Studying an Approach to Query Spatial XML

J. E. Córcoles, P. González

LOuISE Research Group. Dept. Informática
Universidad de Castilla-La Mancha
02071, Albacete (Spain)
{corcoles, pgonzalez}@info-ab.uclm.es

Abstract. A contribution to the Geospatial semantic Web is the definition of an approach for integrating non-spatial resources like HTML, PDF, GIF, etc., with spatial XML resources represented by Spatial XML [5]. However, in order to put this approach into practice it is necessary to solve the problem of developing an integration system for querying spatial XML resources stored in different sources. In this paper, we have implemented an approach for querying spatial and non-spatial information represented in the Geographical Markup Language (GML). A performance study has been carried out and the results are given.

1. Introduction

With the growth of the World Wide Web has come the insight that currently available methods for finding and using information on the Web are often insufficient. Today's retrieval methods are typically limited to keyword searches or sub-string matches, offering no support for deeper structures that might lie hidden in the data or that people typically use to reason; therefore, users may often miss critical information when searching the Web. At the same time, the structure of the posted data is flat, which increases the difficulty of interpreting the data consistently. There would exist a much higher potential for exploiting the Web if tools were available that better match human reasoning. In this vein, the research community has begun an effort to investigate foundations for the next stage of the Web, called *Semantic Web* [1].

A rich domain that requires special attention is the Geospatial Semantic Web [2]. The enormous variety of encoding of geospatial semantics makes it particularly challenging to process requests for geospatial information. In the future, the Geospatial Semantic Web will allow the returning of both spatial and non-spatial resources to simple queries, using a browser. For example, a query "*lakes in Maine*" should return all relational resources with lakes in Maine (pictures, text, ...) in different formats (XML, HTML, JPG, PDF, References, ...) [2].

However, in the same way as with the Semantic Web, in order to approach the Semantic Geospatial Web it is necessary to solve several problems. One of these is the integration of spatial and non-spatial resources with different schemas stored in different sources.

Tackling the integration of spatial information on the Web is not a simple task since there are several High level problems (e.g. heterogeneity for representing spatial information, for using spatial operators, for defining the semantic of the objects, etc. [3]) and Low level problems (e.g. the sources may store large amounts of incomplete spatial data, which may make it necessary to join the results of queries with spatial joins [4]).

In order to contribute to the Geospatial Semantic Web, in this paper we study a prototype for querying spatial XML resources. The main task of this approach is to provide users with a unique interface for querying spatial XML resources with different schemas, independently of their actual organization and location. It provides the infrastructure for formulating structured spatial queries by taking into consideration the conceptual representation of a specific domain in the form of an ontology. The resources are integrated using RDF. This work is based on [5], and we detail how to use RDF for integrating non-spatial resources (HTML, PDF, etc.) with spatial XML resources represented by GML. The most novel and critical feature of this approach is the querying of spatial XML resources, because it uses a different way from that of querying and relating non-spatial resources. Therefore, our prototype is focused on this alone. After proving the viability of this prototype, making a prototype in accordance with [5] is a simply task.

In our study the spatial information is represented by GML because it is an XML encoding for the transport and storage of spatial/geographic information, including both spatial features and non-spatial features. The mechanisms and syntax that GML uses to encode spatial information in XML are defined in the specification of OpenGIS [6]. Thus, GML allows a more homogeneous and flexible representation of the spatial information.

Query mediation has been extensively studied in the literature for different kinds of mediation models and for the capabilities of various sources: in the field of non-spatial integration there are several approaches such as *Tsimmis* [7], *YAT* [8], *Information Manifold* [9]. More directly concerned with the integration of XML resources, it is worth noting C-Web Portal [10] and [11]. C-Web Portal supports the integration of non-spatial resources on the Web, and it provides the infrastructure for formulating structured queries by taking into consideration the conceptual representation of a specific domain in the form of an ontology. On the other hand, [11] proposes a *mediator* architecture for the querying and integration of Web-accessible XML data resources (non-spatial data). Its contribution is the definition of a simple but expressive mapping language, following a *local as view* approach and describing XML resources as local views of some global schema. Both of these approaches have aims in common with our approach, but they are only focused on non-spatial information.

