Proceedings of the International Multiconference on
Computer Science and Information Technology, pp. 525 – 534

ISBN 978-83-60810-14-9
ISSN 1896-7094

Designing new XML Based Multidimensional Messaging Interface for the new XWarehouse Architecture

Ahmed Bahaa Farid
Helwan University, Faculty of Computers and Information, Cairo, Egypt
Email:
{Ahmed.Bahaa@gmail.com}

Prof.Dr.Ahmed Sharaf Aldin Ahmed
Helwan University, Faculty of Computers and Information, Cairo, Egypt
Email:
{Profase2000@yahoo.com}

Prof.Dr. Yehia Mostafa Helmy
Helwan University, Faculty of Computers and Information, Cairo, Egypt
Email:
{Ymhelmy@yahoo.com}

***Abstract*— OLAP (Online Analysis Processing) applications have very special requirements to the underlying multidimensional data that differs significantly from other areas of application (e.g. the existence of highly structured dimensions). In addition, providing access and search among multiple, heterogeneous, distributed and autonomous data warehouses, especially web warehouses, has become one of the leading issues in data warehouse research and industry. This paper proposes a new message interface for a new platform independent data warehouse architecture that can deliver location, platform, and schema transparency for clients that access autonomous data warehouses. The new message interface uses XML in order to provide interoperable way to query and administrate federated data warehouses in addition to compose the multidimensional query result sets.**

I. Introduction

A. Background

SINCE its evolution XML, accompanied with its related technologies (XQuery, XPath, XSL,…etc), has been considered the main standardized technology for the data exchange over information networks [1], [2]. By the time, the XML usage has increased with many applications. Recently, XML has significantly influenced building databases [2]. XML data is generated by applications and it can be consumed by applications. It is not too hard to imagine that some data sources in the enterprise are repositories of XML data or that they are viewed as XML data independently on their inner implementation.

In this case we could try to build a new data warehouse (DW) architecture that uses XML as its base for designing the messaging interface for that new architecture. In [3] a new data warehouse architecture (that is called XWarehouse) has been introduced. This architecture proposes the idea of utilizing XML as well as other design ideas in order to be able to build a platform independent multi- federated DW. In [3] the need for XML based messaging interface that has been called `eXtensible multiDimensional XML XDXML` has been introduced. As being part of the XWarehouse architecture, XDXML purpose is to compose all request, and response messages between the XWarehouse clients and orchestration. This will gives the opportunity to establish a communication between DW clients and a DW server with no regard to the platform compatibility issues. Moreover, XDXML provides the multidimensional format needed for caching the Multi-federated DW data as well as the DW Metadata on the Warehouse orchestration server that is responsible of orchestrating the multiple federated incompatible data warehouse at the backend with the DW query requests by the XWarehouse clients. In consequence, a need for dimensional modeling of XML data is appearing through the introduction of the XDXML. This paper proposes the XML based multidimensional messaging interface (XDXML) that the proposed XWarehouse uses. The paper depicts XDXML design goals, its basic structure, its embedded support for multi dimensions, its new operators, and the active capabilities into it. This XML based interface has been implemented into a real world case study that has been already presented previously [3].

B. Contribution

The contribution of this paper aims to propose a new Multidimensional messaging interface in order to interchange multidimensional data, schemas, queries, and other administrative commands over XWarehouse data warehouse architecture [3]. This interface will be called `eXtensible Multidimensional XML (XDXML)`. The remainder of this study is organized. Specifically; to describes the architecture by means of:

- Overview of the related research work
-The XDXML design objectives.
-The XDXML schema description.
-The XDXML active commands as well as predicates

C. Outline

This paper is organized as follows. At the beginning this paper presents a literature review that has conducted a survey about the previous efforts that tried to tackle using XML in dimensional modeling [4], [5]. The section analyzes each of those efforts and tells what is/are the negative point(s) into each proposed effort. Afterwards, the paper presents the

This work was supported into its practical side with Johnson & Johnson – Medical Egypt

main design objectives that the XDXML tries to fulfill. Prior to that, the paper delves into the XDXML basic multidimensional schema. In order present more details about the XDXML; the paper presents part of a real life implementation that has been used in order to validate the XDXML applicability. This helps in depicting the last part of the paper that depicts the XDXML ability to not only to compose multidimensional data as well as queries but also to send some administrative commands too.

D.Literature Review

The study has made a reviews for finding out the research efforts that has tackled the same problem domain, or part of it. For the time being, and as to the researcher knowledge, only few research efforts have been done regarding utilizing the XML [6], [7] in creating an architectural foundation for storing, administrating, and integrating data warehouse data (i.e. multidimensional data).

1)The Common Warehouse Metamodel (CWM)

The CWM is an Object Management Group (OMG) initiative. Its version 1.0 has been released in Feb. 2001. Through the CWM specifications[7], [8], OMG is targeting the creation of standard format for data warehouse Metadata based on a foundation Metamodel [4]. Based on the UML, CWM builds a complex model for describing data warehouse Metadata. The main goal of its specification is to create a standard interface to data warehouses that every vendor tool can access, e.g. OLAP tools [5], ETL tools etc. [8], [9], The specification concentrate on building a standard between vendor tools for data warehouse interchange based on certain XML format [9]. The data warehouse multidimensional data interchange is out of scope of this specification. The CWM XML package only contains XML based definitions for classes and associations that represent common data warehouse Metadata. Based on this, the CWM just targets to bridge the gaps between data warehouse tools in order to be able to work together but, it doesn't provide message interface for exchanging and querying data in an open environment [8] ,[10],[11], [12]. By other words, CWM is a data interchange format more than being a multidimensional message interface that includes action commands to take remote actions. This is left for the application level not the CWM itself.

