IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITJ2884

Conceptual and Systematic Design Approach for XML Document Warehouses

Vicky Nassis, La Trobe University, Australia
Rajugan Rajagopalapillai, University of Technology, Australia
Tharam S. Dillon, University of Technology, Australia
Wenny Rahayu, La Trobe University, Australia

ABSTRACT

EXtensible Markup Language (XML) has emerged as the dominant standard in describing and exchanging data among heterogeneous data sources. The increasing presence of large volumes of data appearing creates the need to investigate XML Document Warehouses as a means of handling the data. In this paper our focus is twofold. First we utilise Object Oriented (OO) concepts to develop and propose a conceptual design formalism to build meaningful XML Document Warehouses (XDW). This includes: (1) XML (warehouse) repository (xFACT) using OO concepts followed by the transformation of XML Schema constructs and (2) Conceptual Virtual Dimensions (VDims) using Conceptual views (Rajugan, Chang, Dillon, & Feng, 2003, 2004). Secondly we address several important outstanding issues related to our proposed design of an XML Document Warehouse. Specifically we note that the xFACT portion is now a complex structure, involving several entities and relationships as opposed to being a simple FACT table as was the case in relational data warehouses, and the notion of Virtual Dimensions (VDims) has considerably greater complexity.

Keywords: *conceptual design; data modeling; data warehouse; database conceptual design; database design; database logical design; database requirements analysis; database views; information engineering; information requirements analysis; structural modeling; UML; XML*

INTRODUCTION

Data Warehousing (DW) has been an approach adopted for handling large volumes of historical data for detailed analysis and management support. Transactional data in different databases is cleaned, aligned and combined to produce data warehouses. Since its introduction in 1996, eXtensible Markup Language (XML) has become the *defacto* standard for storing and manipulating self-describing information, which creates vocabularies in assisting information exchange between heterogenous data sources over the Web (Pokorny, 2002). Amongst the purposes of

XML is to carry out electronic document handling, electronic storage, retrieval and exchange. It is envisaged that XML will also be used for logically encoding documents for many domains. Hence, it is likely that a large number of XML documents will populate the would-be repository and include several disparate transactional databases.

The need for managing large amounts of XML document data raises the necessity to explore the data warehouse approach through the use of XML document marts and XML document warehouses.

Since the introduction of dimensional modeling (which revolves around facts and dimensions), several design techniques have been proposed to capture multidimensional data (MD) at the conceptual level. Ralph Kimball's Star Schema (Kimball & Ross, 2002) proved very popular, from which the well-known conceptual models SnowFlake and StarFlake were derived. More recent comprehensive data warehouse design models are built using Object-Oriented concepts on the foundations of the Star Schema. In Trujillo, Palomar, Gomez and Song (2001), Lujan-Mora, Trujillo, and Song (2002), and Abello, Samos, and Saltor (2001), two different OO modeling approaches are demonstrated where a data cube is transformed into an OO model integrating class hierarchies. The Object-Relational Star Schema (O-R Star) model (Rahayu, Dillon, Mohammad, & Taniar, 2001) aims to envisage data models and their object features, focusing on hierarchical dimension presentation, differentiation and their different sorts of embedded hierarchies.

These models, both object and relational, have a number of drawbacks if one wishes to use them for XML document warehouses, namely: (1) data-oriented without sufficient emphasis or capturing user requirements, (2) extensions of semantically poor relational models (star, snowflake models), (3) original conceptual semantics are lost before building data warehouses as the operational data source is relational, (4) further loss of semantics results from oversimplified dimensional modeling, (5) time consuming if additional data semantics are required to satisfy evolving user requirements, and (6) complex query design and processing is needed, therefore maintenance is troublesome. Traditional design models lack the ability to utilise or represent XML design level constructs in a well-defined abstract and implementation-independent form.

One of the early XML data warehouse implementations includes the Xyleme Project (Xyleme, 2001). The Xyleme project was successful and it was made into a commercial product in 2002. It has a well-defined implementation architecture and proven techniques to collect and archive Web XML documents into an XML warehouse for further analysis. Another approach by Fankhauser and Klements (2003) explores some of the changes and challenges of a document centric XML warehouse. Coupling these approaches with a well defined conceptual and logical design methodology will help future design of such XML warehouse for large-scale XML systems. In this article we are concerned with the design of a data warehouse rather than the implementation of the XML document warehouse. We propose a conceptual modelling approach for the development of an XML Document Warehouse (XDW), while emphasising the design techniques to build the XDW conceptual model in UML. We also carry out a systematic transformation of this conceptual model into an XML Schema. An in-depth analysis for the design of the xFACT and the Virtual Dimension structures is illustrated using a walk-through with a case study example.

Our Work

UML, a widely adopted standard for Object-Oriented (OO) conceptual models, is the foundation to build this conceptual model for XML document warehousing. The main aspects of our methodology are as follows: (1) *User requirements* (*UR*; assist in determining different perspectives of the document warehouse), (2) *XML Document structure* (W3C Consortium, 2000): Using XML document capability in accommodating and explicitly describing heterogeneous data along with their interrelationships semantics, unlike flat-relational data), (3) *XML Schema:* Describes, validates, and provides semantics for its corresponding instance document; XML Document (W3C Consortium, 2001). Also, it can capture all OO concepts and relationships (Feng, Dillon, & Chang, 2002, 2003) as well as intuitive XML specific constructs such as ordering of components (see section *XML Schema and OO Concepts*), and (4) *Conceptual Views* (Rajugan et al., 2003, 2004; describes how a collection of XML tags contained in an XML document relates to the direct utilisation by a domain user at the conceptual/abstract level."

The rest of the article is organised as follows. In the section *XML Schema and OO Concepts*, we outline some unique XML Schema concepts that must be captured and modeled using the UML document data model. The section of the *XML Document Warehouse Model* is dedicated to in-depth discussion of the XDW conceptual model (with examples). The section of the case study example, *Conference Publication System (CPSys)*, provides a brief description of the example case study used in this paper, of which we will be gradually building the XML warehouse model (using UML). This is followed by a discussion, conclusion and future work.

XML Schema and OO Concepts

A widely used approach for conceptual modeling for developing XML Schemas for XML Data is the OO conceptual model, frequently expressed in UML (Feng et al., 2003). In order to understand the semantics that must be captured in abstract models of XML Data Warehouses, two significant issues must be noted, namely: ordered composition and homogeneous composition. An illustration of each case is provided in the sections that follow using the proposed mapping techniques in Feng et al. (2003) for the transformation of the OO diagrams to XML Schema. The examples are extracted from the complete UML diagram in Figure 10.

Ordered Composition

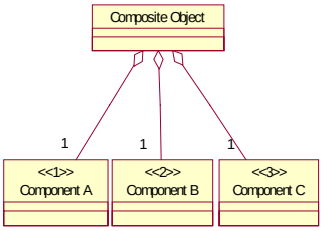
Consider the XML Schema fragment in Box A, modeled applying this UML notation. The composite element `Publ_Papers` is an aggregation of the sub-elements: `Publ_Title`, `Publ_Abstract`, and `Publ_References`. Interpreting this XML Schema fragment we observe that the tag `<xs:sequence>` signifies that the embedded elements are not only a simple assortment of components but these have a specific ordering. Hence we add to UML an annotation that allows capturing of the ordered composition as shown in Figure 1, utilising stereotypes to specify the objects' order of occurrence such as <<1>>, <<2>>, <<3>>,…,<<n>>. Figure 1(b) shows the XML Schema segment above modeled applying this UML notation.

Once we have the XML Schema component, we can generate the corresponding XML instance document (see Box B). This is used to check and verify the validity of the XML Schema document.

