MEDIATED GEOGRAPHIC WEB FEATURE SERVICES

Mehdi Essid and Omar Boucelma
LSIS-CNRS, Université Paul Cézanne Aix-Marseille III
Avenue Escadrille Normandie-Niemen
F-13397 Marseille Cedex 20
{<first>.<last>}@lsis.org

Abstract With the proliferation of Geographic Information Systems (GIS) and spatial resources over the Internet, there is an increasing demand for robust geospatial information services that allow federation/interoperation of massive repositories of heterogeneous spatial data and metadata.

However, interoperating GIS poses several challenges. First, there is no accepted standard for spatial or geographic data representation. Second, each GIS provides its own proprietary format as well as its specific query language; while geographic resources are designed for a variety of different purposes. Finally, orthogonal directions in the design of geographic resources may affect the semantics of the data they contain and impair their integration.

The purpose of this paper is to show how *mediation* - a data integration technique - can help in building a Web-based geospatial service. This technique has been fully implemented in the context of a geographic mediation/wrapper system that provides an integrated view of the data together with a spatial query language. As a proof of concept, we deployed the service in building a prototype for an interoperability application involving several catalogues of satellite images.

Keywords: Mediation, Web, GIS, GML, WFS

1. Introduction

The proliferation of spatial data on the Internet is beginning to allow a much larger audience to share data currently available in various Geographic Information Systems (GIS). As spatial data increases in importance, many public and private organizations need to disseminate and have access to the latest data at a minimum (right) cost and as fast as possible. In order to move to a real Web-based spatial data system, we need to provide flexible and powerful GIS data integration solutions. Indeed, GIS are highly heterogeneous: not only they differ by their data representation, but they also offer radically different query languages. The two main problems resulting from the data integration are the

data modeling (how to integrate different source schemas) and their querying (how to answer correctly to the queries posed on the global schema).

The purpose of this paper is to show how *mediation* - a data integration technique - can help in building a Web-based geospatial service. This technique has been implemented in the VirGIS system [Boucelma et al., 2004] which uses Geography Markup Language (GML) [Cox et al., 2001] for the encoding and the transport of geographic information, and the WFS interfaces [OpenGIS, 2002] to access GIS features.

2. Problem Statement

Interoperating geographic information systems is an important problem that raises many issues. Surveying the research that has been conducted in this area is definitely out of the scope of this paper. Conferences series such [Vckovski et al., 1999] contains a comprehensive list of issues, while a paper such as [Devogele et al., 1998] provides a clear understanding of the (geographic) schema integration problem.

The work described in this paper is directly related to data mediation and its applicability to the GIS world: and this is where our contribution fits. Mediation systems provide users with a uniform interface to access different data sources via a common global (mediated) schema. The main issues of the schema integration problem are as follows : sources heterogeneity, global schema modeling and definition, definition and management of mapping rules that express the correspondence between the global schema and the local source ones, source semantics and schema evolution.

The global schema definition, that provides an uniform view of the different resources, can be done using two different approaches. The first, *Global As View* (GAV), consists in defining the global schema as a set of views made on the local sources. This approach is used, for example, in the TSIMMIS [Garcia-Molina et al., 1997] integration systems. The second, *Local As View* (LAV), used in particular in the projects Information Manifold [Kirk et al., 1995] and Picsel [Lattes and Rousset, 2000], consists in defining the local sources as a set of views made on the global schema.

To compensate the insufficiency of the two approaches, other approaches come into existence like *Global Local As View* (GLAV) [Friedman et al., 1999] or *Both As View*(BAV) [Boyd et al., 2004]. The GLAV approach combines the expressive power of both LAV and GAV, allowing flexible schema definitions independent of the particular details of the sources. The BAV approach is based on the use of reversible schema transformation sequences.

In mediation systems, the result of a user query is impacted by the rewriting process. Several approaches have been followed in different mediation systems like TSIMMIS, PICSEL [Lattes and Rousset, 2000], Xyleme [Aguilera et al.,

2001] or MOMIS [Lattes and Rousset, 2000]. In the VirGIS mediation system, we are concerned about completeness, i.e., we explore the real sources in order to return a maximum number of tuples and to look for the missing properties in some sources.

As in any mediation system, a mapping between real data sources and the mediation schema is needed. One of the main issues faced when integrating geographic domain data is ontological heterogeneity. As a matter of fact, each geographic data source is an abstraction of the real world according to a particular point of view and purpose. A data source may represent regular topographic maps (IGN, Ordinance Survey, etc.) or a statistical survey (national Census), or even a satellite image (weather forecast or land-cover oriented). For each dataset, the underlying motivation leads to a particular representation of geographic objects. As a result, depending on the source point view, its scale, its producer and its final user, the data source has its own way of classifying geographic objects. In the sequel , we do not care about the mapping issues between the real data sources and the mediation schema, then we chose a simple example with (simple) one-to-one mappings.