2) MetaCube-X XML Metadata Foundation

The MetaCube-X is an XML instance of the Metacube concept, [11], [12]. While MetaCube is a conceptual multidimensional data model that some vendors are currently using (e.g. Informix, and Microstrategy), MetaCube-X seeks providing the user with a query mechanism for accessing information on the different web warehouses. This concept of MetaCube-X concentrate on defining an XML based schema for querying multidimensional data from different web warehouses [11]. Based on this MetaCube-X only concentrates on Metadata and doesn't take care of the data itself (the fact data as well as the dimensions)[6]. Moreover, when proposing the data MetaCube-X proposes it tightly coupled with its Metadata. By other words, it doesn't make separation between schema and the multi-dimensional data. The MetaCube-X whole contribution is only targeting querying activities. Similarly to CWM it doesn't utilize the web services technology to provide access to remote web warehouse, and OLAP systems. Finally it doesn't support a specific architecture to implement its model as a complete solution.

3) XML for Analysis (XMLA)

XMLA is an initiative that has been co-sponsored by Microsoft and Hyperion (SAS has joint them in April 2002). Its version 1.0 specification has been released in April 2001 [13], [14], [15]. This specification utilizes the popularity of web services in providing data warehouse users with SOAP and XML based access to remote OLAP systems. While proposing an appropriate architecture, XMLA is concerned with how to query multidimensional data as well as Metadata through the use of the md/XML interface [13]. By other words, XMLA along with md/XML is designed to retrieving data not for manipulating and recapitulating cubes [13][14],. Moreover, XMLA doesn't support querying the galaxy schema for data retrieval

4) XCube

XCube is a family of XML based document templates that aim to exchange data warehouse data (i.e. data cubes) over any kind of networks [17]. In spite of releasing a way to exchange data as well as format, XCube schema is dedicated for the purpose of querying the web warehouse data , not any other purpose (i.e. exploring cube facts and dimensions, and managing a web warehouse) [18]. Not only that but also, the XCube schema is complicated to the extent that it is hard to be processed. This means that it doesn't support multiple Hierarchies dimensions. Within the schema description there is no distinction between the dimension itself and its dimensional hierarchies. XCube main target is to supply a standard format for exchanging data warehouse data, not a complete architectural solution for managing, querying, and exchanging data. There is no support for utilizing XSD for Schema validation or XSLT for presentation layer flexibility. There is no concrete definition for an architecture that defines how to deploy this format, or where its parsers will be deployed. Subsequently, XCube doesn't discuss how could be a level of integration between federated data warehouses, or web warehouses. Finally, it is not concerned with granting a high level of access to the data warehouses through something like Thin OLAP (ThOLAP) that X-Warehouse.

5) INRIA's GEMO Project

GEMO is a three-years project that has been born from the merging of INRIA (Institut National De Rechercha En Informatique Et En Automatique). With Members of another multinational research group. This project depends heavily on the usage of XML related technologies in order to insure data exchange as well as management. One of the main application domains for this project is the data for data warehouses. The Gemo 2006 activity report that has been published by INRIA doesn't talk about tackling the idea of the integration between federated data warehouses [19]. In order to promote their project foundation, the members of INRIA has contributed in releasing many publications regarding there diversified research points.

6) Concluding Remarks

These are all the known scientific contributions regarding the study research point. As has been clarified each one of the five contributions has some weak points that is fulfilled in this paper's contribution. Based on the above review it could be seen that none of the depicted efforts has proposed a complete architecture that enables platform independent data warehouse architecture that can enable integrating multi-federated data warehouses together which could be accessed transparently. While not supported by some of the efforts depicted above, XDXML proposes this through its architecture (XWarehouse). In addition to that, XDXML supports querying Galaxy schema. Moreover, XDXML supports making remote actions over the remote cubes.

II. XDXML Design Goals

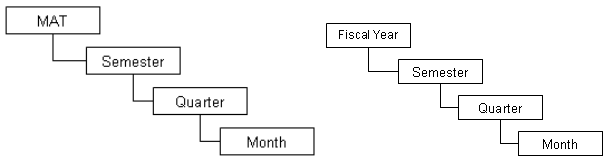
The XDXML format has been designed according to certain objectives. These objectives target functional as well as nonfunctional requirements. At the following are the main five design objectives that based on them the XDXML XML based Multidimensional Message Interface has been designed.

A. Minimizing Size

At the core of the XDXML, the well formed XML resides. The main nature of any well formed XML document is that it is being constructed hierarchal. The hierarchal structure in turn imposes a larger size than other ways of composing documents (e.g. relational). According to that the XDXML should try to minimize the redundancies that may appear into the single document without scarifying any required functional requirement. Minimizing the size of the XDXML data and commands can enhance the performance of sending requests and receiving responses to and from the X-Warehouse server. In addition to that it can minimize the time needed to parse

B. Supporting Multiple Hierarchies

In multidimensional design any cube can support multiple hierarchies [20], [21], [22]. This helps when same fact data is needed to get navigated with specific dimension according to multiple hierarchal points of view. As an example for that, for a pharmaceutical company they need to track the net sold amounts (after calculating all type of discounts) according to the time dimension. They have multiple time hierarchies Fig. 1 a, and b.



(a) Time Hierarchy 1 (b) Time Hierarchy 2

Fig. 1a: Time Dimension Hierarchy 1

The first hierarchy is concerned with dynamic periods. People at the pharmaceutical company have the need to track the sales values over the last 12 months (Moving Annual Time- MAT). This period is divided into two semesters (6 months). Each semester in turn is divided into 2 quarters (3 months) for each (see Fig. 1a). At the same time, people need to navigate the data for the current fiscal year, semester, and quarters. This brings up the second hierarchy (see Figure 1b). Incentives at the pharmaceutical companies are calculated based on each quarter sales and performance. Most of the work that has been proposed previously in this research field neglects delivering a schema that supports multiple hierarchies for the same dimension. This put them in a bad corner when it comes to the real implementation practices [18], [11].XDXML should deliver a multi-hierarchal dimension support.

