Journal of Computing and Information Technology - CIT 17, 2009, 4, 347–358
doi:10.2498/cit.1001354

XML Encoding and Web Services for Spatial OLAP Data Cube Exchange: an SOA Approach

Etienne Dubé[12], Thierry Badard[1] and Yvan Bédard[12]

[1]Centre for Research in Geomatics (CRG), Université Laval, Québec, Canada
[2]NSERC Industrial Research Chair on Geospatial Databases for Decision Support, Université Laval, Québec, Canada

XML and Web Services technologies have revolutionized the way data are exchanged on the Internet. Meanwhile, Spatial OLAP (SOLAP) tools have emerged to bridge the gap between the Business Intelligence and Geographic Information Systems domains. While Web Services specifications such as XML for Analysis enable the use of OLAP tools in Service Oriented Architecture (SOA) environments, no solution addresses the exchange of complete SOLAP data cubes (comprising both spatial and descriptive data and metadata) in an interoperable fashion.

This paper proposes a new XML grammar for the exchange of SOLAP data cubes, containing both spatial and descriptive data and metadata. It enables the delivery of the cube schema, dimension members (including the geometry of spatial members) and fact data. The use of this XML format is then demonstrated in the context of a Web Service. Such services can be deployed in various situations, not limited to traditional client-server platforms, but also ubiquitous mobile computing environments.

Keywords: geospatial business intelligence, Spatial OLAP, XML, Web services, data warehouse, service-oriented architecture (SOA)

1. Introduction

The World Wide Web Consortium's (W3C) XML (eXtensible Markup Language) specification (1) has gained large popularity as a data exchange format on the Internet. Along with the growing acceptance of Service Oriented Architecture (SOA) and Web Services technologies (2), it provides the means to exchange information between heterogeneous systems, independently of programming language, operating system, hardware and software platforms. XML grammars have been developed for various application domains. This is also the case for geographic information systems (GIS), with the Geography Markup Language (GML) specification (3) from the Open Geospatial Consortium (OGC) and International Organization for Standardization (ISO), along with Web Services proposals such as Web Feature Service (WFS) (4), enabling the exchange of geographic features in distributed GIS applications.

In the meantime, SOLAP (Spatial OLAP) tools have been introduced to support the exploration and analysis of geospatial decision-support data. It is described as "a type of software that allows rapid and easy navigation within spatial databases and that offers many levels of information granularity, many themes, many epochs and many display modes that are synchronized or not: maps, tables and diagrams" (5). Based on OLAP (On-Line Analytical Processing) (6) technologies merged with GIS components, these tools provide interactive tabular, graphical and cartographical views of spatial data cubes, which are multidimensional datasets comprising spatial data such as the geometry of countries. This enables analysts to fully exploit the spatial component characterizing about 80% of all enterprise data (7).

Emerging SOLAP applications, especially in mobile computing environments, have prompted the need for an interoperable and reliable way to deliver SOLAP data cubes to heterogeneous clients (8). Unconventional usage patterns, such as disconnected operation when roaming

between mobile networks, and inherent limits of mobile computing such as limited bandwidth and processing capabilities, render traditional client-server OLAP poorly suited for these cases. SOA and Web Services have been considered as a suitable solution for these new applications.

GML and WFS specifications well cover the interchange of transactional geospatial data, traditionally found in operational systems. However, analytical systems such as SOLAP are often based on the multidimensional paradigm, in which data are represented in terms of hypercubes, composed of objects such as dimensions, members and facts. This contrasts with transactional data, typically represented by tables containing columns and rows. Several proposals have been made for the transmission of OLAP data through XML documents and Web Services. XML for Analysis (XMLA) (9), introduced by Microsoft and Hyperion, is targeted towards client-server applications using the MDX ("MultiDimensional eXpressions") query language (10), thus encapsulating MDX queries and result sets in XML messages. It is therefore not suited for the exchange of complete data cubes (data and metadata) for disconnected operation in mobile clients, and it lacks support for spatial data. XCube, a set of XML document templates for OLAP cubes (11), is capable of representing complete cubes, but has no provision for spatial data either. Another research paper proposes a Web Service, named "GML for Analysis" (GMLA) (12), which leverages the capabilities of XMLA and WFS, providing a way to combine multidimensional and spatial queries, and embed spatial data in the result sets. However, like XMLA, this technology is suited for client-server applications using a query-response paradigm, with result sets intended for presentation (i.e. in spreadsheets or other analytical clients), and not for the transmission of complete SOLAP data cubes.

This paper introduces a new XML format for the exchange of multidimensional geospatial data, addressing the shortcomings of existing proposals. This format leverages the GML grammar for the representation of geographic features, and is inspired in some aspects by existing XML and Web Service standards used in the OLAP industry. Possible uses of such a format include the dissemination of SOLAP cubes (or subsets of these cubes) from a central data warehouse to distributed data marts, the technical integration of heterogeneous systems (e.g. migrating SOLAP data between products of different vendors) and the implementation of Web Services for the delivery of SOLAP data to clients on the Internet or to mobile clients over wireless networks. The paper is divided as follows: first, requirements for the XML exchange of multidimensional spatial data are described. This is followed by the technical specification of the XML format itself. An application of this format, in the context of a geospatial decision-support Web Service for the delivery of spatial cubes to mobile devices, is also proposed. Finally, future outlooks and research challenges are discussed.

2. Requirements for a SOLAP Data Cube XML Format

Existing specifications and proposals for the XML exchange of either spatial data (GML, WFS) or multidimensional data (XMLA, GMLA and XCube) do not fully address the issue of delivering complete spatial data cubes, comprising both the metadata (the definition of the cube schema including dimensions, levels and measures) and the data (the dimension members and the fact cells). This prompts the design of a new XML format to overcome limits identified in existing ones. Several requirements have been identified for such a format, and are presented in this section.

