including the design of multidimensional databases for XML data sources and the extension of traditional OLAP techniques to analyze online XML data. They also study approaches that use of XML technology as an integration tool in distributed DW systems, including the use of XML for Exchanging multidimensional data and metadata and to integrate distributed DW. Another interesting study on this area is presented by Ravat et al in [24]. They give a detailed discussion of the different approaches for an XML data warehouse by outlining differences that can then be used to find the appropriate model such as XML data warehouse or XML document warehouse approaches and integration or warehousing architectures.

Our previous study in [14] is on the one hand based on these works, but using appropriate criteria (Abstraction level, models, type of schema, categories, paradigm, automation, storage, query language) identified from techniques used in these approaches, we have presented a more homogenous comparative study that would permit a better comprehension of the field of XML data warehousing. Our scope in this paper is how to generate the schema of DW from the XML data sources in automatic fashion using the transformation rules expressed in *QVT* transformation language

Finally, results presented along this section are summarized in Table I, columns correspond to criteria *(C: Conceptual, L: Logical, P: Physical)* and rows correspond to each of the above-described approaches

3. Proposed XML Data Warehousing Approach

As we have previously presented in the introduction section, the aim of this paper is to automate the design of XML data warehouse schema using open industries standards (i.e. UML and QVT language), which facilitates its implementation and extension. In dimensional modeling, there are two important elements: facts (measures), which are usually numeric values that could be aggregated and dimensions (context), which are business descriptors that specify the facts. Our rules based-approach to automatically build the logical schema for a data mart is described in this section. As a starting point we assume that XML data sources (XML documents) are conformed to XML schema W3C XSD. For example, a database for bibliographic data of computer science (DBLP) contains a large XML document available at DBLP's website is conformed to DBLP.xsd to describe the content and structure of such large XML documents. The design process consists of the following 4 steps:

1. Generating UML class Diagram
2. Reducing UML class diagram
3. Creating the Star schema
    - Extract measures
    - Find candidate fact class
    - For each fact, select dimensions
4. Generating the XML data warehouse schema with XSD

3.1 Automation Degree-Based Comparison

As described above, many approaches to XML DWs systems, including preprocessing, conceptual and logical modeling, outputting in a Native or Relational), have been proposed, with various degrees of automation. Our approach is fully automatic and does not require any manual intervention and expertise of DW system from the user. To compare and evaluate XML DW approaches from the automation degree, we propose a new formula in which the degree of automation is determined as $Aut = \Sigma S_a / \Sigma S_t$, where ΣS_a is the number of automatic steps and ΣS_t is the total number of all steps. The results are shown in the following table II. As we can see, we provide a high degree of automation (*L:Low, M: Medium, H: high, F: Fully automatic*). Due to few works develop a tool based on their proposals; we are not able to compare all approaches

3.1.1 Generating UML class diagrams

Generating UML class diagrams from XML schema is based on the idea that every XML document sources is an instance of XML schema and every XML schema can be graphically modeled using UML class diagram notations. So, here, taking as an input XML schema describing XML data sources (XML documents), XML schema is automatically transformed to UML Class diagram (Table III). UML Classes emerge from complex types and Data type properties emerge from attributes and from simple types. In the first, the XML-Schema is analyzed using XSOM that allows applications to easily parse XML Schema documents.

3.1.2 The basic concepts of QVT

Approaches	Abstraction level			Processes (automation)
	C	L	P	
Golfarelli [4]	✓	✓		semi automatic
Pokorný [16]		✓		semi automatic
Jensen [6]	✓	✓		-
Neimi [13]		✓		-
Hümmer [5]		✓		-
Vrdoljak [28]		✓		semi automatic
Pederson [18]		✓		-
Park [17]		✓	✓	-
Rajugan [25]	✓			-
Trujillo [27]	✓			-
Yu li [26]	✓	✓		semi automatic
Baril [1]		✓	✓	-
Nassis [12]	✓	✓		-
Rusu [24]		✓	✓	-
Boussaid [2]	✓	✓		-
Mahbouni [9]			✓	-
Ravat [22]	✓	✓		semi automatic
Zhang [30]		✓	✓	-
Parimala [21]		✓	✓	semi automatic
Pujolle [19]	✓	✓		semi automatic
Dasgupta [3]	✓	✓		semi automatic
Sarkar [26]	✓	✓		-
Our approach	✓	✓		**Fully automated**

Table 1. The characteristics of xml data warehousing Approaches (|✓| 3 indicates support3)

Query/View/Transformation (QVT) [32] can be considered as the standard model transformation language proposed by the OMG (Object Management Group) in 2007. The QVT transformation is a function with source and target models which must conform to the MOF. It is defined as a set of relations that must hold for the transformation to be successful. These following lines show the basic elements and transformation modes of a relation:

Check-only: to test that models are related in a specified way;

Enforce mode: to modify one of the models so that the set of models will be consistent.

When clause: it indicates the pre-condition under which the relation needs to hold.

Where clause: it indicates the post-condition that must be satisfied by elements in the relation.