In relation to spatial XML integration, the approaches developed by [12], [13] and [14] stand out. [12] presents a mediation system that addresses the integration of GIS data and tools, following a *global-as-view* approach. It has a multi-tier client-server architecture based on WFS and uses standard wrappers to access data, extended by derived wrappers that capture additional query capabilities. [13] extends the MIX wrapper-mediator architecture for integrating information from spatial information systems and searchable databases of geo-referenced imagery. MIX is focused on integrating geo-referenced imagery but our approach is focused on spatial geometries.

On the other hand, [14] designed a novel approach for integrating GML resources. The proposed architecture uses a Catalog expressed by RDF to relate the GML resources. Although [14] has the same aims as [5], it has a different focus.

An overview of the architecture offered in this work is detailed in Section 2, which contains the most important features of our prototype. Section 3 provides some results from our performance studies. Section 4 contains conclusions and projected future work.

2. Overview

The main task of an integration mediator is to provide the users with a unique interface for querying the data, independently of its actual organization and location [15]. This interface, or global schema, is described as an *ontology* expressed with RDF(S). As used here, an *ontology* denotes a light-weight conceptual model and not a hierarchy of terms or a hierarchy of concepts [11]. The global schema can be viewed as a simple object-oriented data model. Hence, a global schema can be viewed as defining a database of objects, connected by roles, with the concept extents related by subset relationships as per the *isA* links in the schema. Since it is an integration schema, this is a *virtual* database. The actual materialization exists in the resources.

Users pose queries in terms of global schema (RDFS). As a consequence, the mediator system must contain a module (*Solve mapping*) that uses the resource descriptions in order to translate a user query into a query that refers directly to the schemas of the GML resources. For this purpose, we establish a correspondence between each resource and the global schema using RDF instances. In the *Solve mapping* module, an execution plan is not necessary because a query will be executed over a source if it fully satisfies all attributes of the query.

In addition, this approach supplies *descriptions* of the resources and specifies constraints on resource contents (e.g. contains information about "Madrid"). These *descriptions* are stored as alphanumeric data in RDF instances (Filter Resources).

A Wrapper receives a query over a schema of the resources from the mediator system. Here, the query is executed and the results of the query are expressed in GML format, which are then returned to the mediator.

This process is a well-known pattern in applications over non-spatial XML documents. However, to apply this architecture efficiently over spatial XML documents it is necessary to take into account the following points:

1. The spatial query language used for the users should have the same semantic as the query language used in the sources, so the translation between both languages is easier. Because of this, we have used the same spatial language to query the ontology and the GML data model on the sources[5]. This query language was born as a spatial query language over spatial semi-structured XML documents. The data model and the algebra underlying the query language are defined in [16]. The query language has a familiar *select-from-where* syntax and is based on SQL. It includes a set of spatial operators (disjoint, touches, etc.), and includes traditional operators (=, >, <, ...) for non-spatial information.

2. Unlike XML, it is very difficult to make efficient queries over GML because the spatial operator requires a spatial index that cannot be created directly over GML. Due to this an efficient method for storing GML documents is necessary. In [17] we studied three storage models for storing and retrieving GML documents over RDBMS. This work concluded that the LegoDB approach [18] obtains the best performance. We have therefore used this approach in our implementation.
3. Obviously, in schema integration the mediator can know when a resource has schema to satisfy a query. However, only when the query is executed in the resources does the mediator know whether the information stored in the resource satisfies the *where clause.* Thus, for example*,* it is possible that 100 resources have schema to satisfy the query, but only 10 have the information that the user wants. In this case, posting the query to each resource (100) is inefficient. To solve this, our approach uses the above-mentioned *description* to obtain *a priori* a set of candidate resources (in the same way as a *Catalog* in [19]).

In the following section the most important features of our prototype are detailed.

2.1 Ontology

The solution proposed in this paper is based on [5], which details how to use RDF(S) for querying spatial XML resources (GML). [5] is based on *Community Web Portal (C-Web)* [20]. *C-Web* essentially provides the means to select, classify and access, in a semantically meaningful and ubiquitous way, various information resources for diverse target audiences. However, in [5] we do not only use the RDF(S) to relate resources (html, pdf, jpg,…) situated in different web sites; we also use it to know how a schema satisfies an ontology. These schemas are always expressed as DTDs for each source. Each site can offer the schemas that satisfy, fully or partially, the ontology defined for an interest community. Therefore, unlike the original C-Web, it is possible to apply spatial operators (comparatives: cross, overlap, touch; analysis: Area, Length) over the resources provided that they represent geometry information with GML.