As an example, consider the global (mediation) relation

satellite(ID, Name, SatID, Elevation, Date, Geom, Url)

describing a catalogue of satellite images. A user may pose a query against the satellite relation schema, asking for a satellite image (stored at Url address) that cover a location described by Geom (coordinates or bounding box), the image (shot) being taken by a satellite named Name and whose id is SatID, at a given Date and a given sun Elevation.

Relation satellite resulted from the integration of three relations stored in three different data sources as illustrated in Figure 1. The three relations are as follows:

- the SPOT database, stored in the geographic shape format, that contains images taken by French SPOT satellite, and whose relation is

  SPOT(key, Satellite, Sat_ID, Sun_elev, Date_, The_Geom).

- the ikonos database stored in PostGIS format, that supplies IKONOS images and whose relation is

  ikonos(key, Satellite, Sat_ID, Sun_el, Date_acqui, Geometry),

- the preview database stored in PostGRES, that contains images, and whose relation is *preview(key, filename)*.

As illustrated in Figure 1, arrows indicate the schematic mappings between the three (local) schemas and the global (mediated) one.

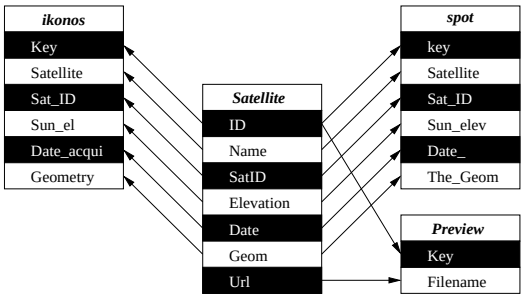

Figure 1. Sources and Mediation Schema Mappings

3. Processing User's Queries

In this section, we first give an overview of the VirGIS integration system and the query languages being used. Then we describe the internals of a user's query processing.

3.1 Mediation System Overview

Figure 3 illustrates the current system architecture, that complies to the approach described in this paper. VirGIS is in charge of processing (geographic) users' queries that are expressed either in GQuery [Boucelma and Colonna, 2004] –a query language we purposely developed, or in XML/KVP (Keyword-Value-Pair) according to the OpenGIS WFS specification. Sources are published on the VirGIS portal via WFS servers. An *admin interface* allows the addition, modification or deletion of sources located by their WFS addresses. Local schemas and capabilities are extracted automatically using WFS operators such as (*GetCapabilities* and *DescribeFeatureType*). The *satellite global schema* is described as an *XML Schema Description* [W3C, a].

The (global) user query is written in terms of the global schema. As this query may include several entities of the global schema, the first step consists in decomposing it into a set of sub-queries dealing with one entity at a time. We call this kind of sub-queries *elementary* queries. The *decomposition* module is in charge of transforming a query into a set of elementary queries, which leads to a a first execution plan called *global execution plan*. Note that the global execution plan is composed of a set of elementary queries expressed in terms of the global schema.

Now, the goal is to produce an execution plan for each elementary query. This is done by the *Rewriting* module in using a set of *mapping rules* and sources' capabilities. The process is repeated for each elementary query and

results in an *elementary execution plan* that corresponds to the set of candidate queries that need to be executed to answer an elementary query. Finally the *final execution plan* (the one that corresponds to the global user query) is obtained by replacing each elementary query of the global execution plan by its elementary execution plan. Figure 2 illustrates the way the final execution plan is calculated. In this figure EQ_1 stands for elementary query1 and EEP_1 stands for elementary execution plan1.

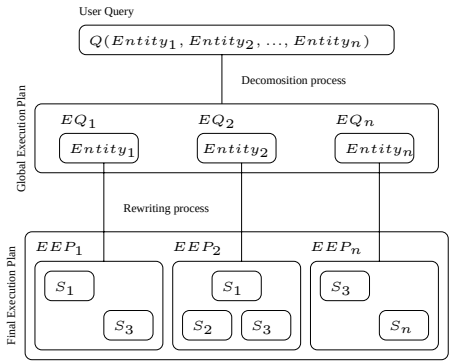

Figure 2. Query Processing

The final execution plan contains three types of queries:

1 HTTP queries are WFS queries in charge of extracting a feature from a local source.

2 GQuery queries are (mainly) used to express spatial operations, and are processed by the GQuery Engine.

3 JOIN queries play a central role in the execution plan. Since we are handling a large amount of data, and because execution of joins in current XQuery implementations is prohibitive, we are using a specific SortJoin algorithm. The WFS join Engine module is responsible for processing this kind of queries.

