A Web-Oriented Approach to Manage Multidimensional Models through XML Schemas and XSLT*

Sergio Luján-Mora, Enrique Medina, and Juan Trujillo

Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante. SPAIN
{slujan,emedina,jtrujillo}@dlsi.ua.es

Abstract. Multidimensional (MD) modeling is the foundation of data warehouses, MD databases, and OLAP applications. In the last years, there have been some proposals to represent MD properties at the conceptual level. In this paper, we present how to manage the representation, manipulation, and presentation of MD models on the web by means of eXtensible Stylesheet Language Transformations (XSLT). First, we use eXtensible Markup Language (XML) to consider main MD modeling properties at the conceptual level. Next, an XML Schema allows us to generate valid XML documents that represent MD models. Finally, we provide XSLT stylesheets that allow us to automatically generate HTML pages from XML documents, thereby supporting different presentations of the same MD model easily. A CASE tool that gives support to all theoretical issues presented in the paper has been developed.

Keywords: Multidimensional modeling, UML, XML, XML Schema, XSLT

1 Introduction

Multidimensional (MD) modeling is the foundation of data warehouses, MD databases, and OLAP applications. These systems provide companies with huge historical information for the decision making process. Various approaches for the conceptual design of MD systems have been proposed in the last few years [1][2][3,4][5]. Due to space constraints, we refer the reader to [6] for detailed comparison and discussion about these models.

On the other hand, a salient issue nowadays in the scientific community and in the business world is the interchange of information. Therefore, a relevant feature of a model should be its capability to share information in an easy and standard form. The eXtensible Markup Language (XML) [7] is rapidly being adopted as a specific standard syntax for the interchange of semi-structured data. Furthermore, XML is an open neutral platform and vendor independent

* This paper has been supported by the Spanish Ministery of Science and Technology, project number TIC2001-3530-C02-02.

meta-language standard, which allows to reduce the cost, complexity, and effort required in integrating data within and between enterprises. However, one common feature of the semi-structured data is the lack of schema, so the data is describing itself.

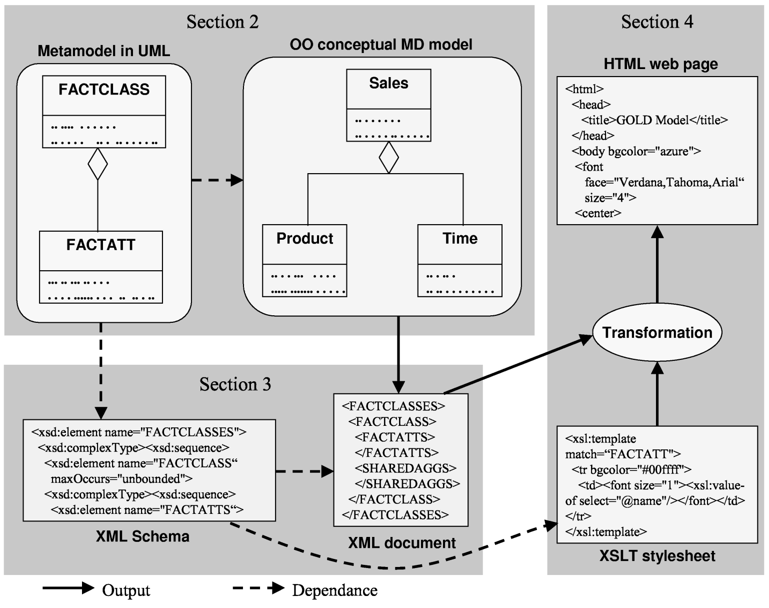
Nevertheless, XML documents can be associated to a Document Type Definition (DTD) or an XML Schema [8], both of which allow us to describe and constraint the structure of XML documents. In this way, an XML document can be validated against these DTDs or XML Schemas to check its correctness. Moreover, thanks to the use of eXtensible Stylesheet Language Transformations (XSLT) [9], users can express their intentions about how XML documents should be presented, so they could be automatically transformed into other formats, e.g. HTML documents. An immediate consequence is that we can define different XSLT stylesheets to provide different presentations of the same XML document.

In this paper, we present how to manage the representation, manipulation, and presentation of the same conceptual MD model by means of XSLT. The conceptual modeling is accomplished by the Object-Oriented (OO) approach presented in [3,4], based on the Unified Modeling Language (UML) [10], as it easily considers main MD properties at the conceptual level such as the many-to-many relationships between facts and dimensions, degenerate dimensions, multiple and alternative path classification hierarchies, or non-strict and complete hierarchies[1]. Each one of these MD properties is represented in a XML Schema. This XML Schema is then used to automatically validate XML documents, so any external application could benefit from the expressiveness of the conceptual MD approach. Furthermore, we use XSLT stylesheets and XML documents in a transformation process to automatically generate HTML pages that can represent different presentations of the same MD model. As an example of the applicability of our proposal, these HTML pages can be used to document the MD models in the web, with the advantages that it implies (standardization, access from any computer with a browser, ease of use, etc.). Moreover, the automatic generation of documentation from conceptual models avoids the problem of documentation out of date (incoherences, features not reflected in the documentation, etc.).

In this context, several proposals have been presented with respect to MD modeling and XML support. All of these proposals make use of XML as the base language for describing data. In [11], an innovative data structure called an XML-star schema is presented with explicit dimension hierarchies using DTDs that describe the structure of the objects permitted in XML data. Another approach is [12], where they propose a semi-automatic approach for building the conceptual schema for a data mart starting from the XML sources. With regard to the presentation of the information, [13] discusses the presentation of multidimensional XML data through multidimensional XSLT stylesheets, thereby allowing the user to view different variants of the same document. To the best of our knowledge, the proposal that comes closest to our goal is [13]. However,

[1] This model is supported by a CASE tool that allows us to semi-automatically generate the implementation of a MD model into a target commercial OLAP tool.

the latter approach focus on the presentation of the multidimensional XML data rather than on the presentation of the structure of the multidimensional conceptual model itself. Furthermore, we use a very recent technology (XML Schemas) which has not been used yet in any of the above-commented approaches.



```
<xsd:element name="FACTCLASSES">
  <xsd:complexType><xsd:sequence>
    <xsd:element name="FACTCLASS“
      maxOccurs="unbounded">
    <xsd:complexType><xsd:sequence>
      <xsd:element name="FACTATTS“>
```

```
<FACTCLASSES>
  <FACTCLASS>
    <FACTATTS>
    </FACTATTS>
    <SHAREDAGGS>
    </SHAREDAGGS>
  </FACTCLASS>
</FACTCLASSES>
```

```
<html>
  <head>
    <title>GOLD Model</title>
  </head>
  <body bgcolor="azure">
   <font
     face="Verdana,Tahoma,Arial“
     size="4">
   <center>
```

```
<xsl:template
match=“FACTATT">
  <tr bgcolor="#00ffff">
    <td><font size="1"><xsl:value-
of select="@name"/></font></td>
</tr>
</xsl:template>
```

Fig. 1. Structure of sections of this paper.

