Development of a Query Language for GML based on XPath[*]

Jesús M. Almendros-Jiménez
University of Almería
Almeria, Spain
jalmen@ual.es

Antonio Becerra-Terón
University of Almería
Almeria, Spain
abecerra@ual.es

Francisco García-García
University of Almería
Almeria, Spain
paco.garcia@ual.es

Abstract

Geography Markup Language (GML) has been established as the standard language for the transport, storage and modelling of geographic information. In this paper we study how to adapt the XPath query language to GML documents. With this aim, we have defined a *semantic based XPath language* which is not based on the *(tree-based) syntactic structure* of GML documents, instead it is based on the *"semantic structure"* of GML documents. In other words, the proposed XPath language is based on the GML schema. We have developed a system called *UALGIS*, in order to implement the approach. Such system stores GML documents by means of the PostGIS RDBMS. In order to execute semantic-based XPath queries we have defined a translation of the queries into SQL. Such translation takes into account the GML schema. Finally, the system allows to visualize the result. With this aim, the result of a query is exported to the *Keyhole Markup Language (KML)* format.

1 Introduction

The *Geography Markup Language* (GML) [7, 20, 21, 8] has been established as the standard language for the transport, storage and modeling of geographic information. GML is a dialect of the *eXtensible Markup Language (XML)* [27], which adapts XML to Geo-spatial data. XML allows to describe the structure of Web data by means of a tree. The tree structure is used to describe relations between data: for instance, a paper tag contains author, title and publisher as subtree tags and the subtree publisher can be described by means of name of the journal, country, editors, etc. The need for querying XML documents has motivated the design of the *XPath* query language [28]. The XPath language allows to specify the path of the XML tree to be retrieved. In addition, XPath allows to constraint the query by means of boolean conditions about the attributes and tags of the selected nodes. For instance, we can specify in a query that we would like to retrieve the editors of the journals in which "Becerra" has published a paper as follows:

$$\texttt{/papers/paper[author} = \text{“}\texttt{Becerra}\text{”}\texttt{]/journal/editor}$$

XPath can be used for retrieving the GML elements. However, due to the usual syntactic structure of GML documents XPath should be adapted to this special case. GML allows to describe spatial objects including, if any, their geometry, together with the coordinate reference system, topology, among others. However, usually, GML documents do not use the tree-based structure of XML documents for the representation of spatial objects. Usually, GML documents store spatial objects as a sequence of XML elements, and they are stored as children of the tree root [18, 30]. It makes the tree-based XPath useless in most of the cases. The reason for storing spatial objects as a sequence of children of the root is that the tree structure is not used for representing spatial relations. One could think that a subtree of a node represents, for instance, the spatial objects enveloped by the node. However, it is not true in general. GML allows the specification of relations between spatial objects. However, GML allows to define a vocabulary of relations between spatial objects. For instance, the European INSPIRE Directive [12] has defined a certain vocabulary of GML whose aim is to create a spatial data infrastructure in the European

[*]This work has been partially supported by the Spanish MICINN under grant TIN2008-06622-C03-03.

Union in order to share information through public organizations. However, the syntatic structure of GML documents do not necessarily take into account the "semantic structure" of such vocabulary. Such vocabulary can be seen as a GML schema in such a way that spatial objects and relations between them conforms the schema. Usually, GML documents contain spatial objects in which spatial relations are specified by means of the linking mechanism of XML documents. Using XPath for GML documents, we could follow the links of the GML document in order to retrieve relationships between spatial objects, however, it makes XPath queries very sophisticated.

In this paper we study how to adapt the XPath query language to GML documents. With this aim, we have defined a *semantic based XPath language* which is not based on the *(tree-based) syntactic structure* of GML documents, rather than, it is based on the *"semantic structure"* of GML documents. In other words, the proposed XPath language is based on the GML schema instead of the syntactic structure.

We have developed a system called *UALGIS*, available via Web in `http://indalog.ual.es/ualgis/testGMaps.jsp` in order to implement the approach. Such system stores GML documents by means of the PostGIS [25]. In order to execute semantic-based XPath queries we have defined a translation of the queries into SQL. Such translation takes into account the GML schema. Finally, the system allows to visualize the result. With this aim, the result of a query is exported to the *Keyhole Markup Language (KML)* format [6].

1.1 Related Work

Spatial data can be handled by well-known relational database management systems (RDBMS) like: *SpatialSQL* [11], *GeoSQL* [13], *Oracle Spatial* [26] and *PostGIS* [25]. Basically, they are based on extensions of the relational model for storing spatial objects and extensions of the SQL query language for the retrieval of spatial queries.