The final execution plan is then processed by the *execution* module which is in charge of sending subqueries to the appropriate engine according to their type. This module uses a cache repository to save temporary results generated by the different queries. When the execution is finished, the *recomposition* module processes the result in removing duplicates (if any), before returning a GML stream as an answer, the stream can be either displayed or a passed to another program.

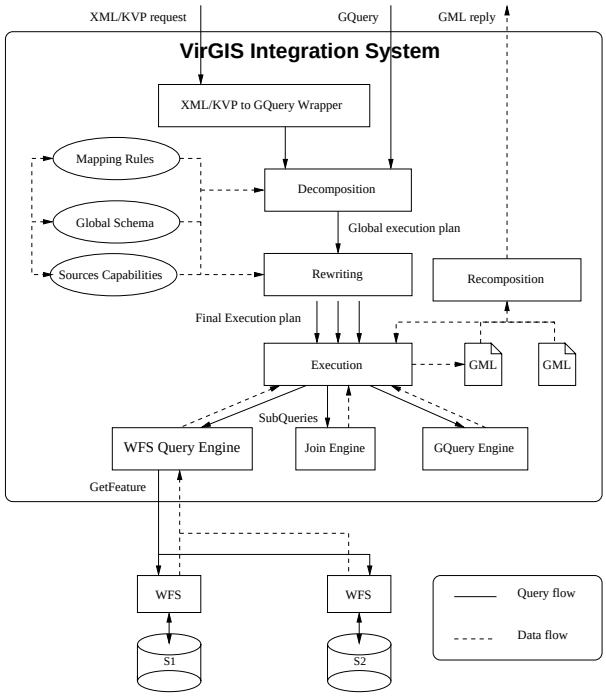

Figure 3. The VirGIS Architecture

3.2 Query Language

We are using GQuery [Boucelma and Colonna, 2004], a query language based on XQuery [W3C, b] that allows users to perform both attribute and spatial queries against GML [Cox et al., 2001] documents. Spatial semantics is extracted and interpreted from XML tags without affecting XQuery syntax, i.e. a spatial query is a pure FLWR (Flower) expression, hence looks like a regular XQuery expression. GQuery may be used either as a stand-alone program, or as a query language in the context of an integration system, to query several heterogeneous GIS data sources disseminated over the Web.

3.3 Obtaining a Global Execution Plan

The Global execution plan is the result of the decomposition of the user query into a set of elementary sub-queries expressed in terms of the global schema. Data sources are queried via WFS which provides access to any geographic repository that supports WFS interface. However, WFS does not handle complex queries, neither it performs geographic data integration. Hence,

execution plans result from a combination (join, union and operation) of elementary queries, each one being posed against a single feature.

Figure 4 returns all SPOT pictures taken on January 1^{th} 2004, that overlap IKONOS images satellite images captured during the same period, for a location having coordinates (8,43 10,41).

```
let $z:=<polygon><coordinates> 8,43 10,41
       </coordinates></polygon>
for $x in document(satellite),
    $y in document(satellite)
where WITHIN($x, $z) = TRUE
      and $x/Date/text() = '01/01/2004'
      and $x/Name/text() = 'SPOT'
      and $y/Date/text() = '01/01/2004'
      and WITHIN($y, $z) = TRUE
      and $y/Name/text() = 'IKONOS'
      and overlap($x/Geom, $y/Geom) = true
return $x
```

Figure 4. Query Q_0

In our approach, features are not retrieved out according to their names but rather according to their references. For example, in Q_0 we have two features (one is referenced by x, the other by y). The decomposition process uses this hypothesis to decompose query into a set of elementary queries.
According to these hypothesis, we can distinguish two features and seven conditions in query Q_0. We have two kinds of conditions:

1 Simple conditions involve properties of the same feature, e.g., *$x/Date/text() = '01/01/2004'*.

2 Complex conditions involve properties from two or more features, e.g., *overlap($x/Geom, $y/Geom) = true*.

As a result, Q_0 is split into three sub-queries. The first sub-query consists in extracting all SPOT images of the Corsica region captured on January 1^{th} 2004. The second one extracts IKONOS images with the same conditions and the last one consists in joining the first query and the second one with the complex condition *overlap($x/Geom, $y/Geom) = true*. At the end of this process, the elementary queries are compared and common sub-queries are unified. Since the first two queries are similar except the condition about the name of the satellite, they are transformed into three queries as follows: the first one consists in extracting all satellite images of Corsica captured on January