2.1. Representation of Both Data and Metadata

The exchange of complete data cubes requires encoding of both cube data and metadata (schema). Query services such as XML for Analysis specify only the transmission of data and a subset of the schema (query axes) describing the resulting cell set. While such a response format is suitable for presentation purposes (e.g. for displaying data in a spreadsheet), it is not sufficient for transmitting the full cube schema, i.e. dimensions, hierarchies and levels.

By including both the full cube schema along with fact and member data, a client application retrieving an XML-encoded data cube will have all the information necessary to use the cube in

an autonomous manner. This is the first reason why the proposed XML encoding must provide for a representation of both data and metadata. In addition, metadata are useful to assess the quality of measures.

2.2. Based on Existing Standards

In order to address interoperability with current and future systems, the XML encoding for SOLAP data cubes should be based on existing standards and specifications.

Geography Markup Language (GML) is a standardized way to encode geographic features in XML form, which can be used for spatial members in SOLAP data cubes. GML elements can be embedded inside the XML cube schema, seamlessly including geographic features. Nevertheless, other XML specifications, such as Scalable Vector Graphics (SVG), KML (as used in Google Earth) or GeoRSS could be substituted for this purpose, depending on the specific application context.

With regard to the XML representation of multidimensional data structures in data cubes, the only existing de facto specification is XMLA. However, as explained in the introduction, it does not fulfill all the requirements for the exchange of complete SOLAP data cubes.

2.3. Independence of Particular Implementations in OLAP and GIS Products

One of the reasons for using open standards is to avoid lock-in to proprietary software platforms. Since the proposed XML format for SOLAP data cubes aims to be open and interoperable, it must not be tied to particular OLAP software. This can be achieved by choosing a common subset of features supported by most OLAP servers. Most of them support some form of a multidimensional data model, composed of dimensions, hierarchies, levels, members, measures and facts; these objects have to be expressed in a homogeneous way in the XML encoding. As mentioned above, using an open standard like GML also ensures interoperability with regards to geographic data. Independence from proprietary query languages and APIs is also a requirement. In this way, systems implementing the proposed XML encoding could be based on almost any OLAP and GIS products from the market. The possibility of exchanging SOLAP data between systems from different vendors is also a benefit of this approach.

2.4. Support for Spatial Dimensions and Spatial Members

SOLAP cubes are characterized by the inclusion of spatial members in geographic dimensions, and sometimes spatial measures (13). The proposed XML encoding must provide the mechanisms for including the spatial component in these members and measures. This requirement is fulfilled by using an XML-based standard such as GML for encoding geospatial objects, as previously explained.

2.5. Support for Shared Dimensions

More than one cube can refer to the same dimension: for example, a "sales" cube and another "payroll" cube could refer to a same "employee" dimension. In the literature, such shared dimensions are often referred to as conformed dimensions (14). In addition, this technique allows one to perform analysis across different cubes, by joining them by their shared dimensions.

The encoding should support this feature by not restricting a dimension to the context of a single cube. Multiple cubes can refer to a shared dimension document, to avoid replicating the dimension schema (hierarchies and levels) and data (members) redundantly across cubes.

2.6. Describing Calculated Measures and Members

Sometimes, calculated measures (or members) are defined to display expressions computed at run-time. Languages such as MDX allow users to define calculated members within a query. Most OLAP products also offer the capability to declare calculated members in the cube schema; in this way, calculated measures and members are accessible to all users of the cube. These predefined calculated members should also be expressed as part of the cube metadata encoded in XML form, so clients would have access to these members. However, the question of calculated spatial measures and aggregate operators

on spatial members is still a research issue, currently being investigated by our research group. By now, only numerical calculated measures are supported.

2.7. Separation of Contents and Presentation

Separation of contents and presentation is a design practice that aims to distinguish between the expression of semantics, as represented in the data, and the presentation of the information to end users. For example, the same data in a SOLAP cube can be presented in different forms: cross-tabs, charts, reports and maps. Moreover, the output device can vary: contents can be displayed in an interactive SOLAP desktop client or in a Web page, it can be printed or it can be displayed on the small screen of a mobile device such as a personal digital assistant (PDA) or a cell phone. Finally, the graphic semiology used to represent the data depends on the context (purposes, users, standards).

Accordingly, the XML cubes should contain only the data and their structure, i.e. cube schema. Any information specifying how to display the data to users, i.e. text format, chart colors and style, classes (ranges) for measure values, map symbols and so on, would be external to the XML data cube documents. The rendering and display format of SOLAP data can vary greatly and are dependant of the capabilities of the client application. Moreover, presentation is subject to customization by the user in some applications. For these reasons, the proposed XML format does not specify a way to encode presentation information.

3. Specification of the XML Encoding

Considering the requirements presented in the last section, a specification of the XML encoding for SOLAP data cubes has been designed. The proposed encoding is inspired by the XCube paper (11), but uses grammar and semantics closer to the data models of most OLAP software (specifically, terminology used in the MDX query language). A SOLAP data cube encoded with this specification would consist of three documents, respectively for the cube schema, dimension members and cell data. Separating the cube contents in three parts allows for progressive discovery of cube metadata and data. There are three root elements in the model, one for each type of document, i.e. *CubeSchema*, *CubeMembers* and *CubeCells*. A W3C XML Schema (WXS) definition (15) was developed for formalizing the semantics of the encoding and for validation of data cube documents. This WXS definition (XSD file) is available at:
`http://geosoa.scg.ulaval.ca/GeoCubeML/`.