In the case of GML data, the *Web Feature Service (WFS)* is a standard of the *OpenGis Consortium (OGC)* [22] for data manipulation of geographic features stored on a Web site (i.e. a *Web Feature Server*) using HTTP requests. The expressiveness of this language is very poor compared with query languages like SQL. *GQuery* [4] is a proposal for adding spatial operators to *XQuery* [5], the standard XML query language. Manipulation of trees and sub-trees are carried out by means of *XQuery*, while spatial processing is performed using geometric functions which use the *JTS Topology Suite* [1]. JTS is an open source API that provides a spatial object model and a set of spatial operators. The *GeoXQuery* approach [14] extends the *Saxon XQuery* processor [16] with function libraries that provide geo-spatial operations. It is also based on *JTS* and provides a GML to *Scalable Vector Graphics (SVG)* [29] transformation library for the XQuery processor in order to show query results.

With respect to *GQuery* and *GeoXQuery*, our proposal can be seen as a specific query language for GML instead of considering ad-hoc mechanisms for querying GML in *XPath* and *XQuery*. Our approach is focused on the *XPath* query language which is a sublanguage of the *XQuery* language. Our work can be seen as the first step to use *XQuery* as query language for GML documents. However, our proposal is based on the semantic structure of GML documents instead of the syntactic one used in *GQuery* and *GeoXQuery*. With respect to geometric functions, we are not still interested in queries involving geometric operations. Our current approach is mainly interested to querying semantic spatial relations expressed in GML documents which do not depend on the geometry of the objects. In any case, we will consider the extension of our work to geometric operations in the future by using the *PostGIS* library. Finally, our system is able to export the result of queries in order to visualize them, but instead of using SVG like in GeoXQuery, we export to KML.

GML Query [19] is also a contribution in this research line that stores GML documents in a spatial RDBMS. This approach performs a simplification of the GML schema that is then mapped to its corresponding relational schema. The basic values of spatial objects are stored as values of the tables. Once the document is stored, spatial queries can be expressed using the *XQuery* language with spatial functions. The queries are translated to their equivalent in SQL which are executed by means of the spatial RDBMS. This approach has some similarities with our. Firstly, the storage of GML documents in a spatial RDBMS. In our system, we store GML documents by means of the *PostGIS* RDBMS. Secondly, in our approach the queries are expressed in XPath and are translated into SQL. However, our XPath-based query language is properly based on the GML schema.

Another problem related to GML is how to visualize GML documents. The are several technologies (i.e. SVG, VRML, HTML, among others) to specify how to show the content of GML documents. KML [6] is an XML-based language focused on geographic data visualization, including annotation of maps and images, as well as controlling the display in the sense of where to go and where to look. From this perspective, KML is complementary to GML and most of the major standards of the OGC including *Web Feature Service* (WFS) [23] and *Web Map Service* (WMS) [24]. We have decided to export our GML query result to KML due to the advantages that offer this technology (i.e. APIs, WFS, WMS) and because it has been approved by the OGC as standard for the exchange and representation of geographic data in three dimensions. In this way the results can be interpreted by different GIS or Earth browsers. Other approaches return the output in SVG format but it has as disadvantage that it is just a graphical format. KML can display the results without losing the GML semantics and allows to include meta-data.

The rest of this article is organized as follows. Section 2 will present the GML data model and schema. Section 3 will define the semantic-based XPath language. Section 4 will describe the system and finally Section 5 will conclude and present future work.

2 GML Data Model

Next, we show an example of GML document representing the center of a city:

```
<CityCenter gml:id=”C1”>
        <gml:name>London </gml:name>
         <geometry>
             <gml:Point>
             <gml:pos>45.256 -71.92 </gml:pos >
             </gml:Point >
         </geometry >
         <cityCenterMember>
            <Building gml:id=”B1”>
                <gml:name>Great Building <gml:/name>
                <belongsTo>
                    <Block xlink:href=”BL1”/ >
                </belongsTo>
            </Building>
         </cityCenterMember>
         <cityCenterMember>
            <Building gml:id=”B2”>
                <gml:name>Small Building <gml:/name>
                <belongsTo>
                    <Block xlink:href=”BL2”/ >
                </belongsTo>
            </Building>
            </cityCenterMember>
            <cityCenterMember>
               <Block gml:id=”BL1”>
                   <gml:name>Grey Block <gml:/name>
                   <nextTo>
                       <Way xlink:href=”W1”/ >
                       <Way xlink:href=”W2”/ >
                   </nextTo>
               </Block>
            </cityCenterMember>
            <cityCenterMember>
               <Block gml:id=”BL2”>
                   <gml:name>Green Block <gml:/name>
               </Block>
            </cityCenterMember>
            <cityCenterMember>
               <Way gml:id=”W1”>
                   <gml:name>6th Street <gml:/name>
               </Way>
            </cityCenterMember>
            <cityCenterMember>
               <Way gml:id=”W2”>
                   <gml:name>5th Street <gml:/name>
               </Way>
            </cityCenterMember>
 </CityCenter>
```