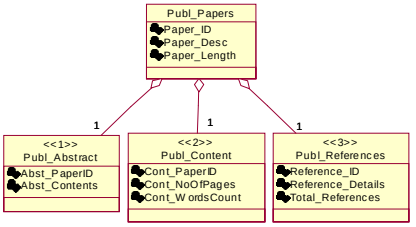
Box A. XML Schema fragment

```
<xs:element name="Publ_Papers" type=" Publ_PaperType" maxOccurs="unbounded"/>
<xs:complexType name="Publ_PaperType">
   <xs:sequence>
      <xs:element name="Paper_ID" type="xs:ID"/>
      <xs:element name="Paper_Desc" type="xs:string"/>
      <xs:element name="Paper_Type" type="xs:string"/>
         <xs:sequence>
            <xs:element name="Publ_Abstract" type="Publ_AbstractType"/>
            <xs:element name="Publ_Content" type="Publ_ContentType"/>
            <xs:element name="Publ_References"  type="Publ_ReferenceType"/>
         <xs:sequence>
      </xs:sequence>
</xs:complexType>
```

Figures 1(a) and Figure 1(b). Ordered composition example



a. UML Stereotype for an Ordered Composition



b. The XML Schema Fragment for an Ordered Composition shown in UML

Box B. XML instance document

```
<Publ_Papers>
     <Paper_ID>  131  </Paper_ID>
     <Paper_Desc>  Conceptual Design for XML Document Warehouses
     </Paper_Desc>
     <Paper_Type>   Long Journal Paper   </Paper_Type>
  <Publ_Abstract>
     <Abst_PaperID>  131  </Abst_PaperID>
     <Abst_Contents>   This is the abstract of the paper  </Abst_Contents>
  </Publ_Abstract>
  <Publ_Content>
     <Cont_PaperID>  131  </Cont_PaperID>
     <Cont_NoOfPages>  25  </Cont_NoOfPages>
     <Cont_WordsCount>  6,900  </Cont_WordCount>
  </Publ_Content>
  <Publ_References>
     <Reference_ID>  1  </Reference_ID>
     <Reference_Details>  These are the details of reference 1
     </Reference_Details>
     <Total_References>  30  </Total References>
  </Publ_References>
</Publ_Papers>
```

Homogeneous Composition

In a homogeneous aggregation, one "whole" object consists of "part" objects, which are of the same type (Rahayu & Chang, 2002). Two important cases are considered when applied to the homogenous composition, which are as follows:

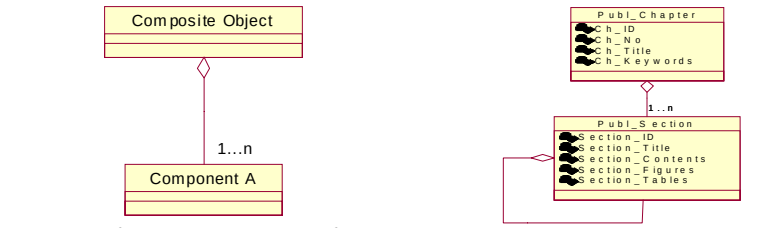
Case of One-to-Many Relationship

Consider the XML Schema segment in Box C of the relationship between the

Box C. XML Schema segment

```
<xs:element name="Publ_Chapter" type="Publ_ChapterType" maxOccurs="unbounded"/>
<xs:complexType name=" Publ_ChapterType">
   <xs:sequence>
      <xs:element name="Ch_ID" type="xs:ID"/>
      . . .
      . . .
         </xs:sequence>
            <xs:element name="Publ_Section" type="Publ_SectionType
maxOccurs="unbounded"/>
         </xs:sequence>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="Publ_SectionType">
   <xs:sequence>
      . . .
      . . .
      <xs:sequence>
        <xs:element name="Section_SubSection" type="Publ_SectionType"
maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:sequence>
</xs:complexType>
```

Figure 2(a) and Figure 2(b). Homogeneous composition (Case 1)



a. UML Notation for a Homogeneous Composition featuring One to Many relationship

b. Homogeneous Composition example

two elements `Publ_Chapter` and `Publ_Section`.

The object `Publ_Chapter` can consist of one or more `Publ_Sections`. We specify the *maxOccurs* value of `Publ_Section` to "unbounded" (default value is "1"), enabling the element `Publ_Section` to occur from one to many times within `Publ_Chapter`. We consider the assumption that a given section *cannot* be contained in any other chapter from the papers submitted. This characteristic is shown in Figure 2(a) using the proposed UML notation while Figure 2(b) shows the model corresponding to the XML Schema fragment in Box D.

In Box D is the equivalent XML document for the homogeneous composition with the one-to-many relationship.

Case of Many-to-Many Relationship

Consider the XML Schema segment in Box E of the relationship between `Publ_Year` and `Publ_Month` elements.

In this category a composite object (`Publ_Year`) may have many components (`Publ_Months`) and each component may belong to many composite objects. The *maxOccurs* value set to "unbounded" in both elements forms the many-

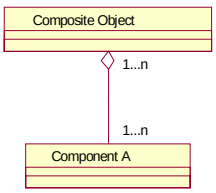
Box D. XML document

```
<Publ_Chapter>
     <Ch_ID>  1A  </Ch_ID>
     <Ch_No>  1  </Ch_No>
     <Ch_Title>  Introduction   </Ch_Title>
     <Ch_Keywords>  Design, Conceptual, Data Warehouse, XML   </Ch_Keywords>
  <Publ_Section>
     <Section_ID>  1.1  </Section_ID>
     <Section_Title>  This is the title of section 1.1 </Section_Title>
     <Section_Contents>  This is the content of section 1.1
     </Section_Contents>
     <Section_Figures>   Figure 1.1a, Figure 1.1b  </Section_Figures>
     <Section_Tables>   Table 1.1a, Table 1.1b  </Section_Tables>
  </Publ_Section>
  <Publ_Section>
     <Section_ID>  1.2  </Section_ID>
     <Section_Title>  This is the title of section 1.2 </Section_Title>
     <Section_Contents>  This is the content of section 1.2
     </Section_Contents>
     <Section_Figures>   Figure 1.2a, Figure 1.2b  </Section_Figures>
     <Section_Tables>   Table 1.2a, Table 1.2b  </Section_Tables>
  </Publ_Section>
</Publ_Chapter>
```

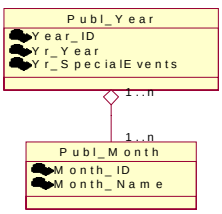
Box E. XML Schema statement

```
<xs:element name="Publ_Year" type="Publ_YearType" maxOccurs="unbounded"/>
<xs:complexType name="Publ_YearType">
      <xs:sequence>
         <xs:element name="Year_ID" type="xs:IDREF"/>
         . . .
            <xs:sequence>
               <xs:element name="Publ_Month" type="Publ_MonthType"
maxOccurs="unbounded"/>
            </xs:sequence>
      </xs:sequence>
</xs:complexType>
```

Figure 3(a) and Figure 3(b). Homogeneous composition (Case 2)



a. UML Notation for a Homogeneous Composition featuring Many to Many relationship



b. Homogeneous Composition example (n:m)

to-many relationship between `Publ_Year` and `Publ_Month` elements. The UML notation illustrating this case is shown in Figure 3(a) while Figure 3(b) shows the model corresponding to the XML Schema fragment in Box F.

In Box F is the XML document for the second case of homogeneous composition involving a many-to-many relationship.

Box F. XML document

```
<Publ_Year>
<xs:element name="Year_ID"/>
     <Year_ID>  3  </Year_ID>
     <Yr_Year>  2004  </Yr_Year>
     <Yr_SpecialEvents>  Conference  </Yr_Special_Events>
  <Publ_Month>
     <Month_ID>  9  </Month_ID>
     <Month_Name>  September  </Month_Name>
  </Publ_Month>
</Publ_Year>
```

PROPOSED MODEL FOR XML DOCUMENT WAREHOUSE (XDW)

The XDW model is shown in Figure 4. To our knowledge, this is unique in that it utilises XML itself (together with XML Schema) to provide: (1) structural constructs, (2) metadata, (3) validity, and (4) expressiveness (via refined granularity and class decompositions). The proposed model is composed of two levels: (1) User Requirement Level, which includes the warehouse user requirement document and OO requirement model, and (2) XML Warehouse Conceptual Level, composed of an XML FACT repository (section on XML FACT Repository, xFACT) and a collection of logically grouped *conceptual views* which employ the dimensions that satisfy captured warehouse user requirements.