C. Supporting Actions Declaration

The XDXML doesn't target only the regular decision makers. By other words it doesn't only provide DW reports and queries requests and responses but, it targets supporting DW administrators too. That is why the Proposed X-Warehouse XDXML format supports two action commands that could be sent by administrators and other querying users to perform certain actions on the server. These commands are *Act* and *Get*

D. Supporting New Dimensional Operators

XDXML should propose new dimensional operators. The new operators should deliver better querying capabilities. The new operators should help to get only data that is really needed.

E. Supporting Galaxy Schema

The multi-fact cubes are not regular cubes but, sometimes the business needs impose having it. This may happen when extending the data warehouse schema with new measures that changes the core approach that the business work on. At this time, it is highly recommended to overcome the problems that may arise at this time. Moreover, the multi fact cube could be beneficial when targeting to build the data warehouse according to the Ralph Kimball's `Bus architecture` [21], [22]. Multi-Fact cube creates a multi-star schema that is called Galaxy Schema, or Fact Constellation schema [23]. In Galaxy Schema same dimensions are utilized by more than one fact table. XDXML X-Warehouse messaging interface should support the Galaxy Schema that is not supported into any of the previous work [24].

III. The XDXML Schema

In order to fulfill the previous design objectives XDXML differentiates between the schema and the data itself. The schema could be explained separately through an XSD based file then the XDXML data comes accompanied with. At the following is an explanation of the XDXML schema, and how the data will get composed to.

A. The XDXML Cube

The XDXML Cube is the key component of the XDXML schema. The cube is the container of the rest of the multidimensional data that will be gotten from the server. For better network performance, the XWarehouse architecture implements the idea of `Cubes Client-Side Cashing`

(3Cs) [3]. The 3Cs feature imposes transferring data into cube format not a report format. This can minimize the size of data transferred which fulfils the first design objective for the XDXML protocol which is Minimizing. Moreover, transferring the requested data as facts and dimensions can give the user more flexibility to make some slicing and dicing operations while not being connected to the server. According to that the *XDXML Cube* consists of *XDXML Facts,* and one or more *XDXML Dimensions.* Fig. 2 shows the main structure of the XDXML Cube.

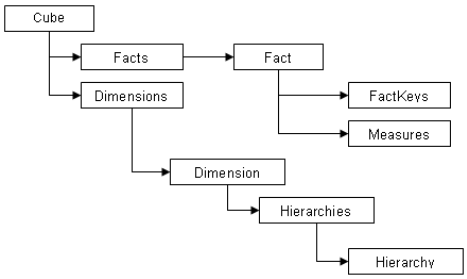


Fig. 2 The XDXML Cube main structural schema

As it is depicted each cube consists of one or many fact as well as one or many dimensions. As has been stated above the first design objective for the XDXML which is Minimizing Size has been fulfilled by transferring the multidimensional data not as cells but as facts and dimensions. This doesn't not only minimize the size of data received from the server but also, gives the flexibility to process the data on the client while being disconnect off the server. It is important to note here that each XDXML Cube node has an attribute called name for the cube name.

Fig. 3 Depicts the XDXML schema elements as classes using the UML based Class diagram.

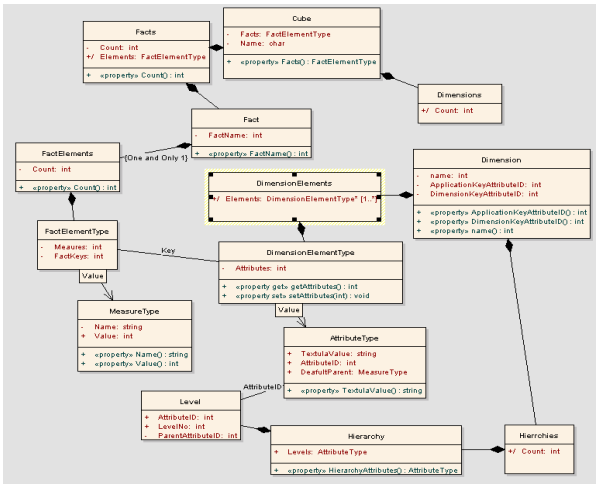


Fig. 3 A UML Representation for the main XDXML schema

Fig. 4 depicts the XML representation of the XDXML cube.

Fig. 4 the XDXML Cube Schema

B. The XDXML Fact

XDXML Fact is the heart of the cube. At most cases, cubes have just one fact table but, at some case their maybe more than one fact table. This happens in what is called galaxy schema. XDXML format supports both cases. Based on that XDXML schema may contain one or many cubes within the XDXML Facts. The Facts contains the sequence of XDXML Fact. Each Fact represents a fact table. Actually, this fulfils the fifth design objective for the XDXML protocol which is; Supporting Galaxy Schema. Each Fact consists of an attribute FactName and a Collection of FactElements. This is a of sequence of FactElement that represent one fact along with its measures and keys. That is why, each FactElement consists of two children elements; Measures element, and Factkeys element. The Factkeys element contains inside it all XDXML FactKey nodes. Each FactKey element node describes one of the FactKeys that connect the facts to their related dimensions. Each FactKey has two describing attributes. The Measures element contains a sequence of Measure nodes. Each Measure element has two attributes. Fig. 5 shows a fragment of the XDXML Cube schema that describes the XDXML Fact.

Figure 5: The XDXML Fact Schema

C. The XDXML Dimension

The Dimension element has a count attribute. This helps in defining explicitly how many dimensions is contained into any XDXML document that can enhance the parsing algorithm performance. Each XDXML Dimension element describes one dimension table. The Dimension element has two children, and three attributes. The first Attributes is the Name attribute (This attributes is not presented into figure 6 because of the space limitation), the second, and third attributes are attributes for declaring the Dimension Key, and the Application Key. These are DimensionKeyAttributeID, and ApplicationKeyAttributeID attributes respectively.