As above mentioned, this GML document describes a center of a city. This description includes tags for *CityCenter* and *cityCenterMember*'s of a city, such as *buildings*, *blocks* and *ways*. However, a GML data model provides mechanisms for structuring GML documents. For instance, the previous GML document can be also represented as follows:

```
 <CityCenter gml:id=”C1”>
         <gml:name >London </gml:name >
         <geometry>
             <gml:Point>
             <gml:pos>45.256 -71.92 </gml:pos >
             </gml:Point >
         </geometry >
         <cityCenterMember>
            <Building gml:id=”B1”>
                <gml:name>Great Building <gml:/name>
                <belongsTo>
            <Block gml:id=”BL1”>
                <gml:name>Grey Block <gml:/name>
                <nextTo>
            <Way gml:id="W1">
                <gml:name>6th Street <gml:/name>
            </Way>
            <Way gml:id="W2">
                <gml:name>5th Street <gml:/name>
            </Way>
                </nextTo>
            </Block>
                </belongsTo>
            </Building>
         </cityCenterMember>
         <cityCenterMember>
            <Building gml:id="B2">
                <gml:name>Small Building <gml:/name>
                <belongsTo>
            <Block gml:id="BL2">
                <gml:name>Green Block <gml:/name>
            </Block>
                </belongsTo>
            </Building>
         </cityCenterMember>
         <cityCenterMember>
 </CityCenter>
```

where instead of using linking mechanisms of XML (i.e. *xlink:href*), the elements of the GML document are nested. However, it is normally recommended to avoid nesting of GML elements in order to do not increase the complexity of GML documents. From the point of view of using XPath for querying GML documents, it makes not easy to express complex queries w.r.t. GML documents. Our approach aims to propose a semantic version of XPath in which the result of the query does not depend on the syntactic structure of the GML document. In other words, nesting of elements is not relevant in our approach because we will follow the GML schema to define queries. Next will present a standard of GML schemas to be used in our approach.

2.1 INSPIRE Directive

The *European INSPIRE Directive* [12] aims to create a spatial data infrastructure in the European Union to share information through public organizations and facilitate public access across Europe. Furthermore, the spatial information considered under this policy is extensive and covers a wide range of areas and topics. INSPIRE is based on several common principles:

- Data should be collected once and should be stored where they can be maintained more efficiently.
- It must be able to easily combine the spatial information from different sources across Europe and share it with other users and applications.
- It should be possible to collect information at some detail level and share it with all levels, e.g. detailed for local analysis, general for global strategic purposes,. . .
- Geographic information should be transparent and readily available.

- It must be easy to find which geographic information is available, how it can be used to meet a specific need, and under which conditions can be acquired and used.

