XPath for Querying GML-Based Representation of Urban Maps*

Jesús M. Almendros-Jiménez, Antonio Becerra-Terón, and Francisco García-García

Information Systems Group
University of Almería
04120-Spain
{jalmen,abecerra,paco.garcia}@ual.es
http://indalog.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 *XPath based query language* which handles the *"semantic structure"* of GML. Our approach focuses on querying urban maps whose representation is based on GML. 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.

Keywords: Query languages, GML, XPath, KML, GIS, SQL.

1 Introduction

The *Geography Markup Language* (GML) [7,19,20,8] 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)* [26] for 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 tree node with tag paper contains author, title and publisher as subtrees 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 [3]. 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

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

B. Murgante et al. (Eds.): ICCSA 2011, Part I, LNCS 6782, pp. 177–191, 2011.

selected nodes. For instance, we can query the editors of the journals in which "Becerra" has published a paper. It can specified as follows:

/papers/paper[author/last = "Becerra"]/journal/editor

GML allows to describe spatial objects, including their geometry, the coordinate reference system, etc. However, in practice, GML does not use the tree-based structure of XML documents. In most of cases, GML documents store spatial data as a sequence of XML elements, and they are stored as children of the tree root [17,28]. It makes XPath not useful for GML querying. In other words, the tree structure is not used for representing *spatial relations*. One could think that the subtrees of a node represent, for instance, the spatial objects enveloped by the node. However, it is not true in general.

GML allows the specification of relations between spatial data. In particular, the *GML schema* allows to define a vocabulary of relations. For instance, the European INSPIRE Directive [12] has defined a certain vocabulary for GML whose aim is to create a spatial data infrastructure in the European Union in order to share information through public organizations. However, the syntatic structure of GML does not necessarily take into account the "semantic structure" of such vocabulary. In practice, GML documents describe spatial relations by means of the linking mechanism (i.e., *xlink:href*) of XML documents. Using the standard XPath, 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 *XPath based query language* which handles the *"semantic structure"* of GML documents. Our approach focuses on querying urban maps whose representation is based on GML. In such a case, urban maps describe city elements: streets, buildings, buildings of interest, shops, etc. Such city elements can be related by several spatial relations: located at, next to, belongs to, etc. Such relationships can be modelled by means of a suitable GML based data model. We have developed a system called *UALGIS*, available via Web in `http://indalog.ual.es/ualgis/index.jsp` in order to implement the approach. Such system stores GML by means of the PostGIS [23]. 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* [24] and *PostGIS* [23]. Basically, they are based on extensions of the relational model for storing spatial objects, and provide an enriched 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)* [21] for data manipulation of geographic features stored on a Web site (i.e., a *Web Feature Server*) using *http* requests. The expressiveness of WFS 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, which includes XPath as sublanguage. Manipulation of trees and sub-trees are carried out by means of *XQuery*, while spatial query processing is performed using geometric functions of the *JTS Topology Suite* [25]. JTS is an open source API that provides a spatial 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)* [27] transformation library 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. 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 operations, we are not still interested in such queries. We are mainly interested to query semantic spatial relations expressed in GML, which do not depend on the geometry of the objects, such as located at, next to, etc. In any case, geometric operations are considered as future work, which will be included thanks to the *PostGIS* libraries. Finally, our system is able to visualize query results, but instead of using SVG like in GeoXQuery, we export to KML.

GML Query [18] is also a contribution in this research line, storing GML data in a spatial RDBMS. This approach, firstly, performs a simplification of the GML schema, and secondly, the GML schema is mapped to the corresponding relational schema. The basic values of spatial data 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 SQL which are executed by means of the spatial RDBMS. This approach has some similarities with our. Firstly, the storage of GML data by the spatial RDBMS taking into account the GML schema. We store GML documents in the *PostGIS* RDBMS. Secondly, in our approach the queries are expressed in XPath and translated into SQL. In contrast, our XPath-based query language is properly based on the GML schema, which makes easier the specification of queries and the translation to SQL.

Another problem related to GML is how to visualize documents. There 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 WFS and the *Web Map Service* (WMS) [22]. We have decided to export query results to KML due to the advantages that offer this technology (i.e., APIs, WFS, WMS). In addition, KML 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. Some of the quoted approaches return the query results in SVG format which is just a graphical format. KML can display the results without losing GML semantics, allowing to include meta-data/schema.