The QVT-Relations has both a textual and a graphical concrete syntax of relations as shown in this section, in which we provide a formal and clear description of transformations that are automatically executed. For example, Figure 1 (R1) below shows the transformation

Approach	[4]	[28,16]	[2]	[3]	[22]	XUML Star
Automation level	3/5 M	4/5 H	1/4 L	2/3 M	2/4 M	4/4 F
Trans-formation Technique	-	-	-	-	-	QVT

Table 2. Automation Degree-based Comparison

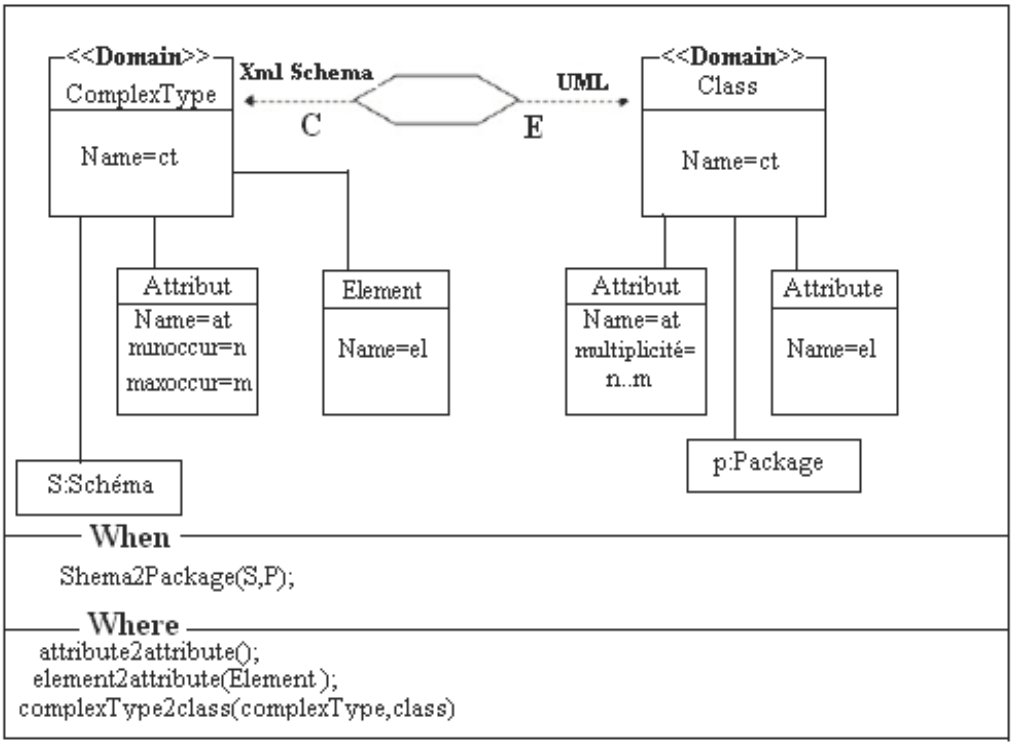
XML Schema	UML Class diagram
(<xs :element>) with (<xs:complexType>)	Classes UML
sub element of the corresponding class complex type	Attribute
<xs:minOccurs> or <xs:maxOccurs>	UML cardinalities
xs :int>, <xs :double>, <xs :float>, <xs:string>	Natives data types such as int, double, float, string,
Simple type attributes and child elements of a complex element	Attributes of the corresponding UML class
Key/Keyref references of elements. Hierarchical relationship. XLink and XPointer.	Associations
Complex type of the subclass is defined as an extension of the complex type of the Super Class	Generalization

Table 3. Transformation of XML Schema To UML Class

rule of ComplexType (source: XML schema) to a UML Class (target: XML schema) uses the function complexType2class ().

• **R1:** Transformation rule of ComplexType to class (textual and graphical syntax)

```
Relation complexType2class (complexType ,class)
{
checkonly domain xml complexType : complexType {
name = complexTypeName ,
set { attribute = aXML: Attribute {name = attributeName ;
type=attributetype, minoccur = n, maxoccur = m}
}
set { element = eleXML: element {name = element Name, Type=type
element, minoccur = n, maxoccur = m,}
}
}
enforce domain uml class : class { classeName = complexTypeName ,
set { attribute = aUML: Attribute {name = attributeName ;type
= attributetype, multiplicité = n..m ;}
set { attribute = aUML: Attribute {name = elementName; type
 = elementType, }
}
When Schema2package(s, p)}
Where
{ attribute2attribute() ;
Element2attribute() ;
complexType2class (complexType,class)
}}
```



3.2 Preprocessing process

This step tends to make some cleaning and conforming for the UML class diagram generated in the previous step to produce compact structure with compulsory classes and relationships. This is achieved by removing inconsistency and reducing their complexity by relaxing some conditions, or by deleting the unnecessary and redundant classes. This process includes data cleaning and transformation. This stage includes six basic rules (Redundant classes (RR), Embedded class (RE), Inline class (RI), Disconnected classes (RD), Trivial classes (RT), and Associations between removed classes (RA)):

• **(RR) Redundant classes:** Two Classes are redundant if they denote the same set of instances and express the same information, In that case, remove one class;

• **(RE) Embedded class:** Removing redundant attributes, this function is applied in the case when one class has all attribute of another class and linked by key

• **(RI)Merges Inline classes:** (One-to-one relationships): Two classes are related by One-to-one relationships that objects involved in the relation are highly dependent. In that case, join together inline classes.

• **(RD)Remove isolated classes (Disconnected classes):** This function verify if there is isolated classes, then removes a class without incoming or outgoing relationships), while the smallest Star schema require two connected classes one for fact and other for dimension,

• **(RT)Trivial classes:** Delete a class without content.

• **(RA) eliminated classes:** Remove relations between eliminated classes: If one class in the relationship has been eliminated, then the relationship must be eliminated or restated in terms of other classes.

As a result, From obtained reduced form, after applying this process, discovering MD elements is an easy process, because the number of classes to check is decrease and the maximum number of attributes for each class with numerous one-to-many relationship between classes, therefore high cohesion level, as a consequence the complexity of the algorithm for searching MD elements decreases from $O(n)$ to $O(k)$ for fact and from $O(n^2)$ to $O(k)^2$ for dimensions, (n is the number of classes and $k < n$).

3.3 Building logical schema (The star schema)

Once the generated UML classes diagram has sufficiently been reduced, the activity to identify the DW schema can start with reduced form. The most natural way to model a data warehouse is as a star schema, which is the simplest DW model. It is called a star schema because the diagram resembles a star in which the center consists of one or more fact classes and the points of the star are the dimension classes. The identification of the all potentially needed multidimensional elements such as fact and dimensions is a compulsory and a tedious process.