1^{th} 2004, the second one consists in keeping only SPOT data, and the last one consists in keeping IKONOS data. Figure 5 illustrates the global execution plan.

```
<Queries>
 <Query id="query0" priority="0">
   let $y:=<polygon><coordinates> 8,43 10,41
           </coordinates></polygon>
   for $x in document(satellite)
   where WITHIN($x,$y) and $x/Date/text() = '01/01/2004'
   return $x
 </Query>
 -- Code for query1 returns SPOT features --
 -- Code for query2 returns IKONOS features --
 <Query id="query3" priority="2">
   for $x in document(query1),
       $y in document(query2)
   where overlap($x/Geom,$y/Geom) = true
   return $x
 </Query>
</Queries>
```

Figure 5. Global Execution Plan for Q_0

Figure 6 illustrates the decomposition algorithm. Functions used in this algorithm have the following meaning:

- ExtractElementaryQueries: extracts elementary queries from the user query, together with their simple conditions.
- SetQueryID: assigns an identifier to the query.
- AddQuery: adds a query to the execution plan.
- RemoveSimpleConditions: removes all the simple conditions from the user query because they are already treated in the elementary queries.
- ReplaceFeatureByQueryID: replaces the features by the id of their corresponding elementary query.

The execution plan is implemented as an XML document whose elements are query nodes. Each query has an id, a priority number which indicates its execution order (queries having the lowest priority are executed first) and other properties that we will describe below.

Algorithm ***DA***(in Query Q0)
ExecutionPlan GEP = ϕ
ElementaryQueries EQ
EQ = ExtractElementaryQueries(Q0)
for each Q in EQ
SetQueryId(Q)
AddQuery(GEP, Q)
endfor
RemoveSimpleConditions(Q0)
ReplaceFeatureByQueryID(Q0)
AddQuery(GEP, Q0)
return GEP

Figure 6. Decomposition Algorithm

3.4 Obtaining a Final Execution Plan

Once the global execution plan is computed, each subquery (except the last one which is used to compute the final result) should be rewritten in terms of the local schemas. To this end, we adapted the approach described in [Amann et al., 2002]. Indeed, for each query, we start in computing its *binding*. A binding of a basic query is the set of mappings that support (totally or partially) the feature of the query. When a source supports all the properties used in the query (totally) we say that this query has a *full-binding* with this source. The binding of the query allows query rewriting into a set of sub-queries expressed in terms of the local schemas.

Let us consider, for instance, an elementary query Q_e and its binding $\mathcal{B} = \{\mathcal{M}_i\ ,\ i \in 1 \ldots n\}$ where n is the number of sources that support the feature queried by Q_e and $\mathcal{M}_i\ ,\ i \in 1 \ldots n$, are their respective associated mapping. To extract the maximum of information located in local sources we rewrite Q_e with a given mapping of the binding, then we try to look for the unsupported properties (if any) using the other mappings of the binding. To do this we define the notion of *prefix query* and *suffix query*. Given a mapping M_i of a source S_i, A prefix query Q_p is a sub-query of Q_e that consists in extracting only the properties supported by S_i. Note that M_i provides a full-binding for query Q_p. The suffix query Q_s, is the sub-query consisting in extracting the remaining properties (plus the key's attributes).

For each mapping $\mathcal{M}_i \in \mathcal{B}$, the query Q_e is processed as follows:

1 compute Q_p and Q_s of the query Q_e and the mapping $\mathcal{M}_i$.

2 let $\mathcal{B}_s = \{\mathcal{M}_j\ ,\ \mathcal{M}_j \in \mathcal{B}\ ,\ j \neq i\}$

3 let $\mathcal{B}_i = \{M_i\}$

4 the elementary execution plan of query Q_e using the binding $\mathcal{B}_i$ is the join between Q_p and the execution plan of Q_s using the binding $\mathcal{B}_s$.

Since queries responsible for extracting information features are executed by WFS interfaces, they are reformulated, using the mapping rules, into queries that are expressed in terms of local schemas (each feature is replaced by the corresponding feature and each property is replaced by the equivalent property). Let us note that these transformed queries(the ones expressed in terms of the local schemas) are sent to the WFS servers that execute them and send the result expressed in the local schemas. Since the final result must be expressed in the global schema, WFS result are translated to this schema. This is done by an additional GQuery expression.