Following these considerations, the remainder of this paper is structured in Fig. 1 as follows: Section 2 describes the basis of the OO conceptual MD modeling approach that this paper is based on. Section 3 presents the XML Schema created from the metamodel of the MD model that will allow us to validate XML documents that store instances of the conceptual model. Then, Section 4 shows how to use XSLT stylesheets to automatically generate HTML pages from XML documents, thereby allowing us to manage different presentations of the model in the web. In Section 5 we present our conclusions and benefits of our proposal. Finally, future works that are currently being considered are presented in Section 6.

2 Conceptual multidimensional modeling

In this section, we will summarize how the conceptual MD modeling approach followed in this paper [3,4] represents both the structural and dynamic parts of MD modeling. In this approach, main MD modeling structural properties are specified by means of a UML class diagram in which the information is clearly separated into facts and dimensions.

Dimensions and facts are considered by *dimension classes* and *fact classes* respectively. Then, fact classes are specified as composite classes in shared aggregation relationships of n dimension classes. Thanks to the flexibility of shared aggregation relationships that UML provides, *many-to-many* relationships between facts and particular dimensions can be considered by indicating the 1..* cardinality on the dimension class role.

By default, all measures in the fact class are considered additive. For nonadditive measures, additive rules are defined as constraints and are also placed in somewhere around the fact class. Furthermore, derived measures can also be explicitly considered (indicated by /) and their derivation rules are placed between braces in somewhere around the fact class.

This OO approach also allows us to define identifying attributes in the fact class, if convenient, by placing the constraint *{OID}* next to a measure name. In this way we can represent degenerate dimensions [14][15], thereby providing other fact features in addition to the measures for analysis. For example, we could store the ticket and line numbers as other ticket features in a fact representing sales tickets.

With respect to dimensions, every *classification hierarchy* level is specified by a class (called a base class). An association of classes specifies the relationships between two levels of a classification hierarchy. The only prerequisite is that these classes must define a Directed Acyclic Graph (DAG) rooted in the dimension class (constraint *{dag}* placed next to every dimension class. The DAG structure can represent both alternative path and multiple classification hierarchies. Every classification hierarchy level must have an *identifying* attribute (constraint *{OID}*) and a *descriptor* attribute (constraint *{D}*). These attributes are necessary for an automatic generation process into commercial OLAP tools, as these tools store this information in their metadata. The multiplicity *1* and *1..** defined in the target associated class role addresses the concepts of *strictness* and *non-strictness*. In addition, defining the *{completeness}* constraint in the target associated class role addresses the completeness of a classification hierarchy. This approach considers all classification hierarchies non-complete by default.

The *categorization of dimensions*, used to model additional features for an entity's subtypes, is considered by means of generalization-specialization relationships. However, only the dimension class can belong to both a classification and specialization hierarchy at the same time.

Regarding the dynamic part, this approach also provides a UML-compliant class notation (called *cube class*) to specify initial user requirements. A cube class is structured into three sections: *measures*, to specify which fact attributes are analyzed; *slice*, to express constraints in terms of filters; and *dice*, to define grouping conditions of the data. Later, a set of basic OLAP operations (e.g. roll-up, drill-down, slice, etc.) is provided to accomplish the further data analysis phase from these cube classes.

Finally, this approach provides a set of modules with adequate transformation rules to export the accomplished conceptual MD model into a commercial OLAP tool, which allows us to check the validity of the proposed approach.

3 XML Schema and XML Document

As said in the introduction, our goal is to provide a common standard format to store and interchange conceptual MD models accomplished by the OO conceptual approach. To achieve this goal, this section presents the XML Schema[2] that allows us to structure XML documents that are instances of the MD models together with all the MD expressiveness commented in Section 2.

3.1 XML Schema

The purpose of XML Schemas is to specify the structure of instance elements together with the data type of each element/attribute. The motivation for XML Schemas is the dissatisfaction with DTDs mainly due to their syntax and their limited data type capability, not allowing us to define new specific data types. Therefore, XML Schemas are a tremendous advancement over DTDs, as they allow us to create enhanced data types and valid references, they are written in the same syntax as instance documents, they can define multiple elements with the same name but different content (namespace), they can define substitutable elements, and many more features (sets, unique keys, nil content, etc.).

In [16] we presented a DTD to validate XML documents that stored the conceptual MD models. In this section, we notably improve our previous proposal by defining an XML Schema instead of the DTD. With respect to the structure of an XML Schema, there are two main possibilities: flat and "Russian doll" designs. The former is based on a flat catalog of all the elements available in the instance document and, for each of them, lists of child elements and attributes. We will use the later design as it allows us to define each element and attribute within its context in an embedded manner. In this sense, the representation of the XML Schema as a tree structure is illustrated in Fig. 2 for the sake of clearness and comprehension. As it can be observed, we denote every node of the tree with a label. Then, every label has its correspondence with one element in the XML Schema. Following a left-right path in the tree, we clearly identify all the MD properties supported by the OO conceptual MD model. More precisely, the root of the tree is tagged as GOLMODEL and corresponds to the following element definition

```
<!-- Main element (root) -->
<xsd:element name="GOLDMODEL">
  <xsd:complexType>
    <xsd:sequence>
<!-- First level: FACTCLASSES -->
      <xsd:element name="FACTCLASSES">
```

[2] The complete definition of the XML Schema has more than 300 lines and it is not completely shown here due to space constraints.

