GMLA: A XML Schema for Integration and Exchange of Multidimensional-Geographical data

ROBSON DO NASCIMENTO FIDALGO[αβ]
JOEL DA SILVA[α]
VALÉRIA C. TIMES[α]
FERNANDO DA F. DE SOUZA[α]
ROBERTO SOUTO M. DE BARROS[α]
[α] Cin-UFPE - Centro de Informática – Universidade Federal do Pernambuco - Recife - PE - Brasil
{rdnf, js, vct, fdfd, roberto}@cin.ufpe.br
[β] CSI-FIR – Curso de Sistemas de Informação - Faculdade Integrada do Recife - Recife - PE - Brasil
rfidalgo@fir.br

Abstract. The integration among DW, OLAP and GIS has been given considerable attention in recent years by many researchers and industrial corporations. This may be a result of: 1) DW/OLAP can improve GIS spatial queries whereas, 2) a GIS can provide better support to deal with the DW/OLAP geographic data. Some research about this integration has already been done. However, these approaches do not deal with open and extensible solutions. In order to address this problem, we use the GOLAPA architecture, which follows Java, XML and Web Service technologies. Based on this architecture, this paper proposes GML for Analysis (GMLA), which provides a XML format to integrate and interchange geographical multidimensional data. GMLA is based on OLAP and GIS XML standards, namely XML for Analysis (XMLA) and Geography Markup Language (GML), respectively.

1 Introduction

Data Warehouse (DW) [1, 2, 3, 4] and On-Line Analytical Processing (OLAP) [1] are traditional technologies for decision support. On the one hand, DW is a database designed especially (e.g. star model) to support decision-making and basically it is organised over two types of tables: 1) dimensions - which mainly have descriptive data and 2) facts - which mainly have measuring data. According to Inmon [2] data in a data warehouse is subject-orientated (store data by theme), integrated (join data from heterogeneous sources), time-variant (maintain the historical data) and non-volatile (infrequently overrides the historical data). On the other hand, OLAP is a specific software category to make strategic queries over the DW data. Particularly, the OLAP queries occur with high performance and interactivity and frequently are named multidimensional, as its results are interpreted in structures such as a hypercube. In practice, this multidimensional structure represents the more relevant intersections among the DW data, which are handled by operations like: (1) Drill-up or Roll-up – data aggregation, (2) Drill-down – data desegregation, (3) Slice & dice – data selection & projection (4) Pivoting – data reorientation.

Another technology for decision support, but specifically inside a spatial context, is the Geographical Information Systems (GIS) [5, 6, 7]. These systems help you acquire, manipulate, examine and present the data which are geographically referenced and normally viewed into maps that can be combined, one above the other, in order to provide layers of geo-information. These layers are frequently known as themes or coverage and basically can be composed of two types of data format: vector and raster [8].

Developers of geographical application have awakened special interest considering the GIS use inside the Date Warehousing environment. This occurs mainly due to: the GIS side, the use of a DW with an OLAP tool can increase the power of spatial queries, and from DW/OLAP side, there is much geographical information (e.g. countries, states, addresses) that need a spatial support to be better discerned. Therefore, we can observe that the integration of these tools (DW, OLAP and GIS) is important for a good decision support, especially because it makes it possible to more properly answer questions that involve “what?”, “who?”, “when?” and “where?”.

An important advance in the Information Technology field, was the creation of the eXtensible Markup Language (XML) [20], which became a standard for publication and exchange of information on the Web. Being an extensible language, XML allows new standards to be specified from it, enabling, in this way, the development of new technologies that operate in conformity with its standard.

A technology which was created from XML is Web Services [9]. These compose a distributed computational architecture based on auto-descriptive services that can be published, located and executed through the Web, which publish interfaces and connections that were defined and described in XML. The syntax of a Web Services can be exposed through Web Services Definition Language (WSDL) [21], which allows you to publish information related to service location and its access mode. The publication of a Web Service is performed through Universal Description, Discovery and Integration (UDDI) [10], where the service provider

makes available its information. The information exchange between a Web Service and a client application is performed through the Simple Object Application Protocol (SOAP) [11], which sends and receives information coded in XML, through the Hypertext Transfer Protocol (HTTP) [12]. In this context, OLAP technologies are related to XML for Analysis (XMLA) [13], whereas geographical technologies are related to Geography Markup Language (GML) [14] and Web Feature Services (WFS) [15].