The structure of the paper is 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 an urban map:

```
<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>
```

This document describes the center of a city. This description includes tags for *CityCenter* and *cityCenterMember*'s of a city, such as *buildings*, *blocks* and *ways*. However, the same semantic content can be expressed in several syntactic ways. 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>
</CityCenter>
```

where instead of using linking mechanisms of XML (i.e., *xlink:href*), the elements of the document are nested. However, it is normally recommended to avoid nesting 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. Our approach aims to propose a semantic version of XPath in which the result of a query does not depend on the syntactic structure. 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 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.

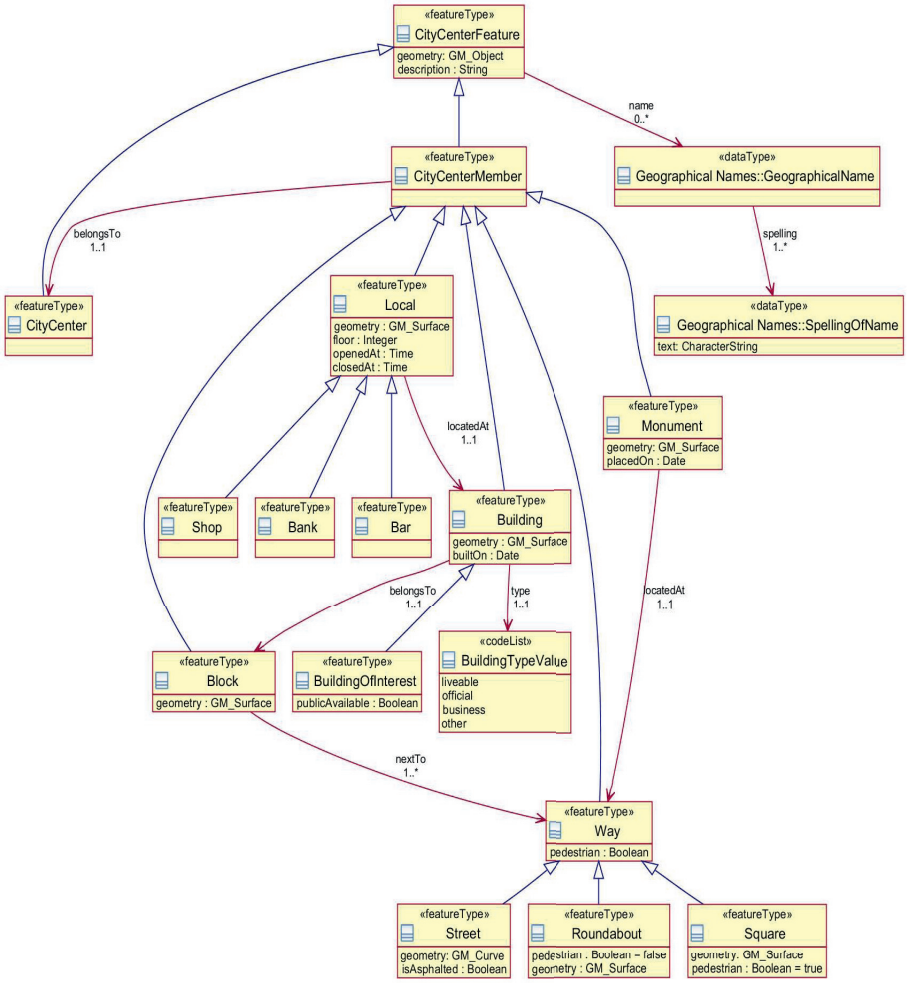

Fig. 1. GML Schema of the Urban Map

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 and extended this directive in our approach in order to make our proposal more interesting in the real world.

For instance, Figure 1 shows an example of GML schema for urban maps. This schema uses an UML Profile for its definition called HollowWorld [9]. It is based on Table E.1 of ISO 19136:2007 (GML 3.2.1) [1] and is used by the INSPIRE Directive. Figure 1 describes the elements (i.e., members) of a *city center*. A city center contains different 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 schema allows to describe entities by means of the *Feature type*. In addition, the INSPIRE directive provides *data types* for geographical entity naming.

3 XPath for Querying GML-Based Representation of Urban Maps

Now, we present the proposed XPath based query language. Basically, the path of the query has to follow the GML schema of Figure 1, and the query can include boolean conditions on 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. *(Pieces of) Ways called "5th street" next to some Block*

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

Query 3. *Ways next to 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 GML 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]]

In summary, the proposed semantic version of XPath makes easier to the user the formulation of queries: (s)he does not need to know the physical structure of the GML document. Rather than it is enough to known the GML schema. The translation of the semantic XPath expression to a syntactical one is hidden to the user.

3.1 Translation XPath to SQL

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

1. The GML schema is transformed into a Relational schema.
2. The GML document is stored in the spatial RDBMS.
3. The XPath query is translated into an equivalent SQL query.
4. The result of the query is exported to GML or KML format.