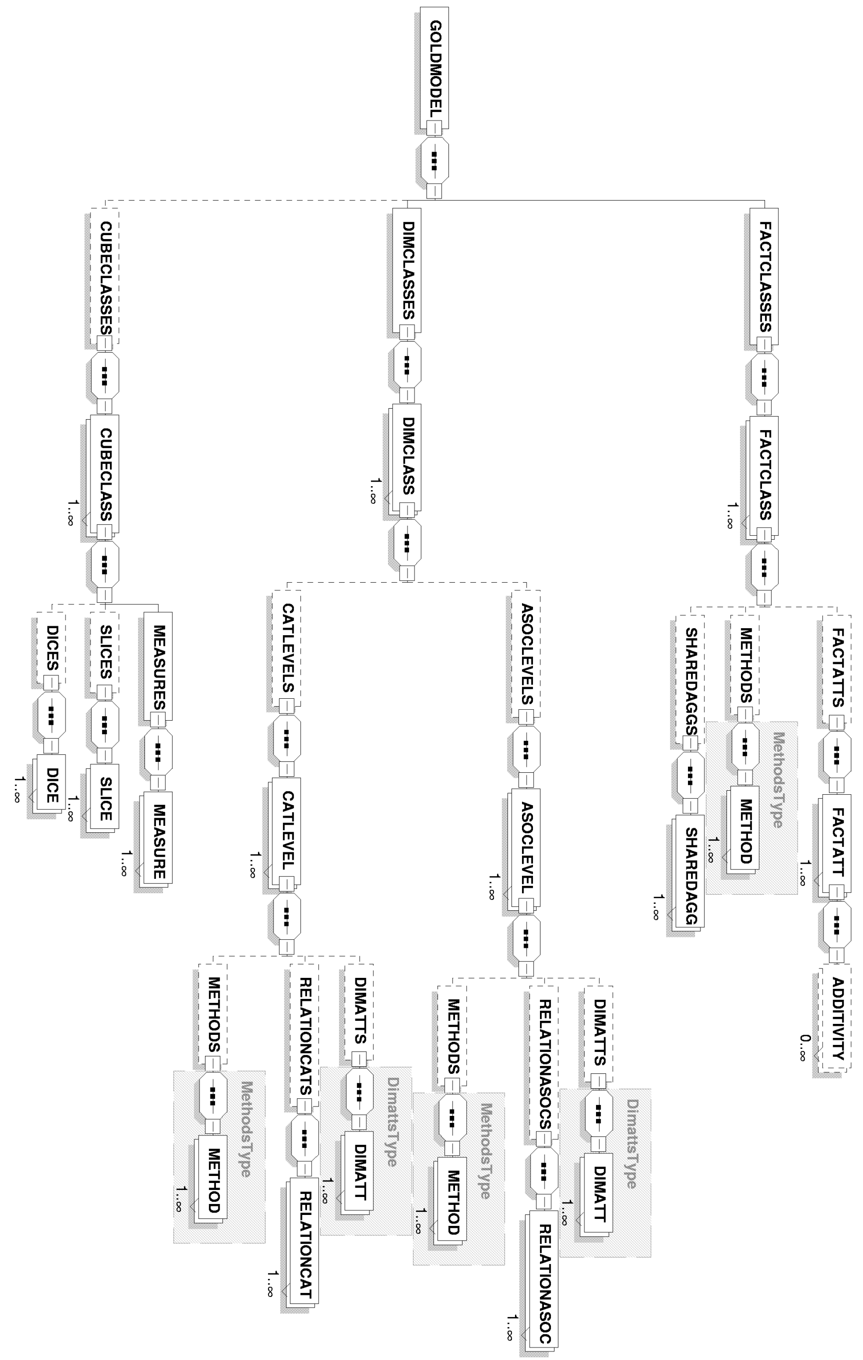


Fig. 2. The XML Schema represented as a tree structure.

```
      ...
      </xsd:element>
<!-- First level: DIMCLASSES -->
      <xsd:element name="DIMCLASSES">
      ...
      </xsd:element>
<!-- First level: CUBECLASSES -->
      <xsd:element name="CUBECLASSES" minOccurs="0">
      ...
      </xsd:element>
    </xsd:sequence>
<!-- GOLDMODEL attributes-->
    <xsd:attribute name="id" type="xsd:ID" use="required"/>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="showAtts" type="xsd:boolean" default="true"/>
    <xsd:attribute name="showMethods" type="xsd:boolean" default="true"/>
    <xsd:attribute name="creationDate" type="xsd:date"/>
    <xsd:attribute name="lastModified" type="xsd:date"/>
    <xsd:attribute name="description" type="xsd:string"/>
    <xsd:attribute name="responsible" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

while the child nodes of a node represent the subelements of an element in the XML Schema[3]. Whether the subelements are optional is indicated by a dashed line in the tree structure. Furthermore, to indicate the multiplicity (cardinality) of the subelements, we use the standard modifiers provided by the XML Schema syntax: minOccurs and maxOccurs, corresponding to minimum and maximum values for the lower and upper bounds respectively. By default, both modifiers have the value 1, but if this value changes, then it is explicitly defined in the schema, as can be seen in Fig. 2. Furthermore, we define additional tags (in plural form) in order to group common elements together, so that they can be exploited to provide optimum and correct comprehension of the model, e.g. tags in plural like FACTCLASSES or DIMCLASSES.

As previously commented, one of the main advances of the XML Schemas is that we can easily declare new data types to define our own attributes and elements. To achieve this, we give a name to the simpleType and complexType elements of the XML Schema. The following fragment shows how the simple data types `Operator` and `Multiplicity` are defined to be used later as types of particular attributes, such as `roleA`, `roleB` or `operator` (notice that user-defined data types are shadowed in Fig. 2):

```
<xsd:simpleType name="Operator">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="eq"/>
    <xsd:enumeration value="lt"/>
    <xsd:enumeration value="gt"/>
    <xsd:enumeration value="let"/>
    <xsd:enumeration value="get"/>
    <xsd:enumeration value="noteq"/>
    <xsd:enumeration value="like"/>
    <xsd:enumeration value="notlike"/>
    <xsd:enumeration value="in"/>
    <xsd:enumeration value="notin"/>
  </xsd:restriction>
</xsd:simpleType>
```

```
<xsd:simpleType name="Multiplicity">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="0"/>
    <xsd:enumeration value="1"/>
    <xsd:enumeration value="M"/>
    <xsd:enumeration value="1..M"/>
  </xsd:restriction>
</xsd:simpleType>
```

[3] The attributes of the elements are not represented in the tree for the sake of simplicity.

Within this XML Schema, fact classes labeled FACTCLASS in Fig. 2, may have no fact attributes to consider fact-less fact tables, as can be observed in the multiplicity (`minOccurs="0"`) for the tag FACTATTS:

```
<!-- Set of fact classes -->
<xsd:element name="FACTCLASSES">
  <xsd:complexType> <xsd:sequence>
    <xsd:element name="FACTCLASS" maxOccurs="unbounded">
      <xsd:complexType> <xsd:sequence>
<!-- Set of fact attributes -->
        <xsd:element name="FACTATTS" minOccurs="0">
          <xsd:complexType> <xsd:sequence>
            <xsd:element name="FACTATT" maxOccurs="unbounded">
              <xsd:complexType> <xsd:sequence>
                <xsd:element name="ADDITIVITY" minOccurs="0" maxOccurs="unbounded">
                  <xsd:complexType>
                    <!-- ADDITIVITY attributes -->
                    <xsd:attribute name="dimclass" type="xsd:IDREF" use="required"/>
                    <xsd:attribute name="isNOT" type="xsd:boolean" default="false"/>
                    <xsd:attribute name="isSUM" type="xsd:boolean" default="false"/>
                    <xsd:attribute name="isMAX" type="xsd:boolean" default="false"/>
                    <xsd:attribute name="isMIN" type="xsd:boolean" default="false"/>
                    <xsd:attribute name="isAVG" type="xsd:boolean" default="false"/>
                    <xsd:attribute name="isCOUNT" type="xsd:boolean"/>
              </xsd:complexType> </xsd:element> </xsd:sequence>
              <!-- FACTATT atributes -->
              <xsd:attribute name="id" type="xsd:ID" use="required"/>
              <xsd:attribute name="name" type="xsd:string" use="required"/>
              ...
              <xsd:attribute name="derivationRule" type="xsd:string"/>
          </xsd:complexType> </xsd:element> </xsd:sequence>
        </xsd:complexType> </xsd:element>
<!-- Set of fact methods -->
        <xsd:element name="METHODS" type="MethodsType" minOccurs="0"/>
<!-- Set of shared aggregations -->
        <xsd:element name="SHAREDAGGS" minOccurs="0">
          <xsd:complexType> <xsd:sequence>
            <xsd:element name="SHAREDAGG" maxOccurs="unbounded">
              <xsd:complexType>
                <!-- SHAREDAGG attributes>
                <xsd:attribute name="dimclass" type="xsd:IDREF" use="required"/>
                <xsd:attribute name="name" type="xsd:string"/>
                <xsd:attribute name="description" type="xsd:string"/>
                <xsd:attribute name="roleA" type="Multiplicity" default="M"/>
                <xsd:attribute name="roleB" type="Multiplicity" default="1"/>
          </xsd:complexType> </xsd:element> </xsd:sequence>
        </xsd:complexType> </xsd:element> </xsd:sequence>
        <!-- FACTCLASS attributes --> ...
</xsd:complexType> </xsd:element> </xsd:sequence> </xsd:complexType> </xsd:element>
```

In case that fact attributes exist in the fact class, derivation rules can be defined for these attributes by means of `derivationRule` in order to express calculated measures. Additivity is also supported in terms of the element ADDITIVITY along with the information about how a measure is aggregated (`isSUM`, `isMAX`, etc.) along a particular dimension. Notice at the end of the previous XML Schema fragment how many-to-many relationships between facts and dimensions can also be expressed by assigning the same value "M" to both attributes `roleA` and `roleB` in the element SHAREDAGG. The type of both attributes is `Multiplicity`, previously defined in the XML Schema.

With respect to dimensions, the following fragment of the XML Schema contains elements to express dimensions and classification hierarchies by means of association relationships:

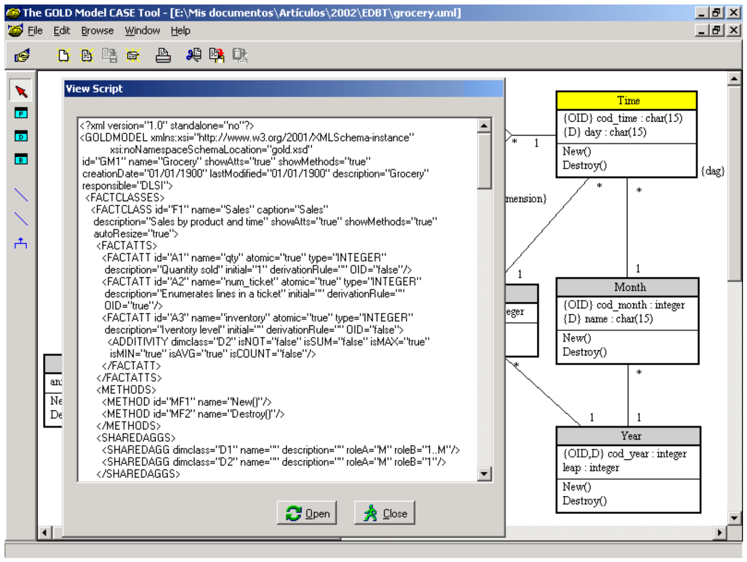
```
<!-- Set of dimension classes -->
<xsd:element name="DIMCLASSES">
  <xsd:complexType> <xsd:sequence>
    <xsd:element name="DIMCLASS" maxOccurs="unbounded">
      <xsd:complexType> <xsd:sequence>
<!-- Set of association levels -->
        <xsd:element name="ASOCLEVELS" minOccurs="0">
          <xsd:complexType> <xsd:sequence>
            <xsd:element name="ASOCLEVEL" maxOccurs="unbounded">
              <xsd:complexType> <xsd:sequence>
<!-- Set of dimension attributes -->
                <xsd:element name="DIMATTS" type="DimattsType" minOccurs="0"/>
<!-- Set of association relationships -->
                <xsd:element name="RELATIONASOCS" minOccurs="0">
                  <xsd:complexType> <xsd:sequence>
                    <xsd:element name="RELATIONASOC" maxOccurs="unbounded">
                      <xsd:complexType>
                        <xsd:attribute name="child" type="xsd:IDREF" use="required"/>
                        <xsd:attribute name="name" type="xsd:string"/>
                        <xsd:attribute name="description" type="xsd:string"/>
                        <xsd:attribute name="roleA" type="Multiplicity" default="1"/>
                        <xsd:attribute name="roleB" type="Multiplicity" default="M"/>
                        <xsd:attribute name="completeness" type="xsd:boolean"/>
                  </xsd:complexType> </xsd:element> </xsd:sequence>
                </xsd:complexType> </xsd:element>
<!-- Set of dimension methods -->
                <xsd:element name="METHODS" type="MethodsType" minOccurs="0"/>
              </xsd:sequence>
              <!-- METHODS attributes -->
          </xsd:complexType> </xsd:element> </xsd:sequence>
        </xsd:complexType> </xsd:element>
        <!-- DIMCLASS attributes-->
        <xsd:attribute name="id" type="xsd:ID" use="required"/>
        <xsd:attribute name="name" type="xsd:string" use="required"/>
        <xsd:attribute name="caption" type="xsd:string"/>
        <xsd:attribute name="description" type="xsd:string"/>
        <xsd:attribute name="isTime" type="xsd:boolean" default="false"/>
</xsd:complexType> </xsd:element> </xsd:sequence> </xsd:complexType> </xsd:element>
```

Notice that attributes within these elements in the XML Schema allow us to express all the MD properties of the model. In this way, non-strictness may be defined by assigning the same value "M" to both attributes `roleA` and `roleB` in the element RELATIONASOC, along with completeness by means of the boolean attribute `completeness`. Moreover, identifying and descriptor attributes within dimensions can be defined using the attributes `OID` and `D`, respectively in DIMATT.

Finally, information about valid references is provided to our XML Schema. This is a very important feature, as we can determine which elements an element is able to reference, thereby allowing the schema to be semantically correct (it is similar to foreign keys in relational databases). It supposes an improvement of our previous proposal [16], as DTD references are not selective and can be applied to any element, although not being semantically correct. For example, the following fragment ensures that the values of the attribute `dimclass` within the ADDITIVITY and SHAREDAGG elements point to identifier attributes (`@id`) in DIMCLASS elements:

```
<xsd:key name="DIMCLASSKey">
  <xsd:selector xpath="DIMCLASSES/DIMCLASS"/>
  <xsd:field xpath="@id"/>
</xsd:key>
<xsd:keyref name="additivityDIMCLASSKey" refer="DIMCLASSKey">
  <xsd:selector xpath="FACTCLASSES/FACTCLASS/FACTATTS/FACTATT/ADDITIVITY"/>
```

```
    <xsd:field xpath="@dimclass"/>
  </xsd:keyref>
  <xsd:keyref name="sharedaggDIMCLASSKey" refer="DIMCLASSKey">
    <xsd:selector xpath="FACTCLASSES/FACTCLASS/SHAREDAGGS/SHAREDAGG"/>
    <xsd:field xpath="@dimclass"/>
  </xsd:keyref>
```


The GOLD Model CASE Tool - [E:\Mis documentos\Artículos\2002\EDBT\grocery.uml]
File Edit Browse Window Help
View Script
Open
Close
Time
{OID} cod_time : char(15)
{D} day : char(15)
New()
Destroy()
Month
{OID} cod_month : integer
{D} name : char(15)
New()
Destroy()
Year
{OID,D} cod_year : integer
leap : integer
New()
Destroy()
{dag}

```
<?xml version="1.0" standalone="no"?>
<GOLDMODEL xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="gold.xsd"
 id="GM1" name="Grocery" showAtts="true" showMethods="true"
 creationDate="01/01/1900" lastModified="01/01/1900" description="Grocery"
 responsible="DLSI">
  <FACTCLASSES>
   <FACTCLASS id="F1" name="Sales" caption="Sales"
    description="Sales by product and time" showAtts="true" showMethods="true"
    autoResize="true">
     <FACTATTS>
      <FACTATT id="A1" name="qty" atomic="true" type="INTEGER"
       description="Quantity sold" initial="1" derivationRule="" OID="false"/>
      <FACTATT id="A2" name="num_ticket" atomic="true" type="INTEGER"
       description="Enumerates lines in a ticket" initial="" derivationRule=""
       OID="true"/>
      <FACTATT id="A3" name="inventory" atomic="true" type="INTEGER"
       description="Iventory level" initial="" derivationRule="" OID="false">
        <ADDITIVITY dimclass="D2" isNOT="false" isSUM="false" isMAX="true"
         isMIN="true" isAVG="true" isCOUNT="false"/>
      </FACTATT>
     </FACTATTS>
     <METHODS>
      <METHOD id="MF1" name="New()"/>
      <METHOD id="MF2" name="Destroy()"/>
     </METHODS>
     <SHAREDAGGS>
      <SHAREDAGG dimclass="D1" name="" description="" roleA="M" roleB="1..M"/>
      <SHAREDAGG dimclass="D2" name="" description="" roleA="M" roleB="1"/>
     </SHAREDAGGS>
```

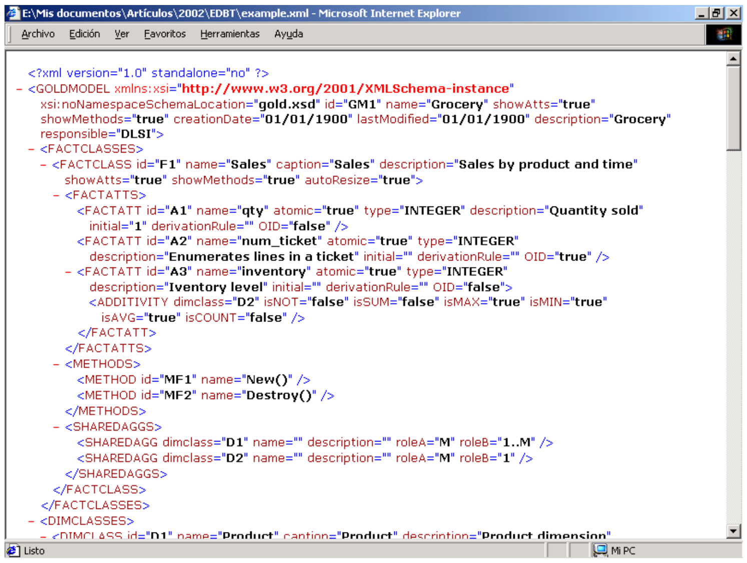
Fig. 3. An XML document generated by the CASE Tool.

3.2 XML Document

Once the specification for the XML Schema is accomplished, XML documents may be generated by the CASE tool to store instances of the OO conceptual models, as reflected in Fig. 3. On the other hand, Fig. 4 shows the presentation of an XML document generated by the CASE tool in a popular web browser (*Microsoft Internet Explorer*). This browser brings the possibility to validate an XML document against a DTD, but not against an XML Schema; in addition, the XML document is not presented in a "pretty" way.

To validate our XML Schema, we used *IBM Schema Quality Checker 1.2.1.003*[4], a program which takes one or more XML Schema files, parses them and reports any violation of rules and constraints defined in the W3C XML Schema specification [8]. On the other hand, we also used *Apache Xerces Java Parser 1.4.4*[5] to validate XML documents against our XML Schema.

[4] Available from `http://www.alphaworks.ibm.com/tech/xmlsqc`.
[5] Available from `http://xml.apache.org/xerces-j/index.html`.


E:\Mis documentos\Artículos\2002\EDBT\example.xml - Microsoft Internet Explorer
Archivo
Edición
Ver
Favoritos
Herramientas
Ayuda
Listo
Mi PC