The research presented in this paper does an analysis of the more important proposals for geographical analytical processing, highlighting their advantages and limitations. In sequence, according to Geographic On-line Analytical Processing Architecture (GOLAPA) [18,28], the meta-schema Geography Markup Language for Analysis (GMLA) is proposed. The remainder of this paper is organized as follows: The second section presents a summary of the main related researches. The third section contains a short presentation about the GOLAPA architecture. The sections 4 and 5 briefly describe XMLA and GML standards, respectively. In turn, the GMLA XML Schema is presented in section 6 and an example of its application is showed in section 7. Finally, some conclusions of this study and future works will be discussed in section 8.

2 Related Works

Some researches about integration of OLAP and GIS were proposed in literature, but we will consider just four of these, because their contributions have showed to be more relevant. They are: MapCube [29], GeoMiner [30], GOAL [31] and SIGOLAP [32, 33].

MapCube is an operator used to perform aggregation and visualization of geographical data in a metaphor that remind an album of maps. MapCube is based on the Cube [16] operator and similar to this, it performs all combinations of aggregation by each data of each DW dimension. The main difference between these operators is the presentation of their results, that is, the Cube operator just presents its results in a tabular view, whereas the MapCube operator can present the results in a tabular and geographic (map) view.

GeoMiner is a project for the mining of geographical data, which is composed by three basic modules. Although, just two modules of these can be considered as related work: the module for geo-cubes construction and the one for geographical OLAP. In [34, 35] Han et al. proposed two algorithms for efficient construction of geographical data cubes: Pointer Intersection and Object Connection. These algorithms provide analytical and spatial cubes basing on criteria for selective materialization of a geographical dataset. The specified criteria were: 1) the geo-dataset access frequency, 2) its storage cost and 3) the support that it will offer the construction of other derivatives geographical cubes.

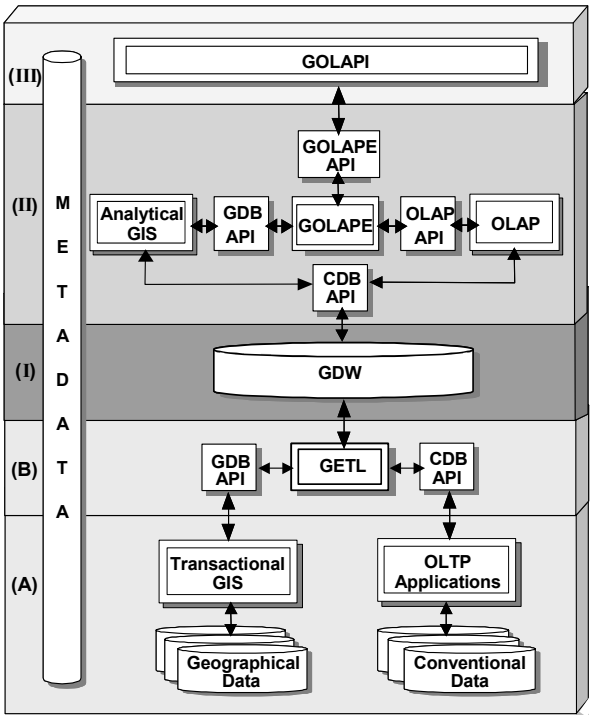
Geographical Information On-Line Analysis (GOAL) is a project that aims to accomplish the integration of DW/OLAP with GIS. To achieve this purpose, Kouba et Al. [36] implemented an architecture where the main component used to manage this task is called Integration Module (IM). In the case study presented in [36] was used the software GT Media98 and ArcView as GIS components, and the MSOLAP server as the OLAP component. Considering this software has different interfaces for communication, the IM had to be adjusted to offer the appropriate support for each software component. The communication to ArcView was executed with Avenue scripts (the native language of ArcView) and Delphi libraries. The communication with GT Media 98 was not presented. The communication with MSOLAP was done from the implementation of an OLE DB for OLAP provider [17] with Visual C++.

SIGOLAP aims to provide the integration between GIS and OLAP by a three ties architecture, which is based on mediation approach. The case study implemented to this architecture was developed using the following technologies: OLAP – MSOLAP Server; GIS - MSAccess and AutoDesk Mapguide Server; Integration model and Mediator- SQL Server and Visual Basic, and, finally, user interface - Visual Basic Script.