In Figure 1 an example of the *Portal Schema* and its instances is shown. Due to RDF's capability for adding new feature and geometry types in a clear and formal manner, this example has been carried out extending the geospatial ontology defined by OpenGIS [6], where the class (*Geometry, LineString, etc*) and properties (*coordinates, PolygonMember, etc*) are defined. The example shows a simplification of a City Model. In this, *CityModel* contains *Blocks* and a *Block* contains *Parcels*, and a *CityModel* has the properties *name*, *population* and *Category*.

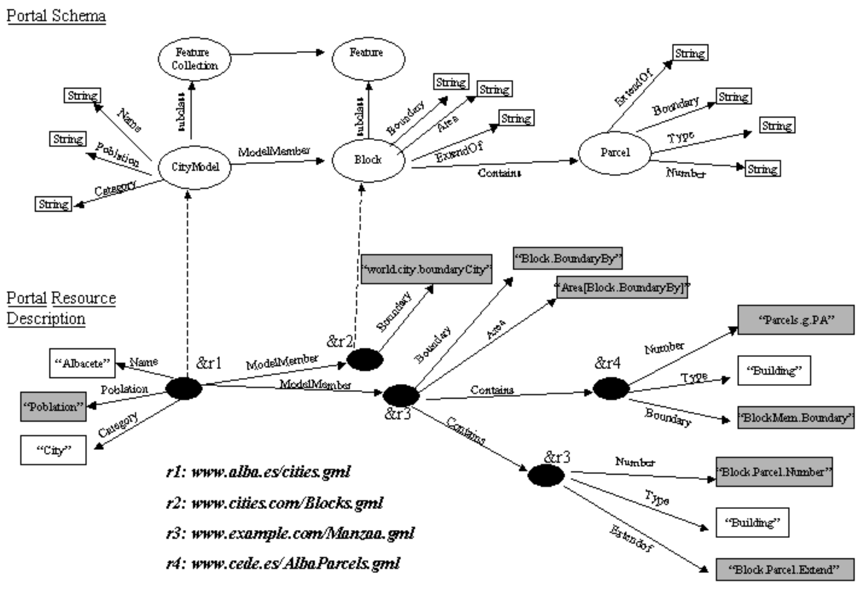


Fig. 1. Portion of Catalog of a CityModel.

The most important feature of this approach is that there are two kinds of properties: (1) properties that describe the data offered by the resources (*descriptions)*, represented by a rectangle filled with white; (2) properties that represent the name of the attributes of a schema (DTD) that have the same semantic, represented by a rectangle filled with gray.

The first set of properties such as *Name* and *Category* in *CityModel* or *Type* in *Parcel* make the *Catalog*. It allows a spatial query to be refined before it is posted to the sources. For example, if we wanted to execute a query about parcels in "Albacete", it would not be necessary to execute the query in all the resources with information about parcels, but only in those resources related to the *CityModel* called "Albacete". At the beginning, users filter resources in accordance with their description in order to get the candidate resources, and then users can apply a spatial query over these alone.

Evidently, a large *Catalog* allows better filtering of a query, but it leads to difficult management. Owing to this, the size of the *Catalog* should be defined by consensus for each problem.

With regard to the second set of properties, in our prototype the attributes are referenced by beginning with the root of the XML documents. For instance, the resource &r2 (*Block*) located in *www.cities.com/Blocks.gml* represents in a DTD the "*Boundary"* property with the attribute "*world.city.BoundaryCity*", and the same property is represented by &r3 with "*Block.BoundaryBy*". Note that *dot notation* is used to describe the attributes in the spatial XML data model. It is the same syntax used in our query language mentioned above. Thanks to this, a translation between notations (query and mapping) is not necessary. In addition, *wildcards* and other features related to semi-structures can be used to define attributes [16].

On the other hand, spatial operators defined with the syntax of our query language can also be used to enrich the meaning of a property in a resource. Thus, for example, the property *Area* in the concept *Parcel* is defined in &r3 as a function "*AREA[Block.BoundaryBy]*". In this case, &r3 does not have an attribute to describe the property *Area*, but it is possible to obtain data with the same meaning applying an operator *Area*[] over *Block.BoundaryBy*. In this way, operators like *Length, Area, Buffer, ConvexHull, Union, Intersection*, etc. can be used. This feature can only be used when it is applied over attributes in the same resource.