While the encoding is generic enough for representation of any OLAP cube, specializations

```
<?xml version="1.0" encoding="UTF-8"?>
<cubeSchema
  xmlns="http://geosoa.scg.ulaval.ca/GeoCubeML"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://geosoa.scg.ulaval.ca/GeoCubeML
      ␣CubeSchema.xsd">

  <include fileLocation="WholesalersDimensions.xml" />

  <cube name="Sales">
    <dimension name="Measures" isMeasure="true">
      <dimensionDef>
        <level name="MeasuresLevel">
          <property name="aggregators">
            <contentType ref="aggregator" minOccurs="1"
              maxOccurs="unbounded" />
          </property>
          <property name="measureType">
            <contentType ref="contentType" />
          </property>
        </level>
      </dimensionDef>
    </dimension>
    <dimension name="Store">
      <dimensionUsage
        defName="WallyWorld␣Stores" />
    </dimension>
    <!-- other dimensions can be defined here ... -->
  </cube>

  <sharedDimensions>
    <dimensionDef defName="WallyWorld␣Stores">
      <hierarchy name="Administrative␣limits"
        default="true"/>
      <hierarchy name="Population␣areas" />
      <level name="(All)" isAll="true" />

      <!-- (Country, Province, Region levels) ... -->

      <level name="City">
        <inHierarchy
          hierarchy="Administrative␣limits" />
        <rollupTo level="Province" />
        <property name="City␣geometry">
          <contentType ref="gml:AbstractSurface"/>
        </property>
      </level>
      <level name="Agglomeration">
        <inHierarchy hierarchy="Population␣areas" />
        <rollupTo level="Province" />
        <property name="Agglomeration␣geometry">
          <contentType ref="gml:AbstractSurface"/>
        </property>
      </level>
      <level name="Retail␣outlet">
        <rollupTo level="City" />
        <rollupTo level="Agglomeration" />
        <property name="Retail␣outlet␣geometry">
          <contentType ref="gml:Point" />
        </property>
        <property name="Number␣of␣floors">
          <contentType type="xs:integer" />
        </property>
      </level>
    </dimensionDef>
  </sharedDimensions>
</cubeSchema>
```

Listing 1. CubeShema document example.

regarding spatial data are presented in this paper. They concern the encoding of vector geometry in spatial members and measures, and also metadata specifying which spatial aggregate functions can be used with these spatial measures. As explained in the following subsections, geometry data types are defined using WXS elements. Consequently, the type system is extensible, and conceivably this encoding could be extended to support any arbitrary XML data types, including complex data.

3.1. CubeSchema Document

The *CubeSchema* document encodes information pertaining to the structure of the cube. It identifies the cube name and all its dimensions with their respective hierarchies, levels and properties. An example CubeSchema document is presented in Listing 1, and the UML class diagram of the WXS definition in Figure 1.

At the root of the document lies the <*cubeSchema*> element. Optional <*include*> elements can be used to refer to external XML documents, to be included in the current cube schema. The <*cube*> element is used for defining the schema of a cube; there can be more than one in a single document. The optional <*sharedDimensions*> element provides for the definition of dimensions shared across cubes. Shared dimensions can come from external files, as specified in <*include*> elements; this allows sharing dimension definitions across many *CubeSchema* documents.

The <*cube*> element has one or more <*dimension*> elements as children. A dimension is identified by its *name* attribute. An *isMeasure* optional attribute specifies if this is the measure dimension, i.e. whose members identify the measures. A dimension can be defined in two manners: either with a <*dimensionDef*> element directly embedded in <*dimension*>, defining a dimension which is local to the cube, or with a <*dimensionUsage*> element, which refers to the name of a shared <*dimensionDef*>, defined in the <*sharedDimensions*> element (child of <*cubeSchema*>).

A <*dimensionDef*> element has one or more <*level*> children, defining which levels compose the dimension. Each <*level*> element can contain zero or more <*rollupTo*> children, identifying the parent level. More than one parent level can be specified when using multi-

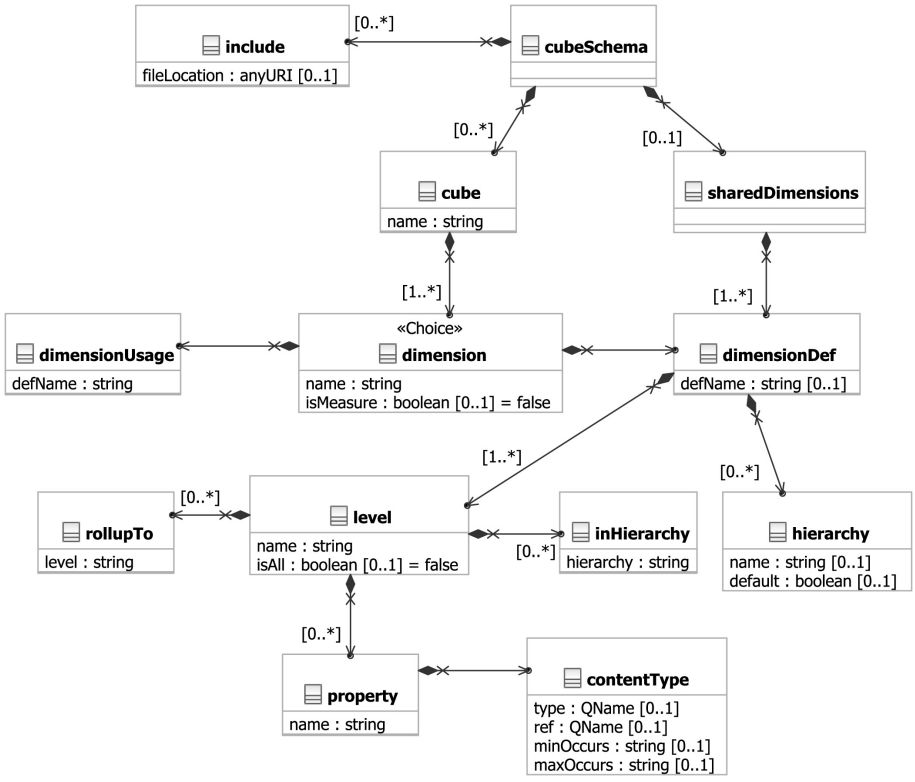

Figure 1. UML class diagram for CubeSchema.

ple hierarchies. In this case, <*hierarchy*> elements must figure under <*cube*>, to declare the hierarchy names and which to use by default; <*inHierarchy*> elements are used under <*level*> to specify which hierarchies (one or more) the level is part of. Omitting <*inHierarchy*> means that a level is shared by all the hierarchies of the current dimension. Levels can also have optional <*property*> elements, declaring member properties (i.e. attributes). Properties have a name, and contain a <*contentType*> element, specifying either which WXS simple data type (with the *type* attribute) or which complex XML elements (with the *ref* attribute, referring to an element declared in a WXS definition, with optional *minOccurs* and *maxOccurs* attributes defining cardinality) can be used as property values. This design choice, leveraging WXS, allows the inclusion of any XML element to be used as member attributes. In the example, properties are declared with "gml:AbstractSurface" and "gml:Point" element references, respectively allowing the use of GML surface (region) or point geometries. Any other XML encoding of geometry could be substituted instead (for example SVG).

