XCube – XML For Data Warehouses

Wolfgang Hümmer
University of Erlangen-Nuremberg
Department of Database Systems
Martensstr. 3, 91058 Erlangen, Germany
huemmer@cs.fau.de

Andreas Bauer
T-Systems
Merianstr. 32, 90409 Nürnberg, Germany
andreas.bauer@t-systems.com

Gunnar Harde
Oldenburger Forschungs- und Entwicklungsinstitut für Informatik-Werkzeuge und Systeme (OFFIS)
Escherweg 2, 26121 Oldenburg, Germany
gunnar.harde@offis.de

ABSTRACT

Data warehouse systems are nowadays a well known and widely spread approach for supporting management decisions. In several companies or even across companies the idea of integrating several data warehouses into a virtual or federated data warehouse is of growing interest. But the technical and semantic problems are very demanding. An essential part for solving this problem is a standardized, vendor independent format for describing multidimensional data. This paper introduces XCube, a family of XML based document templates to exchange data warehouse data, i.e. data cubes, over any kind of network. XCube is organized in a modular fashion, so the multidimensional schema, the descriptions of the single dimensions and the fact data itself can be transmitted in separate steps. In addition to the describing formats XCube also offers two kinds of dynamic document types that can be used to explore the (multidimensional) content of another warehouse in a vendor independent way. They are primarily meant to reduce the amount of data transferred over the network

Categories and Subject Descriptors

H 2.3 [Languages]:
Data description languages (DDL), Query languages

H.2.7 [Database Administration]:
Data warehouse and repository

General Terms

Design, Languages, Management, Standardization

Keywords

Exchange of data cubes, between data warehouses, based on XML documents

1. INTRODUCTION

Data warehouse systems are getting more and more popular and wide spread. In many companies they are an integral technological part of management decision support. In most cases the data in a warehouse are imported from transactional systems (OLTP) of a company. By now the World Wide Web is rarely seen as a potential data source. One reason for sure is that data in the WWW are rather semi structured or not structured at all: information is stored in HMTL format without any semantic meta data, or – in very few cases – in XML files based on different or no grammars (DTDs or XML Schemas [1], [2]). Other information is encoded in pictures or arbitrary binary file formats. Besides these problems of technical heterogeneity the semantic heterogeneity is even worse: different WWW documents might refer to different meanings when using the same expression *year*, e.g. *financial year* or *calendar year*. As long as the semantics of the online documents is not understood it is extremely dangerous to integrate it into a data warehouse with regard to data quality ([3]). Still it is well known that a huge amount of valuable information is spread all over the Internet.

An even better data source for extending a data warehouse is another data warehouse: its data is already cleaned and verified. The situation that several data warehouses have to be integrated can happen inside or across companies, e.g. after a merger or for cooperation reasons. The integration of information systems is an important issue in enterprise application integration (EAI) and B2B integration ([4], [5]).

A standard format for online data that could be useful for data warehouses is highly desirable and could solve the technical problems completely and the semantic problems at least to a certain degree. As XML ([1]) is more and more seen as the lingua franca for exchanging data over any kind of networks, this paper introduces XCube: XCube ([6]) is an open, manufacturer independent and XML based family of document templates to store, exchange and query data warehouse data, i.e. data cubes. The benefits of this approach are manifold: the advantages of using standards are well known and by using XML it is easy to adapt the standard documents to the needs of every single data warehouse in a modular fashion. To enable a data warehouse system to handle XML documents can be seen as rather simple – nearly every standard database system is XML enabled nowadays ([7]). Furthermore, lots of tools exist to transform XML data into the necessary local formats and vice versa (e.g. XSLT, [8]). One disadvantage of using XML for exchanging data cubes is the fact that XML documents tend to be rather large. But with constantly growing network bandwidth and the use of efficient, transparent compression methods in mind the advantages prevail.

Structure of the paper

The paper is structured as follows: in the next section several scenarios are described where an open, independent standard for exchanging data warehouse data would be useful. Further a

DOLAP'03, November 7, 2003, New Orleans, Louisiana, USA.

brief overview over related work is given. Section 3 starts introducing the family of XCube formats. Besides the basic standards for storing dimensions, classification hierarchies and facts also some advanced, dynamic standards are dealt with, that allow to build a basic service infrastructure. Section 5 closes with an outlook how XCube can be combined with Web Services and some remarks on our prototype.

2. MOTIVATING USE CASES

Before explaining XCube this section is meant to sketch some typical use cases that could benefit from our approach. The driving force between all the scenarios presented below is that many data warehouses exist containing lots of valuable information spread over different departments of the same company or even over different companies. The exchange of data between these heterogeneous systems could be done much easier if there was a unique format for describing the data warehouse data in a standardized way.