With regard to the researches discussed above, we shall now conclude that: the two first approaches, MapCube and GeoMiner, just provide multidimensional analytical processing over geographical data without considering the GIS operators (e.g. distance and adjacency). By contrast, the two last works, GOAL and SIGOLAP, certainly offer an approach to integrate OLAP and GIS, whereas, the use of proprietary technologies prevent these works to be multi-platform and mainly minimize the extensibility and data interchange of these approaches.

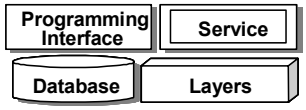
3 GOLAPA - Geographic On-Line Analytical Processing Architecture

GOLAPA [18, 28] arose from the necessity of providing an open and extensible architecture. To support these properties we decided to choose Java and XML as the standard languages in developing our architecture. This choice was made because these languages are very well known (especially on Web) and also can be used with other technologies (e.g. CORBA and Web Services). The five layers of GOLAPA are distributed as Figure 1. There are three essential layers for the multidimensional-geographical process: layers (**I**), (**II**) and (**III**), that, respectively, provide data, services and graphical interface of a GIS/OLAP system and also two other support layers: layers (**A**) and (**B**) which, respectively, handle operational data and build/maintain the layer (**I**).



Legend

- **Icons:**



- **Layers:**
  - **(A) Operative Data**
  - **(B) Data Access, Extraction, Transformation,Validation and Load**
  - **(I) Strategic Data**
  - **(II) Geographical Multidimensional Decision**
  - **(III) Graphical User Interface**
- **Main Acronyms:**
  - **GIS - Geographical Information System**
  - **OLAP - On-Line Analytical Processing**
  - **OLTP - On-Line Transaction Processing**
  - **API - Application Program Interface**
  - **GOLAPI - GOLAP Interface**
  - **GOLAPE - GOLAP Engine**
  - **GDB - Geographical Database**
  - **CDB - Conventional Database**
  - **GDW - Geographical DW**
  - **GETL - Geographical ETL**

Figure 1: GOLAPA architecture.

The layers of *GOLAPA* are briefly described below:

(A) Operative Data – the layer responsible for dealing with operative data. Note that all tools inside this layer are inherently for OLTP processing.

(B) Data Access, Extraction, Transformation, Validation and Load – the layer responsible for the translation from operational environment to decision environment.

(I) Strategic Data – the layer based on GML [14] and traditional DW star schema. This layer explicitly divides the transactional environment of the decision environment, because it prevents that any operation occurred on transactional environment will not be reflected on the decision environment. Moreover, as the GDW has geographical properties defined by GML, it guarantees that any GIS which conforms with GML can be used.

(II) Geographical Multidimensional Decision– The layer responsible for manipulating the GIS, OLAP and GIS/OLAP functionalities. This layer is the most important, because it is actually responsible for the execution of geographical and analytical processing. In this layer, the component that will address this processing is GOLAPE and the contribution of this paper is the definition of its XML schema (i.e. GMLA) for data exchange and integration, which will be encapsulated in the GOLAPE API component.

(III) Graphical User Interface – the layer responsible for data visualization, which can be a local graphical interface as well as a remote client or a web browser.

With regard to main components of GOLAPA, these can be summarized as follows:

GETL – It represents a traditional Extract Transform Load (ETL) [4] tool with a geographical support. If this tool does not exist, a similar service can be provided from the joining of an exclusive geographical ETL (e.g. FME[1]) with a traditional ETL (e.g DTS[2]).

GDW – It is similar to the traditional DW schemas (e.g. star schema), however, the geographical data also contains its geometries archived.

Analytical GIS – as the traditional GIS is inherently OLTP (i.e. allows updates and deletes of data), which contrast with the decision support approach, we decided to explicitly define a service that just allows load and query of data, which is namely of analytical GIS.

GDB/CDB API – It corresponds to programming interfaces used to access geographical and conventional services (e.g. FME e JDBC), respectively.

OLAP – It constitutes an OLAP service.

GOLAPE – It represents the main component for GIS and OLAP service integration, which will receive and classify a valid request as: geographical only, analytical only or both (geographical and analytical) and, afterwards, it will process the query and return its result.