3.2. CubeMembers Document

Once a cube schema is specified with the *CubeSchema* document, the members in its dimensions have to be described. This is done using the *CubeMembers* document (Listing 2 and UML class diagram in Figure 2). Members are represented as <*member*> elements, and are contained in successively embedded <*cubeRef*>, <*dimensionRef*> and <*levelRef*> elements, referring to the cube, dimension, and level defined in *CubeSchema*. A member has a *name* attribute, uniquely identifying it, and can contain <*rollupTo*> elements, describing the roll-up relationships to other members. These can be to members of a higher level, or to members of the same level as it is the case in parent-child hierarchies. More than one <*rollupTo*> element can be specified for a member, to account for multiple hierarchies (i.e. a member in a level rolling-up to members in two alternate levels, in different hierarchies). In this case the parent level must be specified (with the *level* attribute). Values for member properties are represented with <*propertyValue*> elements. This element contains data in any form authorized by XML, and constrained by the WXS type specified for the property in *CubeSchema*. The client application is responsible for ensuring conformity to the property's data type (i.e. by validating <*propertyValue*> contents against the WXS definition for the namespace of the data type.) Finally, optional <*include*> elements can also be used under <*cubeMembers*>, to specify members stored in external files (e.g. for shared dimensions).

In the example, we see a special case of member in the "Measures" dimension. These are measures, defined using <*member*> elements, and having the following special properties which

```
<?xml version="1.0" encoding="UTF-8"?>
<cubeMembers
  xmlns="http://geosoa.scg.ulaval.ca/GeoCubeML"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://geosoa.scg.ulaval.ca/GeoCubeML
      ␣CubeSchema.xsd">
  <cubeRef name="Sales">
    <dimensionRef name="Measures">
      <levelRef name="MeasuresLevel">
        <member name="Sales␣amount">
          <propertyValue property="aggregators">
            <aggregator><sum /></aggregator>
          </propertyValue>
          <propertyValue property="measureType">
            <contentType type="xs:decimal" />
          </propertyValue>
        </member>
        <!-- other measures defined here ... -->
        <member name="Quantity␣in␣stock">
          <propertyValue property="aggregators">
            <aggregator><sum /></aggregator>
            <aggregator dimension="Time">
              <avg />
            </aggregator>
          </propertyValue>
          <propertyValue property="measureType">
            <contentType type="xs:integer" />
          </propertyValue>
        </member>
      </levelRef>
    </dimensionRef>
    <dimensionRef name="Store">
      <levelRef name="(All)">
        <member name="All␣Stores" isAll="true" />
      </levelRef>
      <!-- members in other levels (city, region, ...) -->
      <levelRef name="Province">
        <member name="Prince␣Edward␣Island">
          <rollupTo member="Canada" />
          <propertyValue property="Province␣geometry">
            <gml:MultiPolygon>
              <gml:polygonMember>
                <gml:Polygon>
                  <gml:outerBoundaryIs>
                    <gml:LinearRing>
                      <gml:coordinates>
                        -63.1714669996, 46.1263050001
                        <!-- ... -->
                        -63.1722950003, 46.1198730006
                      </gml:coordinates>
                    </gml:LinearRing>
                  </gml:outerBoundaryIs>
                </gml:Polygon>
              </gml:polygonMember>
              <!-- ... -->
            </gml:MultiPolygon>
          </propertyValue>
        </member>
        <!-- other members ... -->
      </levelRef>
    </dimensionRef>
  </cubeRef>
</cubeMembers>
```

Listing 2. CubeMembers document example.

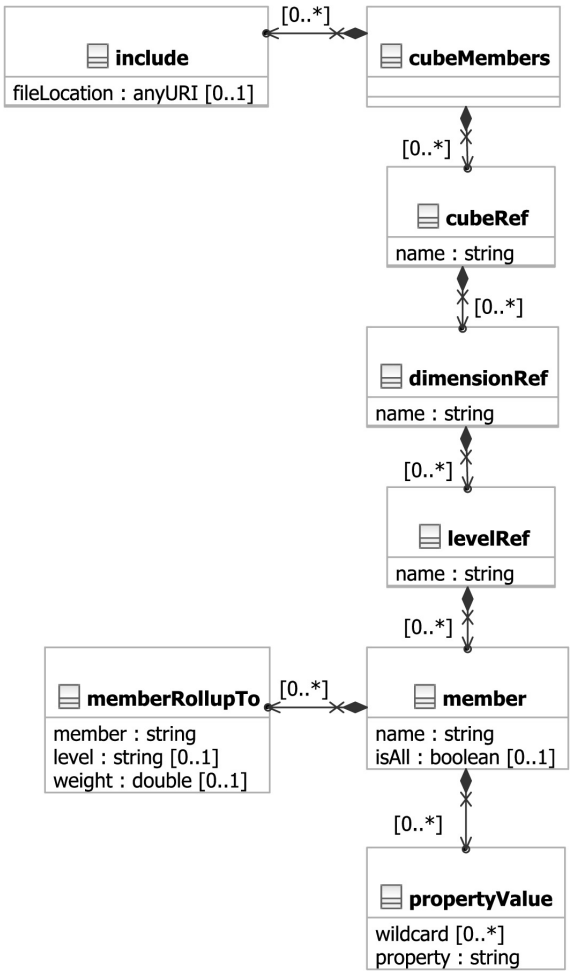

Figure 2. UML class diagram for CubeMembers.

are to be interpreted by the client application. The "aggregator" property contains <*aggregator*> elements, defining the type of operator used for aggregation (sum, average, etc.). More than one <*aggregator*> element can be specified, to account for non-additive or semi-additive measures (for example stocks or population, which cannot be summed across time) (14). Additional aggregators must specify which dimension they apply to (using the *dimension* attribute), indicating the client application which aggregative formula to use, instead of the default one, when summarizing along this specific dimension. Finally, the "measureType" property is used to specify the WXS data type or element for cell values corresponding to this measure, using the same <*contentType*> element introduced for the *CubeSchema* document. This allows for a great flexibility in the representation of non-numerical measures, for example geometric spatial measures (13).