2.1 The "Download" Use Case

A web server offers data warehouse cubes for download (figure 1). Every client data warehouse interested in this data can download it and integrate it into its local database. First of all the description of the multidimensional schema of the cube is downloaded, in a second step the master data is exchanged, i.e. the classification hierarchies of the single dimensions. Finally the transaction data is sent to the requestor and integrated into its local data warehouse. The integration on the client side is easy as long as any expression in the new cube does not conflict with those of existing data cubes. If on the other hand for example a dimension already exists the exact relationship between the two has to be carefully checked, which leads to the problem of semantic integration. An interesting tool for integrating data from various sources could be the use of standardized reference dimensions.

2.2 The "Query" Use Case

As the amount of data of a multidimensional cube is usually rather large (especially the transaction data part) the possibility to analyze data cubes online, i.e. on the (web) server side, is highly desirable to reduce the consumption of network capacities. In the online query case only schema and master data have to be downloaded from the web server completely which are smaller by magnitudes (steps 1 and 2 in figure 2). With this description in hand the client application (not only data warehouses but e.g. OLAP tools) can decide which subset of the data cube it needs and can then send an according query to the web server (step 3). The server then computes the desired transaction data subcube and sends it to the client (step 4). If the client application is a data warehouse that wants to integrate the result into its own database, this use case is a generalization of the download case presented in section 2.1.

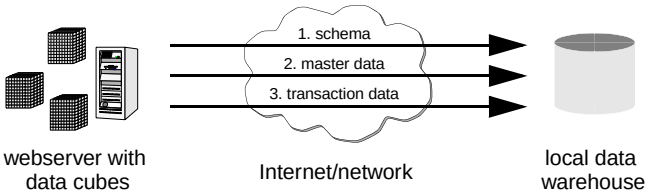

Figure 1. Use Case "Download"

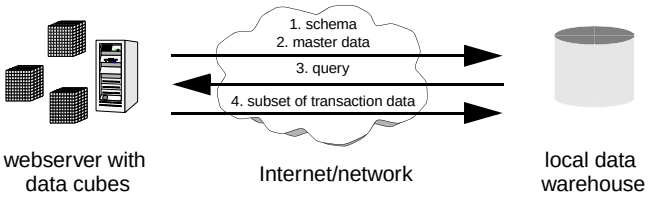

Figure 2. Use Case "Query"

For the query scenario it is important to think about an update strategy for schema and master data on the client side because when the client keeps this data for a longer time it can become outdated. The problem of conflicting expressions has to be treated in the same way as mentioned above.

2.3 The "Generating" Use Case

The generating case is about how to create data cubes that can be offered by a web server. In principle data of any format from all kinds of data sources can be converted to a multidimensional cube. The conversion process is the same as when integrating a new source into a warehouse: the data has to undergo the complete ETL workflow.

The more interesting case is generating an online cube from an existing data warehouse. Here the expensive integration task is already done and it only has to be decided which subset of the data is to be published. The transformation of multidimensional data (independent of relational or multidimensional storage) is simple and can be done automatically. For implementing this case the introduction of a data mart holding only that subset of data that is meant for going online might be useful. Another interesting design choice for implementation is if transforming the warehouse data into online cubes is done statically or dynamically on demand by a set of SQL queries.

2.4 Requirements for Representing Online Data Cubes

The last three subsections gave a set of examples motivating the necessity of a web based format for exchanging data cubes. This paper describes XCube, a family of XML schemas to precisely describe these online cubes. As the cubes are supposed to be transferred over the Internet and contain highly structured data, using XML ([1], [2]) seems reasonable. Before explaining the XCube standards in the next section a list of requirements to a format expressing online data cubes is introduced which can be easily derived from the scenarios presented above.

1. The format has to support a multidimensional data model.
2. The conceptual distinction between the description of schema, master or dimension data and transaction or fact data has to be supported.
3. The format has to be transportable over a network, primarily over the Internet.
4. To achieve a high level of flexibility and reuse the format has to support linking and inclusion concepts.
5. The format should be extensible to be able to adapt to different data models or to introduce new concepts.

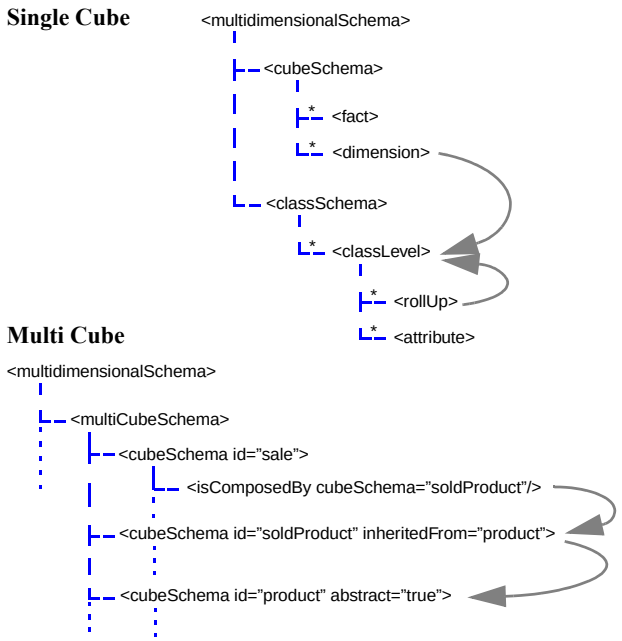

Figure 3. Two structures of XCubeSchema documents

6. The format must be easily convertible to and from various data sources and formats.
7. The format could possibly allow online analytical processing (OLAP) to reduce the amount of data to be transferred over the network.