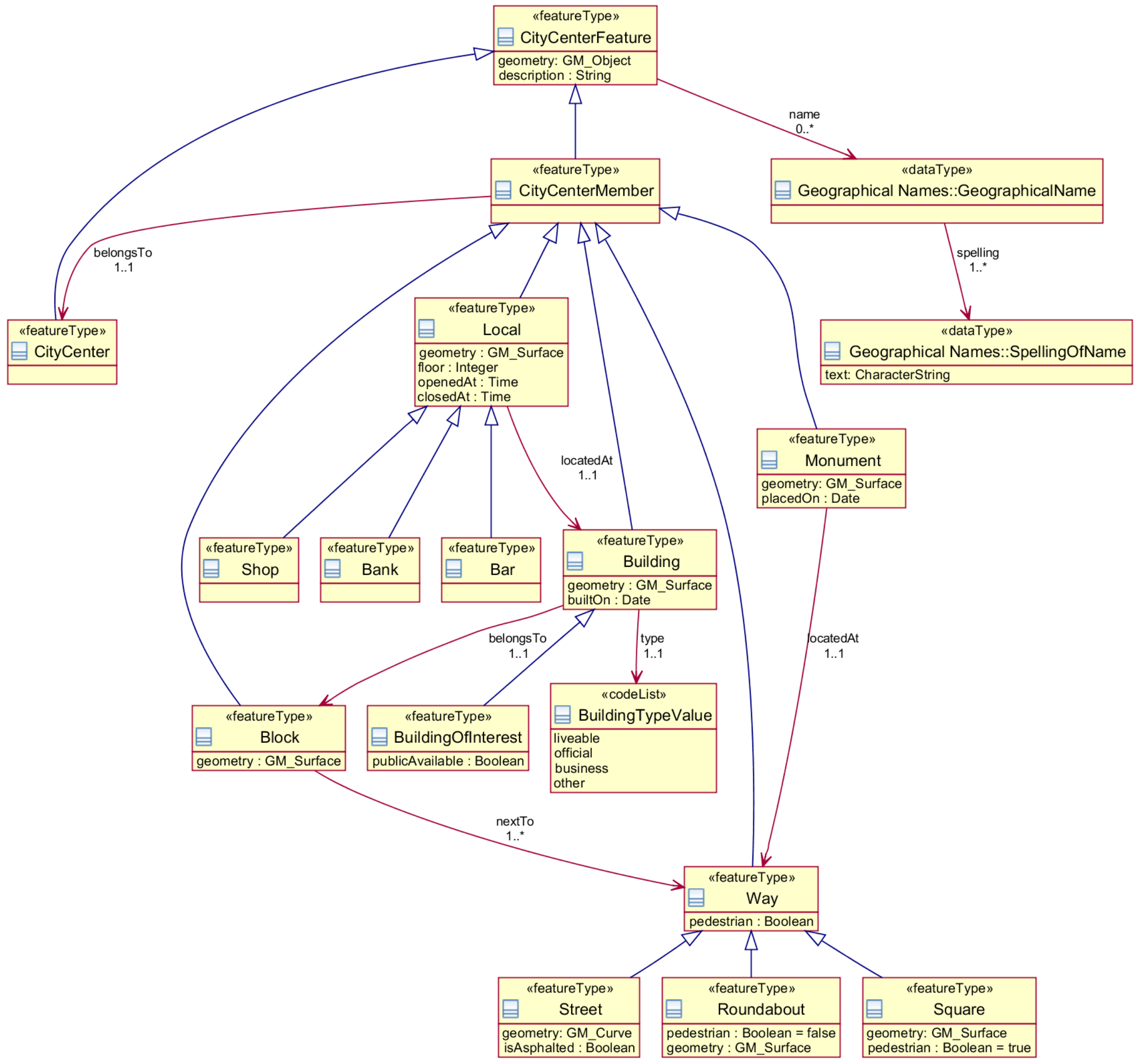


Figure 1: GML Schema of the City Center

The INSPIRE directive defines 34 topics on spatial data needed for application development. The INSPIRE directive defines GML schemas for each one of these topics. We have followed this directive in our approach in order to make our proposal more interesting in the real world.

For instance, the Figure 1 shows an example of GML schema for cities. This schema uses an UML Profile for its definition called HollowWorld [9] which is based on Table E.1 of ISO 19136:2007 (GML 3.2.1) [2] and is used by the INSPIRE Directive. It describes the elements (i.e. the members) of a *city center*. A city center contains differents entities like *locals*, *buildings*, *blocks*, *monuments* and *ways*. Locals can be *bars*, *shops* and *banks*. Buildings can be *buildings of interest*, and ways can be *roundabouts*, *streets* and *squares*.The GML schema also includes *spatial relationships* between the entities: locals are *'locatedAt"* buildings that *"belongsTo"* blocks which can be *"nextTo"* Ways. Monuments can be also *"locatedAt"* a way. GML allows to describe entities by means of the *Feature type*. In addition, the INSPIRE directive provides *Data types* for geographical entity naming.

In our system, we have used data available from *IDEAndalucía* [15] which is part of the geo-services of the INSPIRE directive. This is an *Andalusian Cartographic System Geo-portal* available to search, locate, view, download or request some type of geographic information referring to the territory of Andalusia in Spain. IDEAndalucía provides various services, such as WMS (Web Map Server) and WFS (Web Feature Service) with data available in GML format. By a *GetFeature* request to a WFS server a GML document is returned with all the Features of a selected type that are within the limits defined in a geographical rectangle or *GML Bounding Box*.

3 Development of a Query language for GML based on XPath

Now, the proposed GML query language is as follows. Basically, the path of the query has to follow the GML schema, and the query can include boolean conditions over the elements of the GML schema. For instance, we can express the following queries w.r.t. the running example:

Query 1. *Buildings of the Block named "Grey Block"*

/Building[belongsTo/Block/gml:name="Grey Block"]

Query 2. *Ways called "5th street" next to some Block*

/Block/nextTo/Way[gml:name="5th Street"]

Query 3. *Ways nextTo a Building called "Great Building"*

/Building[gml:name="Great Building"] /belongsTo/Block/nextTo/Way

The semantic version of the *XPath* query language is syntactically similar to the tree-based version. However, the semantic version can specify paths starting from any point of the schema (i.e. the root of the XPath expression can be any of the features of the GML document). The XPath expression alternates features with spatial relations. For instance, starting from *Building* we can build the following (semantic) XPath expression:

/Building[gml:name="Great Building"]/belongsTo/Block/nextTo/Way

by following the sequence *Building*, *belongsTo*, *Block*, *nextTo* and *Way*, where *Building*, *Block* are Features and *belongsTo* and *nextTo* are relationships among Features.