Geometric spatial measures require specific aggregate operators, different from the classical ones used for numerical measures (16). Examples of such aggregators include geometric union, centroid, minimum bounding rectangle, convex hull and concatenation (creating a multi-part geometry from single parts). Other possible functions yield a numerical value from sets of geometries (e.g. total area of polygons, or length of line segments). Such spatial aggregators can be specified in <*aggregator*> elements. Since the realization of these operators can be complex, it is left to the implementation to define which set of spatial aggregate functions are supported. Ongoing research work[1] aims to clarify the use of geometry aggregation in SOLAP applications.

3.3. CubeCells Document

CubeCells, the last type of document, is shown in Listing 3 and UML diagram in Figure 3. It contains the individual cells representing the facts of the cube. A <*cubeCellsCubeRef*> element identifies the cube (described in a

```
<?xml version="1.0" encoding="UTF-8"?>
<cubeCells
  xmlns="http://geosoa.scg.ulaval.ca/GeoCubeML"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://geosoa.scg.ulaval.ca/GeoCubeML
      CubeSchema.xsd">
  <cubeCellsCubeRef name="Sales">
    <gcml:slicerMembers>
      <gcml:tuple>
        <gcml:memberRef dimension="Measures"
          level="MeasuresLevel" member="Sales amount" />
      </gcml:tuple>
    </gcml:slicerMembers>
    <cells>
      <cell>
        <tuple>
          <memberRef dimension="Product" level="Item"
            member="Fuzzy dice" />
          <memberRef dimension="Time" level="Day"
            member="2007-08-23" />
          <memberRef dimension="Store"
            level="Retail outlet"
            member="Downtown Quebec City" />
        </tuple>
        <cellValue>120.00</cellValue>
      </cell>
      <cell>
        <tuple>
            <memberRef dimension="Product" level="Item"
              member="Fuzzy dice" />
            <memberRef dimension="Time" level="Day"
              member="2007-08-23" />
            <memberRef dimension="Store"
              level="Retail outlet" member="Sainte-Foy" />
        </tuple>
        <cellValue>210.00</cellValue>
      </cell>
      <!-- other cells ... -->
    </cubeRef>
  </cells>
</cubeCells>
```

Listing 3. CubeCells document example.

[1] within the NSERC Industrial Research Chair on Geospatial Databases for Decision Support, at Laval University

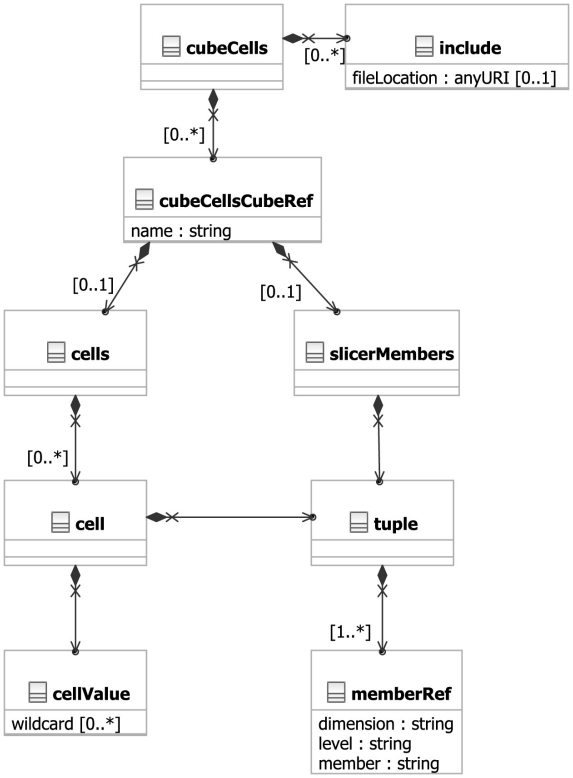

Figure 3. UML class diagram for CubeCells.

CubeSchema document) to which the cells belong. The <*slicerMembers*> optional element identifies "slicer members", which are members of dimensions excluded from a query result's axes. In other words, members in the tuple under <*slicerMembers*> apply to every cell in the *CubeCells* document. This element is only relevant when the *CubeCells* document is generated from the result of an OLAP query (i.e. it contains a subset of an existing cube). The next element is <*cells*>; each cell is represented under it as a <*cell*> element, composed of a <*tuple*> with several <*memberRef*> elements (each one referring to a member in each dimension) and a <*cellValue*> for the value of the fact. A value can be of either a simple data type (number, string) or a complex one (e.g. a GML geometry representing a spatial measure). Usually, a *CubeCells* document will contain only the cells corresponding to leaf members of each dimension, or in other words the most granular data. The client application can dynamically rebuild all the cube cells at less detailed levels, by applying the aggregation operators defined in *CubeMembers* for each measure. Cells corresponding to non-leaf members can also be included if needed (to include pre-computed aggregates for performance reasons or for cells corresponding to non-summarizable measures which cannot be obtained from an aggregative formula).