XCube is a family of XML based formats that fulfills these requirements. Especially points 3 to 6 suggest XML as the meta format of choice. Standards like XLink ([9]) and XPointer ([10]) allow to connect remote data sources and schemas. By applying XSLT-Stylesheets the XML presentation of an online data cube can be converted to a vast number of other data formats.

The most challenging point above is the first one: unfortunately there is no unique multidimensional data model. Interesting proposals for a standard multidimensional data model can be found e.g. in [11], [12] or [13]. XCube is designed especially with the features of mUML ([13]) in mind, but because of XCube's flexibility it is compatible with other data models as well.

As a direct consequence of the second and the fifth point an appropriate format for representing data cubes cannot be a single format but rather a family of specialized formats that are linked together. These formats will be described in detail in the following sections. As a format like this is first of all meant for exchanging data between several data warehouses point 7 is only suggested as optional, but a certain subset of OLAP functionality could be useful.

2.5 Related Work

To the authors' knowledge only few research has been done on creating an XML standard for storing multidimensional data and what benefits such a standard could offer.

With the Common Warehouse Metamodel (CWM, [14]) the Object Management Group (OMG, [15]) creates a standard format for data warehouse meta data. Based on the UML it offers a complex model for describing data warehouse meta data; dimension and fact data are completely left out. The main goal of CWM is not the exchange of data cubes between data warehouses but to create a standard interface to data warehouses that every kind of tools can access, e.g. OLAP tools, ETL tools etc. More detailed information on CWM can be found in [16].

Another proposal for a meta data standard for data warehouses is MetaCube-X ([17]) which is based on XML. Similarly to CWM it only concentrates on meta data and does not take care of the fact data, i.e. the data cube itself. There is no separation between schema and dimension data, and because dimension data is encoded as XML element names, reusability is aggravated.

Following the growing popularity of Web Services, XML for Analysis ([18], [19]) allows SOAP and XML based access to remote OLAP systems. The current version of XML for Analysis is an XML wrapping of MDX, so at the moment this approach is limited to warehouses built on SQL Server. It allows to discover a server on several levels of detail, e.g. an overview of the available data cubes, which dimensions are involved and finally the fact data itself. However it is not sure if XML for Analysis will be accepted as a standard because of its tight relations to Microsoft based systems.

3. BASIC XCUBE FORMATS

The basis of XCube consists of XCubeSchema, XCubeDimension and XCubeFact. These three formats allow to completely describe a data cube with XCubeSchema holding the multidimensional schema, XCubeDimension the hierarchical structure of the dimensions involved and XCubeFact containing the fact data i.e. the cells of the cube.

3.1 XCubeSchema

XCubeSchema is the central format for describing the multidimensional structure of the data cube: it ties together the dimensions and the measures contained in a cube. The basic structure of an XCubeSchema document can be seen in figure 3 in the upper half. The dashed lines express a parent-child relationship, the asterisks indicate that the child can appear an arbitrary number of times (even zero times). Under the root, `multidimensionalSchema`, a `cubeSchema` block and a `classSchema` block can be found. The `cubeSchema` section contains a collection of facts and dimensions, while `classSchema` describes the classification levels of the dimensions involved. Besides `attribute` which determines the current level, `rollUp` points to the next higher classification level, thus implicitly defining the complete hierarchy. The connection between a dimension and the classification hierarchy is achieved by a reference from `cubeSchema/dimension` to `classSchema/class-Level`. A simple example of an XCubeSchema document according to the grammar presented above is depicted in figure 4.

The example describes a data cube named `sale` containing two facts (`sales` and `revenue`) set up by two dimensions (`geography` and `article`). For both dimensions their finest granularity is given (`branch` resp. `article`) which is a reference to the following classification schemas as mentioned before. The classification schema for `geography` consists of the `branch` level containing a `manager` attribute. It rolls up to the higher `city` level, that again rolls up to

```
<multidimensionalSchema version="0.4"
        xmlns="http://www.xcube-open.org/V0_4/XCubeSchema.xcsd">

        <cubeSchema id="sale">
                <fact id="sales"/>
                <fact id="revenue"/>
                <dimension id="geography" granularity="branch"/>
                <dimension id="product" granularity="article"/>
        </cubeSchema>

        <classSchema>
                <!-- geography -->
                <classLevel id="branch">
                        <attribute id="manager"/>
                        <rollUp toLevel="city"/>
                </classLevel>
                <classLevel id="city">
                        <rollUp toLevel="region"/>
                </classLevel>
                <!-- ... -->

                <!-- product -->
                <classLevel id="article">
                        <attribute id="articleName"/>
                        <attribute id="brand"/>
                        <rollUp toLevel="productGroup"/>
                </classLevel>
                <classLevel id="productGroup">
                        <rollUp toLevel="productFamily"/>
                </classLevel>
                <!-- ... -->
        </classSchema>

</multidimensionalSchema>
```