In the literature there are several techniques to facilitate such process. However, they do not provide a standard approach to correctly identify all potentially elements in a fully automatic manner using formal specifications, and therefore, there is a need for Automatable rules based on standard transformation language. In this step, we propose an automatic algorithm for selecting suitable fact according to the number of additive attributes they have, then, for every fact; we extract their corresponding

dimensions, in which both fact and dimensions can be considered as potential star schema based on the following transformation rules expressed using QVT. The main idea underlying this step is that candidate's fact classes emerge from those many-to-many relationships in the UML CD with numeric fields and additive non key facts and candidates dimensions are identified by means of (* - 1) and (1-1) relationships from the candidate fact. Figure 2 (R2) below shows the top relation to transform UML class diagram (source: obtained from XML schema) to a Star schema (target: Dimensional model/Kimball model) using the function Top relation UML2UMLstar () with their embedded functions.

• R2: Main: Transformation of UML into UML star

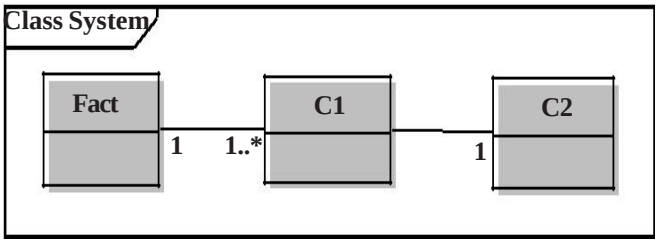
```
Top relation UML2UMLstar
{
checkonly domain uml u:Package {name = ps}
enforced domain umlstar us:Package {name = ps}
where { class2fact(class, fact);
class2dimension(class, dimension);
attribute2measure(attribute)}
}
```

3.3.1 Find Measures

The first step is defining measures, the numerical attributes:

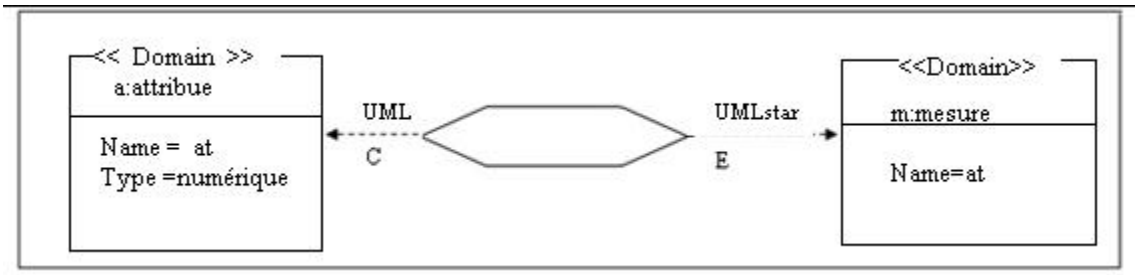
• Every numerical no-key attribute and additive is potential measure of class,

• Every numerical no-key attribute of a class linked with potential fact class by one-to-one or one-to-many is potential measure for fact class.



R3: Transformation of attributes into measures

```
Relation Attribut 2 Measure(attribut)
{matrice min [M][M], max[M][M]:int;
fact,c1,c2:class; rel12:Booleen;//relationship between classes c1 and c2
checkonly domain uml c:Class { name = className ,
set { attribute :a {name = attributeName , Type = typeattribute,}}
Where {if ((min[fact][C1] = 1 and max [fact][C1] = 1) and (min[C1][fact] = 1 and max[C1][fact] = N))
then Attribute2mesurefact (fact, c1) ;
endif
if ((rel12 = true and min[C2][C1] = max[C2][C1] = 1)and( min[fact][C1] = 1 and max[fact][C1] = 1)) then Attribute2measurefact(fact,c2) ;
endif }}
relation Attribute2measurefact (fact, c1)
{enforce domain umlstar fact: factclass { name = className ,
set { measure :m{name = attributeNamec1, Type = typeattributec1,}
}}
```

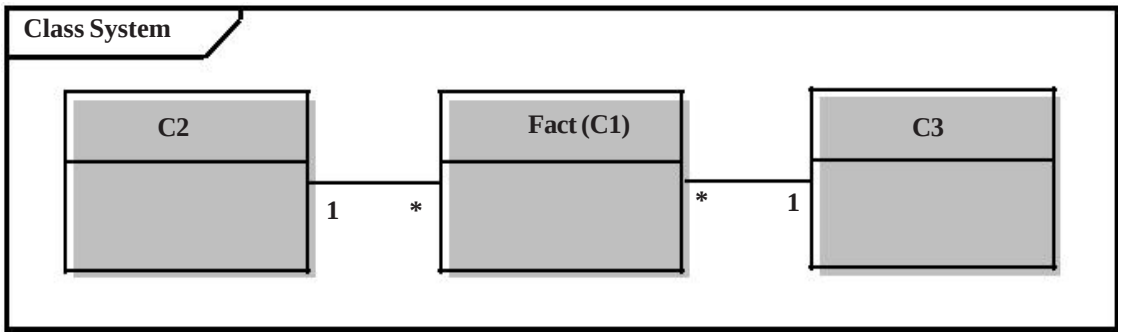


3.3.2 Find Facts

• Each class containing numerical values of interest is also identified as a fact

• Class with a large number of numeric attributes is most probable to be a potential fact class For example: An important type of relationships used in our case is shown in the figure below:

• A class $C1$ in UML class diagram is transformed into fact class, if for each classes C_I and the cardinality is C_I (1-1) and Fact class (C_1) (0 - N)