Finally, let us see the syntactic version of the above Query 1, which is considerably more complex than the proposed semantic one:

/CityCenter/cityCenterMember/Building[belongsTo/Block
[@xlink:href=/CityCenter/cityCenterMember/Block[gml:name="Grey Block"]/@gml:id]]

3.1 Translation XPath to SQL

In order to implement the proposed XPath-based query language, we have to proceed as follows:

1. A GML schema is transformed into a Relational schema.
2. A GML document is stored in the spatial RDBMS.

3. A XPath query is translated into an equivalent SQL query.
4. The result of the query is exported to GML or KML format.

3.1.1 Transforming the GML Schema into a Relational Schema

For data storage the first thing to do is to transform the GML schema into a relational schema of PostGIS. For this transformation we proceed as follows:

- A table is created for each element of Feature type.
- Attributes of elements of Feature type are mapped to columns of the tables.
- Geometric attributes of elements of Feature type are mapped to columns of PostGIS geometry type.
- Spatial relations between elements of Feature type are represented in two ways:
  - A one to one relationship is mapped to a column that references the primary key of the elements in the spatial relation.
  - A one to many relationship is mapped to a table, with the name of the spatial relation, with columns containing the primary keys (i.e. foreign keys) of the elements in the spatial relation.
  - A many to many relationship is mapped as two one-to-many relationships, one for each direction of the relationship.
- Feature inheritance is represented by table inheritance provided by PostGIS.

Figure 2 shows the result of the transformation of some elements of the GML schema represented in Figure 1.

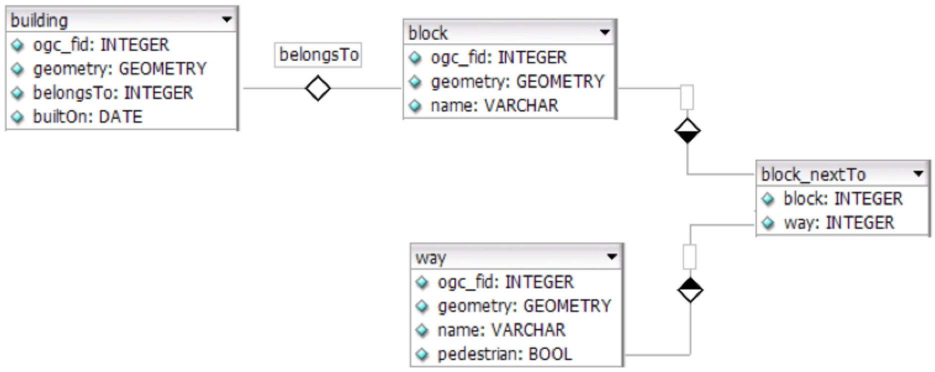


Figure 2: A Fragment of Relational schema of the City Center

3.1.2 Storage of GML documents in the Spatial RDBMS

The GML documents are stored in the spatial RDBMS as follows. Firstly, features instances are added to tables. Secondly, spatial relations are added to columns (in the case of one to one relationships) and to tables (in the case of one to many and many to many relationships). Next, we show the table instances of the running example:

Table: Building

ogc_fid	name	belongsTo
B1	Great Building	BL1
B2	Small Building	BL2
...	...	...

Table: Block

ogc_fid	name
BL1	Grey Block
BL2	Green Block
...	...

Table: Way

ogc_fid	name
W1	6th Street
W2	5th Street
...	...

Table: Block_nextTo

Block	Way
BL1	W1
BL1	W2
...	...

3.1.3 Translation of XPath into SQL

We can now define the translation of XPath-based queries into SQL queries, using the transformation of the GML Schema into the Relational Schema. The translation is as follows:

1. Case */A/p/B* where *p* is a one to one relationship: *Select B.* From A,B Where A.p = B.id*

2. Case */A/p/B* where *p* is a one to many relationship or a many to many relationship: *Select B.* From A,A_p,B Where A_p.A = A.id and A_p.B = B.id*

3. Case */A/p1/B/p2/C* where *p1,p2* are one to one relationships: *Select C.* From (Select B.* From A,A_p1,B Where A_p1.A = A.id and A_p1.B = B.id) B, C Where B.p2=C.id*

4. Case */A/p1/B/p2/C* where *p2* is a one to one relationship and *p1* is a one to one relationship: *Select C.* From (Select B.* From A,A_p1,B Where A_p1.A = A.id and A_p1.B = B.id) B, B_p2,C Where B_p2.B = B.id and B_p2.C = C.id*

5. Case */A[cond]/p/B[cond2]* where *p* is a one to one relationship: *Select B.* From A,B Where A.p = B.id and cond and cond2*

6. Case */A[cond]/p/B[cond2]* where *p* is a one to many relationship: *Select B.* From A,A_p,B Where A_p.A = A.id and A_p.B = B.id and cond and cond2*

7. Similarly, the rest of the cases

Now, we show the translation into SQL expressions of the XPath queries of Section 3 w.r.t. the GML schema represented by Figure 1.