Figure 4. Simple XCubeSchema document

`region` and so on. The same is done for the `product` dimension starting with the `article` level.

Besides expressing simple multidimensional data cubes as shown above XCubeSchema is enriched by a set of special features some of which will be briefly introduced in the following:

- Instead of simple cubes XCubeSchema can also represent Multi Cubes by replacing the `cubeSchema` element under `multidimensionalSchema` in the upper half of figure 3 by the `multiCubeSchema` element that in turn collects a number of standard `cubeSchema`s. Thus it is possible to store several related cubes in one schema. This is especially interesting because XCube allows to create `cubeSchema`s by inheritance or composition of other `cubeSchema`s. The Multi Cube in the lower half of figure 3 contains a `sale` cube that is composed of the `soldProduct` cube which in turn is inherited from an abstract `product` cube. Please note that the Multi Cube structure depicted in the lower half of figure 3 is not a pure schema but an instance.

- XCube also allows the definition of data types and units for measures, classification nodes and attributes. It is possible to use the data types defined by XML Schema ([2]) or to create own types and measures by using simpleType-elements as defined by XML Schema.

- Further interesting features for a multidimensional data model provided by XCube is the possibility to define several aggregation operations based on the measure or the dimension, the ability to use computed measures and the special treatment of time dimensions. For further details the reader is referred to [20].

3.2 XCubeDimension

While XCubeSchema describes the multi-dimensional aspects of a data cube, XCube-Dimension ([21]) is meant to formalize the dimensional structures. In essence an XCube-Dimension document contains the nodes belonging to the classification levels defined in XCubeSchema thus instanciating the hierarchies of the dimensions. The basic structure of every XCubeDimension document is depicted in figure 5. The root element, `dimensionData`, has two children, `units` for importing units defined in the XCubeSchema (see above), and `classification`. The latter carries the real payload which is a set of `node`s for each classification level `level`. Besides the set of attributes that are characteristic for a certain node, the node also has to implement the roll-up-relationships given by the corresponding XCubeSchema document which is done by the `rollUp`-element linking to another node on a higher classification level. It is left to the application logic to enforce that the roll-up-relationships of the nodes are consistent with the roll-up-relationships of the classification levels. Please

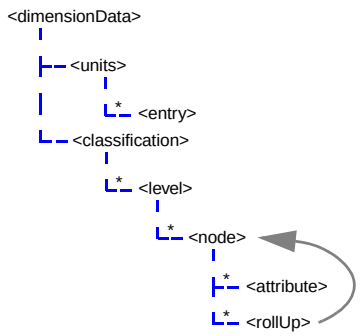

Figure 5. Structure of an XCubeDimension document

```
<dimensionData version="0.4"
    xmlns="http://www.xcube-open.org/V0_4/
    XCubeDimension_base.xcsd">

    <units>
        <entry unitType="currency" unit="EUR"/>
    </units>

    <classification>

        <!-- dimension: geography -->
        <level id="country">
            <node id="Germany"/>
            <node id="Switzerland"/>
            <node id="France"/>
            <!-- ... -->
        </level>

        <level id="region">
            <node id="Northern Germany">
                <rollUp toNode="Germany" level="country"/>
            </node>
            <node id="Western Germany">
                <rollUp toNode="Germany" level="country"/>
            </node>
            <node id="Eastern Germany">
                <rollUp toNode="Germany" level="country"/>
            </node>
            <node id="Southern Germnamy">
                <rollUp toNode="Germany" level="country"/>
            </node>
            <!-- ... -->
        </level>

        <!-- ... -->

    </classification>

</dimensionData>
```

Figure 6. Example of an XCubeDimension document

note that though the roll-up-relationships in figure 5 look very similar to those in figure 3, they are not redundant at all: while the roll-ups in XCubeSchema documents express relationships between classification levels e.g. *branch – city*, those in XCubeDimension documents express the relationships between nodes, i.e. instances of the levels as with *Northern Germany – Germany*.

An example of an XCubeDimension document related to the XCubeSchema from figure 4 can be found in figure 6. After defining EUR as a specialized currency unit, parts of the geography dimension are presented: *Germany*, *Switzerland* and *France* are instances of the country classification level. Several parts of Germany are shown as instances of the region level all rolling up to the country node *Germany*.