The first child is the `Attributes` element of `Attributes` type. The Attributes type is a sequence of XDXML Attribute element that compose up the dimension table. Each `Attribute` element usage purpose is depicted into a XML attribute called Description describing what this is used for. In addition to that each attributes has four describing nodes. The `Name` node element states the attribute name. The `Value` node contains the attribute value. The `AttributeID` node grants each attribute an integer id. This ID helps in referring to each attribute for many purposes that will appear later. Using integer ID instead of the string attribute id (attribute name) to refer for the XDXML attribute can enhance the parsing performance as well as minimizing the XDXML size. It is important here to state that the Dimension Key as well as the Dimension Application Key [21], [22] are included as two XML attribute elements. The values of these two attributes refer to the values of the AttributeID element node value within the two XDXML Attributes that act as Dimension ID and Application ID. The fourth attribute is The `Default Parent` node element is the AttributeID of the upper level for each attribute in the default hierarchy. Fig. 6 shows the XDXML Dimension Schema. The second child for *the XDXML Dimension* element node is the Hierarchies of a Hierarchies type. The Hierarchies Type contains a sequence of *XDXML Hierarchy* element nodes.

D. The XDXML Hierarchy

The `XDXML Hierarchy` is no more than a definition for how data will get organized. It has no more user data than that already exists into the dimension attributes. Instead, it organizes the existing attributes by specific organization only into certain granularity levels. According to that, the `XDXML Hierarchy` is a part of the `XDXML Dimension` (see Fig. 6). As it is apparent at the figure, the XDXML Hierarchy schema contains one direct child node; the `Levels` element node of Levels type. In turn, the Levels element is a sequence of Level element nodes type. Each Level element node has one attribute that describe the Level number of each level in the hierarchy. The description of the attribute that resides at each level comes for two element nodes; the AttributeID and the ParentAttributeID that contains the AttributeID value for the attribute above it in the Hierarchy. This approach of describing the multi-hierarchal schema in the XDXML schema makes it distinctive of the other related work that talked about the multi-hierarchal schema because; it makes the Multi-hierarchal description on the schema itself not on the data, as others do [7],[9], [11], [18],. Again, this can minimize the size aggressively. The XDXML Hierarchy Schema fulfills the second design objective for that XDXML that has been stated above which is: *Supporting Multiple Hierarchies.*

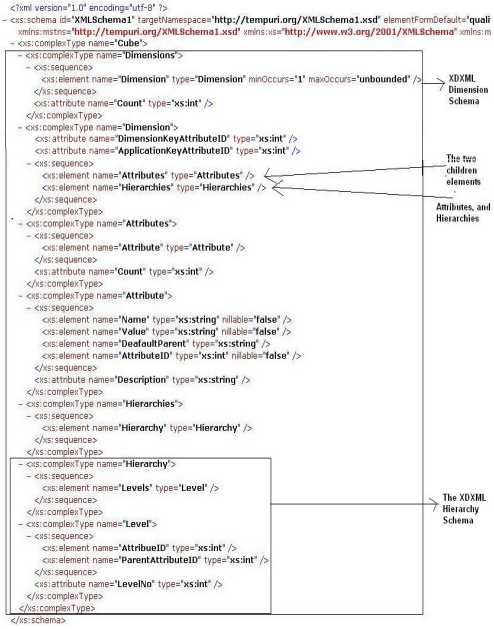


Figure 6: The XDXML Dimension Schema

IV. Applying the XDXML Schema on a Real Multidimensional Data

In order to validate this architecture, a case study has been conducted in one of the Medical equipment multinational companies in Egypt (Johnson & Johnson-J&J- medical Egypt), the architectural components have been belt using .NET C# code and then got deployed. The company hosts two data marts. The first one is the In Market Sales data marts (this includes facts about selling the devices and items from the distributor to the end-user) that keep the sales team distribution, Sales budgets and sales actual achievements. This data mart is about 2.54 GB in size. The Financial data mart is a 3.2 GB data mart that is hosted on an AS/400 server that is located on Europe (this includes the financial facts as well as the facts of sales between the company and distributors). This data mart is related to the internal sales between the Egyptian subsidiary and the EMEA headquarter. Previously all strategic planning used to be done on the numbers of the To Market sales. According to some deficiencies that has been discovered during the last two years, the corporate has changed its global strategy to make both the To-market Sales, and the In-Market Sales equally important. Based on this, new measures have been emerged based on the new interest of having the ability to process the internal To-Market sales(J&J/ Distributor) along with the In-Market Sales(distributor/end customer) . That is why a new need has been emerged to navigate to both of measurements together. All used examples below are extracted out of this case study as a try to highlight practical examples for using the

XDXML in exchanging a subset of a multidimensional data of a multinational medical company. This multinational medical company is specialized in producing and selling the medical materials (bondage, medical threads, plasters…). Fig. 7 depicts an overview of the multidimensional data that is retrieved from the orchestration server of this company. As it is clear from the figure, the cube is composed of one Fact called *SalesFact* and two dimension elements. The first one is the *Product_Dim* dimension while the second is the *Date_Dim*.

```
<?xml version="1.0" encoding="utf-8" ?>
- <Cube name="Sales">
  - <Facts Count="1">
    - <Fact FactName="SalesFact">
      + <FactElements count="1">
      </Fact>
    </Facts>
  - <Dimensions count="2">
    - <Dimension name="Product_Dim" DimensionKeyAttributeID="1">
      + <DimensionElements>
      - <Hierarchies>
        + <Hierarchy name="Franchise_ProductCode">
        </Hierarchies>
      </Dimension>
    - <Dimension name="Date_Dim" DimensionKeyAttributeID="1">
      + <DimensionElements>
      </Dimension>
    </Dimensions>
  </Cube>
```