User Requirement Level

The first level of the XDW model captures the warehouse end-user requirements. As opposed to the classical data warehouse models, this requirement model does not consider the transactional data as the focal point in the design of the warehouse. The XDW conceptual level data model is designed based on the user requirements. This level is further divided into two more sub-components, which are discussed next.

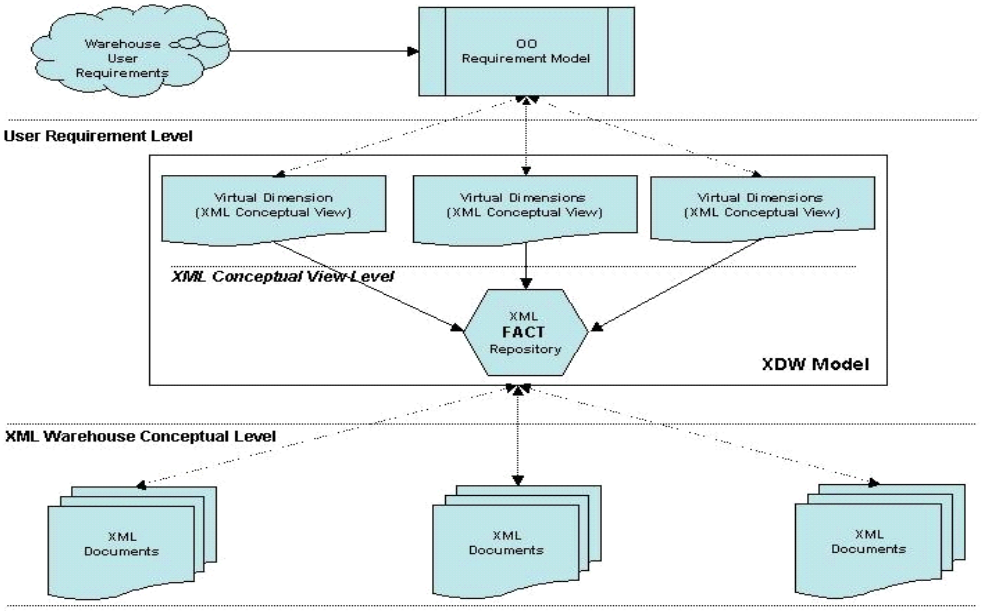
Warehouse User Requirements

The warehouse user requirements correspond to the written, non-technical outline of the XML warehouse. These are usually the typical or the predictable results expected of the XML Document Warehouse (XDW). Also, these user requirements can be further refined and/or enhanced by the participation of domain experts or the users of the transactional system in question. In addition, further refinement can be added to this model by using approaches that generate user requirement documents such as analysing frequent user query patterns (Zhang, Ling, Bruckner, & Tjoa, 2003) or other automated approaches.

OO Requirement Model

The OO requirement model transforms all non-technical user requirements into technical, software model specific concepts using UML (actors, use-case, and objects). The OO requirement model can be further refined through the involvement of domain experts, system analysts, programmers or the operators of the transactional system. This OO requirement model, by making more precise the representation of the requirements allows domain experts and users to move more effectively, evaluate their requirements and identifying requirements or misrepresentations. It helps XDW model developers in providing a less

Figure 4. XDW Context Diagram



ambiguous and more meaningful model on which to base the data models.

XML Data Warehouse Conceptual Level in UML

The process of deriving the XDW conceptual data model involves taking the formalised user requirements expressed in UML and then validating them against the XML transactional systems, from which the data in the XML data warehouse is assembled, to check for data availability. The case of lacking transactional data necessary to assemble certain dimensional or xFACT data highlights an ambiguous or non-achievable user requirement, which then has to be re-defined and/or modified. At the stage of transactional system maintenance, these will be considered as future user requirements to define what data needs to be collected.

The xFACT is a snapshot of the underlying transactional system(s) for a given *context*. A *context* is more than a measure (Golfarelli, Maio, & Rizzi, 1998; Trujillo, Palomar, Gomez, & Song, 2001) but instead is an item that is of interest for the organisation as a whole.

The role of conceptual views is to provide perspectives of the document hierarchy stored in xFACT repository. These can be grouped into logical groups, where each one is very similar to that of a given *subject area* (Dillon & Tan, 1993; Coad & Yourdon, 1991) appearing in Object-Oriented conceptual modeling techniques. Each subject-area in the XDW model is referred to as a *Virtual Dimension* (VDim) in accordance with the language of dimensional models. VDim is called *virtual* since it is modeled using a *conceptual view* (Rajugan et al., 2003) (which is an *imaginary* XML document) behaving as a dimension to a given perspective from the xFACT. The following sections discuss in detail the modeling of VDims, xFACT re-

pository and the issues likely to be encountered during this process.

XML FACT Repository (xFACT) and Meaningful FACT

In building the xFACT Repository, we note that it differs from a traditional relational data warehouse, which has a flat FACT table that is normally modeled as an ID packed FACT table with its associated data perspectives as dimensions. But, in regard to XML, a *context* refers to the presence of embedded declarative semantics and relationships including ordered and homogeneous compositions, associations and inheritance as well as non-relational constructs such as set, list, and bag (see Figure 10). Therefore, we argue that a plain FACT does not provide semantic constructs that are needed to accommodate an XML *context*.

At the logical level we utilize the XML Schema definition language to capture the xFACT semantics (i.e., classes, relationships, ordering and constraints). Therefore, modeling of the xFACT is constrained only by the availability of XML Schema elements and constructs. Below we introduce a case study example and provide an illustrative demonstration, which goes through the steps involved in the formation of our xFACT conceptual model.

Case Study: Conference Publication System (CPSys)

The main component of a conference publication comprises of a collection of papers. All the papers are documents originally written by one or more authors and which are then submitted to the appropriate publication chief editor for review. The structure of the manuscript has three components, namely: Abstract, Content *and* References. *In the review process, papers are distributed to two or four referees with expertise in the subject area. The given feedback will assist in the selection process of papers to be included in the proceedings book. A conference is not limited to one area of research and may encompass several areas, therefore workshops are formed to cover each or closely related subject topics. Usually the conference participants can originate from any part of world and belong to various internationally located institutes. The conference organisers' responsibility is to develop the schedule and inform the participants, as well as to ensure that events unfold in an orderly manner.*

Taking the above outlined points into consideration, we are able to grasp and represent these through a conceptual model using UML. Concentrating most importantly on the interaction amongst the objects will help determine their relationship configuration and cardinality, which will be reflected in the conceptual design. An examination of our complete UML conceptual model (Figure 10) emphasises the structural complexity of the xFACT in which real-world objects can be hierarchically decomposed into several sub-elements where each of these can include further embedded elements. Such decompositions are necessary to provide representation of a real-world object with appropriate granularity and if needed, additional semantics are added at different levels of the hierarchy. For example, the `Publ_Conference` class hierarchy is decomposed with additional semantics such as `Publ_Region`, `Publ_Country` and `Publ_City`. It is important to make the xFACT as expressive as possible whilst retaining the overriding objective of relevance.

Virtual Dimensions

A user requirement, which is captured in the OO Requirement Model, is trans-

formed into one or more *Conceptual Views* (Rajugan et al., 2003) also referred to as *Virtual Dimension(s)*, in association with the xFACT. These are typically *views* involving aggregation or *perspectives* of the underlying stored documents of the transaction system(s). A valid user requirement is such that it can be satisfied by one or more XML conceptual views for a given context (i.e., xFACT). But in the case where for a given user requirement there is no transactional document or data fragment to satisfy it, further enhancements are necessary to make the requirement feasible to model with a certain xFACT. Therefore modeling of VDims is an iterative process where user requirements are validated against the xFACT in conjunction with the transactional system. It might be simply the information required was present in the transactional systems but not included in the xFACT or alternatively it was not present in the transactional systems.