Transforming the GML Schema into a Relational Schema. For data storage the GML schema has to be transformed into a relational schema of PostGIS. For this transformation we proceed as follows:

- A table is created for each element of Feature type of the GML schema.
- Attributes of elements of Feature type are mapped to columns of the corresponding 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 as follows:
  - 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, named as the name of the feature + name of 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.
- Feature inheritance is represented by table inheritance in PostGIS.

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

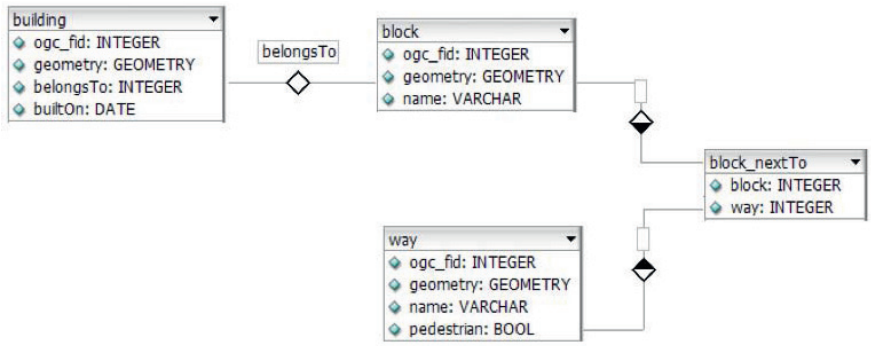

Fig. 2. A Fragment of Relational schema of the Urban Map

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
...	...

Translation of XPath into SQL. We can now define the translation of XPath-based queries into SQL queries. It is based on 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.ogc_fid*
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.ogc_fid and A_p.B = B.ogc_fid*
3. Case */A[cond1]/p/B[cond2]* where p is a one to one relationship: *Select B.* From A,B Where A.p = B.ogc_fid and cond3 and cond4*, where *cond3* and *cond4* are the translation of *cond1* and *cond2*, respectively.
4. Case */A[cond1]/p/B[cond2]* where p is a one to many relationship: *Select B.* From A,A_p,B Where A_p.A = A.ogc_fid and A_p.B = B.ogc_fid and*

cond3 and cond4, where *cond3* and *cond4* are the translation of *cond1* and *cond2*, respectively.

5. 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 of 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.ogc_fid and Block.name = "Grey Block"

The result of the query in GML format is as follows:

GML Output

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

Query 2. *(Pieces of) 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.ogc_fid and Block_nextTo.Way = Way.ogc_fid and Way.name="5th Street"

Query 3. *Ways next to 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.ogc_fid and Building.name = "Great Building") Block_0 where Block_nextTo.Block = Block_0.ogc_fid and Block_nextTo.Way = Way.ogc_fid

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

- AsGML: Returns the geometries as GML elements, allowing the selection of the spatial reference system.
- AsKML: It works similarly to AsGML but returning the geometries as KML geometries. We cannot select the spatial reference system because it is fixed to *WGS84*.

These functions are only responsible for generating the geometries in GML / KML format but it is still required the exporting of 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 and table names, and column names of geometric data and their spatial reference system.
- *Spatial_ref_sys* table: It contains a collection of spatial reference systems and stores information about projections for transforming from one system to another.

4 *UALGIS* System