3.4. Relevance of the Encoding for Spatial Data and SOA

The proposed encoding has two characteristics making it suitable for spatial data and SOA: first, it takes into account the specificity of spatial data in a generic and extensible manner. By considering that any complex XML data can be part of members and facts, this encoding can leverage external XML Schemas such as GML, SVG and Google KML, allowing for the representation of rich spatial data and extensibility for future versions of these standards. Second, its structure is particularly suited to coarse-grained, document-oriented Web services, which have a central role in Service Oriented Architectures (SOA) (17). An example of such a service is presented in the next section. By having a loosely-coupled way to integrate SOLAP systems, promoting component reuse and interoperability, productivity gains in development can be achieved.

4. Use of the XML Encoding in a Web Service

The described XML format was first designed for a Web Service for the delivery of SOLAP mini-cubes to mobile clients (18). A mini-cube is a data cube of reduced size, extracted from existing cubes, and adapted with respect to the constraints of mobile computing (i.e. reduced CPU speed and storage size of PDAs or cell phones, limited bandwidth and coverage of wireless networks).

The relevance of exchanging mini-cubes to mobile clients arise in the context of disconnected operation: while a traditional query/response mechanism is sufficient when network availability is guaranteed, wireless networks are sometimes limited in their coverage and bandwidth. Thus, by requesting mini-cubes when network connectivity is available and caching these mini-cubes locally on the client, using the mobile SOLAP application is still possible outside of the network coverage area.

4.1. Service Contract

The Web Service is invoked using the SOAP protocol (19). The service contract, i.e. the operations accessible to clients, is described in WSDL (Web Services Description Language) (20). The following operations are supported:

1. *GetCapabilities*: this operation informs the client about the service version and gives information on the identification (name and address of the organization, etc.).
2. *DescribeSchema*: this operation is used to get the metadata describing cubes, dimensions, hierarchies, levels and measures, returned in a <*cubeSchema*> element.
3. *GetMembers*: this operation allows querying of cubes' dimensions to retrieve their members. It must support the navigation into a hierarchy of members, i.e. get the parents or children of a specific member, get the members of a level, etc. This operation also offers mechanisms to select members based on their spatial attributes, with metric or topological operators. The selected members are returned in a <*cubeMembers*> element.
4. *GetCells*: this operation is intended to query the facts of a data cube. The argument of this operation forms a multidimensional "select query" that defines the set of cells to be returned as part of a <*cubeCells*> element.
5. *GetCube*: this operation is used to retrieve a complete cube. It is based on the *GetCells* operation to get the data, and adds all the members that correspond to the mini-cube selection and the schema description of this mini-cube (metadata, i.e. dimensions, hierarchies, levels and measures structure). The XML document that is returned is a complete representation of the requested mini-cube (data and metadata), i.e. regrouping <*cubeSchema*>, <*cubeMembers*> and <*cubeCells*> elements.

As an example, a request for the *GetMembers* operator is presented in Listing 4 (omitting the SOAP headers for briefness). This request queries members in level "Province" of the "Store" dimension (in the same "Sales" cube presented in the previous section) for which the geographic bounding box (i.e. the smallest rectangle fully containing the limits of the region) intersects a box delimited by longitude 85°*W* to 55°*W* and latitude 44°*N* to 66°*N* (as seen in the GML <*Box*> element). The *GetMembers* operation leverages the Open Geospatial Consortium (OGC) Filter Encoding (21) XML notation for expressing spatial queries such as this one.

```
<GetMembersRequest
  xmlns="http://geosoa.scg.ulaval.ca/MinicubeWS">
  <cubeSelection name="Sales">
    <dimensionSelection name="Store">
      <levelSelection name="Province"
        include="levelonly" />
      <memberSelection>
        <memberSelectors>
          <selectMembersByProperty
            property="Province␣geometry">
            <GMLPropertySelector
              GMLPropertyName="the_geom">
              <ogc:Filter
                xmlns:ogc="http://www.opengis.net/ogc">
                <ogc:BBOX>
                  <ogc:PropertyName>
                    the_geom
                  </ogc:PropertyName>
                  <gml:Box
                    xmlns:gml="http://www.opengis.net/gml">
                    <gml:coordinates>
                      -85.0,44.0 -55.0,66.0
                    </gml:coordinates>
                  </gml:Box>
                </ogc:BBOX>
              </ogc:Filter>
            </GMLPropertySelector>
          </selectMembersByProperty>
        </memberSelectors>
      </memberSelection>
    </dimensionSelection>
  </cubeSelection>
</GetMembersRequest>
```

Listing 4. GetMembers request.

A response to this request is shown in Listing 5 (again omitting the SOAP headers). It consists of a <*GetMembersResponse*> element encapsulating a <*cubeMembers*> element, using the same syntax as described in subsection 3.2. The members returned in this document satisfy the

```
<mcws:GetMembersResponse
  xmlns:mcws="http://geosoa.scg.ulaval.ca/MinicubeWS">
  <gcml:cubeMembers
    xmlns:gcml="http://geosoa.scg.ulaval.ca/GeoCubeML"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:gml="http://www.opengis.net/gml"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://geosoa.scg.ulaval.ca/
         GeoCubeML␣CubeSchema.xsd">
    <gcml:cubeRef name="Sales">
      <gcml:dimensionRef name="Store">
        <gcml:levelRef name="Province">
          <gcml:member name="Quebec">
            <gcml:rollupTo member="Canada" />
            <gcml:propertyValue
              property="Province␣geometry">
              <!-- GML geometry of Quebec ... -->
            </gcml:propertyValue>
          </gcml:member>
          <!-- other members returned by the query ... -->
        </gcml:levelRef>
      </gcml:dimensionRef>
    </gcml:cubeRef>
  </gcml:cubeMembers>
</mcws:GetMembersResponse>
```

Listing 5. GetMembers response.

multidimensional (level and dimension selection) and spatial (bounding box query) criteria of the request.