With the Portal Schema and the Portal Resources Description we have all the information necessary to know what resources there are and how to satisfy a query.

3. A Performance Study

In this section, we describe the experimental studies that were conducted in order to assess the efficiency of querying a GML document with our approach. We focus on the effectiveness of this approach in terms of translation of queries over an ontology into queries supported by the wrappers, translation of these queries into spatial SQL queries, query processing and generation of GML documents from the results. All the experiments were conducted on an 2300Mhz PC with 512Mb RAM, 40Gb hard disk with operating system Windows XP professional. The RDBMS used was *Oracle9i Spatial 9.0.1*, which we selected because it allows the storage of spatial objects and the application of spatial operators over them. Oracle's Spatial object-relational model was used (SDO_GEOMETRY object) [21] for simplicity. However, as mentioned above, Oracle's relational model or another RDBMS may be used. All experiments were conducted on the same computer, using different databases to the mediator and the wrappers in the same Oracle DBMS. In this way, we obtained a greater system load.

We carried out our experiments using our own benchmark and data sets. We used the features needed by a benchmark for XML , detailed by [22]. We did not use other existing benchmarks for XML documents such as [23] because these benchmarks are oriented to work with XML documents (only alphanumeric data) and the retrieval and application of spatial objects was not contemplated. In the same way, we did not use the data sets (DBLP Bibliography, Bosak Shakespeare collection, etc.) used in the majority of related works because they only have alphanumeric data. On the other hand, we did not use our benchmark design in [17] because these experiments had different aims. In addition, some features of our query language included in the benchmarks are not supported in the approach prototype presented in this paper (e.g. group by, having, etc.).

Nevertheless, the data set used in these experiments respects the data set used in [17]. This data set represents a *City model* where a city has several blocks, each block has several parcels and each parcel has an owner. We used a data set with 7.1 Mb approx. and 5000 rectangular parcels. In the mediator, we used three data sets: D1

(2.1Mb approx.) with 1250 resources, D2 (4.4Mb) with 2500 resources and D3 (8.4Mb approx.) with 5000 resources.

For this test, on the DBMS the same relations shown in [17] were created. We used the simplest mapping proposed by [18]. The relations stored the data model shown in Figure 4. Indexes on relational tables were also properly built to improve query processing as follows: the attributes: *TState(state_id), TCitymember(Citymember_id), TBlock(block_Id), TBlockmember(blockmember_id), TParcel (parcel_id), TArquitet (arquitect_id)* are primary key. *TCitymember(parent_state_id), TBlock(parent_Citymember_id), TBlockmember(parent_block_id), TParcel (parent_blockmember_id), TArquitet (Parent_parece_id)* are foreign key. The Spatial Attributes *TState(BoundaryBy), TBlock(BoundaryBy,Extendof)* and *TParcel (extendof)* have been indexed with *R-Tree* [21]. In order to carry out this mapping efficiently the RDF is stored in a RDBMS following [19].

3.1. Elapsed time

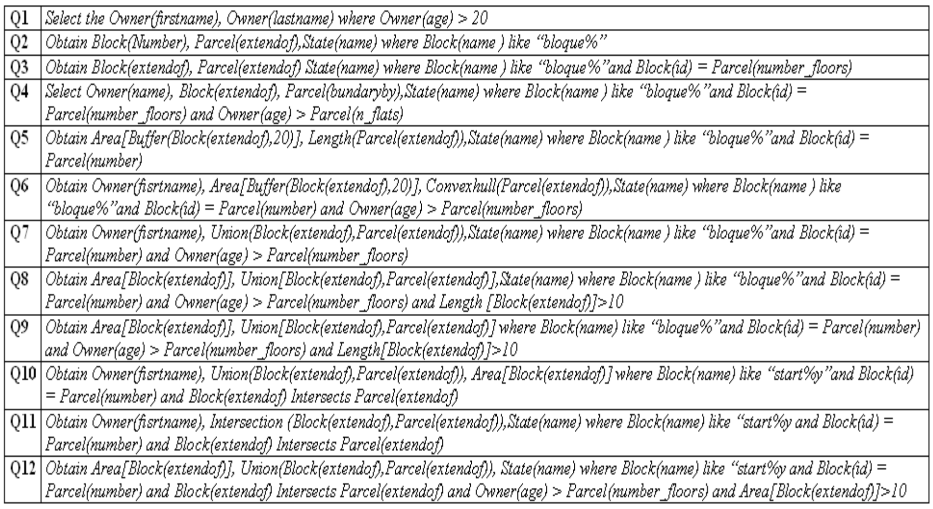
In our study, we have focused our attention on the following aspects involved in assessing performance:

1. Translating queries over an ontology into queries supported by the wrappers has an elapsed time that should be studied [15].
2. Translating queries from our data model to the relational model (spatial SQL) needs a mapping process. It depends on the path length between the entities in the GML document [17].
3. Since we are dealing with an XML query language, a very large number of joins is necessary. It is one of the more important performance problems in the relational implementation of these query languages [24]. In addition, storing and querying spatial data requires a larger amount of resources than storing and querying alphanumeric data [25]. This elapsed time was studied in [17] in isolation from a real system, but now we study how it works inside our system.

For these aspects we have studied the following elapsed times: T1 is associated to the translation of queries over an ontology into queries supported by the wrappers (parser and mapping). T2 is associated to translation of these queries into spatial SQL queries (parser, mapping, generation of *from* clause and translation to SQL). T3 is the elapsed time for the execution of SQL sentences.

Firstly, a set of queries is used to study the behavior of these approaches, involving only alphanumeric data (users may only wish to query alphanumeric data of a GML document). Secondly, a set of queries with spatial and alphanumeric operators is used. The queries are listed in Table 1: We define the complexity of a query according to the number of properties included in the query, the number of relations involved in the final spatial SQL and the number of joins in SQL. Table 2 shows the number of joins and relations involved in the final SQL query. All queries return 2500 objects approx. We have used the data set D2 in the mediator. In the execution of the query only one resource satisfies the query.

Table 1. Queries studied



Q1	*Select the Owner(firstname), Owner(lastname) where Owner(age) > 20*
Q2	*Obtain Block(Number), Parcel(extendof),State(name) where Block(name) like "bloque%"*
Q3	*Obtain Block(extendof), Parcel(extendof) State(name) where Block(name) like "bloque%"and Block(id) = Parcel(number_floors)*
Q4	*Select Owner(name), Block(extendof), Parcel(bundaryby),State(name) where Block(name) like "bloque%"and Block(id) = Parcel(number_floors) and Owner(age) > Parcel(n_flats)*
Q5	*Obtain Area[Buffer(Block(extendof),20)], Length(Parcel(extendof)),State(name) where Block(name) like "bloque%"and Block(id) = Parcel(number)*
Q6	*Obtain Owner(fisrtname), Area[Buffer(Block(extendof),20)], Convexhull(Parcel(extendof)),State(name) where Block(name) like "bloque%"and Block(id) = Parcel(number) and Owner(age) > Parcel(number_floors)*
Q7	*Obtain Owner(fisrtname), Union(Block(extendof),Parcel(extendof)),State(name) where Block(name) like "bloque%"and Block(id) = Parcel(number) and Owner(age) > Parcel(number_floors)*
Q8	*Obtain Area[Block(extendof)], Union[Block(extendof),Parcel(extendof)],State(name) where Block(name) like "bloque%"and Block(id) = Parcel(number) and Owner(age) > Parcel(number_floors) and Length [Block(extendof)]>10*
Q9	*Obtain Area[Block(extendof)], Union[Block(extendof),Parcel(extendof)] where Block(name) like "bloque%"and Block(id) = Parcel(number) and Owner(age) > Parcel(number_floors) and Length[Block(extendof)]>10*
Q10	*Obtain Owner(fisrtname), Union(Block(extendof),Parcel(extendof)), Area[Block(extendof)] where Block(name) like "start%y"and Block(id) = Parcel(number) and Block(extendof) Intersects Parcel(extendof)*
Q11	*Obtain Owner(fisrtname), Intersection (Block(extendof),Parcel(extendof)),State(name) where Block(name) like "start%y and Block(id) = Parcel(number) and Block(extendof) Intersects Parcel(extendof)*
Q12	*Obtain Area[Block(extendof)], Union(Block(extendof),Parcel(extendof)), State(name) where Block(name) like "start%y and Block(id) = Parcel(number) and Block(extendof) Intersects Parcel(extendof) and Owner(age) > Parcel(number_floors) and Area[Block(extendof)]>10*