Due to the lack of space the extended features of XCubeDimension are only briefly mentioned here. When constructing dimensions naming collisions might occur, therefore XCubeDimension introduces the concept of keys: every node can be provided with a unique key thus resolving the naming problem. Several data warehouses support shared roll-ups, i.e. a certain node, e.g. a modern mobile phone, can be rolled up to several higher nodes, e.g. *mobile phone* and *digital camera* ([22]). Although situations like that should be avoided by appropriate modelling XCubeDimension allows to specify weights for different higher classification nodes. Similarly to XCubeSchema XCubeDimension supports default time dimensions.

3.3 XCubeFact

After describing the dimensions and how they play together, the cells of the data cubes can be defined by using the XCubeFact schema ([23]) which is outlined in figure 7. For each `cube` a collection of `cell`s is defined. Each `cell` consists of two parts, `dimension` representing the multidimensional coordinates, i.e. the links to the dimensions and their corresponding classification nodes, and the `fact` values themselves (as each multidimensional cube can store several facts, also each cell has to be able to hold several facts).

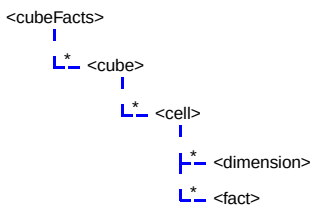

Figure 7. Structure of an XCubeFact document

A sample document further clarifying the use of XCubeFact can be seen in figure 8. For the sale cube constructed in the code examples above two cells are shown: sales and revenue values are given for the product *MA-450* in two different shops (*branch48* and *branch75*) on 24th July 2003.

3.4 Summary

In this section the core of XCube is presented. It consists of a set of three XML Schemas responsible for expressing the multidimensional schema, the single dimensions and the fact values. One reason for this decomposition is the possibility of reusing some of these documents, e.g. an XCubeDimension document can be shared by several cubes or even applications. Another reason is the attempt to deal with the different multidimensional terminologies. Perhaps it is possible to integrate these diversified streams by introducing an open XML standard for data warehouses.

```
<cubeFacts version="0.4"
    xmlns="http://www.xcube-open.org/V0_4/XCubeFact_base.xcsd">

    <cube id="sale">
        <cell>
            <dimension id="geography" node="branch48"/>
            <dimension id="product" node="MA-450"/>
            <dimension id="time" node="2003-07-24"/>
            <fact id="sales" value="3"/>
            <fact id="revenue" value="960"/>
        </cell>


        <cell>
            <dimension id="geography" node="branch75"/>
            <dimension id="product" node="MA-450"/>
            <dimension id="time" node="2003-07-24"/>
            <fact id="sales" value="2"/>
            <fact id="revenue" value="640"/>
        </cell>
        <!-- ... -->
    </cube>

    <!-- ... -->

</cubeFacts>
```

Figure 8. Example of an XCubeFact document

It is further important to note that the three schemas expose no redundancies. Though several elements appear in all schemas they are realized as references (here XML is quite similar to relational database systems).

4. XCUBE EXTENDED STANDARDS

Besides the basic description of data cubes XCube contains advanced formats like XCubeText, XCubeQuery and XCubeFunction. XCubeText allows to add textual descriptions and comments to XCubeSchema- and XCube-Dimension-files. While the formats so far are static and meant for describing a cube, XCubeQuery and XCubeFunction are dynamic. XCubeQuery allows to issue queries to a data cube and its meta data over the network, while XCubeFunction is designed to find out the capabilities of a server holding a cube.

4.1 XCubeText

XCubeText ([24]) is meant to add textual descriptions and comments to nearly every element of XCubeSchema- and XCubeDimension documents. The current version 0.4 of XCube only allows to insert these text blocks directly into these structures; a future version will allow to extract these descriptions into its own files. XCubeText allows several degrees of detail by applying the elements `short`, `medium`, `long` and `html`.

The decision to collect all XCubeText-comments in one or more external files gives rise to several new possibilities, e.g. multi-language support or different comments for different application domains.

4.2 XCubeQuery

XCubeQuery ([25]) is one of the dynamic formats of the XCube family. With Web Services ([26]) growing more and more popular the idea of interactively asking for (static) XCube documents is not far away. XCubeQuery is meant to organize this interactive dialog between client and server. However XCubeQuery is not supposed to compete with full-fledged OLAP tools or XML query languages (XQuery etc.). Rather it might deliver a basis for a more efficient exchange of

data by narrowing down the amount of data needed on the client side. XCubeQuery consists of seven different query formats which are oriented along the process of data exploration and will be described in the following.