Query 1. *Buildings of the Block named "Grey Block"*

XPath Query	SQL Expression
/Building[belongsTo/Block/gml:name="Grey Block"]	Select Building.* from Building,Block where Building.belongsTo = Block.id and Block.name = "Grey Block"

Next, we show the result of the query in GML format:

GML Output

```
<Building gml:id="B1">
    <gml:name>Great Building <gml:/name>
    <belongsTo>
        <Block xlink:href="BL1"/ >
    </belongsTo>
</Building>
```

Query 2. *Ways called "5th Street" next to some Block*

XPath Query	SQL Expression
/Block/nextTo/Way[gml:name="5th Street"]	Select Way.* from Way, Block_nextTo, Block where Block_nextTo.Block = Block.id and Block_nextTo.Way = Way.id and Way.name="5th Street"

Query 3. *Ways nextTo a Building called "Great Building"*

XPath Query	SQL Expression
/Building[gml:name="Great Building"]/belongsTo/Block/nextTo/Way	Select Way.* from Way, Block_nextTo, (Select Block.* from Building, Block where Building.belongsTo = Block.id and Building.name = "Great Building") Block_0 where Block_nextTo.Block = Block_0.id and Block_nextTo.Way = Way.id

3.1.4 Exporting to GML and XML

PostGIS natively provides several functions for the conversion of stored geometries to GML and KML formats:

- AsGML. Returns the geometry as a GML element. We can choose the spatial reference system.

- AsKML. It works similarly to AsGML but returning it as a KML geometry. We cannot choose the spatial reference system since it is fixed to *WGS84*.

These functions are only responsible for generating the geometry in GML / KML format but it is still required the export tables as GML and KML elements. With this aim, *PostGIS* provides two tables as follows:

- *Geometry_columns* table: It stores a catalog with the schema names, table names, column names of the geometric data and their spatial reference system.

- *Spatial_ref_sys* table: It contains a collection of spatial reference systems and stores information about the projections for transforming from one system to another.

4 *UALGIS* System

For validating the proposed query language a *Web Geographic Information System*, called *UALGIS*, has been implemented (available in `http://indalog.ual.es/ualgis/testGMaps.jsp`). The main features of the system can be summarized as follows:

- Storage of GML documents with PostGIS.

- GML / KML document creation using *dom4j* [17].

- Querying of elements of *feature* type of the database.

- XPath-based querying.

- *Google Maps*-based client for displaying the resut of queries.

The system has been built by using the *Java Eclipse Galileo* [10] as *IDE*. For project management, we have used *Maven 2* [3], which has been used for handling library dependences required for the construction of the *GIS*. The system architecture includes *PostGIS* version *1.4* server for data storage and a *Tomcat* version *6.0* server that handles the logic and presentation layers of the application. For the presentation layer, pages are programmed using *JSP*, *HTML* and *AJAX* for interacting with the server. Figure 3 shows the architecture.