GOLAPE API – It corresponds to GOLAPE programming interface. As previously said, the

[1] http://www.safe.com/products/fme/

[2] http://www.microsoft.com/sql/

contribution of this paper is to define the GMLA, which is a XML schema [19] for integration and exchange of geographical and multidimensional data. GMLA is based on the proposals of XMLA [13] and GML [14] (see sections four and five, respectively).

GOLAPI – It represents the graphical interface used to query and display either geographical, or analytical, or geographical-analytical results. The focus of this component is to provide a visual metaphor to query and represent the geographical and/or analytical information.

4 XML for Analysis – XMLA

The XML For Analysis (XMLA) [13] is a XML API based on SOAP [11] created by an initiative of Microsoft Corporation [23] and Hyperion Solutions Corporation [24] to provide an open access for multidimensional databases. This standardized access enables a non proprietary communication between client applications and OLAP data servers through the Internet.

The XML for Analysis implementation is based on the Web Services architecture, and the description of its service is defined with a WSDL document. XMLA provides two methods for controlling their functionalities: Discover and Execute, which enable the access of data and metadata that are stored in a multidimensional OLAP Server. These methods are invoked through SOAP and, thus, the input and output are XML documents.

As XMLA is an open technology based on standards such as XML and Web Services, we have chosen this technology to provide the OLAP support in GMLA. The main elements of a XMLA document are defined in its XML Schema named *MDDataSet.xsd* and will be described in the next paragraph. The other elements and XML Schema are not so relevant and therefore will not be described here, although these can be downloaded from *http://www.cin.ufpe.br/~golapa/schemas/xmla*.

The *root* element is the first of the hierarchy, which is composed of: 1) metadata (*OlapInfo, AxesInfo* and *CellInfo* elements) of the structure of an OLAP Sub-cube; (2) multidimensional axes (*Axes* element) which correspond the dimensions and (3) cell(s) (*CellData* and *Cell* elements) which compose the facts and each cell of a Sub-cube. The metadata that describes the axes of a Sub-cube (*AxesInfo* element) are decomposed by each axis (*AxisInfo* element) and its respective hierarchies (*HierarchyInfo* element) with properties such as: unique name (*Uname* element), title (*Caption* element), level name (*Lname* element), level number (*Lnum* element), exhibition position (*DisplayInfo* element) and anything else that needs to be added (*any* element). Each axis (*Axis* element) is a set of members (*Members* element) and can be implemented as a cartesian product (*CrossProduct* element) or a set of tuple (*Tuples*). Then, each member (*Member* element) has the same set of properties as defined in a *HierarchyInfo* element.

5 Geography Markup Language - GML

GML [14] is the OpenGis [27] *XML Schema* for exchange and storage of the descriptive and geometric properties that define geo-referenced data. *GML* is based on the abstract model [26] of the *OpenGIS* consortium and its last stabilized version is 2.1.2, which just supports vector geo-data. Although the version 3.0 of GML has been launched recently, this is still under evaluation, and for this reason we have decided to define *GMLA* according to GML 2.1.2, which continues to be supported by the version 3.0 [22] and, mainly, because it is already an evaluated and stabilized version.

GML is a meta-schema that offers a set of elements and attributes for the construction of geographical application schemas. Differently from XMLA web service, *GML* is only a vocabulary defined in *XML Schema*, therefore, the use of *WFS* is needed, or another tool, to manipulate geo-data in *GML* (e.g. *FME*). The main elements of a GML document are defined in its XML Schema named *feature.xsd* and will be described in the next paragraph. The remaining elements and other *XML Schemas* used by *GML* are not so relevant and therefore will not be described here, although these can be downloaded from *http://www.opengis.org/techno/implementation.htm*.

The key element of GML is *_Feature*. This represents a geographical object, which is not obligated to have a geometric property (*_geometryProperty* element), for example, a customer without address coordinates. The geometric properties of the features are coded starting from: 1) primitive geometries, that is, point (*Point* element), line (*LineString* or *LinearRing* elements) or polygon (*Polygon* element); 2) homogeneous collection of primitive geometries, that is, points (*Multipoint* element), lines (*MultiLineString* element) or polygons (*MultiPolygon* element) or 3) heterogeneous collection of primitive geometries (*MultiGeometry* element). Moreover, the geometric properties can also be defined starting from more expressive names, which are synonymous for primitive geometries, for example, for a polygon which has *coverage* or *extentOf* elements and for several polygons which have *multiCoverage* or *multiExtentOf* elements. *_Feature* is an abstract XML element and it can have a description (*description* element), a name (*name* element) or a bound (*boundedBy* element).