Definition 1: *One Virtual Dimension is composed of one (or more logically grouped) conceptual view, satisfying one (or more logically related) user document warehouse requirement(s)* (Rajugan et al., 2003, 2004).

We introduced a new UML stereotype called `<<VDim>>` to model the virtual dimensions at the XDW conceptual level (Nassis et al., 2004). This stereotype is similar to a UML class notation with a defined set of attributes and methods. The set of methods here can have either constructors (to construct a VDim) or manipulators (to manipulate the VDim attribute set).

In Figures 5 and 6, the relationship between the VDims is modeled with a dashed, directed line, denoting the `<<construct>>` stereotype. Though VDims can have additional semantic relationships such as generalisation, aggregation, and association (Rajugan et al., 2003, 2004), these can be shown using standard UML notation. In addition to this, two VDims can also have `<<construct>>` relationships with dependencies such as those shown in Figure 6, between VDims `Abstracts_By_Year` and `Abstracts_By_Keyword`.

Dimensions as UML Packages

We stated that semantically related conceptual views could be logically grouped together as grouping classes into a subject area. Further, a new view-hierarchy and/or constructs can be added to include additional semantics for a given user requirement. In the XDW conceptual model, when a collection of similar or related *conceptual views* are logically grouped together, we called it grouped *Virtual Dimension* (Figures 5-6), implying that it satisfies one or more logically related user requirement(s). In addition, we can also construct additional conceptual view hierarchies such as shown in Figures 5-6. These hierarchies may form additional structural or dependency relationships with existing conceptual views or view hierarchies (grouped VDims) as shown in Figures 7-8. Thus it is possible that a cluster of dimensional hierarchy(ies) can be used to model a certain set of user requirement(s). Therefore, we argue that this aggregate aspect can give us enough abstraction and flexibility to design a user-centred XDW model.

In order to model an XML view hierarchy or VDim and capture the logical grouping among them, we utilise the *package* construct in UML. According to OMG specification:

"A package is a grouping of model elements. Packages themselves may be nested within

Figure 5. A conceptual view hierarchy with <<construct>> stereotype

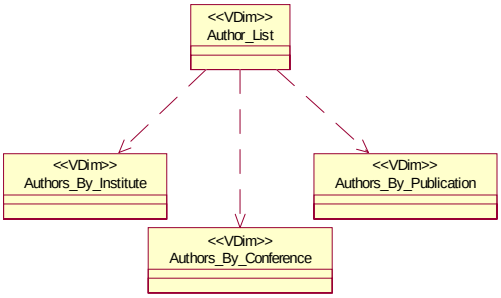


Figure 6. VDim "Abstract" package contents

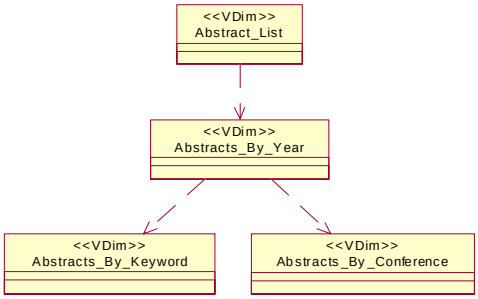
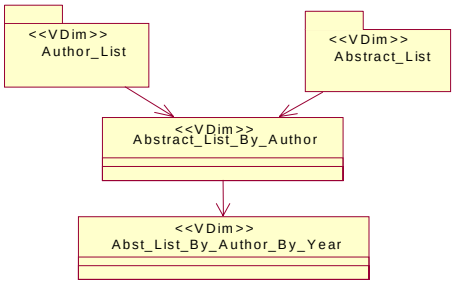


Figure 7. A VDim hierarchy (derived from grouped VDims)



other packages. A package may contain subordinate packages as well as other kinds of model elements. All kinds of UML model elements can be organized into packages" (OMG, 2003, Part 3, Section 3.13.1, p. 416 of 736).

The definition of Virtual Dimension here is a package. This in practice describes our logical grouping of XML conceptual views and their hierarchies. Thus we utilise packages to model our connected dimensions (Figure 9).

Following the arguments similar to these above, we can show that, the xFACT (shown in Figure 8) can be grouped into one logical construct and can be shown in UML as one package. In Figures 8 and 9, we show our case study XDW model with

Figure 8. "Conference_List", "Abstract_List" and "Author_List" packages

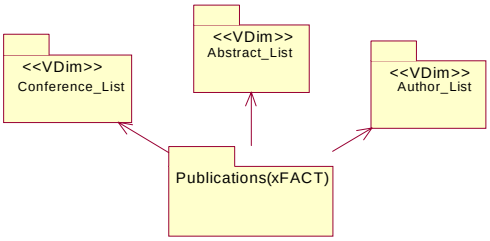
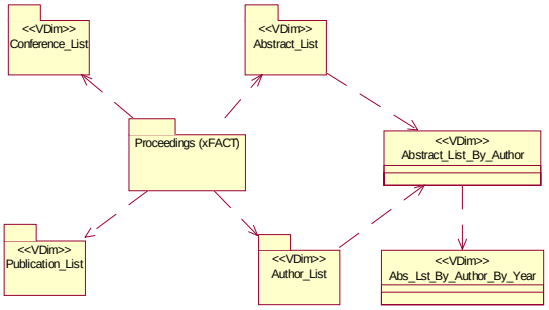


Figure 9. XDW Conceptual Model (in UML)



xFACT and VDims connected via `<<construct>>` stereotype.

Transformation of xFACT OO Conceptual Model to XML Schema

Using the generic rules (Feng et al., 2003), we are able to accomplish the systematic transformation of our OO conceptual model into XML Schema, which is the logical model. Initially we envisage that the xFACT table will be an entire major document of its own containing elements, further embedded elements, and relationships amongst these, which would translate into smaller complex or simple structured components.

We assume that class C, corresponds to the composite document (xFACT) containing additional classes $C_{1,...}C_n$. The steps to transform this into XML Schema segment are given next (Feng et al., 2003).

Step 1: Create an element named C with a ComplexType, *CType*

```
<xs:element name="C" type="CType"/>
```

Step 2: For each of the embedded class C_i of C, create a sub-element C_i with a ComplexType, C_i*Type*. `<xs:element name="C_i" type="C_iType"/>` Elements can also have a simple structure when they carry single valued attributes from the in-built data types of the XML Schema (e.g., string, integer). To apply these rules and illustrate this in more detail, we will refer to our OO conceptual model of xFACT (Figure 10). The real-world object `Publications` (root element) is hierarchically decomposed into `Publ_Conference` and `Publ_Papers`. This can be translated into XML Schema as shown in Box G.

The corresponding instance document for the XML Schema segment in Box G is shown in Box H.