```
  <?xml version="1.0" standalone="no" ?>
- <GOLDMODEL xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="gold.xsd" id="GM1" name="Grocery" showAtts="true"
    showMethods="true" creationDate="01/01/1900" lastModified="01/01/1900" description="Grocery"
    responsible="DLSI">
  - <FACTCLASSES>
    - <FACTCLASS id="F1" name="Sales" caption="Sales" description="Sales by product and time"
        showAtts="true" showMethods="true" autoResize="true">
      - <FACTATTS>
          <FACTATT id="A1" name="qty" atomic="true" type="INTEGER" description="Quantity sold"
            initial="1" derivationRule="" OID="false" />
          <FACTATT id="A2" name="num_ticket" atomic="true" type="INTEGER"
            description="Enumerates lines in a ticket" initial="" derivationRule="" OID="true" />
        - <FACTATT id="A3" name="inventory" atomic="true" type="INTEGER"
            description="Iventory level" initial="" derivationRule="" OID="false">
            <ADDITIVITY dimclass="D2" isNOT="false" isSUM="false" isMAX="true" isMIN="true"
              isAVG="true" isCOUNT="false" />
          </FACTATT>
        </FACTATTS>
      - <METHODS>
          <METHOD id="MF1" name="New()" />
          <METHOD id="MF2" name="Destroy()" />
        </METHODS>
      - <SHAREDAGGS>
          <SHAREDAGG dimclass="D1" name="" description="" roleA="M" roleB="1..M" />
          <SHAREDAGG dimclass="D2" name="" description="" roleA="M" roleB="1" />
        </SHAREDAGGS>
      </FACTCLASS>
    </FACTCLASSES>
  - <DIMCLASSES>
    - <DIMCLASS id="D1" name="Product" caption="Product" description="Product dimension"
```

Fig. 4. An XML document displayed in Microsoft Internet Explorer without XSLT.

4 XSLT and the Web

Another relevant issue of our approach was to provide different presentations of the MD models in the web at a conceptual level. However, nowadays only some web browsers partly support XML (*Microsoft Internet Explorer 5*, *Netscape Navigator 6*). In addition to this inconvenience, XML documents only contain data, so that no information about presentation aspects is included. To solve this problem, XSL Formatting Objects[6] (XSL FO) [17] is a recent technology that allows us to define the presentation for XML documents, although there is no current browser that supports it. As a consequence, we are currently forced to transform XML documents to HTML pages[7] in order to publish them in the web.

The best method to accomplish this task is the use of XSLT [9], as a language for transforming XML documents into other XML documents. XSLT stylesheets describe a set of patterns (templates) to match both elements and attributes defined in an XML Schema, in order to apply specific transformations for each considered match. Thanks to XSLT, the source document can be filtered and reordered in constructing the resulting output. Therefore, our XML documents

[6] XSL FO describes how XML documents will look when displayed or printed.

[7] Actually, the XML documents are transformed to XHTML 1.0. In addition, we also use Cascading Style Sheets (CSS), because it gives us more control over how pages are displayed.

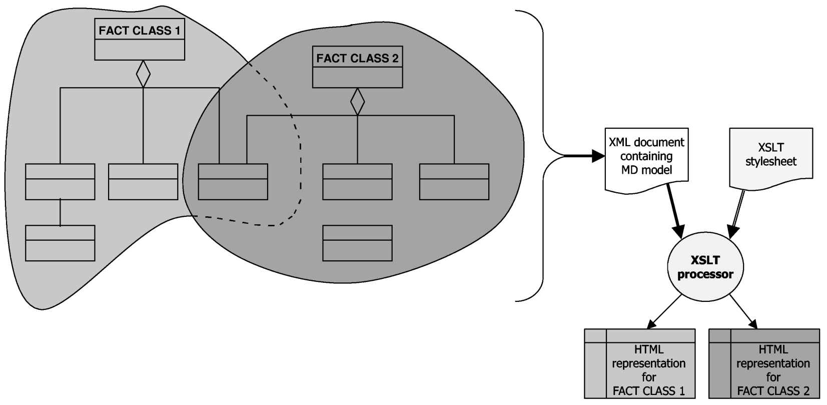

FACT CLASS 1
FACT CLASS 2
XML document containing MD model
XSLT stylesheet
XSLT processor
HTML representation for FACT CLASS 1
HTML representation for FACT CLASS 2

Fig. 5. Generating different presentations in HTML from the same MD model.

can be tailored to represent different presentations of the same MD model using one XSLT stylesheet for each presentation[8].

As an example, Fig. 5 illustrates the overall transformation process for a MD model composed of two fact classes sharing common dimensions. The MD model is stored in an XML document and an XSLT stylesheet is provided to generate different presentations of the MD model as different HTML pages. These presentations correspond to FACT CLASS 1 and 2 respectively and they only contain relevant information about each fact class in the model, i.e. dimensions in the MD model not belonging to a particular fact class are not shown in the corresponding presentation.

In our work, we have used two different XSLT processors: *Microsoft XML Core Services (MSXML) 4.0*[9] (formerly known as the *Microsoft XML Parser*) and *Instant Saxon 6.5*[10]. Both of them are full compliance with XSLT 1.0. However, *Instant Saxon* also allows the use of XSLT 1.1[11] features, including `xsl:script` and `xsl:document` (this new instruction allows us to create different outputs from the same XML document). Therefore, we have considered two approaches: one for version 1.0 that generates an only HTML page with internal links, and the other for version 1.1 that has the benefit of generating a collection of HTML pages with links between them (the number of pages depends on the number of fact classes and dimension classes defined in the model).

Due to space constraints, it is not possible to include the complete definition of the XSLT stylesheet here. Therefore, we only exhibit some fragments of the

[8] It can be also used an only XSLT stylesheet that receives a parameter.

[9] The MSXML 4.0 features are available in Internet Explorer only via scripting. It can be downloaded from `http://msdn.microsoft.com`.

[10] Available from `http://saxon.sourceforge.net`.

[11] The status of this version is "working draft". Besides, XSLT 2.0 is in "requirements working draft".

XSLT. The first example shows the instructions that generate the HTML code to display information about fact attributes (FACTATT):