Another important element of GML is *_FeatureCollection*. This is also an abstract and it corresponds to a collection of features, which are individualized by the *featureMember* element. Note that *_FeatureCollection* element is responsible for implementing the concept of geographical layers/themes.

6 Geography Markup Language for Analysis - GMLA

GMLA was specified according to XML Schema syntax and this is based on XMLA and GML standards, that is, GMLA imports, extends, and integrates the previous XML Schemas. With GMLA, data that were described by two distinct schemas, now can be described and semantically integrated by just one schema, which is used by the GOLAPE API component to describe and validate the geographical-multidimensional data that are integrated by GOLAPE. We highlight that the namespace of elements defines whether it was directly imported or extended/defined by GMLA, that is, any GMLA element that starts with *xmla* or *gml* namespace was directly imported from XMLA and GML, respectively. By contrast, any element without namespace was extended or defined by GMLA

GMLA can be used as a schema (like XMLA), but it can also be used as a meta-schema (like GML). In the first case, data are described exactly as defined by GMLA XML Schema and, in the second case, GMLA plays a role as a base vocabulary used to describe other geographical-multidimensional data schemas. The advantage of the second approach is the fact that it can be used to build schemas that are semantically customized, which cannot occur in the first case, because it is a generic schema. For example, in the first case, all geographical features are *members* elements, which are organized in a *featureType* element. However, in the second case, we can define some geographical features as a special type (e.g. *residenceMember*), which can be organized in a special theme (e.g. *residenceTheme*). The elements of a GMLA document are defined in its *geoMDResult.xsd* XML Schema, which can be downloaded from www.cin.ufpe.br/~golapa/schemas/gmla

In the *geoMDResult.xsd*, the first element of the hierarchy is *geoMDResult.* This is similar to XMLA *root* element, whereas, *geoMDResult* includes the optional element *GeoView*, which is responsible for describing the layers/coverages (*featureType* element) and the respective geometries of their geographical members (*featureMember* element). These are described by the *member* element. In turn, the *member* element represents a unique geographical feature and, for this reason, it extends the GML *_Feature* element. However, as the *featureType* element describes a set of geographical members, it extends the GML *_FeatureCollection* element. With regard to *isGeographic* element, it identifies whether the member is geographical or not. If it is, the member will have one of the GML geographical properties (e.g. *position, centerLineOf or coverage*), generalized by a *_geometryProperty* element. Otherwise, it will have just members properties of XMLA (e.g. *UName*, *Caption*, *LName*, *LNum*, *DisplayInfo*), encapsulated by a *memberProperties* element

The GMLA metadada described by *OLAPInfo* and *Axes* elements are similar to those presented by XMLA *MDDataSet*.xsd. However, differently from XMLA metadata, each GMLA *member* element, as previously said, can represent a geographical feature. We point out that an axis is not a geographical theme, although it can have several geographical members, this occurs, because inside the same axis can have several features of different themes. For example, after successive "drill down" operations on a geographical hierarchy, an axis can have members of different themes (*e.g.* countries, state, districts and addresses). Thus, the members of an axis can even own an associate geometry whereas only in GMLA *GeoView* element will these be organized by a *featureType* element.

Considering GMLA elements, the only one that did not suffer any alteration was *CellData*, which continues to describe facts of the generated sub-cube. In the next section, the use of GMLA will be discussed.