Box G. XML Schema

```
Step 1
<xs:element name="Publications" type=" Publ_PublicationType "/>

Step 2
<xs:complexType name="Publ_PublicationType">
   <xs:sequence>
      <xs:element name="Publ_INDEX" type="xs:ID"/>
      . . .
      . . .
      <xs:sequence>
         <xs:element name="Publ_Conference" type="Publ_ConferenceType"
maxOccurs="unbounded"/>
         <xs:element name="Publ_Papers" type="Publ_PaperType"
maxOccurs="unbounded"/>
      </xs:sequence>
</xs:sequence>
</xs:complexType>
<xs:complexType name="Publ_ConferenceType">
   <xs:sequence>
      <xs:element name="Conf_Conference_ID" type="xs:ID"/>
      . . .
      . . .
            <xs:sequence>
               <xs:element name="Publ_Region" type="Publ_RegionType"/>
               <xs:element name="Publ_Schedule" type="Publ_ScheduleType"
   maxOccurs="unbounded"/>
            </xs:sequence>
   </xs:sequence>
</xs:complexType>
```

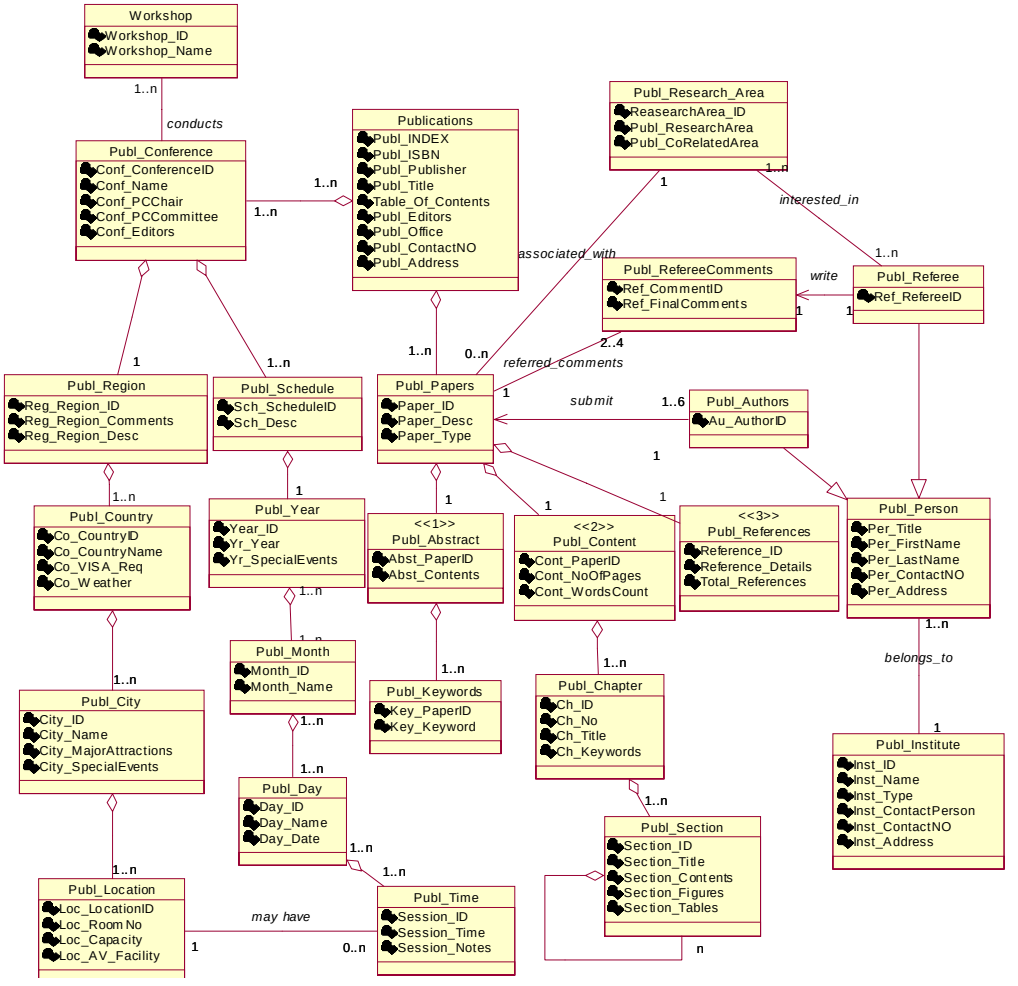
Box H. Instance document for XML Schema in Box G

```
<Publications>
     <Publ_INDEX>  This is the index  </Publ_INDEX>
     <Publ_ISBN>  This is the ISBN of the publication   </Publ_ISBN>
     <Publ_Publisher>  This is the publisher name </Publ_Publisher>
     <Publ_Title>  This is the publication title  </Publ_Title>
     <Table_Of_Contents>  This is the publisher name </Table_Of_Contents>
     <Publ_Editors>  Editor list of the publication </Publ_Editors>
     <Publ_Office>  Office name  </Publ_Office>
     <Publ_ContactNO>  +61-3-1234-5678  </Publ_ContactNO>
     <Publ_Address>  This is the address  </Publ_Address>
</Publication>
<Publ_Conference>
     <Conf_Conference_ID>  Identication number of conference
     </Conf_Conference_ID>



     <Conf_Name>  Name of conference   </Conf_Name>
     <Conf_PCChair>   Chair List  </Conf_PCChair>
     <Conf_PCCommitte>  Committee List  </Conf_PCCommitte>
     <Conf_Editors>  Editor list   </Conf_Editors>
</Publ_Conference>
```

Figure 10. The complete xFACT of the Conference Publication System (CPSys) case study



Another important relationship existing in the xFACT conceptual model is the generalisation relationship for which classes are categorised according to their differences and similarities. As seen from Figure 10, class `Publ_Person` is an abstract class, which creates instance classes such as `Publ_Author` and `Publ_Referee`. In Box I is the transformation of class `Publ_Person` (superclass) also showing the connection of `Per_BelongsTo` with class `Publ_Institute`.

In Box J is the transformation of the sub-class `Publ_Author` into XML Schema.

A child class can inherit all the properties from its parent class. In this case `Publ_Author` class has all the attributes of the super class and in addition `Au_AuthorID` is included, which is exclusive only in this class. The tag `<xs:extension base="Publ_PersonType">` shows that the current class created as an extension of an existing class. An instance of a corresponding XML document is given in Box K.

Box I. Transformation of class Publ_Person (superclass) also showing the connection of Per_BelongsTo with class Publ_Institute

```
<xs:element name="Publ_Person" type=" Publ_PersonType "/>
<xs:complexType name="Publ_PersonType">
   <xs:sequence>
      <xs:element name="Per_Title" type="xs:string"/>
      . . .
      . . .
      <xs:element name="Per_belongsTo">
         <xs:complexType>
            <xs:all>
               <xs:element name="Publ_Institute" type="Publ_InstituteType"/>
            </xs:all>
         </xs:complexType>
      </xs:element>
   </xs:sequence>
</xs:complexType>
```

Box J. Transformation of the sub-class Publ_Author into XML Schema

```
<xs:element name="Publ_Author" type=" Publ_AuthorType "/>
<xs:complexType name="Publ_AuthorType">
   <xs:complexContent>
      <xs:extension base="Publ_PersonType">
         <xs:sequence>
            <xs:element name="Au_AuthorID" type="xs:ID"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

Box K. XML document