List of available cubes

The first question a client might ask is which cubes a server can offer. This task is accomplished by the `getCubeSchema-List` element which does not need any further elements or attributes. The server replies with a list of `cubeSchema` elements known from XCubeSchema enumerating the IDs of the available cubes. An example can be seen in figure 9, where the server offers the cubes *sale*, *purchase* and *stock*.

```
<request>
        <getCubeSchemaList/>
</request>
```

request

response

```
<cubeSchema id="sale"/>
<cubeSchema id="purchase"/>
<cubeSchema id="stock"/>
```

Figure 9. Example for *getCubeSchemaList* and response

Getting the schema of a special cube

The client might want to explore one of the cubes from the result of the previous `getCubeSchemaList` query in more detail. That is done by sending an XCubeQuery document containing the `getCubeSchema` element with the ID of the desired cube (figure 10). The result consists of the major part of an XCubeSchema document containing a list of `facts` and a list of `dimensions` including their granularities. Additionally the `dataTypes` and `unitTypes` involved are included.

In contrast to a complete XCubeSchema document as introduced in section 3 the classification schema is missing. The reason for that is to minimize the amount of data to be transferred over the network as much as possible. The client still might decide that the cube he is just querying is not applicable to his needs.

```
<request>
        <getCubeSchema id="sale">
</request>
```

request

response

```
<cubeSchema id="sale"
        xmlns:xs="http//www.w3.org/2001/XMLSchema">
        <fact id="sales">
                <defaultAggregate>
                        <aggregation operator="sum"/>
                        <aggregation operator="max"/>
                        <aggregation operator="min"/>
                </defaultAggregate>
        </fact>
        <fact id="revenue">
        <dimension id="geography" granularity="branch"/>
        <dimension id="product" granularity="article"/>
        <dimension id="time" granularity="xs:date" stdLevel="true"/>
</cubeSchema>
<dataTypes/>
<unitTypes/>
```

Figure 10. Example of *getCubeSchema* and response

```
<request>
        <getClassNodes level="branch"/>
</request>
```

request

response

```
<level id="branch">
        <node id="branch48">
                <rollUp toNode="Frankfurt" level="city"/>
                <attribute id="manager" value="Meier"/>
        </node>
        <node id="branch75">
                <rollUp toNode="Frankfurt" level="city"/>
                <attribute id="manager" value="Bauer"/>
        </node>
        <!-- ... -->
</level>
```

Figure 11. Example of *getClassNodes* and response

Querying the Classification Schema

The next step in exploring the data cube consists of asking for the detailed classification schemas of the dimensions. Therefore the client sends a document containing the `getClassSchema` element to the server containing a list of desired dimensions (this list has to be a subset of the set of dimensions returned by the previous `getCubeSchema` query). A possible request together with a valid response can be found in figure 12: a client is interested in the classification schemas for the dimensions *time* and *geography*. As a result the server returns the standard time classification levels *month* (*xs:gYearMonth*) and *quarter* for the *time* dimension. For *geography* the levels *branch*, *city*, *region* and *country* are returned. Further the response includes the definition of the

```
<request>
        <getClassSchema>
                <dimension id="time"/>
                <dimension id="geography"/>
        </getClassSchema>
</request>
```

request

response

```
<classSchema xmlns:="http://www.w3.org/2001/XMLSchema">
        <stdTimeClassLevel id="xs:gYearMonth">
                <rollUp toLevel="quarter"/>
        </stdTimeClassLevel>
        <timeClassLevel id="quarter" timeBase="quarter">
                <rollUp toLevel="xs:gYear" stdLevel="true"/>
        </timeClassLevel>

        <classLevel id="branch">
                <attribute id="manager"/>
                <rollUp toLevel="city"/>
        </classLevel>
        <classLevel id="city">
                <rollUp toLevel="region"/>
                <addKey level="region"/>
        </classLevel>
        <classLevel id="region">
                <rollUp toLevel="country"/>
        </classLevel>
        <classLevel id="country"/>
</classSchema>
<dataTypes>
        <dataType name="quarter">
                <xs:restriction base="xs:gYearMonth">
                        <xs:pattern value="[0-9]{4}-0[1-4]"/>
                </xs:restriction>
        </dataType>
</dataTypes>
<unitTypes/>
```

Figure 12. Example of *getClassSchema* and response

data type *quarter*, which is a restriction of *xs:gYearMonth*. The `unitTypes` clause in this example is empty.

Querying Classification Nodes

After having found out the dimensions and their classification hierarchies a client has gathered enough knowledge to request nodes of certain classification levels. This can be achieved by sending an XCubeQuery document consisting of a `getClassNodes` element to find out all nodes of a given level, or containing a `getClassNode` element only asking for detailed information on a single node. An example of the first choice is given in figure 11 returning information on all branches including their attributes and roll-up-relationships.