4.2. Prototype Implementation

A prototype implementation of this Web Service has been developed (see Figure 4). It is based on Mondrian, an open source OLAP server, using a PostgreSQL DBMS (with PostGIS spatial extensions) for the data warehouse back-end. It uses Mondrian to query data cubes and extract a subset of these (i.e. a sub-cube) corresponding to a user-defined selection, done with operations from the service contract. This sub-cube is then serialized to the XML encoding, in order to deliver a mini-cube for use with a mobile client. The mobile SOLAP client must de-serialize the mini-cube to an embedded database prior to use, since XML is useful as an exchange format, but is not suitable for direct query (due to parsing overhead and lack of indexes).

We have done performance testing of this implementation to determine if its response time is adequate. We have used a data cube representing the population of Canadian administrative subdivisions, with a geographic dimension containing the subdivisions' boundaries (i.e. polygon geometries). The fact table has approximately 1.6 million rows at the most granular level of detail. Typical test queries were written for each Web Service operation. The *GetMembers* query applies, among other filters, a topological predicate for selecting spatial members within a rectangular area, and yields 21 members in total. The *GetCells* query returns a total of 297 cells. The set of four queries[2] was tested twice: once immediately after start-up of the Web Service (cold start) and again after executing the first set of queries (warm start). This was done to assess the impact of member and cell value caching (in RAM) within the OLAP server. The configuration used for testing is presented in Table 1 and the results in Table 2.

CPU	AMD Athlon 64 X2 3800+ (2 GHz, dual core)
RAM	2 GB (DDR)
Hard drive	400 Gb, 7200 RPM, SATA (Seagate ST3400832AS)
Operating system	Microsoft Windows XP Professional (Service Pack 2)
Java runtime	Sun Java JRE 1.5.0_12

Table 1. Test environment configuration.

Request	Response time (cold start)	Response time (warm start)
GetCapabilities	4.14 s	0.26 s
DescribeSchema	0.28 s	0.31 s
GetMembers	30.34 s	16.47 s
GetCells	5.31 s	0.30 s

Table 2. Response time of Web Service tests.

The first *GetCapabilities* query, run at cold start, took about 4 seconds to execute, whereas the second execution was much quicker. This is explained by the time required for the Web Service to be fully initialized (class loading and establishing DBMS connections). The *GetCells* query also shows significant improvement between the first and second execution: when run at cold start, the OLAP server must retrieve cell values from the fact table in the DBMS, and any

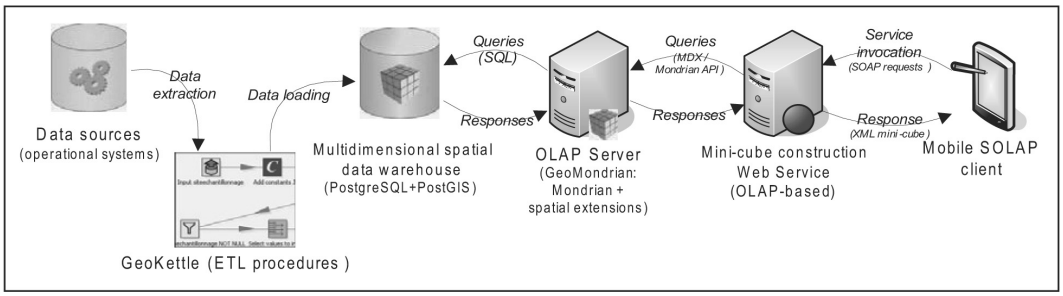

Figure 4. Architecture of the Web service for SOLAP mini-cube delivery.

[2] run in the following sequence: *GetCapabilities*, *DescribeSchema*, *GetMembers* and *GetCells*

required aggregate computation (for combinations of aggregate cell values which are not in an aggregate table or materialized view) occurs at this time. The results are put into Mondrian's in-memory cell cache, so further queries invoking cells already in this cache are faster to execute. This also shows that the XML serialization of the response, occuring every time the service is invoked, does not incur a heavy penalty for the *CubeCells* document.

The run time of the first *GetMembers* query is relatively long (30 seconds) while the second query takes about half the time. Spatial members, which are quite voluminous in data size, must be retrieved from the DBMS; at warm start these members will already be in the in-memory member cache. The remaining processing time (16 seconds) comprises evaluation of the topological predicate and serialization of the spatial data to GML. The topological filter is applied by the Web Service on all spatial members of the chosen level in the geographic dimension. This could be done more efficiently by extending the OLAP server so that all spatial functions can be processed natively, using advanced capabilities of the spatial DBMS such as spatial indexing to speed up the response time. However, this was not implemented in the current prototype. Serialization to GML could also be improved; for the moment this is handled by an external library, which produces GML documents in the form of character strings. These documents have to be parsed to a DOM (Document Object Model) representation, in order to integrate the geometry elements in the *CubeMembers* document returned as a response. Eliminating these inefficiencies would lead to a better response time.

In the light of this analysis, we feel that the response time of such a service, especially with an improved and optimized prototype, would be adequate for the use case mentioned at the beginning of this section.

5. Conclusion and Outlooks

The requirements and the specification of an XML encoding for SOLAP data cubes have been presented. It allows the exchange of geospatial decision-support data in an interoperable manner. It enables the representation of cube metadata and data, and supports spatial members and measures. Its use was also illustrated in a Web Service for delivering SOLAP cubes to mobile clients. Other uses include any situation where cubes have to be exchanged between different computing environments, as often required when integrating heterogeneous systems within an organization.

It should be noted that XML data size and verbosity could pose problems with regard to the *CubeCells* documents. Since a data cube can contain millions of facts, encoding all the individual cells with references to members would yield a very large document. This can be mitigated with data compression. However, this solution does not address parsing overhead. A better approach could consist of a more compact encoding: for example, a multidimensional cell set can be stored by first defining axes with ordered positions referring to members, and then sequentially writing all cell values. In addition, binary XML formats such as EXI (22) or Fast Infoset (23) could be used to further improve parsing speed and reduce data size, especially for mobile computing environments.