```
<Publ_Author>
     <Per_Title>  Mrs.  </Per_Title>
     <Per_FirstName>  Helen    </Per_FirstName>
     <Per_LastName>   Smith  </Per_LastName>
     <Per_ContactNO>  +61-3-1234-3458 </Per_ContactNO>
     <Per_Address>  LaTrobe University, Bundoora, 3086, Australia
     </Per_Address>
     <Au_AuthorID>  12345  </Au_AuthorID>
  </Publ_Author>
```

User Requirements (URs) and the Systematic Querying Approach for Virtual Dimension/s (VDim/s)

Previously for the conversion of xFACT into XML Schema we used the transformations discussed in the papers by Feng et al. (2003) and Xiaou, Dillon, Chang, & Feng (2001). In the case of VDims, this is actually the transformation between the *conceptual views* into *XML View* (Rajugan et al., 2003, 2004) Schemas. An XML View is defined as:

Definition 2: *An XML View is an imaginary XML Document which points to a collection of semantically related XML tags from an XML domain and satisfies a Conceptual View definition from*

the target XML conceptual domain (Rajugan et al., 2004, p. 8).

Definition 2 indicates that for each conceptual view there is a corresponding XML document, which may contain more embedded XML documents. Querying the resulting XML document using any query language for XML with language specific syntax allows one to extract the information needed in order to meet all the conditions of each user requirement. The query process is well suited for simple, straightforward requirements but it tends to get very complex especially when dealing with user requirements having a wide-ranging context.

Logical Implementation of the Virtual Dimension

Considering the above facts regarding the query process, in the segment of queries that follow, the preferred query language is XQuery (W3C Consortium, 2004). However XQuery at this stage proves limited regarding "group by" and aggregate functions for data warehouse operations (need to use nested loops which are difficult to operate) and are still in progress for future development (Fankhauser et al., 2003). Due to this, and in order to illustrate the purpose of querying XML documents in relation to VDim, using the notion of XQuery, in this article we also propose a *generic query algorithm***,** which will be the foundation to build smaller algorithmic segments to suit the structure of the case queries from common cases and hence enable full capture of each user requirement.

We use the terms defined in W3C, namely `Query`, `Context`, and `Return` Context, extracted from the illustrated use case queries, in order to guide us in deriving and defining the main parameters to be used in our proposed algorithm. We consider that each class involved in the search query can be of simple or complex structure, meaning that it can contain only attributes, only elements and sometimes a combination of both. What follows is a list of the conditions and explanations of the *keywords* that will aid to design our *generic query algorithm*.

- **Keyword 1: `Query Context`:** This specifies the full path of classes used to locate the attributes and elements to be queried. Classes can have further embedded attributes and/or elements.
- **Keyword 2: `Query Context Value`:** This is the parameter value (specified by the user) used to execute the query and is located in the last occurring attribute or element from the class specified in the `query context` path. The value can be a precise word, phrase, number or include a group of values denoted in the query by the word `All`.
- **Keyword 3: `Return Context`:** The results are obtained having that all the query's conditions have been met. While the search is conducted within the class(es) specified in the `query context`, it is possible that the required attribute(s) or element(s) might originate from a different class. Also in some cases the outcome may be comprised of new assembled attribute(s) and/or element(s).
- **Keyword 4: `Sort`:** This function applies to the queries requiring the 'ordering' of resulting records. This is performed based on the `subject factor`**,** which is directly related to the user's need regarding the display of the query outcome. A `subject factor value` is defined by a name, a number, or it can be of any value provided it ex-

ists within the class' attribute(s) and/or element(s) involved in the query.

- **Keyword 5: `Merge`:** The merging function facilitates in combining separate query outcomes together. The `subject factor` as previously stated is used to sort records based on a value `type`. When records are ordered, it is likely that there may be more than one entry that belongs under the same `subject factor value`. In order to present the results in a more uniform layout, we `merge` the records occurring under the common subject factor value and remove duplicates.

Regardless of what each VDim aims to fulfil, we categorise four different types of VDims. A description and diagram provides illustration of each of the different types of VDims, namely: *Selection*, *Sorting*, *Implicit Join*, and *Explicit Join*.

Selection Virtual Dimension

This consists of selecting and extracting instance documents originating from one or more classes. The required instance documents correspond to what has been specified in the `query context value`, which can be an exact term or involve a group of values. The records obtained construct a new "partial" class, signifying that only a part of the original class(es) are required and therefore extracted. This is shown in Figure 11.

Sorting Virtual Dimension

In the Selection VDim the resultant document list would be displayed in random order, but the *Sorting VDim* requires for this to appear in a certain order. Common instance cases highlight the fact that sorting is to be done in alphabetical or chronological order. Figure 12 demonstrates that the resulting class A1 is now sorted.

Implicit Join Virtual Dimension

The concept of class hierarchical decomposition proves necessary to provide granularity to a real-world object. Therefore it is likely that a VDim will involve joining of two or more classes/elements, which may appear at different levels within a hierarchy. An *Implicit Join VDim* (Figure 13) applies when classes originating from different hierarchical paths and share a common parent class, are joined to form

Figure 11. Selection VDim

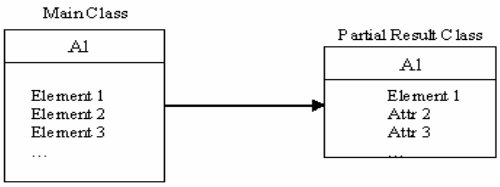


Figure 12. Sorting VDim

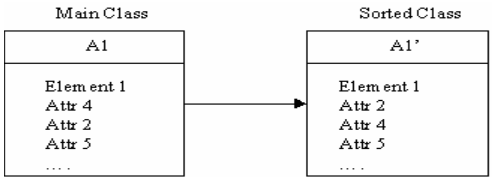


Figure 13. Implicit Join VDim

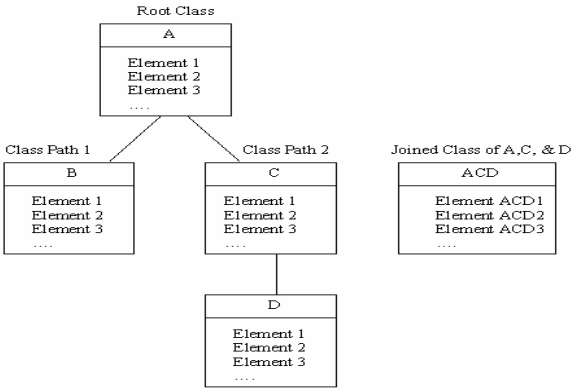
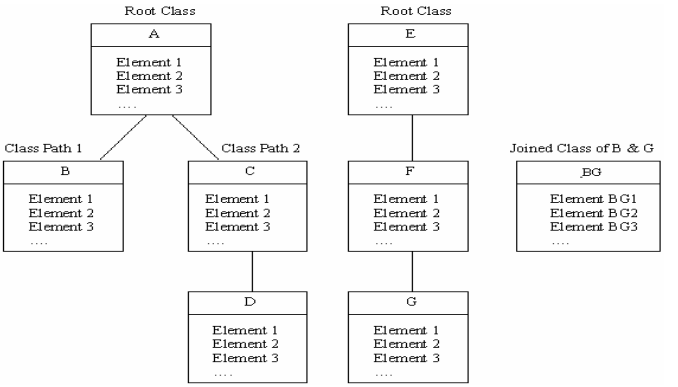


Figure 14. Explicit Join VDim



a combined class; in other words, the the process carried out is analogous to that of *Path Traversing* (Bertino, 1994).

Explicit Join Virtual Dimension

This type of VDim denotes that joining is not limited to occurring within the hierarchical paths originating from the same parent. Now it can also emerge from classes belonging to different source components, which are not directly related. For instance, in Figure 14 we are able to join the components B and G.

Using the *keywords* presented and the four types of *Virtual Dimensions* presented, we are able to construct a *generic query algorithm* as shown in Box L**,** encompassing all these existing components. Note that "[ ]" indicates that the contained function is optional.

Table 1 provides a full illustration of the algorithm's purpose by conveying its main capabilities. These have been applied to a sample set of queries based on our case study example, *Conference Publication System (CPSys)*, including the design and implementation of the most suited algorithm for each case. Each type of the Virtual Dimension so far discussed is also demonstrated.

The three main keywords; `Query Context`, `Query Context Value`, and `Return Context`, are treated as

Box L. Generic Query Algorithm

```
Get Query Context Value
For All | Each value/s within the attribute/s or element/s of the Query
Context path
   If the value is empty
   Go to the next value
[Check for exact match
  If there is no match
  Go to the next value]