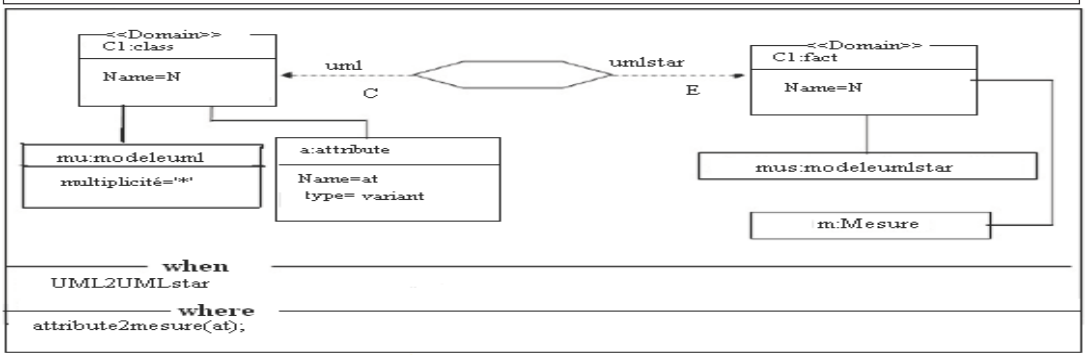


R4: Transformation of class into fact

```
Relation class2fact(class, fact)
{ i, a, M: integer;
matrice min [M] [M], max [M] [M]: integer; rel Fact_i: Booleen
fact:class; myclasses: = list {C1, C2,…..Cm};
checkonly domain uml class : Class {  name =  className,
set { attribute :a {name = attributeName , Type = typeattribute,} }
umlModel :mu {id_model = string, type = string} }
enforce domain uml star fact: fact { factName = className , set {
measure: m{ mesureName = attributeName , measuretype =
attributetype}}
umlstarModel : mus{id _starmodel = id_ model, type_starmodel =
type_umlmodel, }}}
When Uml2UMLstar
Where {for Each  ((i|i < m) &&(a! = 0))
{if ((rel_fact, i = true) and (min [fact] [Ci] = 0) and (max [fact] [Ci]
= N) and min [Ci] [fact] = max [Ci] [fact] = 1) else a = 0;
}

if (a = = 1)   Attribute2measure (a, m);

}}
```



3.3.3 Find Dimensions

Dimension classes: information for which facts will be analysed.

• For each fact class, dimensions are identified by means of many-to-one and one-to-one relationships

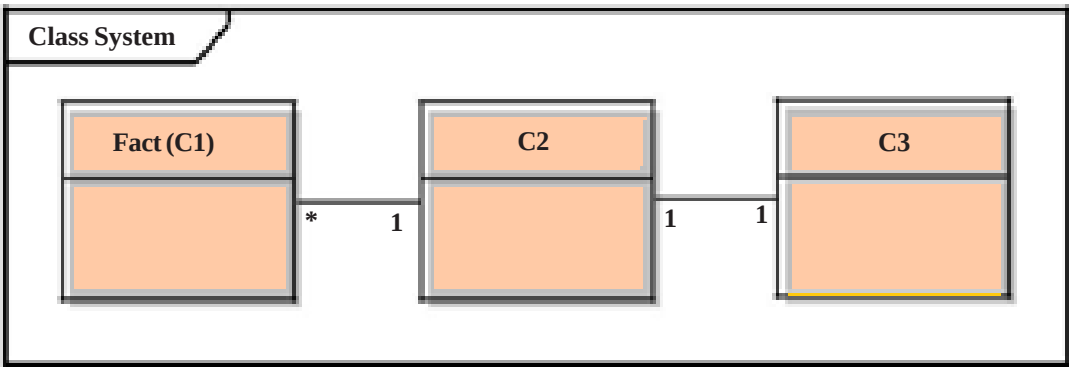
• In addition, time is an important dimension in DW applications.

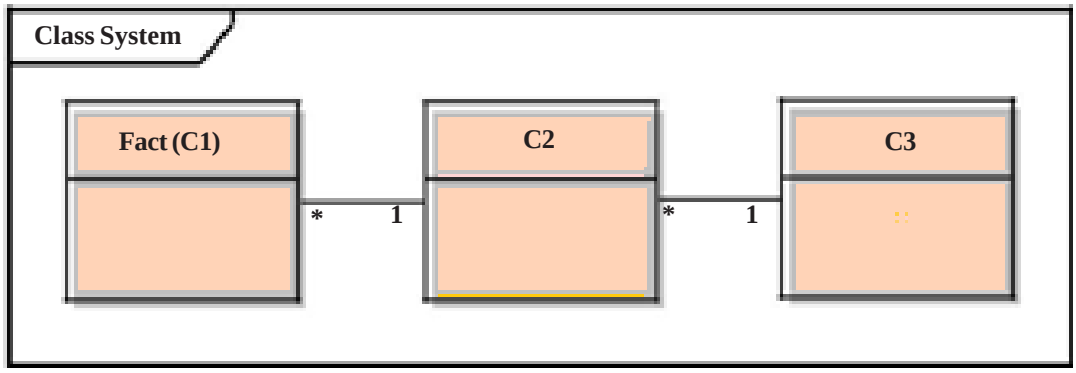
For example:

two important types of relationships used in our star schema, Hierarchies for the dimensions are represented in the dimensional class itself.