Requesting the Facts

At this point the client should have gathered all necessary data to decide which part of the data cube he needs. The `getFacts` request displayed in figure 13 is obviously the most complex of all XCubeQuery elements. It contains an arbitrary number of cubes, a set of dimensions optionally restricted to certain classification nodes, and a number of facts. The example in figure 14 asks for the subsection of the *sale* cube defined by the dimensions *geography* and *product*; only the *revenue* fact is of interest and *geography* is restricted to two branches, *branch48* and *branch75*. The functionality of `getFacts` is rather similar to the OLAP operations slice & dice ([27]). But it is important to know that these restrictions can only be applied to the basic granularity of the cubes, i.e. roll-up or drill-down operations

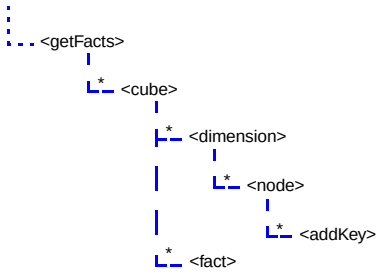

Figure 13. Structure of XCubeQuery.getFacts

```
<request>
    <getFacts>
        <cube id="sale">
            <dimension id="geography" all="false">
                <node id="branch48"/>
                <node id="branch75"/>
            </dimension>
            <dimension id="product" all="true"/>
            <fact id="revenue"/>
        </cube>
    </getFacts>
</request>
```

request

response

```
<cube id="sale">
    <cell>
        <dimension id="geography" node="branch48"/>
        <dimension id="product" node="MA-450"/>
        <fact id="revenue" value="960"/>
    </cell>
    <cell>
        <dimension id="geography" node="branch75"/>
        <dimension id="product" node="MA-450"/>
        <fact id="revenue" value="640"/>
    </cell>
    <!-- ... -->
</cube>
```

Figure 14. Example of *getFacts* and response

XCube format	Description
XCubeSchema	stores the description of the multidimensional schema of a data cube
XCubeDimension	stores hierarchical dimension data
XCubeFact	format for describing the fact data of a data cube
XCubeText	format to hold text blocks to allow multi-language support
XCubeQuery	query language for data cubes
XCubeFunction	allows to find out which services the data source offers

Table 1. Overview over the XCube family

are not supported. As mentioned before XCube is not meant to be another OLAP dialect but only a data format for exchanging multidimensional data over the network. The result of the `getFacts` request is a set of `cell` elements consisting of multidimensional coordinates (`dimension`s with `node`s) and the according `fact` value(s).

4.3 XCubeFunction

The latest format of XCube is XCubeFunction which is still under development, so here only a brief outline can be given. The main idea behind this format is the ability to query an XCube server about its functionality. Primitive servers might only be able to deliver complete cubes, i.e. no restriction with regards to facts, dimensions or classification nodes is possible. Highly sophisticated servers on the other hand might even be able to process full-fledged OLAP queries including all kinds of multidimensional operations.

5. CONCLUSION AND OUTLOOK

In this paper we presented XCube, an open, vendor independent and modular family of XML based document templates for describing data cubes as they are common in data warehouse systems. A short overview is given in table 1. The most important structures are XCubeSchema, XCubeDimension and XCubeFact because these are the common ingredients of every multidimensional model. However all three of them are flexible enough to express special features like shared roll-ups or multicubes. By splitting the description of a data cube into three parts, a client can easily explore if a certain cube is interesting for him or not, especially because schema and dimension data are by far smaller than the huge amount of fact data.

As XCube is first of all meant to describe existing, multidimensional structures most of its templates are static (XCubeSchema, XCubeDimension, XCubeFact and XCubeText); but with XCubeQuery and XCubeFunction two dynamic document types are introduced that allow to discover the cubes and their structures step by step. They form a sound basis for a more complex and efficient infrastructure.

We think that this representation of cubes can be very useful, because the cubes are now easily transferrable from one data warehouse to another. The approach can easily be combined with the new Web Service paradigm (the XCube documents simply have to be included into the body of SOAP messages, and XCubeQuery has to be translated to WSDL code). At least the technical heterogeneity problems when attempting to

integrate data warehouses can be solved completely by applying XCube.

Another interesting point in XCube is the evolution of XCubeText. Extracting all descriptive texts into special files gives rise to supporting several languages or dialects for different application domains and thus making the same data cube useful for different professions.

A last idea for using XCube sketched here is to create new data warehouses according to the XCube standards. Originally XCube is meant to work on top of an existing warehouse, thus the system dependent representation of the data has to be transformed to XCube. For new data warehouse projects it might be worth thinking about using XCube right from the beginning for the conceptual modelling phase. The advantage would be an even easier way to integrate this new data into other warehouses. As a first step towards this direction one could think of introducing standard dimensions similar to the standard time dimension.

Finally we want to remark that XCube is not only a mere specification but that OFFIS and the University of Erlangen-Nuremberg have jointly developed a set of prototypes. We have written an export component that extracts the TPC-H scenario ([28]) from a MS SQL-Server 2000 to XCube; further we have developed a plug-in oriented import component named XCubeLoader that is able to import the TPC-H data in XCube representation into several database systems, among them MS Access and Oracle 9iR2. By writing new plug-ins against a well defined interface nearly every database system can be made XCube aware.