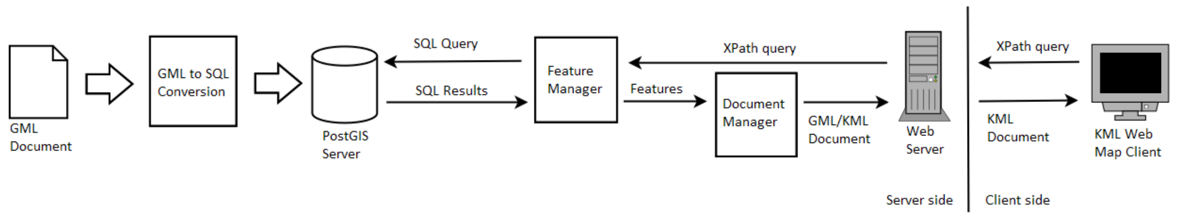
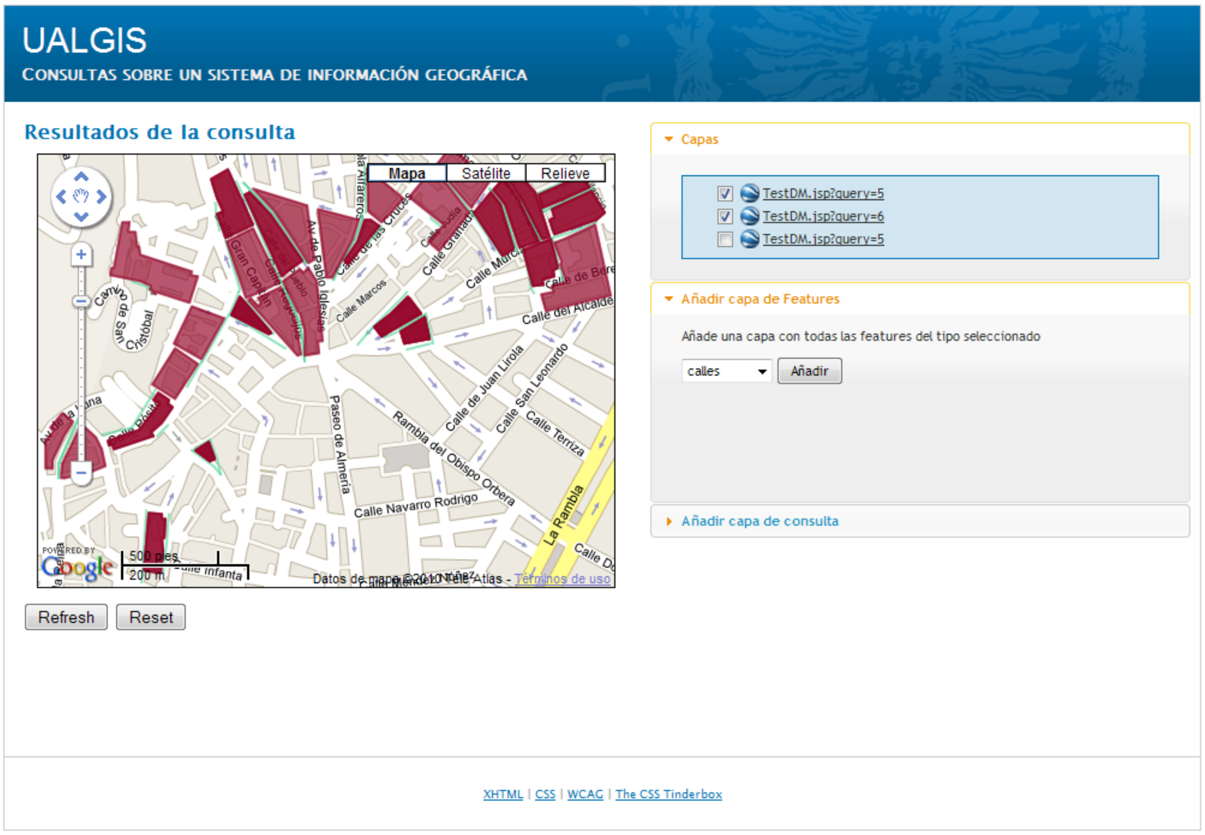


Figure 3: *UALGIS* System Architecture

The *FeatureManager* component is responsible for the management of *Features* stored in the database. It is responsible for translating *XPath* queries into *SQL* queries. The *FeatureManager* is also responsible for transforming the database rows to an object model that can be processed by the *DocumentManager* component. The *DocumentManager* component is responsible for handling documents in *GML/KML* format. It is also responsible for transforming the objects returned by *FeatureManager* onto KML /

GML documents. It also supports the *reverse* process, i.e. the extraction of features from GML/KML documents. For creating and reading documents the *dom4j* library is used. This is an *Open Source XML Framework* for *Java* that allows reading, writing and navigation of XML documents. It also includes a processing model based on events for large documents or XML streams and includes support for XML Schema validation types. On the client side, the *Web GIS* is a map browser with support for KML. It is based on the *Google Maps API* including a fast and efficient *2D* browser that provides features like zooming, panning, searching and displaying geographical names and information about geographic entities. Figure 4 shows a snapshot of the *UALGIS* system.

Figure 4: Snapshot of *UALGIS*


UALGIS
CONSULTAS SOBRE UN SISTEMA DE INFORMACIÓN GEOGRÁFICA
Resultados de la consulta
Mapa
Satélite
Relieve
Refresh
Reset
Capas
TestDM.jsp?query=5
TestDM.jsp?query=6
TestDM.jsp?query=5
Añadir capa de Features
Añade una capa con todas las features del tipo seleccionado
calles
Añadir
Añadir capa de consulta
XHTML | CSS | WCAG | The CSS Tinderbox

5 Conclusion and Future Work

In this paper we have studied how to adapt the XPath query language to GML documents. With this aim, we have defined a semantic based XPath language which is not based on the (tree-based) syntactic structure of GML documents, instead it is based on the "semantic structure" of GML documents. We have developed a system called *UALGIS*, in order to implement the approach. Such system stores GML documents by means of the PostGIS RDBMS. In order to execute semantic-based XPath queries we have defined a translation of the queries into SQL. Such translation takes into account the GML schema. Finally, the system allows to visualize the result. With this aim, the result of a query is exported to the KML format. As future work different techniques as GML filtering, indexing, etc.. will be studied in order to improve the performance of the *UALGIS* system. On the other hand, we would like to extend our work to the *XQuery* language. The extension should be also based on the semantic of GML. Finally, we would like to combine our GML query language with ontologies. Its use would improve the kind of queries and answers obtained from GML documents.