Fig. 7 Practical Example for a XDXML Cube

A. The SalesFact XDXML Fact

For the sake of simplicity this fact has only one measure which is the *Qty*. Actually this depicts the quantity sold from the referred product at the referred time. These three pieces of data compose all together one fact that is recorded as one `FactElement`. Fig. 8 depicts the `SalesFact` XDXML fact. As it is opposed, This XDXML fact has just one fact element (this done to simplify the example) The fact element shows up that this XDXML has two keys that refer to two dimensions (remember that the cube has two dimensions that are referred to by these two keys). The first `FactKey` is the *Product* key that refers to the dimension key number 1 in the `Product_Dim` dimension. The second key is the Date key that refers to the dimension key number 1 in the *Date* dimension.

```
- <Fact FactName="SalesFact">
  - <FactElements count="1">
    - <FactElement>
      - <FactKeys>
          <FactKey name="Product" value="1" />
          <FactKey name="Date" value="1" />
        </FactKeys>
      - <measures>
          <Measure name="Qty" value="20" />
        </measures>
      </FactElement>
    </FactElements>
  </Fact>
```

Figure 8 The *SalesFact* XDXML Fact

B. Product_Dim XDXML Dimension

Fig. 9 depicts the Product_Dim XDXML dimension. As it is apparent from the figure, this dimension has one default hierarchy and another defined hierarchy. For the sake of simplicity just one dimension element has been defined here. As it is clear from the figure the only dimension element available has a dimension key with value 1. This is the same value for the fact key at the SalesFact XDXML fact. It is important here to highlight that the DefaultParentID element is used to represent the default hierarchy.

```
- <Dimension name="Product_Dim" DimensionKeyAttributeID="1">
  - <DimensionElements>
    - <DimensionElement DimensionKeyValue="1">
      - <Attributes>
        - <Attribute>
            <ID>0</ID>
            <name>"DimensionKey"</name>
            <Value>1</Value>
            <DefaultParentID>0</DefaultParentID>
          </Attribute>
        - <Attribute>
            <ID>1</ID>
            <name>"WWFranchise"</name>
            <Value>"Ethicon"</Value>
            <DefaultParentID>0</DefaultParentID>
          </Attribute>
        - <Attribute>
            <ID>11</ID>
            <name>Franchise</name>
            <Value>"GYNECARE"</Value>
            <DefaultParentID>1</DefaultParentID>
          </Attribute>
        - <Attribute>
            <ID>111</ID>
            <name>Major</name>
            <Value>"Uterine Surgery"</Value>
            <DefaultParentID>11</DefaultParentID>
          </Attribute>
        - <Attribute>
            <ID>1111</ID>
            <name>Minor</name>
            <Value>"Versapoint"</Value>
            <DefaultParentID>111</DefaultParentID>
          </Attribute>
        - <Attribute>
            <ID>11111</ID>
            <name>ProductCode</name>
            <Value>"480"</Value>
            <DefaultParentID>1111</DefaultParentID>
          </Attribute>
        </Attributes>
      </DimensionElement>
    </DimensionElements>
  + <Hierarchies>
  </Dimension>
```

Fig. 9 The Product_Dim XDXML Dimension

V. Establishing Actions Using XDMQueries

The third XDXML design objective imposes Supporting Actions Declaration. In order to fulfill this objective XDXML has introduced a way to query and administer the XWarehouse. This is done through `XDMQueries`**.** Actually, XDMQuery is part of the XDXML. This type of queries is sent within a XDMQuery XDXML node that can include *Get* and/or *Act* statements. The Get statement is primarily concerned with querying data and schemas, while the Act statement is concerned with performing some administrative actions on the remote cubes. Each query is sent within a request. The request may contain *Get* or *Act* statements. The answer is received within a XDMQuery Response tag.

A. The XDMQuery Get Statement

Using the `XDMQuery` Get command it is possible to explore data as well as schema. This is done through sending an XDMQuery Request and receiving the response using the XDXML protocol for the schema or/and the data. Get statement works by using some XDMQuery predicates. The first predicate is <GetServerCubes>. If the request is containing a query with a response containing a schema(Metadata) then the statement should contain the predicates inside a `XDMSchema` tag. If the query needs to respond by data, not a schema then, the query predicates should be contained inside a `XDMData` tag. As depicted as Fig.10 the `XDMSchema` and `XDMData` tags are containing the predicates. This helps in containing multiple predicates at the same request some to receive data and others to receive Metadata.

1) GetServerCubes Predicate(Schema only)

Fig.10 depicts the first XDMQuery Get statement predicate; the `GetServerCubes` predicate. This predicate is responsible of acquiring the server cubes available at specific

server. This could be unlimited to all available cubes on the server or limited to the cubes last updated at specific time.

```
- <XDMQuery Server='192.123.14.1' UID='username' PWD='password'>
  - <Request>
    - <Get>
      - <XDMSchema>
        - <GetServerCubes All='true/false'>
            <LastUpdateDate />
          </GetServerCubes>
        </XDMSchema>
      </Get>
    </Request>
  </XDMQuery>
```

(a) GetServerCubes Predicate Query

```
- <XDMQuery Server="192.123.14.1">
  - <Response>
    - <Get>
      - <XDMSchema>
        - <GetServerCubes>
          - <CubesList>
              <Cube name="sales" />
              <Cube name="Marketing" />
            </CubesList>
          </GetServerCubes>
        </XDMSchema>
      </Get>
    </Response>
  </XDMQuery>
```

(b) Query Response

Fig. 10 The GetServerCubes Request and Response in Get Statement

2) GetCube Predicate(Schema and data)