For validating the proposed query language a *Web Geographic Information System*, called *UALGIS (University of ALmería Geographic Information System)*, has been implemented. It is available from `http://indalog.ual.es/ualgis/index.jsp`. In summary, the main features of the UALGIS system are the following:

- XPath-based querying.
- *Google Maps*-based client for displaying the resut of queries.
- Querying of elements of *feature* type of the database.

For testing 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 geographic information referring to the territory of Andalusia in Spain. IDEAndalucía provides several services, such as WMS (Web Map Server) and WFS (Web Feature Service), with data available in GML format. By a *GetFeature* request to the WFS server a GML document is obtained, with all the features enclosed in a rectangle (i.e., *GML Bounding Box*) of a selected type.

The system has been built by using the *Java Eclipse Galileo* [10] as *IDE*. For project management, we have used *Maven 2* [2] for handling library dependences. Figure 3 shows the system architecture.

- The system architecture includes the *PostGIS* server, version *1.4*, and the *Tomcat* server, version *6.0*, for handling logic and presentation layers. For the presentation layer, pages are programmed using *JSP*, *HTML* and *AJAX*.

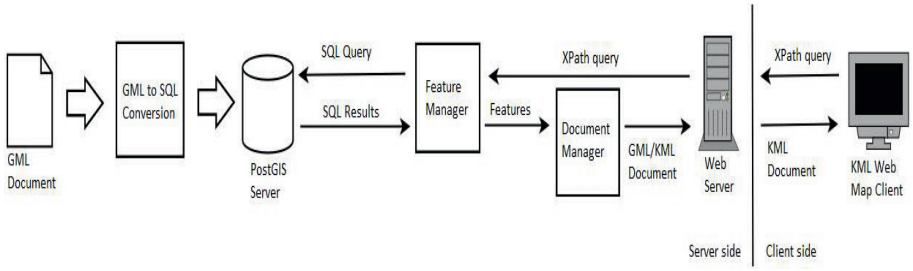

Fig. 3. *UALGIS* System Architecture

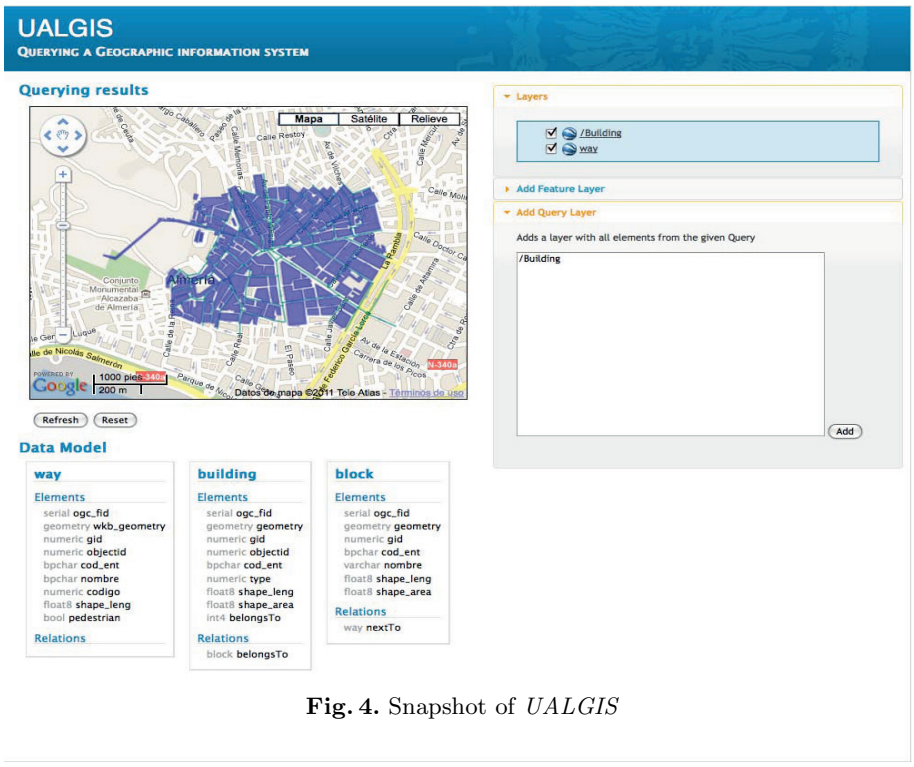

Fig. 4. Snapshot of *UALGIS*

– The *FeatureManager* component is responsible of the management of the features stored in the database. It is also responsible of the translation of *XPath* queries into *SQL* queries, and the transformation of 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 obtained from the *FeatureManager* into 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.
- 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.

5 Conclusion and Future Work

In this paper we have studied how to adapt the XPath query language to GML documents representing urban maps. With this aim, we have defined a XPath-based query language which handles 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 in KML format. As future work, firstly, we would like to integrate GML filtering and indexing, among others, in the UALGIS system in order to improve the performance. Currently, UALGIS has been tested with small examples. We would like to have a good benchmark study to compare UALGIS with similar systems. Secondly, we would like to extend our work to the *XQuery* language. Finally, we would like to combine our GML query language with spatial ontologies. Its use would improve the kind of queries and answers obtained from GML documents.

References

1. Iso 19136, Encyclopedia of Database Systems (2009)
2. Apache Software Foundation. Apache Maven, `http://maven.apache.org/`
3. Berglund, A., Boag, S., Chamberlin, D., Fernandez, M.F., Kay, M., Robie, J., Siméon, J.: XML path language (XPath) 2.0. W3C (2007)