References

[1] Java Topology Suite (JTS). In Shashi Shekhar and Hui Xiong, editors, *Encyclopedia of GIS*, page 601. Springer, 2008.

[2] Iso 19136. In Ling Liu and M. Tamer Özsu, editors, *Encyclopedia of Database Systems*, pages 15–75. Springer US, 2009.

[3] Apache Software Foundation. Apache Maven. `http://maven.apache.org/`.

[4] O. Boucelma and F.M. Colonna. GQuery: a Query Language for GML. In *Proc. of the 24th Urban Data Management Symposium*, pages 27–29, 2004.

[5] D. Chamberlin, Denise Draper, Mary Fernández, Michael Kay, Jonathan Robie, Michael Rys, Jerome Simeon, Jim Tivy, and Philip Wadler. *XQuery from the Experts*. Addison Wesley, Boston, USA, 2004.

[6] OpenGIS Consortium. KML 2.2 Reference - An OGC Best Practice. `http://www.opengeospatial.org/standards/kml/`, 2008.

[7] OpenGIS Consortium. GML Specifications. `http://www.opengeospatial.org/standards/gml/`, 2010.

[8] J.E. Córcoles and P. González. GML as Database. *Handbook of Research on Geoinformatics*, page 1, 2009.

[9] Simon Cox. HollowWorld. `https://www.seegrid.csiro.au/twiki/bin/view/AppSchemas/HollowWorld`, 2009.

[10] Eclipse Foundation. Eclipse. `http://www.eclipse.org/`.

[11] Max J. Egenhofer. Spatial SQL: A Query and Presentation Language. *IEEE Trans. Knowl. Data Eng.*, 6(1):86–95, 1994.

[12] European Union. InspireE. `http://inspire.jrc.ec.europa.eu/`.

[13] F.Wang, J.Sha, H.Chen, and S.Yang. GeoSQL:a Spatial Query Language for Object-Oriented GIS. In *Proc. of the 2nd International Workshop on Computer Science and Information Technologies*, 2000.

[14] C.H. Huang, T.R. Chuang, D.P. Deng, and H.M. Lee. Building GML-native web-based geographic information systems. *Computers & Geosciences*, 2009.

[15] Junta de Andalucia. IDEAndalucia. `http://www.andaluciajunta.es/IDEAndalucia/IDEA.shtml`.

[16] M. Kay and S. Limited. Ten reasons why Saxon XQuery is fast. *IEEE Data Engineering Bulletin*, 1990.

[17] C. Kochmer and E. Frandsen. *JSP and XML: integrating XML and web services in your JSP application*. Sams, 2002.

[18] R. Lake. *Geography mark-up language (GML)*. Wiley, 2004.

[19] Yuzhen Li, Jun Li, and Shuigeng Zhou. GML Storage: A Spatial Database Approach. In Shan Wang, Dongqing Yang, Katsumi Tanaka, Fabio Grandi, Shuigeng Zhou, Eleni E. Mangina, Tok Wang Ling, Il-Yeol Song, Jihong Guan, and Heinrich C. Mayr, editors, *ER (Workshops)*, volume 32–89 of *Lecture Notes in Computer Science*, pages 55–66. Springer, 2004.

[20] C.T. Lu, R.F. Dos Santos, L.N. Sripada, and Y. Kou. Advances in GML for geospatial applications. *Geoinformatica*, 11(1):131–157, 2007.

[21] A.P. Need. Querying GML. *Handbook of Research on Geoinformatics*, page 11, 2009.

[22] OpenGis Consortium (OGC). OpenGis Specifications. `http://www.opengeospatial.org`, 2003.

[23] OpenGis Consortium (OGC). WFS Specifications. `http://www.opengeospatial.org/standards/wms`, 2008.

[24] OpenGis Consortium (OGC). WMS Specifications. `http://www.opengeospatial.org/standards/wms`, 2008.

[25] PostGis. PostGis, Geographic Objects for Postgres. `http://postgis.refractions.net`, 2003.

[26] Siva Ravada and Jayant Sharma. Oracle8i Spatial: Experiences with Extensible Databases. In Ralf Hartmut Güting, Dimitris Papadias, and Frederick H. Lochovsky, editors, *SSD*, volume 16–51 of *Lecture Notes in Computer Science*, pages 355–359. Springer, 1999.

[27] W3C. Extensible Markup Language (XML). Technical report, 2007.

[28] W3C. XML Path Language (XPath) 2.0. Technical report, 2007.

[29] W3C Recommendation. Scalable Vector Graphics (SVG) 1.0 Specification, 2001.

[30] S. Wei, G. Joos, and W. Reinhardt. Management of Spatial Features with GML. In *Proceedings of the 4th AGILE Conference on Geographic Information Science, Brno, Czech Republic*, pages 370–375, 2001.