Return Context attribute/s and/or element/s value/s
[Sort by Subject Factor Value]
[If the Subject Factor Value is a duplicate
  Merge entries under one common Subject Factor Value
  Delete duplicated Subject Factor Value
ELSE ( )]
```

mandatory for any algorithm designed and `Subject Factor Value` is applicable only when the functions `Sort` and `Merge` are required. The format of each case query, shown in Table 1, will be presented as follows: `Query Number`, `Query Name`, `Query Context`, `Query Context Value`, `Return Context`, and `Comments` (written explanation of the query highlighting the involved classes and the connection amongst these). Note that upper case letters signify classes and lower case attributes and elements. A full reference to each class's components and relationships involved in the queries is shown in Figure 10.

CONCLUSION AND FUTURE WORK

XML has become an increasingly important data format for storing structured and semi-structured text intended for dissemination and ultimate publication in a variety of media. It is a mark up language that supports user-defined tags and encourages the separation of document content from presentation. In this article, we present a coherent way to integrate a conceptual design methodology to build a native XML document warehouse. The proposed XML design methodology consists of user requirement level, warehouse conceptual level and XML Schema level, which capture warehouse user requirements and the warehouse model respectively. The only model not discussed in detail is the OO user requirement model, as this is part of our impending work.

We also investigated a number of the issues in the conceptual design of XML data warehouses rather than dealing with its implementation. We used existing generic rules (Feng et al., 2003) to develop the formal steps and demonstrate the transformation of the xFACT conceptual model into XML Schema. Simultaneously we highlighted two cases of semantic involvement within the xFACT, namely *ordered* and *homogeneous* compositions. The *User Requirements (URs)* are important components and aimed to be fully captured at the design level. We proposed a systematic querying approach to access the data warehouse considering the need to accommodate XML semantics such as aggregate functions. Also, we formally defined four

Table 1. Illustration of VDims logical implementation

VDim Type & Requirement	Query Context	Query Context Value	Subject Factor Value	Return Context	Comments	Query Algorithm
Selection VDim on an Exact Value **Author list based on given institute name**	./PUBL_PAPERS /PUBL_AUTHORS /PUBL_INSTITUTE /Inst_Name	La Trobe University	N/A	./PUBL_PAPERS /PUBL_AUTHORS/	It is required for all authors' details to be listed according to their associated institute, which is "La Trobe University." A simple association between the two classes Publ_Authors and Publ_Institute enables one to extract the required data.	*Get* **La Trobe University** *For* **each** *value of the element* **Inst_Name** *If the value is empty* *Go to the next value* *Check for* **exact match** *If there is no match* *Go to the next value* **Return Context** values of **all** the elements in the **PUBL_AUTHORS** class
Selection VDim on a Group of Values **List of conference names**	./PUBL_CONFERENCE /Conf_Name	All	N/A	./PUBL_CONFERENCE /Conf_Name	In order to generate the required list of all the conferences names, the proposed query would perform a search within the class Publ_Conference and more specifically in the element Conf_Name. A group of values is included, as opposed in the previous case checking for a match based on a specific term value.	*Get* **All** *values of element* **Conf_Name** *For* **All** *values of the element* **Conf_Name** *IF the value is empty* *Go to the next value* **Return Context** *all values in the element* **Conf_Name**
Implicit Join VDim **Alphabetical list of conference names**	./PUBL_CONFERENCE /Conf_Name	ALL	Conference Name	./**Publ_Conference_Sort_By_Name**	In the previous algorithms, the list of results would be shown in a random order. It is now required for the results to be presented in a specific order, based on a subject factor value (alphabetical) specified by the user. In this case the result is the newly formed element Publ_Conference_Sort_By_Name.	*Get* **All** *values of element* **Conf_Name** *For* **All** *values of the element* **Conf_Name** *IF the value is empty* *Go to the next value* **Return Context** *of all values of the element* **Conf_Name** **Sort Conference Name** *list in alphabetical order*

Table 1. Illustration of VDims logical implementation (continued)

Implicit Join VDim **Chronological list of conferences**	./PUBL_CONFERENCE /Conf_Name ./PUBL_CONFERENCE /PUBL_SCHEDULE /PUBL_YEAR /Yr_Year	All	Year	./PUBL_CONFERENCE /Conf_Name ./PUBL_CONFERENCE /PUBL_SCHEDULE /PUBL_YEAR /Yr_Year **./Publ_Conference _Sort_By_Year**	The outcome list is likely to have more than one conference name occurring under the same `year`, which means having many instance documents under the same `subject factor value`. Applying the **merge** algorithm ensures that all conferences taking place in the same year are to be grouped and appear under one common `year` heading.	*Get* ***All*** *values of elements* ***Conf_Name*** *and* ***Yr_Year*** *For* ***All*** *values of the elements* ***Conf_Name*** *and* ***Yr_Year*** *If the value is empty* *Go to the next value* ***Return Context*** *of all values in elements* ***Conf_Name*** *and* ***Yr_Year*** ***Sort Conf_Name*** *by* ***Year*** *If the* ***Year*** *value is a duplicate* ***Merge*** *entries under one common* ***Year*** *Value* ***Delete*** *duplicate* ***Year*** *Value* Else ( )
Explicit Join VDim **Chronological listing of conference names with cross reference to author details**	./PUBL_CONFERENCE /Conf_Name ./PUBL_CONFERENCE /PUBL_SCHEDULE /PUBL_YEAR /Yr_Year	All	Year	./PUBL_CONFERENCE /Conf_Name ./PUBL_CONFERENCE /PUBL_SCHEDULE /PUBL_YEAR /Yr_Year ./PUBL_PAPERS /PUBL_AUTHORS **./Publ_Conference _Sort_By_Year**	A chronological listing is again required with the distinction to include cross-referencing from the class `Publ_Authors`. This shows that the return context is not bound to be within the classes that have internal direct connections. Instead at times it may be necessary to go through several intra-connections with external objects to obtain the required values. Therefore the algorithm including the ***sort*** and ***merge*** functions is applied.	*Get* ***All*** *values of elements* ***Conf_Name*** *and* ***Yr_Year*** *For* ***All*** *values of the elements* ***Conf_Name*** *and* ***Yr_Year*** *If the value is empty* *Go to the next value* ***Return Context*** *of all values in elements* ***Conf_Name***, ***Yr_Year*** *and all values in class* ***Publ_Authors*** ***Sort Conf_Name*** *by* ***Year*** *If the* ***Year*** *value is a duplicate* ***Merge*** *entries under one common* ***Year*** *Value* ***Delete*** *duplicate* ***Year*** *Value* *Else ( )*

different types of Virtual Dimensions at the conceptual level.

For future work, the following subject matter deserve investigation: (1) development of formal semantics to automate mapping between XML Data and XDW Schema where the view is defined more precisely, (2) incremental update of materialised views, (3) derivation of user requirements by investigating frequent path queries (Zhang et al., 2003), and (4) investigation of performance issues upon query execution in relation to accessing the data warehouse.

REFERENCES

Abelló, A., Samos, J., & Saltor, F. (2001). Understanding facts in a multidimensional object-oriented model. In *Proceedings of the Fourth International Workshop on Data Warehousing and OLAP (DOLAP)*, Altlanta, November, (pp. 32-39) .

Bertino, E. (1994). A survey on indexing techniques for object-oriented databases. In Freytag et al. (Eds.), *Query Processing for Advanced Database Systems*. Morgan Kaufmann Publisher.

Coad, P., & Yourdon, E. (1991). *Object-oriented analysis*. Australia: Prentice Hall.

Dillon, T.S., & Tan P.L. (1993). *Object-oriented conceptual modeling*. Australia: Prentice Hall.

Elmasri, R., & Navathe, S.B. (2000). *Fundamentals of database systems* (3rd Ed.). Addison-Wesley.

Fankhauser, P., & Klement, T. (2003). XML for data warehousing changes & challenges. In *Data Warehousing and Knowledge Discovery, Proceedings of the Fifth International Conference (DaWaK 2003)*, Prague, Czech Republic, September 3-5, *Lecture Notes in Computer Science*, (Vol. 2737, pp. 1-3). Berlin: Springer-Verlag.