This predicate targets querying both; part of the specific cube data and/or its Metadata. As explained before, in order to define whether you need the schema for the required query or the Metadata you Should use either the XDM-Schema or the XDMData tags. Fig. 11(a) shows how to use the GetCube predicate within the Get Statement in order to query the Cube Metadata. Fig. 11(b) shows an Example for the GetCube predicate when is used to retrieve data. If nothing else has been specified, the default is to return the data with the most granular dimensional level. The example that is depicted below doesn't mention any granularity levels for the Product_Dim dimension. If nothing else has been mentioned, the default hierarchy is retrieved, and the most granular level is used. If nothing else has been specified, the default is to return the data with the most granular dimensional level. The example that is depicted above doesn't mention any granularity levels for the Product_Dim dimension. If nothing else has been mentioned, the default hierarchy is retrieved, and the most granular level is used.

```
- <XDMQuery Server='192.123.14.1' UID='username' PWD='password'>
  - <Request>
    - <Get>
      - <XDMSchema>
        - <GetCube name='sales'>
          - <FactsList>
              <Fact name='SalesFact' />
            </FactsList>
          - <DimensionsList>
              <Dimension name="Product_dim" />
            </DimensionsList>
          </GetCube>
        </XDMSchema>
      </Get>
    </Request>
  </XDMQuery>
```

(a)Retrieving Schema

```
- <XDMQuery Server='192.123.14.1' UID='username' PWD='password'>
  - <Request>
    - <Get>
      - <XDMData>
        - <GetCube name='sales'>
          - <Fact name='SalesFact'>
            - <Measures>
                <Measure name='Qty' />
              - <Constraint>
                  <Predicate>'>'</Predicate>
                  <Value>12</Value>
                </Constraint>
              </Measures>
            </Fact>
          - <DimensionsList>
              <Dimension name="Product_dim" />
              <Dimension name="Date_dim" />
            </DimensionsList>
          </GetCube>
        </XDMData>
      </Get>
    </Request>
  </XDMQuery>
```

(b) Retrieving Data

Figure 11 GetCube Predicate Request

B. The XDMQuery Act Statement

The *Act* is primarily directed to the administrators. By using the `Act` statement, it is possible to take action over existing cubes. One of the most important activities that the administrator may need to accomplish is to update his cubes. The `UpdateCubeData` is a predicate that could be used in order to initiate this task by the administrator remotely. For the time and resources limitation, this study presents only this predicate with the `Act` statement. More predicates could be provided for both the `Get` and the `Act` statements during future work.

Fig.12 depicts an example for the Act statement with the UpdateCubeData operator. As it is clear from the figure, the Act statement is enclosed inside a transaction. This to tell that all the statements contained inside the transaction elements are in just one transaction. The XDMQuery transaction can contain Act as well as Get statements. Enclosing Get statement inside a Transaction maybe helpful because that the XDMQuery may work in asynchronous mode. Based on this enclosing the Get statement inside a transaction will be a declaration that all enclosed statements will work as one batch. If one fails all will fail too. The Transaction purpose is clearer in Act statement. If one fails all actions taken on the cubes will rollback.

The *UpdateCubeData* predicate has one attribute to define the cube name. In addition to that it has one child element to define how data will be updated inside the cube. Whether data will be completely deleted and added again, or just the modified data will be overwritten. The former could be used if the cube data has changed massively at the base data warehouse since last update while, the later could be useful if there is no massive change in the data warehouse data.

Separating the update of each cube in separate Act statement rather than having Cubes child element will give the opportunity to make one cube got updated when others fail to perform their updates. If *All or Nothing* is needed it is possible to use the *XDMQuery Transaction* processing is needed. It is important the mention here that the cube updating alternative is supported if on ly the server side cubes management system supports this feature otherwise; the default cube updating method will take place.

```
- <XDMQuery Server="192.123.14.1" UID="username" PWD="password">
  - <Request>
    - <Transaction>
      - <Act>
        - <UpdateCubeData CubeName="sales">
            <Data UpdateType="OverriteChanges" />
          </UpdateCubeData>
        </Act>
      - <Act>
        - <UpdateCubeData CubeName="Finance">
            <Data UpdateType="Drop_Fill" />
          </UpdateCubeData>
        </Act>
      </Transaction>
    </Request>
  </XDMQuery>
```

Fig. 12 a Sample for the XDMQuery Act Statement

C. Presenting the Pump-Up and Dump-down new Multidimensional Operators for Get command

The *XDQuery* supports two new operators for better multidimensional querying. These operators are directed to minimize the hierarchy declarations. Using these operators, the user can ask for a retrieving the data based on certain schema but without having all its intermediate levels. As an example for this, the hierarchy shown in figure 10(a) is depicting the default dimension hierarchy which is; WWFranchise/Franchise/Major/Minor/ProductCode. The question her is, what if the decision maker needs to retrieve the data according to the following Hierarchy; WWFranchise/Franchise/ProductCode? In order to fulfill this requirement, it is needed to define a

new hierarchy with this structure. The new XDQuery operators help at this case. The DumpDown operator is responsible of aggregating data to one of the indirect parent of specific level. This means that when using the PumpUp operator, the query should define the most granular level and which parent level is direct aggregating the data. By other words using the DumpDown and PumpUp operators, the same required results could be gotten but DumpDown will retrieve some extra data that are not retrieved by PumpUp operator. This is means that using the same facts/dimensions/ level names Dump-Down data set is s upper se t of the PumpUp data set. The other difference is the hierarchy level that is used to refer to the data; whether it is an upper level or lower one. Figure 13 shows a sample hierarchy of the case study in subject of this study. This hierarchy is based on the same data of the previous figures.

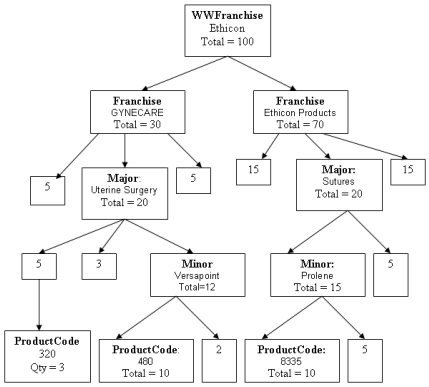


Fig. 13 Sample Data Hierarchy for the XDXML Cube Presented in figure 7 and all its subsequent figures

1) The Pump-Up XDMQuery Operator

Fig.14 depicts syntax for using the PumpUp operator. The *Get* is the *XDMQuery's Get statement*. *With* is a reserved word for defining specific dimensions. *For* is another reserved word for defining specific value for the required hierarchy level.