4. Boucelma, O., Colonna, F.M.: GQuery: a Query Language for GML. In: Proc. of the 24th Urban Data Management Symposium, pp. 27–29 (2004)
5. Chamberlin, D., Draper, D., Fernández, M., Kay, M., Robie, J., Rys, M., Simeon, J., Tivy, J., Wadler, P.: XQuery from the Experts. Addison Wesley, Boston, USA (2004)
6. OpenGIS Consortium. KML 2.2 Reference - An OGC Best Practice (2008), `http://www.opengeospatial.org/standards/kml/`
7. OpenGIS Consortium. GML Specifications (2010), `http://www.opengeospatial.org/standards/gml/`
8. Córcoles, J.E., González, P.: GML as Database. Handbook of Research on Geoinformatics (2009)
9. Simon Cox. HollowWorld (2009), `https://www.seegrid.csiro.au/twiki/bin/view/AppSchemas/HollowWorld`

10. Eclipse Foundation. Eclipse, http://www.eclipse.org/
11. Egenhofer, M.J.: Spatial SQL: A Query and Presentation Language. IEEE Transactions on Knowledge and Data Engeneering 6(1), 86–95 (1994)
12. European Union. InspireE, http://inspire.jrc.ec.europa.eu/
13. Wang, F., Sha, J., Chen, H., Yang, S.: GeoSQL:a Spatial Query Language for Object-Oriented GIS. In: Proc. of the 2nd International Workshop on Computer Science and Information Technologies (2000)
14. Huang, C.H., Chuang, T.R., Deng, D.P., Lee, H.M.: Building GML-native web-based geographic information systems. Computers & Geosciences (2009)
15. Junta de Andalucia. IDEAndalucia, http://www.andaluciajunta.es/IDEAndalucia/IDEA.shtml
16. Kay, M., Limited, S.: Ten reasons why Saxon XQuery is fast. IEEE Data Engineering Bulletin (1990)
17. Lake, R.: Geography mark-up language (GML). Wiley, Chichester (2004)
18. Li, Y., Li, J., Zhou, S.: GML Storage: A Spatial Database Approach. In: ER (Workshops) On Spatial Database Approach. LNCS, vol. 3289, pp. 55–66. Springer, Heidelberg (2001)
19. Lu, C.T., Dos Santos, R.F., Sripada, L.N., Kou, Y.: Advances in GML for geospatial applications. Geoinformatica 11(1), 131–157 (2007)
20. Need, A.P.: Querying GML. Handbook of Research on Geoinformatics, 11 (2009)
21. OpenGis Consortium (OGC). OpenGis Specifications (2003), http://www.opengeospatial.org
22. OpenGis Consortium (OGC). WMS Specifications (2008), http://www.opengeospatial.org/standards/wms
23. PostGis. PostGis, Geographic Objects for Postgres (2003), http://postgis.refractions.net
24. Ravada, S., Sharma, J.: Oracle8i Spatial: Experiences with Extensible Databases. In: Güting, R.H., Papadias, D., Frederick Lochovsky, H. (eds.)SSD, LNCS, vol. 1651, pp. 355–359. Springer, Heidelberg (2001)
25. Shekhar, S., Xiong, H.: Java Topology Suite (JTS). In: Encyclopedia of GIS, p. 601. Springer, Heidelburg (2008)
26. W3C. Extensible Markup Language (XML). Technical report, W3C (2007)
27. W3C Recommendation. Scalable Vector Graphics (SVG) 1.0 Specification (2001)
28. Wei, S., Joos, G., Reinhardt, W.: Management of Spatial Features with GML. In: Proceedings of the 4th AGILE Conference on Geographic Information Science, pp. 370–375. Brno, Czech Republic (2001)