Feng, L., Dillon, T., & Chang. E. (2002). A semantic network based design methodology for XML documents. *ACM Transactions on Information Systems, 20*(4), 390-421.

Feng, L., Chang, E., & Dillon, T. (2003). Schemata transformation of object-oriented conceptual models to XML. *International Journal of Computer Systems Engineering* (CSSE), *18*(1), 45-60.

Golfarelli, M., Maio, D., & Rizzi, S. (1998). The dimensional fact model: A conceptual model for data warehouses. *International Journal of Cooperative Information Systems* (Invited proceeding), 7-2-3.

Kimball, R., & Ross, M. (2002). *The data warehouse toolkit: The complete guide to dimensional modeling* (2nd Ed.). Wiley Computer Publishing.

Lujan-Mora, S., Trujillo, J., & Song, I-Y. (2002). Extending the UML for multidimensional modeling. In *UML 2002: The Unified Modeling Language, Proceedings of the Fifth International Conference*, Dresden, Germany, September 30-October 4, (pp. 290-304). *Lecture Notes in Computer Science* (Vol. 2460). Berlin: Springer-Verlag.

Lujan-Mora, S., Trujillo, J., & Song, I-Y. (2002). Multidimensional modeling with UML package diagrams. In *Conceptual Modeling: ER 2002, Proceedings of the 21st International Conference on Conceptual Modeling,* Tampere, Finland, October 7-11, *Lecture Notes in Computer Science* (Vol. 2503, pp. 199-213). Berlin: Springer-Verlag.

Nassis, V., Rajugan, R., Dillon, T.S., & Rahayu, W. (2004). Conceptual design of XML document warehouses. In *Proceedings of the Sixth International Conference on Data Warehousing and Knowledge Discovery (DaWak 2004),* Zaragoza, Spain, September 1-3, (pp. 1-14).

Nassis, V., Rajugan, R., Dillon, T.S., & Rahayu, W. (2005). A systematic design approach for XML-view driven Web document warehouses. *International Workshop on Ubiquitous Web Systems and Intelligence (UWSI '05),* Singapore, May 9-12, (pp. 914-924).

Object Management Group. (n.d.). *Data warehousing, CWM™and MOF™resource page.* Retrieved from *http://www.omg.org/cwm/*

Object Management Group. (2003, March). *Unified modeling language specification.* Version 1.5 formal/03-03-01. Retrieved from *http://www.omg.org/technology/documents/formal/uml.htm*

Pokorny, J. (2002). *XML data warehouse: Modelling and querying.* The Netherlands: Kluwer Academic Publishers.

Rahayu, W.J. et al. (2002). Aggregation versus association in object modeling and databases. In *Proceedings of the ACS Conference on IS,* (pp. pp 521 – 532). Australian Computer Society Inc..

Rahayu, W.J., Dillon, T.S., Mohammad, S., & Taniar, D. (2001). Object-relational star schemas. *The Thirteenth IASTED International Conference Parallel and Distributed Computing and Systems.* Anaheim, California: IASTED/ACTA Press.

Rajugan, R., Chang, E., Dillon, T.S., & Feng, L. (2003). XML views: Part I. In *Proceedings of the 14th International Conference on Database and Expert Systems Applications (DEXA 2003),* Prague, Czech Republic, September 1-5, (pp. 148-159).

Rajugan, R., Chang, E., Dillon, T.S., & Feng, L. (2004). *XML Views, Part II: Modelling Conceptual Views Using XSemantic Nets. Workshop & Special Session on Industrial Informatics, The 30th Annual Conference of the IEEE Industrial Electronics Society (IECON '04),* South Korea, November.

Trujillo, J., Palomar, M., Gomez, J., & Song, I-Y. (2001). Designing data warehouses with OO conceptual models. *Computer, IEEE Computer Society,* (12), 66-75.

W3C Consortium. (2000). *EXtensible Markup Language*s. Retrieved from *http://www.w3.org/XML/*

W3C Consortium. (2001). *XML schema.* Retrieved from *http://www.w3c.org/XML/Schema*

W3C Consortium. (2003). *XML Query (XQuery) requirements.* Retrieved from *http://www.w3.org/TR/xquery-requirements/*

W3C Consortium. (2004). *XQuery1.0 and XPath 2.0 Full-text specification.* Retrieved from *http://www.w3.org/TR/2004/WD-xquery-full-text-20040709/*

W3C Consortium. (2004).*XQuery1.0 and XPath 2.0 Full-text use cases.* Retrieved from *http://www.w3.org/TR/2004/WD-xmlquery-full-text-use-cases-20040709/*

Xiaou, R., Dillon, T.S., Chang, E., & Feng, L. (2001). *Modeling and transformation of object-oriented conceptual models into XML schema.* In *Proceedings of the 12th International Workshop on Database and Expert Systems Applications*

(DEXA), Munich, Germany, September 3-7, (pp. 795-804).

Xyleme Project. (n.d.). *Publications Web site*. Retrieved from *http://www.xyleme.com/*

Xyleme, L. (2001). A dynamic warehouse for XML data of the Web. *IEEE Data Engineering Bulletin, 24*(2), 28, 40-47.

Zhang, J., Tok, W.L., Bruckner, R.M., & Tjoa, A M. (2003). Building XML data warehouse based on frequent patterns in user queries. In *Data Warehousing and Knowledge Discovery, Proceedings of the Fifth International Conference (DaWaK 2003),* Prague, Czech Republic, September 3-5, *Lecture Notes in Computer Science* (Vol. 2737, pp. 99-108). Berlin: Springer-Verlag.

Miss Vicky Nassis received her Bachelor of Information Systems and Bachelor of Business from La Trobe University and is currently a PhD student within the school of Computer Science and Computer Engineering at the same university. Her research interests include object-oriented conceptual models, XML and data warehousing. Her thesis is concerned with the conceptual design of data warehouses in integration with XML (eXtreme Markup Language). She has been examining techniques to overcome some of the problems encountered during the conceptual design process. She has also been involved in exploring the issues of XML and its use in handling publications over the web. She has had publications in the proceedings of international conferences, in the field of data warehousing, knowledge discovery as well as in web systems and intelligence.

Mr. Rajugan Rajagopalapillai holds a bachelor's degree in information systems (BInfoSys) from La Trobe University, Australia. He has worked in the industry as chief application/database programmer in developing sports planing and sports fitness & injury management software and as database administrator. He was also involved in developing an e-commerce solution for a global logistics (logistics, cold-storage and warehousing) company as a software engineer/architect. He is currently a PhD student at University of Technology, Sydney (UTS) Australia. He has published research articles which have appeared in international refereed conference and journal proceedings. His research interests include object-oriented conceptual models, XML, data warehousing, software engineering, database and e-commerce systems. He is a member of the IEEE, a member of the ACM and an associate member (AACS) of the Australian Computer Society.

Tharam S. Dillon is the dean of the Faculty of Information Technology at the University of Technology, Sydney (UTS). His research interests include data mining, internet computing, e-commerce, hybrid neuro-symbolic systems, neural nets, software engineering, database systems and computer networks. He has also worked with industry and commerce in developing systems in telecommunications, health care systems, e-commerce, logistics, power systems, and banking and finance. He is editor-in-chief of the International Journal of Computer Systems Science and Engineering *and the* International Journal of Engineering Intelligent Systems, *as well as co-editor of the* Journal of Electric Power and Energy Systems. *He is on the advisory editorial board of* Applied Intelligence *(Kluwer, US) and* Computer Communications *(Elservier, UK). He has published more than 400 papers in international and national journals and conferences and has written four books and edited five other books. He is a fellow of the IEEE, fellow of the Institution of Engineers (Australia), and fellow of the Australian Computer Society.*

Dr. Rahayu is an associate professor at the Department of Computer Science and Computer Engineering, LaTrobe University, Australia. She has been actively working in the areas of database design and implementation covering object-relational databases, Web and e-commerce databases, semi-structured databases and XML, Semantic Web and ontology, data warehousing, and parallel databases. She has worked with industry and expertise from other disciplines in a number of projects including bioinformatics databases, parallel data mining, and e-commerce catalogues. She has been invited to give seminars in the area of e-commerce databases in a number of international institutions. She has edited three books which forms a series in web applications, including Web databases, Web information systems and Web Semantics. She has also published over 70 papers in international journals and conferences.