7 GMLA Application

As the *GOLAPE* component is in development, the construction of a GMLA Instance does not occur in an automatic way yet. However, this fact does not invalidate the GMLA proposal, because: 1) it is a *XML Schema* vocabulary, thus any tool that has a *XML* processor, which conforms to *XML Schema* standard, can be used to validate the GMLA Schema and its instances and 2) it is non dependent of *GOLAPE*, that is, it can be used as XML Schema of any component that wishes to integrate and describe geographical-multidimensional data

According to the previous paragraph, to exemplify the use of GMLA we generated, by the use of the tool *XML Spy,* a GMLA Instance that integrates *XMLA* and *GML*. The XMLA Instance was generated from the cube *Sales,* which is used to test the MSOLAP server. The *OLAP* query which was carried out describes the units sold of products types as per stores in states and country. In consideration the GMLA Instance, it describes a geographical vision of country and its states, and it was generated from the extension of ArcView called *ToWKT*, which generates *GML* and *SVG* (*Scalable Vector Graphics*) documents. Figure 2 summarizes the integration process achieved by *GMLA*. In the sequence, Code 1 lists the sketch of the *GMLA* instance that contains the data integration of *XMLA* and *GML*. The complete instances used on *XMLA, GML* and *GMLA* and its XML Schema, can be downloaded from *http://www.cin.ufpe.br/~golapa/instances/app* and http://*www.cin.ufpe.br/~golapa/schemas/app*, respectively. We emphasize that *GMLA* is a XML vocabulary and, thus, its focus is on the content and not on the data presentation. For the data presentation, at this moment, we are investigating the *GOLAPI* component of *GOLAPA*

```
<?xml version="1.0" encoding="UTF-8"?>
<!--The Namespaces of used schemas -->
<geoMDResult>
  <OlapInfo>
    <AxesInfo> <!-- Description of axes metadata --> </AxesInfo>
    <CellInfo>  <!-- Description of cells metadata --> </CellInfo>
  </OlapInfo>
  <Axes>
    <!--Description of axes data -->
    <Axis name="Axis0">
      <!--  Description of column axis data  -->
      <crossProduct>
        <members Hierarchy="Store">
          <member fid="USA">
            <geometry> <!-- Description of USA geometry --> </geometry>
            <memberProperties>      <!-- Description of USA member--> </memberProperties>
          </member>
          <member fid="CA">
            <geometry> <!-- Description of CA geometry --></geometry>
            <memberProperties>      <!-- Description of USA member --></memberProperties>
          </member>
          . . .
        </members>
      </crossProduct>
    </Axis>

    <Axis name="Axis1">
      <!--  Description of row axis data -->
      <crossProduct>
        <members Hierarchy="Product">
          <member> <memberProperties> <!-- Description of products  members --> </memberProperties> </member>
          . . .
        </members>
      </crossProduct>
    </Axis>
    <Axis name="SlicerAxis">
      <!--  Description of restrictive axis data -->
      <crossProduct>
        <members Hierarchy="Measures">
          <member>
            <memberProperties>      <!--Description of units sold --> </memberProperties>
          </member>
        </members>
        . . .
      </crossProduct>
    </Axis>
  </Axes>
  <CellData> <!-- Description of sub-cube cells --> </CellData>
  <GeoView>
    <geoTheme fid="COUNTRY">     <!--  Descrioption of coutry theme  --></geoTheme>
    <geoTheme fid="STATES"><!--  Description of states theme  -->     </geoTheme>
  </GeoView>
</geoMDResult>
```