Table 2. Number of joins and relations in each query

Query	Nº Joins	Relations		Query	Nº Joins	Relations
Q1	*0*	1		**Q7**	*5+1+1*	*6*
Q2	*4*	5		**Q8**	*5+1+1*	*6*
Q3	*4+1*	5		**Q9**	*3+1+1*	*4*
Q4	*5+1+1*	6		**Q10**	*3+1+1s*	*4*
Q5	*2+1*	3		**Q11**	*5+1+1s*	*6*
Q6	*5+1+1*	6		**Q12**	*5+1+1s+1*	*6*

The elapsed query times for these queries are shown in Figure 4. Q1-Q4 are queries with only alphanumeric operators. For Q1-Q2 additional joins are not included. They have a progressive number of joins and relations. Q3-Q4 are queries with more properties in *Select* and *Where* clause and 1 additional *joins* in Q3 and 1 additional *joins* and 1 *none equi-joins* in Q4.

Q5-Q12 are queries with spatial operators, some of which have spatial and alphanumeric operators. Q5-Q6 are spatial queries with spatial operators (*Area, Length* and *Buffer*). The number of joins and relations is doubled in Q6.

The elapsed query times for these queries are shown in Figure 2. The elapsed time for the first mapping (mapping mediator - T1) depends on the number of resources and the number of properties involved in each query. However, the results are not very sensitive to the increment in the number of properties. The second mapping (mapping wrapper – T2) has a longer elapsed time than the first mapping. The most expensive function in this mapping is looking for all relations involved in the query and the relations between them to construct the from *clause*. However, the depth of our data model limits this elapsed time. Note that Q4 and Q6 have the highest elapsed time because they have the highest number of properties and involve entities that are highly-nested in the schema. The time of Execution (T3) of the SQL statement depends on the number of joins and the number of spatial operators involved. Note that this elapsed time increases with several joins and spatial operators. However, as is mentioned above, the number of joins can be limited by applying more specific heuristics [26].

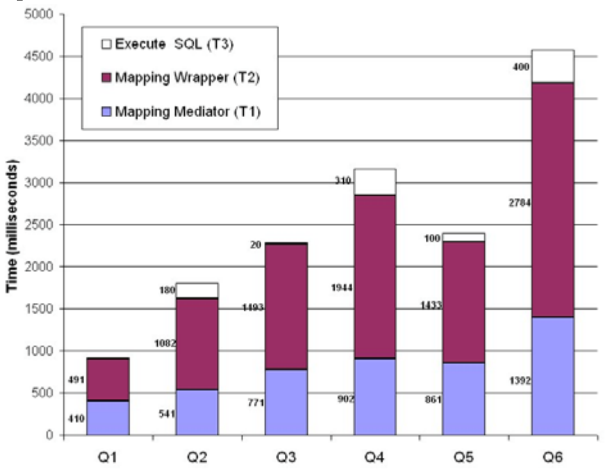


Fig. 2: Elapsed query time (Q1-Q6)

Q7-Q12 use spatial operators which are more expensive in time than Q5-Q6. Q7 uses a spatial *Union* and has a high number of joins. Q8-Q9 use spatial operators in *select* clause (*Union, Area*) and comparison with spatial operators (*Length*) in *where* clause. Q10-Q12 are the most complex queries with a high number of joins and spatial joins. Q12 has *Union* and comparison with spatial operators (Area) in *where* clause for the spatial joins.