1. Appropriate star schema

Each generated star schema is composed of one fact and several dimensions. An appropriate star schema *(SS)* is identified by using a formula that computes *(Val)* the total number of measures and dimensions as follows:

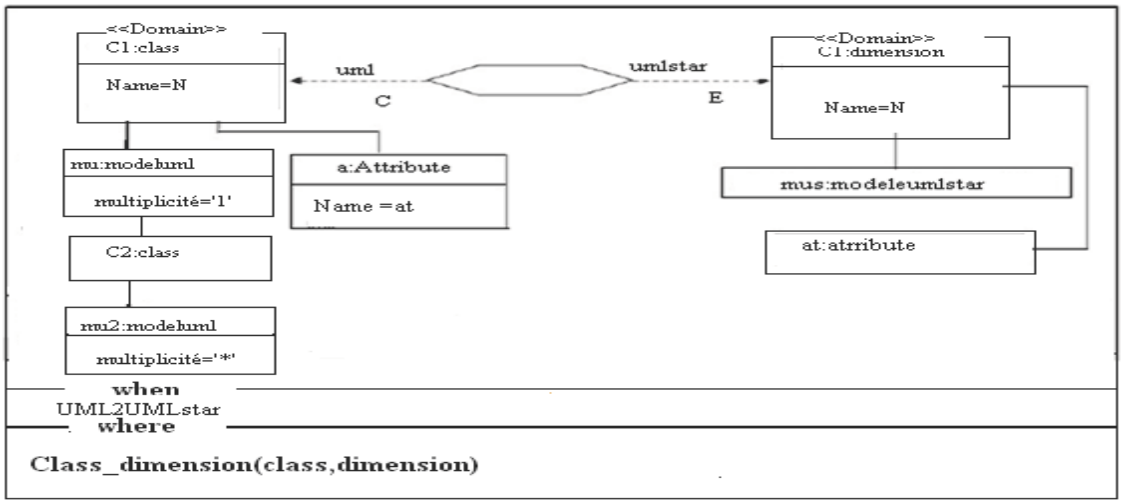




R5: Transformation of class into dimension

```
Relation class2dimension(class,dimension)
      { M:integer;
      matrice min[M][M],max[M][M]:integer; fact;C2;C3:class;
      rel23:Booleen;// A relationship between classes C2 and C3
checkonly domain uml class : Class { name = className , set { attribute
            :at {name = attributeName , Type=typeattribute,}}
            umlmodel:mu {id_ model =string, type=string}
            when UML2UMLstar ;
            Where { if ((rel23=true) and(min[fact][C2]=0 and
max[fact][C2]=N) and (min[C2][fact]=max[C2][fact]=1)) then
If( min[C2][C3]=max[C2][C3]=min[C3][C2]=max[C3][C2]=1) then
Dimention_class (fact, C3);
else if ( (min[C2][C3]=0)and(max[C2][C3]=N)and
( min[C3][C2]=max[C3][C2]=1)) then Dimention_class(c2,c3) ;
endif
endif
endif
} }
```

```
Relation Dimention_class(class1,class2)
{ enforce domain uml star class2: class { Name = className ,
      set { mesure: m{ mesureName = attributeName , mesuretype
      =attributetype}}
      umlstarmodel : mus{id _starmodel = class2+'asoc'+class1,
      type_starmodel=type_umlmodel, }
}
```



SSi = {*Name*, *F* (*m*), *D*}∀*i*∈ [1. ...*n*], is a star schema where
F is a fact,
m is a measure of the fact *F*,
D is a dimension
Val (*SSi*) = *Nbr* (*m*) + *Nbr D*

This formula is executed and repeated for each schema *SSi* generated from each identified fact. It is important to note generally that, a star schema with a higher number of dimensions and measures is most likely to be the most appropriate model.

3.3 Generating XML Star Schema (XML DW structure)

Finally, for this last step of the design process, we present a set of rules based also on QVT to automatically transform a select star UML model into XML schema, which serve as PSM model equivalent. Thus, this final star XML schema contain all the necessary XML Data warehouse structure that composed of multiple schema sub-documents, one part to describe fact data and other parts to clearly describe every dimensions data. Following, we explain in each transformation rules for the most widely used UML Class Diagram elements (class, attribute, association, generalization and built-in data type).

A UML class is transformed into complex types (*<xs:complexType>*) and elements of these types

- UML class attributes are transformed to either XML elements or XML attributes.

- UML association is transformed into child elements of complex types in XML schema.

- Multiplicity is represented by *minOccurs* and *maxOccurs* elements

- Built-in data types (int, double, and string) are transformed into *<xs:int>*, *<xs:double>*, and *<xs:string>* in XML schema

- Not restricted attributes in the UML class into *<xs:all>* in XML schema and restricted attributes in the UML class into *<xs:sequence>*.

- Other association types like dependency, composition and aggregation are treated as general associations.

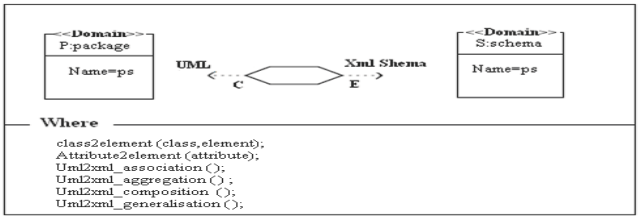
Fragment of the generated Star XML schema is shown as follow.

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xs:schemaxmlns:xs=
"http://www.w3.org/2001/XMLSchema">
< - - FACT - ->
<xsd:element name = "Fact">
<xsd:element name = "id_invoice" type = "int"/>
<xsd:element name = "id_customer" type = "int"/>
<xsd:element name = "id_product" type = "int"/>
<xsd:element name = "montant" type = "float"/>
<xsd:element name = "qte" type = "int"/>
<xsd:element name = "Ass_1" type = "customer"/>
<xsd:element name = "Ass_2" type = "date"/>
<xsd:element name = "Ass_3" type = "product"/>
</xsd:element>
< - - Dimension customer - ->
<xsd:element name = "customer">
<xsd:element name = "id_ customer " type = "int"/>
<xsd:element name = "name" type = "string"/>
<xsd:element name = "street" type = "string"/>
<xsd:element name = "city" type = "string"/>
<xsd:element name = "state" type = "string"/>
<xsd:element name = "zip" type = "string"/>
<xsd:element name = "Ass_1" type = "fact" minOccurs = "0" maxOccurs = "unbounded"/>
</xsd:element>
```