Code 1: Resume of a GMLA instance

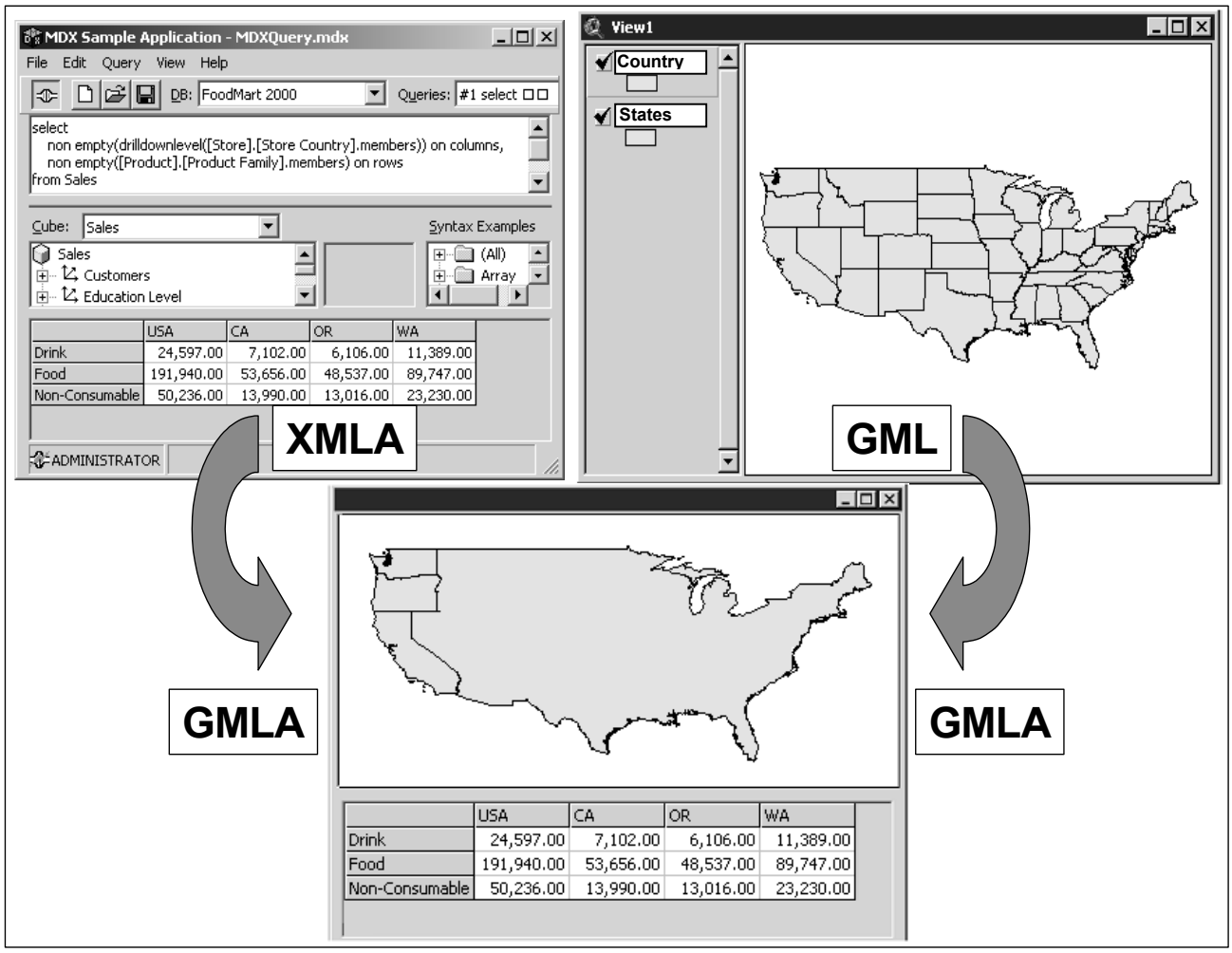


Figure 2: Demonstrative Graphic of GMLA integration

8 Conclusions and Future Work

The integration of DW, OLAP and GIS in a unique environment, provides a wider context for decision support, because it aggregates the following: a specific data model for decision support (DW), the flexibility and the analysis agility on a great volume of data (OLAP) and the power of the visualization and spatial queries (GIS).

About the research of Geographical *OLAP*, we concluded that does not exist a solution that is independent of format or platform. This fact motivates us to implement the GOLAPA architecture, which has as a premise the use of format and platform independent technologies (i.e. Java and XML)

Concerning the independence of data format of GOLPA, this article proposes a XML Schema named GMLA. It is defined to support the integration and exchange of two different XML schema, one for multidimensional data (XMLA) and another for geographical data (GML), in a unique XML Schema. GMLA is encapsulated by the API GOLAPE component of GOLAPA. However, GMLA is independent of the component GOLAPE, and could be used by another component that needs an XML Schema to integrate geographical-multidimensional data. We highlight that GMLA is the first initiative to integrate and exchange data from OLAP and GIS tools in an open format.

To conclude the metadata model, the implementation of *GOLAPE and GOLAPI* components and support the GML 3.0 specification, are the future work of the *GOLAPA* project.

Acknowledgements

We would like to express our gratitude to Juliano de Souza Freitas for his valuable assistance in the GOLAPA Project.