The proposed encoding addresses the requirements listed in Section 2, except for 2.6. Further work is required to enable the representation of calculated members: a new language, with a corresponding XML encoding, will have to be designed to describe calculated members in an interoperable manner. In addition, while the XML format provides for technical interoperability between SOLAP systems, it does not solve the problem of semantic interoperability, i.e. the integration of heterogeneous data cubes with conflicts in meaning, interpretation and intended uses. Ongoing research work aims to address these issues (24). Another research novelty which could eventually extend the XML cube model is the concept of raster-based spatial data cubes, based on a continuous grid division of space instead of discrete (vector) geometric features (13).

6. Acknowledgment

This work has been made possible with financial support from the NSERC Industrial Research Chair in Geospatial Databases for Decision Support (`http://mdspatialdb.chair.scg.ulaval.ca`) and the GeoSOA research group (`http://geosoa.scg.ulaval.ca`). We would also like to thank the Canadian Institute of Geomatics Scholarship Program.

References

[1] World Wide Web Consortium (W3C), Extensible Markup Language (XML) 1.0 (Fourth Edition), 2006.

[2] E. CERAMI, Web Services Essentials, O'Reilly, 2002.

[3] ISO TC/211, Geographic information – Geography Markup Language (GML) (Committee Draft, version 3.1.0), Open Geospatial Consortium, 2004.

[4] OGC (Open Geospatial Consortium), Web Feature Service Implementation Specification, version 1.1.0, 2005.

[5] S. RIVEST, Y. BÉDARD, M. J. PROULX, M. NADEAU, F. HUBERT, J. PASTOR. 2005. SOLAP: Merging Business Intelligence with Geospatial Technology for Interactive Spatio-Temporal Exploration and Analysis of Data. Journal of ISPRS "Advances in spatio-temporal analysis and representation", vol. 60, no. 1, pp. 17–33.

[6] E. F. CODD, S. B. CODD, C. T. SALLEY. 1993. "Providing OLAP to User-Analysts: An IT Mandate." Codd Ass.

[7] C. FRANKLIN, An introduction to geographic information systems: linking maps to databases. *Database*, vol. 15, no. 2, pp. 13–21, 1992.

[8] T. BADARD, Y. BÉDARD, F. HUBERT, E. BERNIER, E. DUBÉ, Web Services Oriented Architectures for Mobile SOLAP Applications. *International Journal of Web Engineering and Technology (IJWET)*, (in press), 2008.

[9] Microsoft Corporation, Hyperion Solutions Corporation, XML for Analysis Specification (version 1.1), 2002.

[10] Microsoft Corporation, Multidimensional Expressions (MDX) Reference, 2007.

[11] W. HÜMMER, A. BAUER, G. HARDE, XCube - XML For Data Warehouses. In: *DOLAP'03*, ACM, pp. 33–40, 2003.

[12] J. DA SILVA, V. C. TIMES, R. N. FIDALGO, R. S. M. BARROS, Providing Geographic-Multidimensional Decision Support over the Web. In *Proc.: Web Technologies Research and Development – APWeb 2005*, 7th Asia-Pacific Web Conference (Shanghai, China), pp. 477–488, 2005.

[13] Y. BÉDARD, T. MERRETT, J. HAN, Fundamentals of Spatial Data Warehousing for Geographic Knowledge Discovery. In: *Geographic Data Mining and Knowledge Discovery*, 2nd edition, Chapter 3, Taylor & Francis, (in press), 2008.

[14] R. KIMBALL, M. ROSS, The Data Warehouse Toolkit. Second Edition, New York, John Wiley & Sons, 2002.

[15] World Wide Web Consortium (W3C), XML Schema, 2004.

[16] E. ORT. 2005. "Service-Oriented Architecture and Web Services: Concepts, Technologies, and Tools." Sun Microsystems (online).

[17] E. DUBÉ, T. BADARD, Y. BÉDARD, Web Service for the Real-Time Delivery of SOLAP Mini-Cubes to Mobile Clients – A Service Oriented Architecture for Mobile Use of Geospatial Decisional Data. In *Proc.: AGEO/GQFD-Géo*, Clermont-Ferrand, France, 18 p., (in French), 2007.

[18] World Wide Web Consortium (W3C), Simple Object Access Protocol (SOAP) 1.1, 2000.

[19] World Wide Web Consortium (W3C), Web Services Description Language (WSDL) 1.1, 2001.

[20] World Wide Web Consortium (W3C), Efficient XML Interchange (EXI) Format 1.0 (W3C Working Draft), 2007.

[21] ITU-T, Information technology – Generic applications of ASN.1: Fast infoset (ITU-T Rec. X.891), 2005.

[22] T. SBOUI, Y. BÉDARD, J. BRODEUR, T. BADARD, A Conceptual Framework to Support Semantic Interoperability of Geospatial Datacubes. ER Workshops 2007, LNCS 4802, pp. 378–387, 2007.

Received: September, 2008
Revised: August, 2009
Accepted: September, 2009

Contact address:
Etienne Dubé
Department of Geomatic Science
Pavillon Louis-Jacques-Casault, Université Laval
Québec QC G1V 0A6 Canada
e-mail: etienne.dube@scg.ulaval.ca

ETIENNE DUBÉ is a research assistant at the Department of Geomatics Sciences, Laval University (Québec City, Canada). His areas of interest include geospatial databases, business intelligence and Web services.

THIERRY BADARD is Professor of geoinformatics at the Department of Geomatics Sciences, Laval University. He leads the GeoSOA research group and he is also a full time researcher of the Centre for Research in Geomatics (CRG) and of the GEOIDE NCE. His research interests deal with geospatial Business Intelligence, (Web) Services-oriented Architectures, location-based and context-aware services and the design of intelligent mobile applications for better decision support. For further details: http://geosoa.scg.ulaval.ca.

YVAN BÉDARD is Professor of GIS and Spatial Databases at Laval University. He holds the Canada NSERC Industrial Research Chair in Geospatial Databases for Decision Support (http://mdspatialdb.chair.scg.ulaval.ca). He is an active member of the Centre for Research in Geomatics where he acted as Director for 7 years, and of GEOIDE NCE. His research interest focuses on geospatial databases UML design methods, spatial datacubes and Spatial OLAP.