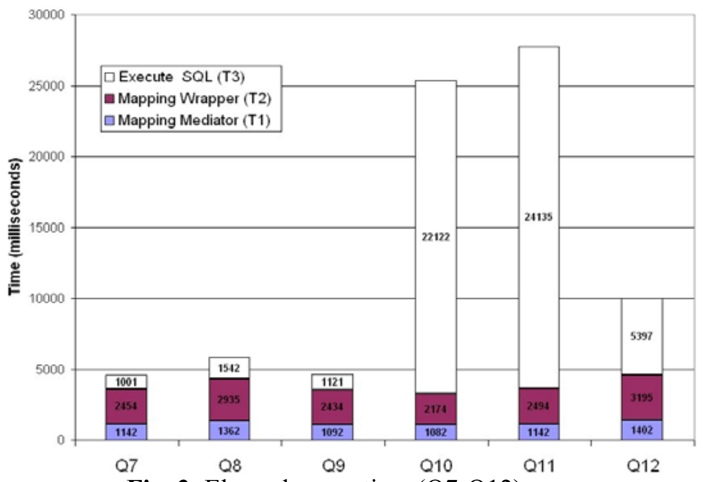


Fig. 3: Elapsed query time (Q7-Q12)

The elapsed query times are shown in Figure 3. Note that when there are complex spatial operators with a higher number of joins, execution time increases. Q7, Q8 and Q9 include, in the *select* clause, *Union* operators, combined with *Area* operator in Q8 and Q9. The elapsed time of these queries is greater than the previous set. Q10-Q12 use spatial joins (*Intersect*). Evidently, this kind of queries gets the worst elapsed time. In addition, the elapsed time to generate GML documents increases when spatial joins are involved. This is due to the process used to adapt the geometries mentioned above.

In view of this, with complex spatial queries, applying a specific study (following on from [18]) for storing GML documents over relational databases is necessary. In order to obtain a good performance querying spatial XML documents, it is preferable to increase the size of relations, and to reduce the number of joins between relations. For each source a particular heuristic must be applied, depending on the kind of information stored on them. In this way, we can reduce the number of joins in each query. This will reduce the elapsed time to execute spatial SQL and the elapsed time to get the *from* clause in the wrapper mapping. In any case, T2 plus T3 is much lower than the elapsed time for querying directly over GML [17].

4. Conclusions

In this paper a prototype of a mediation system for querying XML spatial resources is studied. The main feature of this approach is the possibility of applying spatial operators over the resources that are represented in GML format. We studied three elapsed times: regarding the elapsed time for the mapping in the mediator, we concluded that it depends on the number of resources involved in the query. For the same number of resources, the number of properties is not determinative. The elapsed time for the mapping in the wrapper is more expensive. Moreover, it is more variable because it depends on the heuristic used to represent a GML data model over the relational model in each resource. An heuristic with several relations offers a worse elapsed time in the mapping . The same problem can be applied over the execution of the query. The elapsed time for execution obtains better results with fewer relations.

Future work foresees we will extend this study with a scalability study. Furthermore, we will included more elapsed time (e.g. generating GML documents from the results). In addition, we are developing a prototype to integrate GML resources and any other kind of resource (HTML, PDF, etc). Thus, with this approach it is possible to discover spatial and non-spatial resources which are interrelated semantically on the Web. The study presented in this paper will be used to check the efficiency of this future approach.

Acknowledgment

Research Project: “Metodologías de desarrollo de interfaces de usuario dinámicas” – (JCCM PBC-03-003). Junta de Comunidades de Castilla-La Mancha.(Spain)

References

[1] J. Berners-Lee, J. Hendler and O. Lassila. The Semantic Web, Scientific American, vol 184, no. 5, pp. 34-43, 2001.

[2] J. Egenhofer. Toward the Semantic Geospatial Web. ACM-GIS 2002. 10th ACM International Symposium on Advances in Geographic Information Systems. McLean (USA). 2002.

[3] OpenGis Consortium. Specifications. http://www.opengis.org/techno/specs.htm.1999

[4] H. Samet. Applications of spatial data structures. Computer Graphics, Image processing and GIS. Addison – Wesley. 1990.

[5] J. Córcoles and P. González. Querying Spatial Resources. An Approach to the Semantic Geospatial Web. CAiSE'03 workshop "Web Services, e-Business, and the Semantic Web (WES): Foundations, Models, Architecture, Engineering and Applications". To Appear in Lecture Notes in Computer Science (LNCS) *by Springer-Verlag*. 2003.

[6] OpenGIS. Geography Markup Language (GML) v3.0. http://www.opengis.org/techno/documents/02-023r4.pdf. 2003

[7] Y. Papakonstantinou, H. García-Molina and J. Widom. Object Exchange Across Heterogeneous Information Sources. In Proc. ICDE Conf. TSIMMIS project: http://www-db.standford.edu/tsimmis. 1995.