References

[1] S. Chaudhuri, U. Dayal. Decision support, Data Warehouse, and OLAP, in Tutorials of VLDB, 1996

[2] W. H. Inmon. Building the Data Warehouse. 2nd edition. John Wiley & Sons, 1997

[3] R.Kimball. The Data Warehouse Toolkit., John Wiley&Sons, Inc, 1996

[4] R. Kimball, L. Reeves, M. Ross, and W. Thornthwaite. The Data Warehouse Lifecycle Toolkit. Wiley, 1998

[5] G. Câmara, et al. Anatomia de Sistemas de Informação Geográfica. 10ª Escola de computação, 1996.

[6] N. Chrisman. Exploring Geographic Information Systems. John Wiley and Sons, 1996

[7] P. A. Longley, et al. Geographical Information Systems: Principles, Techiniques, Applications and Managemente. 2nd Edition, John Wiley and Sons, 1999

[8] M. F. Goodchild. Geographical Data Modeling. Computers & Geosciences. 1992

[9] W3C, Web Service official home page, last visit on 2003 (http://www.w3.org/TR/2002/WD-ws-arch-20021114/)

[10] UDDI, UDDI official home page, last visit on 2003, (http://www.uddi.org/)

[11] W3C, SOAP official home page, last visit on 2003 (http://www.w3.org/TR/2002/CR-soap12-part0-20021219/)

[12] W3C, HTTP official home page, last visit on 2003 (http://www.w3.org/Protocols/)

[13] XMLA.org, XML For Analysis Specification, Version 1.1, Microsoft Corporation & Hyperion Solutions Corporation. 2002

[14] OpenGIS® Geography Markup Language (GML) Implementation Specification, version 2.1.2, 2002

[15] OpenGIS© Web Feature Service Implementation Specification, Version: 1.0.0., 2002

[16] J. Gray, et al. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. In Proc. 12th ICDE, 1996.

[17] Microsoft OLE DB 2.0. Program Reference and Data Access SDK, Microsoft Press, 1998

[18] R. N. Fidalgo, V. C. Times, F.F. Souza – GOLAPA: Uma Arquitetura Aberta e Extensível para Integração entre SIG e OLAP. In Proc. GeoInfo 2001, 2001

[19] W3C, XML Schema official home page, last visit on 2003 (http://www.w3.org/)

[20] W3C, XML official home page, 2003, (http://www.w3.org/XML)

[21] W3C, WSDL official home page, last visit on 2003 (http://www.w3.org/TR/2001/NOTE-wsdl-20010315)

[22] OGC, *GML* 3.0.0, 2003 (http://www.opengis.org/techno/implementation.htm)

[23] Microsoft Corporation (www.microsoft.com)

[24] Hyperion Solutions Corporation (www.hyperion.com)

[25] XML Spy Editor,(http://www.xmlspy.com)

[26] OpenGis. The Abstract OpenGIS Specification. OpenGIS Document Nunber 99-100. Jun., 1999.

[27] OpenGis (www.opengis.org)

[28] R. N. Fidalgo. Uma infraestrutura básica para integração de dados multidimensionais e geográficos em um ambiente de Data Warehouse, PhD Thesis proposal, CIn-UFPE, 2003.

[29] S. Shekhar, et al. Map Cube: A Visualization Tool for Spatial Data Warehouse. 2000 (http://www.cs.umn.edu/Research/shashi-group/Mapcube/mapcube.html)

[30] J. Han, K. et al. *GeoMiner*: a system prototype for spatial data mining. Proc. ACM SIGMOD, 1997

[31] GOAL Project official home page. Last visit on 2003 (http://krizik.felk.cvut.cz/goal/)

[32] A.C. Ferreira, et al. An Architecture for Spatial and Dimensional Analysis Integration. In Proc. of World Multiconference on Systemics, Cibernetics and Informatics (SCI 2001), 2001

[33] A.C. Ferreira. Um Modelo para Suporte à Integração de Analises Multidimensionais e Espaciais. Dissertação de Mestrado – UFRJ, 2002

[34] J. Han, et al. Selective materialization: An efficient method for spatial data cube construction. In PAKDD, 1998

[35] N. Stefanovic. Design and implementation of on-line analytical processing (*OLAP*) of spatial data. M.Sc. Thesis, Simon Fraser University, Canada, September 1997 http://citeseer.nj.nec.com/stefanovic97design.html

[36] Z. Kouba, K. Matousek, P. Miksovsky. On Data Warehouse and GIS integration, DEXA, 2000