```
<xsl:template match="FACTATT">
  <tr bgcolor="#00ffff">
    <td><font size="1"><xsl:value-of select="@name"/></font></td>
    <td><font size="1"><xsl:value-of select="@atomic"/></font></td>
    <td><font size="1"><xsl:value-of select="@type"/></font></td>
    <td><font size="1"><xsl:value-of select="@description"/></font></td>
    <td><font size="1"><xsl:value-of select="@initial"/></font></td>
    <td><font size="1"><xsl:value-of select="@derivationRule"/></font></td>
    <td><font size="1"><xsl:value-of select="@OID"/></font></td>
  </tr>
</xsl:template>
```

Notice that XSLT instructions and HTML tags are intermingled. The XSLT processor copies the HTML tags to the transformed document and interprets any XSLT instruction encountered. Applied to this example, the value of the attributes of the element FACTATT are inserted in the resulting document in an HTML table; we can notice the attributes commented in Section 2: atomic, derivation rule, OID, and so on.

As above commented, XSLT 1.1 allows us to create different HTML outputs and connect them from an only XML document. The following instructions show how we create the links between the main page, the page that represents the MD model, and the pages that correspond to the fact classes:

```
<xsl:template match="FACTCLASS">
<xsl:variable name="url" select="@id"/>
 <tr>
  <td></td>
  <!-- The link to the fact class page -->
  <td><font size="1"><a href="{$url}.html"><xsl:value-of select="@name"/></a></font></td>
  <td><font size="1"><xsl:value-of select="@description"/></font></td>
 </tr>
<!-- New document -->
<xsl:document href="{$url}.html">
 <html>
 <head><title>Fact class: <xsl:value-of select="@name"/></title></head>
 <body bgcolor="mintcream">
```

As we have defined the `@id` attribute of FACTCLASS as `ID` in the XML Schema, and thus, it has a unique value:

– We store it in a variable:
  `<xsl:variable name="url" select="@id"/>`
– We use it as the name of new documents:
  `<xsl:document href="{$url}.html">`
– We use it as the target document in the links:
  `<a href="{$url}.html">`

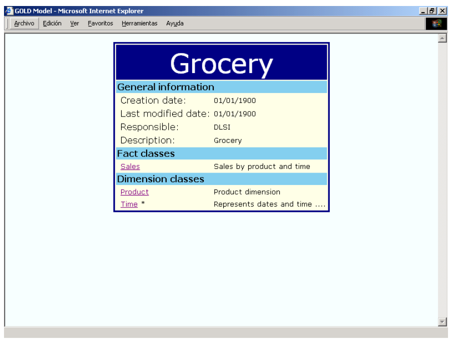
4.1 Managing the model in a web browser

The resulting HTML pages allow us to navigate through the different presentations of the model on a web browser. All the information about the MD properties of the model is represented in the HTML pages.

For example, in Fig. 6 we show an example of navigation for one of the presentations. The first page is showed in Fig. 6.1 and contains the general description of the model: name, creation date, last modified, and the names of the fact classes and dimension classes, which are active links.

Whenever it is possible, there is a link connecting different pieces of information. For example, if the fact class `Sales` is selected, the page showed in Fig. 6.2 is loaded. This page shows the information about the selected fact class: name, measures, methods, and shared aggregations. In the example, `Sales` contains three measures, `inventory`, `num_ticket` and `qty`, and the first one is an active link because it has additivity rules. If this link is selected, a new floating page is showed (Fig. 6.3) with the corresponding additivity rule.

Finally, Fig. 6.4 shows the definition of the dimension class `Time`: attributes, methods, association levels and categorization levels. In this example, this dimension class has two association levels: `Month` and `Week`, which are again active links and allow us to continue exploring the model. This page can be reached from pages in Fig. 6.1, 6.2, and 6.3, because there exists a relationship between them.


GOLD Model - Microsoft Internet Explorer
Archivo Edición Ver Favoritos Herramientas Ayuda

Grocery

General information

Creation date:	01/01/1900
Last modified date:	01/01/1900
Responsible:	DLSI
Description:	Grocery

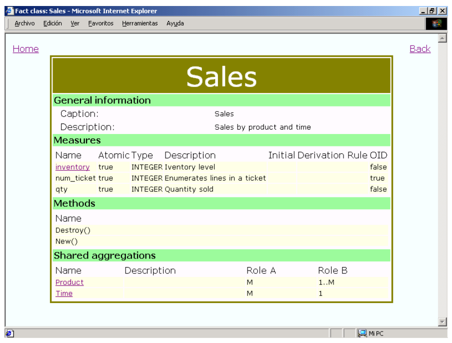
Fact classes

Sales	Sales by product and time

Dimension classes

Product	Product dimension
Time *	Represents dates and time ....

6.1


Fact class: Sales - Microsoft Internet Explorer
Archivo Edición Ver Favoritos Herramientas Ayuda
Home
Back
Mi PC

Sales

General information

Caption:	Sales
Description:	Sales by product and time

Measures

Name	Atomic	Type	Description	Initial	Derivation Rule	OID
inventory	true	INTEGER	Iventory level			false
num_ticket	true	INTEGER	Enumerates lines in a ticket			true
qty	true	INTEGER	Quantity sold			false

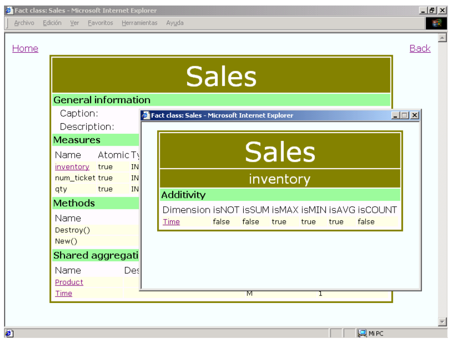
Methods

Name
Destroy()
New()

Shared aggregations

Name	Description	Role A	Role B
Product		M	1..M
Time		M	1

6.2


Fact class: Sales - Microsoft Internet Explorer
Archivo Edición Ver Favoritos Herramientas Ayuda
Home
Back
Fact class: Sales - Microsoft Internet Explorer
Mi PC

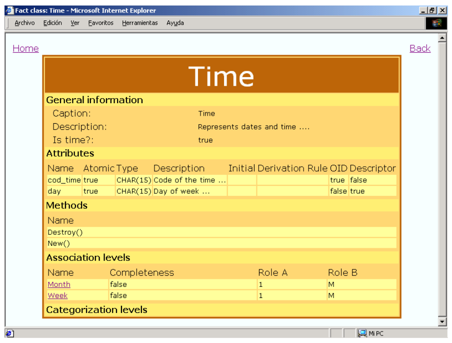
Sales

inventory

Additivity

Dimension	isNOT	isSUM	isMAX	isMIN	isAVG	isCOUNT
Time	false	false	true	true	true	false

6.3


Fact class: Time - Microsoft Internet Explorer
Archivo Edición Ver Favoritos Herramientas Ayuda
Home
Back
Mi PC

Time

General information

Caption:	Time
Description:	Represents dates and time ....
Is time?:	true

Attributes

Name	Atomic	Type	Description	Initial	Derivation Rule	OID	Descriptor
cod_time	true	CHAR(15)	Code of the time ...			true	false
day	true	CHAR(15)	Day of week ...			false	true

Methods

Name
Destroy()
New()

Association levels

Name	Completeness	Role A	Role B
Month	false	1	M
Week	false	1	M

Categorization levels

6.4

Fig. 6. Model navigation in the web using a browser.

5 Conclusions