```
< - - Dimension date- - >
<xsd:element name = "date">
<xsd:element name = "date_days" type =
"date"/>
<xsd:element name = "month" type = "int"/
>
<xsd:element name = "year" type = "int"/>
<xsd:element name = "Ass_2" type = "fact"
minOccurs = "0" maxOccurs = "unbounded"/>
</xsd:element>
< - - Dimension product- ->
<xsd:element name = "product">
<xsd:element name = "id_product" type =
"int"/>
<xsd:element name = "nom_prod" type =
"string"/>
<xsd:element name = "Ass_3" type = "fact"
minOccurs = "0" maxOccurs = "unbounded"/>
</xsd:element>
</xsd:schema>
```

• R6: Top relation for the transformation of UML Star into XML Star

```
Top relation umlStar2xmlStarPackage
{
packageName , className : String ;
checkonly domain uml1 class : Class { name
= className , namespace
= package : Package{ name = packageName}}
;
enforce domain xml1 ns : NameSpace { name
= packageName
{ child = element : Element { tagName =
className }} ;
where { uml2xml Class ( class , element )
;
uml2xml Aggregation ( class , element ) ;
uml 2 xml Association ( class , element )
;
uml 2 xml Generalization ( class , element
) ;
}}
```



• R7: Transformation rule of UML class to ComplexType (textual syntax)