[8] D. Christophides, S. Cluet and J. Siméon. On Wrapping Query Languages and Efficient XML integration. In Proc. Of ACM SIGMOD Int. Conference on Management of Data Dallas (USA). 2000.

[9] A. Y. Levy, A. Rajaraman and J. Ordille. Querying Heterogeneous Information Sources Using Source Description. In Proc. of the Int. Conference on Very Large Databases, pp.25-262. India. 1996

[10] B. Amann, I. Fundulaki and M. Scholl, C. Beeri, A-M. Vercoustre. Mapping XML Fragments to Community Web Ontologies. In Proc. Fourth International Workshop on the Web and Databases. 2001

[11] B. Amann, C. Beeri, I. Fundulaki, and M. Scholl. Ontology-Based Integration of XML Web Resources. In *International Semantic Web Conference (ISWC)*, Sardinia, Italy, 2002.

[12] O. Boucelma, M. Essid and Z. Lacroix. A WFS-Based Mediation System for GIS Interoperability. ACM-GIS 2002. 10th ACM International Symposium on Advances in Geographic Information Systems. McLean (USA). 2002

[13] A. Gupta, R. Marciano, I. Zaslavsky and C. Baru. Integrating GIS and Imagenery through XML based information Mediation. Integrated Spatial Databases: DigitalImages and GIS. Lecture Notesin Computer Science. Vol1737. Pp. 211-234. Springer-Verlag. 1999

[14] J. Córcoles, P. González and V. López-Jaquero. Integration of Spatial XML Documents with RDF. International Conference on Web Engineering. Spain. To Appear in Lecture Notes in Computer Science (LNCS) *by Springer-Verlag*. 2003.

[15] A.Y. Levy. Logic-Based Techniques in Data Integration. In Jack Minker, editor, Logic Based Artificial Intelligence, pages 575-595. Kluwer, 2000.

[16] J. Córcoles and P. González. A Specification of a Spatial Query Language over GML. ACM-GIS 2001. 9th ACM International Symposium on Advances in Geographic Information Systems. Atlanta (USA). 2001

[17] J. Córcoles and P. González. Analysis of Different Approaches for Storing GML Documents ACM-GIS 2002. 10th ACM International Symposium on Advances in Geographic Information Systems. McLean. (USA) 2002

[18] P. Bohannon, J. Freire, P. Roy and J. Simeon. From XML Schema to Relations: A Cost-Based Approach to XML Storage. 18th International Conference on Data Engineering (ICDE2002). 2002.

[19] S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis, K. Tolle. "*The ICSFORTH RDFSuite: Managing Voluminous RDF Description Bases*". In Proceedings of the 2nd International Workshop on the Semantic Web (SemWeb'01), in conjunction with WWW10, pp. 1-13, Hong Kong., 2001.

[20] Karvounarakis, G., Christophides, V., Plexousakis, D., and Alexaki, S. Querying community web portals. Technical report, Institute of Computer Science, FORTH,Heraklion, Greece. 2000 http://www.ics.forth.gr/proj/isst/RDF/ RQL/rql.pdf.

[21] Oracle9i Database Documentation. http://otn.oracle.com:80/docs/products/oracle9i/content.html. 2002.

[22] A.R. Schmdit, F. Waas, M. Kerste, D. Florescu, M. Carey, I. Manolescu and R. Busse. Why and How to Benchmark XML Databases. SIGMOD Record. nº3 vol. 30. 2001.

[23] A.R. Schmdit, F. Waas, M. Kerste, D. Florescu, I. Manolescu, M. Carey and R. Busse. The XML Benchmark Project. Technical Reports INS-R0103. 2001.

[24] S. Abiteboul, P. Buneman and D. Suciu. Data on the Web. From Relations to Semistructured Data and XML. Morgan Kaufmann Publishers. 2000.

[25] P. Rigaux, M. Scholl and A. Voisard. Spatial Databases with Application to GIS. Morgan Kaufmann Publishers. 2002.

[26] J. Shanmugasundaram, K. Tufte, G. He, C. Zhang, D. DeWitt, and J.Naughton. Relational Databases for Querying XML Documents: Limitations and Opportunities. In Proc. of the Int'l. Conf. On Very Large Data Bases, pages 3002-314. 1999.