Get Sales**:** SalesFact(Qty)
With *Product-Dim*
On ProductCode ***PumpUp*** Franchise
For ProductCode = 8335

Fig 14 an example for Using PumpUp XDQuery Operator

According to figure 15, the result for this query will be as depicted in figure 16 but using XDXML. Figure 17 shows the same result but in XDXML The importance of using DumpDown and DumpUp operators comes from the fact that they minimize the amount of data that could be retrieved because it gives the ability to omit the unwanted intermediate levels of dimensional levels

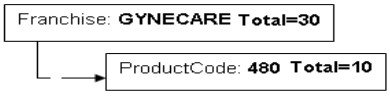


Fig. 15 Block Diagram for the Result of Query Shown in Fig.14

Fig 16 The XDXML Cube of the Get Statement at Fig 15

2) Dump-Down XDMQuery Format

Fig. 17 shows the *DumpDown* operator. As presented, the *DumpDown* operator is a way that can be used to omit a lot of unneeded data. Fig.17 expresses that the user needs only to get the Qty Measure that is inside the SalesFact using the Franchise Level and its children up to the Product Code directly at the Product_Dim dimension. The result for this query is depicted in Fig. 18. As it is clear the main difference between *DumpDown* and *PumpUp* operators is that, *PumpUp* operator is used to get certain upper level data along with its all children at the dimensional hierarchy, while the *PumpUp* operator retrieves certain measurement value according to its value for certain leaf node value at specific hierarchy along with its defined parent at certain level. According to that, *DumpDown* will most probably returns amount of data that is larger in size of that returned by the PumpUp if same level names have been used at both statements.

Get Sales: SalesFact(Qty)
With Product-Dim
On Franchise ***DumpDown*** ProductCode
For Franchise = GYNECARE

Fig. 17 An example for Using DumpDown XDMQuery Operator

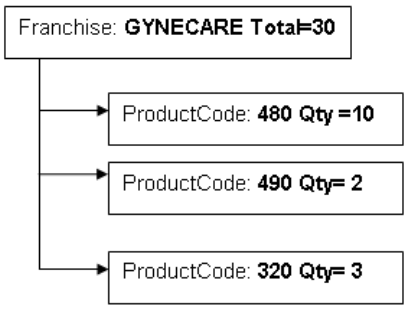


Fig.18 The XDXML Cube that represents the result of the Get Statement at Fig. 17

VI. Future Work and Conclusion

A. Future Work

This scientific work could be extended at the future by many ways. The following are just some examples of the potential future work:

-Designing more predicates for the existing Get and Act commands. These predicates could be related to more administrative tasks (e.g. building and dropping cubes)

-Taking in considerations the security aspect of XDXML as this paper doesn't discuss that aspect.

-Implementing optimized parsing algorithm for parsing XDXML at the XWarehouse client-side as well as service-side.

-Extending XDMQuery in order to be able to query available mining models.

-Extending XDXML to include querying data mining models [23].

B. Conclusion

This work presented a message Interface for exchanging data as well as commands over the XWarehouse Architecture [19] using XML as well as its related technologies (e.g. XSD, XSLT,…). The XWarehouse architecture is a newly proposed data warehousing architecture that uses Web Warehousing infrastructure for building better platform independent and more integrated and accessible data warehouses. Web warehousing refers to whether the use of the power of web infrastructure in architecting data warehouses, or the use of data warehouse techniques into keeping historical data about the click stream backlog [22]. In the case of XWarehouse first meaning is the intended and meant.

The paper used the XML technology as the base for building up the XDXML. XDXML used inside XWarehouse to get used by clients and servers for exchanging their data, schemas, and query as well as administrative commands. This paper is not concerned with the security details of the XDXML. XWarehouse uses http and SOA (Service Oriented Architecture) in order to build interoperable, vendor neutral data warehouse architecture that can enable clients as well as servers that depends on different platforms, DWMS (Data Warehouse Management Systems), and Operating systems to interoperate together transparently from the client. Moreover X-Warehouse architecture provides a way in order to integrate multi-federated data warehouse to act transparently as one logical data warehouse in front of the X-Warehouse clients.

This research has introduced the XDXML by providing a review for the related work that used XML for encapsulating data warehouse data and queries. Moreover, this paper tried to unveil the weak points for each effort that XDXML tries to overcome [23]. Moreover, the paper clarified what is the major design objectives that has been build based on. Presenting the design objectives has illustrated how does XDXML overcomes all the depicted weak points at the related research efforts surveyed. Prior to that, the paper began to delve into the detailed design of the XDXML by explain the key constructing components of the XDXML data schema. This paper has used the same case study at which the XWarehouse has been implemented in order to apply the XDXML. Part of the sample data of this case study has been used as an example for depicting examples of the XDXML Cube, XDXML Fact, and XDXML dimension schemas. All these schemas are mainly based on the XSD (XML Schema Definition). In addition to that the paper showed how XDXML can express action either for querying multidimensional data or to do some administrative actions at the server side through what has been called XDMQuery. XDMQuery is a dual-verb XML based language that is considered part of the XDXML. The two verbs are *Get* for querying data, and *Act* for sending action requests including administration actions. Get verb can use to newly proposed operators at this paper in order better retrieve what is needed accurately without any extra dimensional data member levels. Thus, reducing the retrieved the size of the XDXML which is one of its design goals.

It is important to mention here that the case study has been conducted to apply the X-Warehouse along with its XDXML messaging interface using a real life environment that is brought from the medical industry of one of the multinationals in Egypt In order to verify the applicability of the architecture [3]. The result showed that:

-The X-Warehouse Architecture is possible to get implemented and fulfilling its design goals.