Multidimensional (MD) modeling is the foundation of data warehouses, MD databases, and OLAP applications. We have presented how to manage the representation, manipulation, and presentation of MD models accomplished by a OO conceptual approach, based on UML, in XML documents. In this way, we are able to consider the main MD properties from a conceptual level such as many-to-many relationships between facts and dimensions, additivity, non-strictness and so on.

To validate these XML documents, we have defined an XML Schema that represents the structure of XML documents. The benefit of this proposal is that we make use of standard and recent technology, that is increasingly becoming accepted, together with the task of MD modeling by means of the OO conceptual approach.

Furthermore, we have provided XSLT stylesheets that allow us to generate different presentations on HTML pages of the same MD model. Therefore, management of different presentations could be easily achieved. As an example of the applicability of our proposal, these HTML pages can be used to document the MD models in the web, with the advantages that it implies. Moreover, the automatic generation of documentation avoids the problem of documentation out of date. A CASE tool that gives support to all theoretical issues presented in the paper has been developed.

6 Future Work

Due to the rapid evolution of the used technology, we are considering to adapt our proposal to the new emerging standards. With respect to the presentation, XSL FO can be used to specify in deeper detail the pagination, layout, and styling information that will be applied to XML documents. However, to the best of our knowledge, there are no current tools that completely provide support for XSL FO.

From the architecture perspective, we run the transformation process using a client-server technology, i.e. the XSLT stylesheet is applied to the XML document in the server and the HTML is returned to the client browser. In the future, when the browsers completely support XML and XSLT, the transformation will be able to be performed in the browser. This last approach has the advantage of removing some of the processing load from the server.

Finally, we are currently studying the Common Warehouse Metamodel [18] as a common framework to easily interchange warehouse metadata between distributed heterogenous environments. This proposal provides designers and tools with common definitions but lacks the complete set of information an existing tool would need to fully operate. Therefore, another future research line would be to extend the given definitions to provide the CASE tool with specific definitions required for operation.

References

1. Golfarelli, M., Rizzi, S.: A methodological Framework for Data Warehouse Design. In: Proc. of the ACM 1st Intl. Workshop on Data warehousing and OLAP (DOLAP'98), Washington D.C., USA (1998) 3–9
2. Sapia, C., Blaschka, M., Höfling, G., Dinter, B.: Extending the E/R Model for the Multidimensional Paradigm. In: Proc. of the 1st Intl. Workshop on Data Warehouse and Data Mining (DWDM'98). Volume 1552 of LNCS., Springer-Verlag (1998) 105–116
3. Trujillo, J., Gómez, J., Palomar, M.: Modeling the Behavior of OLAP Applications Using an UML Compilant Approach. In: Proc. of the 1st Intl. Conf. On Advances in Information Systems (ADVIS'00). Volume 1909 of LNCS., Springer-Verlag (2000) 14–23
4. Trujillo, J., Palomar, M., Gómez, J., Song, I.Y.: Designing Data Warehouses with OO Conceptual Models. IEEE Computer, special issue on Data Warehouses **34** (2001) 66–75
5. Tryfona, N., Busborg, F., Christiansen, J.: starER: A Conceptual Model for Data Warehouse Design. In: Proc. of the ACM 2nd Intl. Workshop on Data warehousing and OLAP (DOLAP'99), Kansas City, Missouri, USA (1999)
6. Abelló, A., Samos, J., Saltor, F.: A Framework for the Classification and Description of Multidimensional Data Models. In: Proc. of the 12th Intl. Conference on Database and Expert Systems Applications (DEXA'01), Munich, Germany (2001) 668–677
7. World Wide Web Consortium (W3C): eXtensible Markup Language (XML) 1.0 (SE). Internet: http://www.w3.org/TR/2000/REC-xml-20001006 (2000)
8. World Wide Web Consortium (W3C): XML Schema. Internet: http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/ (2001)
9. World Wide Web Consortium (W3C): XSL Transformations (XSLT) Version 1.0. Internet: http://www.w3.org/TR/1999/REC-xslt-19991116 (1999)
10. Object Management Group (OMG): Unified Modeling Language (UML). Internet: http://www.omg.org/cgi-bin/doc?formal/01-09-67 (2001)
11. Pokorný, J.: Modelling Stars Using XML. In: Proc. of the ACM 4th Intl. Workshop on Data warehousing and OLAP (DOLAP'01), Atlanta, GA USA (2001)
12. Golfarelli, M., Rizzi, S., Vrdoljak, B.: Data warehouse design from XML sources. In: Proc. of the ACM 4th Intl. Workshop on Data warehousing and OLAP (DOLAP'01), Atlanta, GA USA (2001)
13. Gergatsoulis, M., Stavrakas, Y., Karteris, D.: A Web-Based System for Handling Multidimensional Information through MXML. In Albertas Caplinskas and Johann Eder, ed.: Proc. of the 5th East European Conference on Advances in Databases and Information Systems (ADBIS'01). Volume 2151 of LNCS., Vilnius, Lithuania, Springer-Verlag (2001)
14. Giovinazzo, W.: Object-Oriented Data Warehouse Design. Building a star schema. Prentice-Hall, New Jersey, USA (2000)
15. Kimball, R.: The data warehousing toolkit. 2 edn. John Wiley (1996)
16. Luján-Mora, S., Medina, E., Trujillo, J.: From Object-Oriented Conceptual Multidimensional Modeling into XML. Technical report, DLSI - Univ. of Alicante (2001) Internet: http://gplsi.dlsi.ua.es/almacenes/files/oomd-mm-dtd.pdf.
17. World Wide Web Consortium (W3C): Extensible Stylesheet Language (XSL) 1.0. Internet: http://www.w3.org/TR/2001/REC-xsl-20011015/ (2001)
18. Object Management Group (OMG): Common Warehouse Metamodel (CWM). Internet: http://www.omg.org/cgi-bin/doc?ad/2001-02-01 (2000)