```
Relation class2element(class,element)
{ checkonly domain uml class : Class {
name = className, namespace = p:Package {
},
set { attribute = aUML: Attribute {name =
attributeName, type = attributetype,
multiplicity = n . .m, }
}}
enforce domain xml element : Element {
elementName = className,
set { child = aXML: Element{ elementName =
attributeName
elementtype = attributetype ,
minoccur = n, maxoccur = m,}
} } ;
when package2schema(p,s)
where uml2xmlAttribute ( attribute );
}
```

In this fragment, we present a part of the Star XML schema of the Star model obtained in the previous step. This schema can be accordingly imported in the native XML database representing the schema of XML Data warehouse which is populated with XML data sources by applying XSLT stylesheets to transform the original XML sources into another presentation according to the structure of this schema.

4. Implementation

To demonstrate the applicability of the previously described our approach based on transformation rules introduced in each step of design process, we have developed A prototype tool for Automatically Generating Star Schema from XML schema using UML called "*XUML Star tool*" that allows designers to design and to create an XML data warehouse. It is developed using the Eclipse platform, which is one of the most well known software development environment (a free and open-source project) and evolved into a multi-language, highly-extensible framework, thanks to a powerful plug-in system and the integration of MDE concepts in the platform. Implemented as a set of Eclipse plugins, our prototype can be integrated with Eclipse based-BI tools.

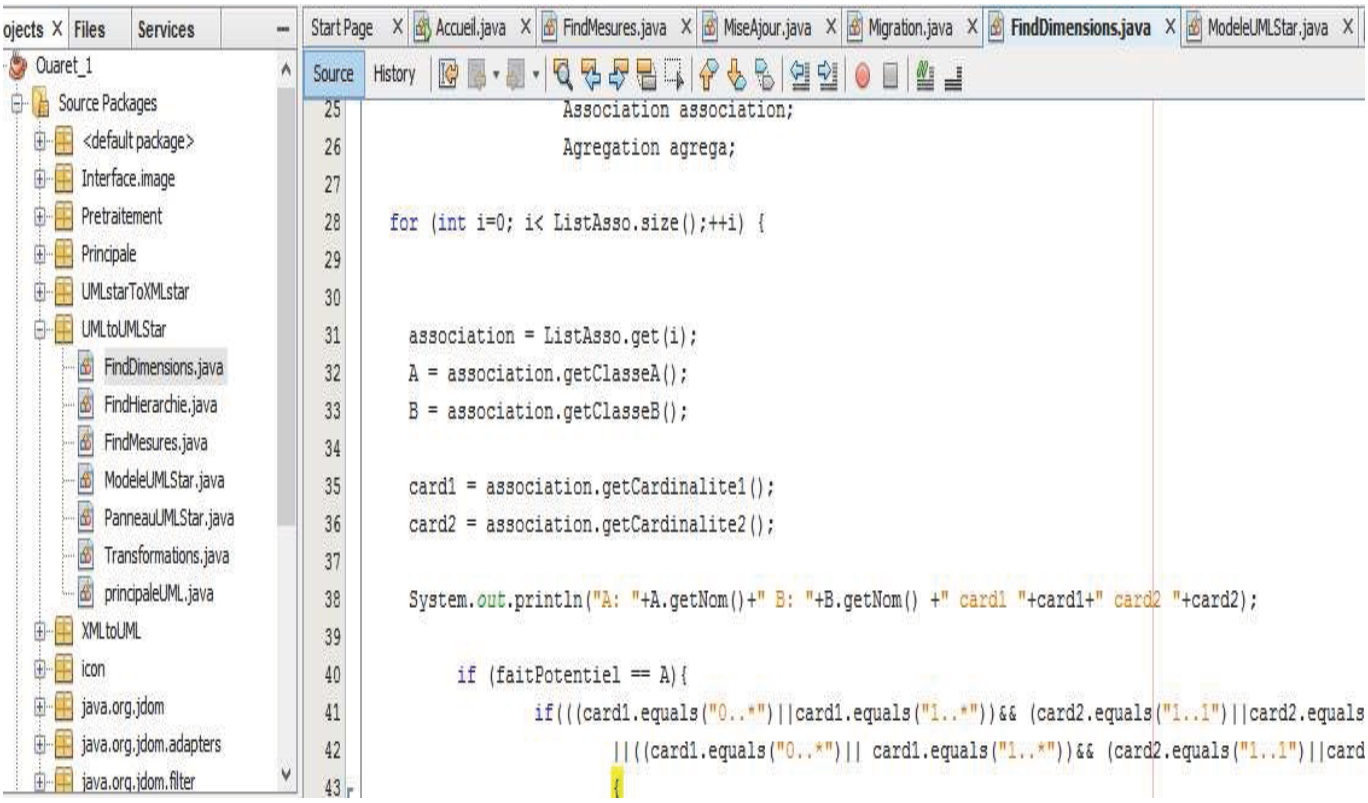
The prototype is composed of three main elements developed as set of graphical user interfaces (GUI), which itself are often composed of many smaller sub- modules

– UML Modelling interface: our tool generate automatically the UML class diagram from a XML schema and remove inconsistencies. This graphical interface allows designers to visualise each XML data sources in UML and then in compact form which is principally a graphical, visual language where as XML schema has essentially a complex textual presentation. In this part, we adopt reverse engineering technique and the best practices of UML design.

– Multidimensional Modelling interface: the system create automatically Star UML model from a UML class diagram generated in the first step. this a graphical interface allows designers to visualise, check and endorse the obtained Star schema. In this part, we adopt some traditional data warehouse design techniques.

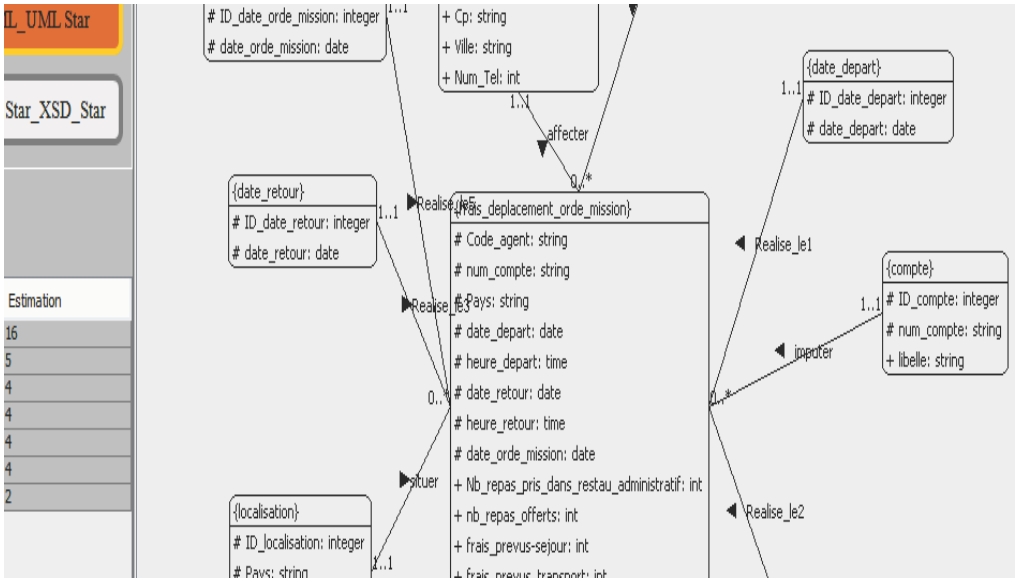
– Schema generation interface, the system generate automatically the XML schema (or XML data warehouse

1. Java-Editor with the XUML Star Tool main class file



schema), in which one part for describing fact and another part for describing dimension. This graphical interface allows designers to visualise then upload the generated Star XML schema to a native XML database, which must populated with XML data sources according to the . In this part, we adopt some traditional data warehouse design techniques. For example, following figure (Figure 9), we can see a screenshot from XUML Star tool that shows the obtained DW schema in UML model Star from XML schema.

2.The following figure illustrates the obtained Star Schema:



5. Conclusions and Future Work

In this paper, we introduced a new rule-based model transformation approach and an efficient tool to automatically design data warehouses starting from XML Schemas. We established and formalized relationship between the three levels of data warehouses (DW) using Query/Views/Transformation (QVT), which is the OMG standard language for specifying model transformations. We defined a set of transformation rules to generate UML class model from XML schemas model (a source model), and then automatic generation of multidimensional model (output star model) based on Well-formed transformation rules. The XML data warehouse schema description using XML schema language is derived automatically from UML star schema

An initial prototype called "*XUML Star Tool*" is developed based on our approach and conducted experiments on a large XML schema as data sources. Our prototype operates as a helpful support for creating an XML data warehouse conceptual and logical schema, with multiple options at each step. We believe that using our tool for XML data warehouse design may reduce design time and the risk of design errors.

In future work, we plan to extend the capabilities of our approach and the prototype tool to automatically generate multiple logical schema from multiple schema (DTDs, XSDs,...etc), with a new ranking function to select the most appropriate schema according to their relevance. Moreover, we plan to integrate user requirements in design process. Thus, a mixed approach for XML Data Warehouse design, that takes into account user requirements as well as data sources.

References

[1] Baril, X., Bellahsène, Z. (2003). Designing and Managing an XML Warehouse. XML Data Management: Native XML and XMLenabled Database Systems. Addison Wessley, p. 455-473.