6. REFERENCES

[1] Bray, T.; Paoli, J.; Sperberg-McQueen, C. M.; Maler, E.: *Extensible Markup Language* (XML) 1.0 (Second Edition) W3C Recommendation 6 October 2000 http://www.w3.org/TR/REC-xml

[2] Fallside, D.C.: *XML Schema Part 0: Primer* W3C Recommendation 2 May 2001 http://www.w3.org/TR/xmlschema-0/

[3] Jarke, M.; Jeusfeld, M.A.; Quix, C.; Vassiliadis, P.: Architecture and Quality in Data Warehouses: An extended Repository Approach. In: *Information Systems*, 24(3), 1999

[4] Linthicum, D.S.: *Enterprise Application Integration*. Addison-Wesley, Boston, 2000

[5] Linthicum, D.S.: *B2B Application Integration*. Addison-Wesley, Boston, 2001

[6] http://www.xcube-open.org

[7] Bourret, R.: *XML Database Products*. 2003 http://www.rpbourret.com/xml/XMLDatabaseProds.htm

[8] Clarke, J.: *XSL Transformations* (XSLT) Version 1.0 W3C Recommendation, 16 November 1999 http://www.w3.org/TR/xslt

[9] DeRose, S.; Maler, E.; Orchard, D.: *XML Linking Language* (XLink) Version 1.0 W3C Recommendation 27 June 2001 http://www.w3.org/TR/xlink/

[10] Grosso, P.; Maler, E.; Marsh, J.; Walsh, N.: *XPointer Framework* W3C Recommendation 25 March 2003 http://www.w3.org/TR/2003/REC-xptr-framework-20030325/

[11] Albrecht, J.; Bauer, A.; Deyerling, O.; Günzel, H.; Hümmer, W.; Lehner, W.; Schlesinger, L.: Management of Multidimensional Aggregates for Efficient Online Analytical Processing. In: *IDEAS*, Montreal, Canada, 1999

[12] Sapia, C.; Blaschka, M.; Höfling, G.; Dinter, B.: Extending the E/R Model for the Multidimensional Paradigm. In: *Advances in Database Technologies*, Lecture Notes in Computer Science, Vol. 1552, Springer, Heidelberg, 1999

[13] Herden, O.; Harren, A.: MML und mUML – Sprache und Werkzeug zur Unterstützung des konzeptionellen Data Warehouse-Designs. In: *DMDW*, Magdeburg, Germany, 1999

[14] N.N.: *Common Warehouse Metamodel (CWM) Specification*. Version 1.0, OMG, 2001 http://www.omg.org/docs/ad/01-02-01.pdf

[15] http://www.omg.org

[16] Poole, J.; Chang, D.; Tolbert, D.; Mellor, D.: *Common Warehouse Metamodel: An Introduction to the Standard for Data Warehouse Integration*. John Wiley & Sons, New York, 2002

[17] Nguyen, T.B.; Tjoa, A.M.; Mangisengi, O.: MetaCube-X: An XML Metadata Foundation for Interoperability Search among Web Warehouses. In: *DMDW*, Interlaken, Switzerland, 2001

[18] N.N.: *XML for Analysis Specification*. Version 1.0. Microsoft Corporation, Hyperion Solutions Corporation, 2001 http://xmla.org/download.asp?id=2

[19] http://xmla.org

[20] Harde, G.: *XML Schemata for XCubeSchema*, version 0.4. 2002 http://www.xcube-open.org/V0_4/XCubeSchema.xcsd

[21] Harde, G.: *Basic XML Schemata for XCubeDimension*, version 0.4. 2002 http://www.xcube-open.org/V0_4/XCubeDimension.xcsd

[22] Hümmer, W.; Lehner, W.; Bauer, A.; Schlesinger, L.: A Decathlon in Multidimensional Modeling: Open Issues and Some Solutions. In: *DAWAK*, Aix-en-Provence, France, 2002

[23] Harde, G.: *Basic XML Schemata for XCubeFact*, version 0.4. 2002 http://www.xcube-open.org/V0_4/XCubeFact.xcsd

[24] Harde, G.: *XML Schemata for XCubeText*, version 0.4. 2002 http://www.xcube-open.org/V0_4/XCubeText.xcsd

[25] Harde, G.: *XML Schemata for XCubeQuery*, version 0.4. 2002 http://www.xcube-open.org/V0_4/XCubeQuery.xcsd

[26] http://www.w3.org/2002/ws/

[27] Kimball, R.; Ross, M.: *The Datawarehouse Toolkit*, 2nd ed. John Wiley & Sons, Chichester, 2002

[28] N.N.: *TPC Benchmark H* (Decision Support) Standard Specification, Revision 2.0.0. Transaction Processing Performance Council, 2002 http://www.tpc.org/tpch/spec/tpch2.0.0.pdf