Figure 19 depicts the deployed X-Warehouse architecture. As it is apparent, in addition to the regular desktop as well as web based access, this architecture permits mobile devices from accessing web warehouses data.

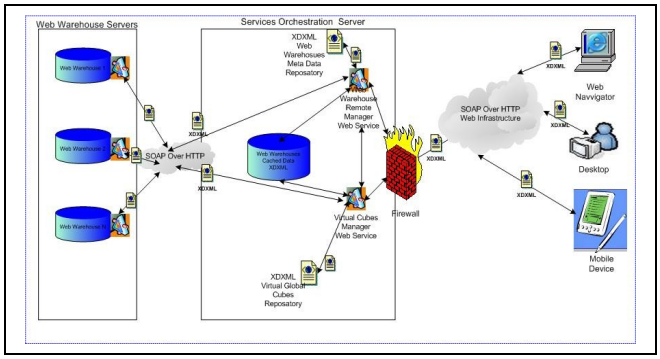


Fig 19 The X-Warehouse Architecture

-Using Caching in both server side as well as client side of the X-Warehouse architecture enhance the performance in average by 313%

-The performance overhead of using the X-Warehouse is in average just 5.11% which could be worthy to be incurred in order to solve the compelling problems that X-Warehouse tackles.

- The `ThOLAP` could be implemented based on the same architecture using the XDXML application protocol.

To conclude this work has successfully implemented the XDXML XML based XWarehouse Messaging Interface. A future studies will be made in order to more predicates to the Get and Act verbs for performing other administrative tasks. The XDXML at it is current state doesn't tackle how to query the data mining structures out from the server side.

This is expected to be the subject of one of the future research efforts.

REFERENCES

[1] Xiaogang Li and Gagan Agrawal, "*Efficient Evaluation of XML Over Streaming Data",* Proceedings of the 31st VLDB Conference, Trondheim, Norway, 2005.

[2] Wanhung Xu, Z. Meral Ozsoyoglu, *"Rewriting XPath Queries Using Materialized Views",* Proceedings of the 31st VLDB Conference, Trondheim, Norway, 2005

[3] Ahmed Bahaa, Ahmed Sharaf, and Yahia Helmy, *"Towards a New Platform Independent Data Warehouse Architecture Using Web, and XML Technologies",* 2008

[4] Bert Scalzo, "*Oracle® DBA Guide to Data Warehousing and Star Schemas",* Prentice Hall. 2003.

[5] E. F. Codd, S. B. Codd, and C. T. Salley, "*Providing OLAP (Online Analytical Processing) to User Analysts: An IT Mandate",* White Paper, Arbor Software Corporation, 1993.

[6] Dalamagas Theodore, etal, *"A methodology for clustering XML documents by structure, Information Systems",* Information Systems Journal No. 31, 2006, Elsevier B.V. P.187 -228.

[7] Pokorny Jaroslav. "*Modeling Stars Using XML",* Proceedings of DOLAP'01 ACM. Atlanta, United States, November 9, 2001.

[8] Kumpon Farpinyo, "*Designing and Creating Relational Schemas with a CWM-Based Tool",* ACM digital library, 2003.

[9] N. N.: "*Common Warehouse Meta Model Specification. Version 1.0",*.OMG Feb. 2001. TTTT http://www.omg.org/cgi-bin/apps/doc?ad/01-02-01.pdf

[10] Claudio Seidman, "*Data Mining with Microsoft® SQL Server™ 2000 Technical Reference",* Microsoft. 09/2001.

[11] T. B. Nguyen, A M. Tjoa, R. R. Wagner, "*Conceptual Multidimensional Data Model Based on Object-Oriented MetaCube",* Proceedings of the 2001 ACM Symposium on Applied Computing Las Vegas, Nevada, United States ,2001.

[12] Thanh Binh Nyguyen, A Min Tjoa, and Oscar Mangisengi, "*MetaCube-X: An XML MetaData Foundation for Interoperability Search Among Web Warehouses",* Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'2001). Switzerland, June 2001.

[13] N. N: "*XML for Analysis Specification. Version 1.0",* Microsoft Corporation, Hyperion Solutions Corporation, 2001.

[14] John Mikesell, *"Implementing the XML for Analysis Provider for SQL Server 2000 Analysis Services",* Microsoft Corporation. 2004.

[15] N. N: *"Designing and Implementing OLAP Solutions with Microsoft® SQL Server™ 2000, Course# 2074 Curriculum",* Microsoft Corporation, 2001

[16] N. N: *"Populating a Data Warehouse with Microsoft SQL Server™ 2000 Data Transformation Services",* Course# 2092 Curriculum, Microsoft Corporation, 2001

[17] Wolfgang Hümmer, Andreas Bauer, and Gunnar Harde., *"X- Cube-XML for Data Warehouses. Proceedings of the 6th ACM international Workshop on Data Warehousing and OLAP",* November 2003.

[18] R. A. Moeller. Distributed Data Warehousing Using Web Technology. AMACOM. 2001.

[19] N. N: Project-Team gemo: "*Management of Data and Knowledge Distributed Over the Web Activity Report".* INRIA, 2004

[20] Kimball Ralph, *"Laura Reeves, Margy Ross, and Warren Thornthwaite. The Data Warehouse Lifecycle Toolkit",* Second Edition John Wiley & Sons 2002.

[21] Kimball Ralph, and Margy Ross, *"The Data Warehouse Toolkit",* Second Edition John Wiley & Sons 2003.

[22] Kimball Ralph, and Richard Merz. *"The Data Webhouse Toolkit (Building the Web Enabled Data Warehouse)",* John Willey & Sons, 2000.

[23] Panos Vassiliadis, Timos Sellis. *"A Survey of Logical Models for OLAP Databases",* SIGMOD Record, Vol. 28, No. 4, December 1999.

[24] Micheleline Han, *"Data Mining Concepts and Techniques", Morgan Kaufmann",* 2004.