[2] Boussaïd. O., BenMessaoud, R., Choquet, R., Anthoard, S. (2006). XWarehousing: an XML-Based Approach for Warehousing Complex Data, 10th East-Euro Conf on Advances in Databases and Information Systems.

[3] Dasgupta, S., Sen, S., Chaki, N. (2011). A Framework To Convert XML Schema to ROLAP; *In*: Proc. of the Second International Conference on Emerging Applications of Information Technology (EAIT 2011), Kolkata, India, ISBN : 978-1-4244-9683-9

[4] Gofarelli, M., Rizzi, S., Vrdoljak, B. (2001). Data Warehouse Design from XML Sources. 4th ACM Intl Workshop DOLAP. 40–47, Atlanta.

[5] Hümmer, W., Bauer, A., Harde, G. (2003). Xcube : Xml for data warehouses. DOLAP '03 : 6th ACM intl workshop on Data warehousing and OLAP, NY, USA, p. 33–40.

[6] Jensen, M. R., MÆller, T. H., Pedersen, T. B. (2001). Converting XML Data To UML Diagrams For Conceptual Data Integration, In Proc. The 1st Intl Workshop on Data Integration Over The Web, p. 17–31.

[7] Kimball, R. (1996). The data warehouse toolkit. Wiley.

[8] Kurze. C., Gluchowski. P., Computer-Aided Warehouse Engineering (CAWE): Leveraging MDA and ADM for the Development of Data Warehouses. AMCIS 2010:282

[9] Mahboubi, H., Ralaivao, J. C., Loudcher, S., Boussaid, O., Bentayeb, F., Darmont, J. (2009). X-WACoDa: An XML-based approach for Warehousing and Analyzing Complex Data, Advances in Data Warehousing and Mining, IGI Publishing, 2009.

[10] Mazon J. -N., Trujillo. J. (2008). An MDA approach for the development of data warehouses. Decis. Support Syst., 45 (1):41–58.

[11] Nassis, V., Rajugan, R., Dillon, R., Rahayu, W. (2004). Conceptual Design of XML Document Warehouses, In Proc 6th International Conference, DaWaK, pp. 1–14, Spain.

[12] Nassis, V., Rajugan, R., Dillon, T. S., Rahayu, J. W. (2005). Conceptual and Systematic Design Approach for XML Document Warehouses. Int Journal of Data Warehousing & Mining 1 (3), 63– 86.

[13] Niemi, T., Niinimaki, M., Nummenmaa, J., Thanisch, P. (2002). Constructing an OLAP cube from distributed XML data. In the 5th ACM Int workshop, DOLAP '02, pages 22–27, NY, USA, 2002.

[14] Ouaret. Z., Chalal. R., Boussaid, O., An overview of XML warehouse design approaches and techniques: Int. J. of Information and Coding Theory 2013 Vol.2, No.2/ 3.140 - 170

[15] Pardillo. J, Mazón J -N, Trujillo. J. (2001). An MDA Approach and QVT Transformations for the Integrated Development of Goal- Oriented Data Warehouses and Data Marts. J. Database Manag 22 (1):43-68 (2011)

[16] Pokorny, J. (2001). Modelling Stars Using XML, In Proc. The 4th ACM Int Workshop DOLAP, p. 24-31, , 2001.

[17] Park, B. -K., Han, H., Song. Il- Y. (2005). XML-OLAP: A Multidimensional Analysis Framework for MLWarehouses. DAWAK(2005) Springer Berlin / Heidelberg, p. 32-42

[18] Pedersen D. and. Pedersen T. B. Achieving adaptivity for OLAP-XML federations. In Proceedings of the DOLAP, pages 25– 32, 2003

[19] G. Pujolle, F. Ravat, O. Teste, R. Tournier, G. Zurfluh. Multidimensional Database Design from Document-Centric XML Documents: (DaWaK2011), Toulouse, Springer-Verlag, LNCS, p 51- 65.

[20] Perez, J. M., Berlanga, R., Aramburu, M. J., & Pedersen, T. B.. Integrating data warehouses with web data: A survey. Knowledge and Data Engineering, IEEE Transactions on, 20; 20(7), 940-955. (2008)

[21] Parimala, N., Payel pahwa (2009). From XML schema to cube International Journal of Computer Theory and Engineering Vol 1 No 3 August.

[22] Ravat, F., Teste, O., Tournier, R., Zurfluh, G. (2009). : Designing and Implementing OLAP Systems from XML Documents. New Trends in Data Warehousing and Data Analysis : Annals of Information Systems V3,p 1-21

[23] Ravat, F., Teste, O., Tournier, R., Zurfluh. G. (2010). Finding an Application-Appropriate Model for XML Data Warehouses. In, Information Systems, Elsevier, Vol. 36 N. 6, p. 662- 687.

[24] Rusu, L. I., Rahayu, W., Taniar, D. A. (2005). methodology for Building XML Data Warehouses, IJDM,

1 (2), p.67-92, - June 2005

[25] Rajugan, R., Chang, E., Dillon, T. S. (2005). Conceptual design of an XML FACT repository for dispersed XML document warehouses & XML marts, The 5th Intl Conf Computer and Information Technology, Page(s): 141 – 147.

[26] Sarkar, A., Choudhury, S., Debnath, N. C. (2012). Graph Semantic Based Design of XML Data Warehouse: A Conceptual Perspective", 10th IEEE (INDIN 2012), Beijing, China, P 992 – 997, July 25 – 27.

[27] Trujillo, J., Lujµan-Mora, S., et I. (2004). Song. Applying UML and XML for Designing and Interchanging Information for Data Warehouses and OLAP Applications. JDM 15 (1), 41-72.

[28] Vrdoljak, B., Banek, M., Rizzi, S. (2003). Designing Web Warehouses from XML Schema, Data Warehousing and Knowledge Discovery, 5th International Conference DaWak 2003, Prague, Czech Republic, Sept.3-5.

[29] Li, Y., An. A. (2005). Representing UML Snowflake Diagram from Integrating XML Data Using XML Schema. IEEE International Workshop on Data Engineering Issues in E-Commerce (DEEC '2005), Tokyo, Japan, p. 103–111.

[30] Zhang, Ji., Wei, Wang., Han, Liu., Sheng, Zhang. (2004). X-Warehouse : Building Query Pattern-driven Data ACM.

[31] Zepeda, L., Celma, M., Zatarain. R. (2008). A Mixed Approach for Data Warehouse Conceptual Design with MDA. In ICCSA, pages 1204–1217. Springer-Verlag.

[32] Object Management Group (OMG), QVT /MOF 2.0 Query/View/Transformation.http://www.omg.org/spec/MOF/2.0/PDF