Let us now discuss constraints of the prefix and the suffix queries. It is easy to see that the prefix query contains only the subset of conditions expressed over its properties. However, the suffix query contains the subset of conditions over properties of the suffix query extended by the set of conditions of the prefix query. This extended set is used to refine the prefix query by adding more conditions. This allows us to minimize the number of extracted tuples, hence minimize the execution time. Finally, when a condition is not supported by a data source, it is added to the GQuery expression which is in charge of translating the result in terms of the global schema.

Figure 7 illustrates the rewriting algorithm which, given an elementary query Q_e and its binding $\mathcal{B}$, computes the execution plan expressed in terms of the local sources .

```
Algorithm RW(in ElementaryQuery Qe, in Binding B)
 ElementaryExecutionPlan tmpEMP, EEP = φ
 for each Mi in B
    if ( fullbinding(Qe, Mi))
   tmpEP = Qe
    else {
        Qp = getPrefixQuery(Mi)
        Qs = getSuffixQuery(Mi)
        Bs = {Mj in B , j > i}
       tmpEP = join(Qp, RW(Qs, Bs))
   }
    endif
   EEP = union(EEP, tmpEP)
 endfor
 return EEP
```

Figure 7. Rewriting Algorithm

Note that, in the rewriting algorithm, $\mathcal{B}_s$ is defined as $\{\mathcal{M}_j \in \mathcal{B}\ ,\ j > i\}$ and not as $\{\mathcal{M}_j \in \mathcal{B}\ ,\ j \neq i\}$ as mentioned in the beginning of this section. We have made this change in order to eliminate duplicated answers. In fact, the subset of mappings $\{\mathcal{M}_k \in \mathcal{B}\ ,\ k < i\}$ has already been treated in previous steps. Thus, it is useless to recompute it.

To compute the final execution plan, we just rewrite the elementary queries of the global execution plan.

As an example, let us process the first elementary query of global execution plan of Q_0. This consists in extracting all satellite images of Corsica captured on January 1^{th} 2004. The catalog illustrated in figure 1 shows that neither SPOT, nor IKONOS properties do map the Url property. Hence, the query is divided into two sub-queries. The first one (prefix query) extracts the Key, Satellite, Sat_ID, Sun_elev, Date_ and The_Geom properties satisfying the condition Date_ = '01/01/2004' and WITHIN($x,$y), from the SPOT data source. The second one (suffix query) extracts the ID (the key) and the Url properties from the preview data source. A join is then performed between the results of the prefix and the suffix queries in order to build the result. The same process is done with the IKONOS and the preview data sources. Finally an union is performed on the two results.

4. Conclusion

With the proliferation of GIS data and resources over the Internet, there exists a huge demand for robust geospatial information services that allow interoperation of massive repositories of heterogeneous data and metadata.

In this paper we described a solution and a system that address effective integration needs expressed by the GIS community, in a Web bases environment. To avoid the pitfalls of a complex query language, we developed a user interface (demonstrated at ICDE'04 [Boucelma et al., 2004]) that allows naive user queries, i.e., there is no need for a non skilled user to invest time in learning GQuery.

The work described in this paper fully complies with Web and GIS consortia: it combines both XQuery technology, with OpenGIS ones such as WFS and GML.

References

[Aguilera et al., 2001] Aguilera, V., Cluet, S., Veltri, P., Vodislav, D., and Wattez, F. (2001). Querying a Web Scale XML Repository. In *SEBD*, pages 105–118.

[Amann et al., 2002] Amann, B., Beeri, C., Fundulaki, I., and Scholl, M. (2002). Querying XML Sources Using an Ontology-Based Mediator. In *On the Move to Meaningful Internet Systems, 2002 - DOA/CoopIS/ODBASE 2002 Confederated International Conferences DOA, CoopIS and ODBASE 2002*, pages 429–448. Springer-Verlag.

[Boucelma and Colonna, 2004] Boucelma, O. and Colonna, F.-M. (2004). GQuery: a Query Language for GML. In *Proc. 24th Urban Data Management Symposium*, Chioggia-Venice, Italy.

[Boucelma et al., 2004] Boucelma, O., Essid, M., Lacroix, Z., Vinel, J., Garinet, J.-Y., and Betari, A. (2004). VirGIS : Mediation for Geographical Information Systems. In *Proc. ICDE 2004, Boston*.

[Boyd et al., 2004] Boyd, M., Lazanitis, C., Kittivoravitkula, S., Brien, P. M., and Rizopoulos, N. (2004). AutoMed: A BAV Data Integration System for Heterogeneous Data Sources. In *Advanced Information Systems Engineering 16th International Conference, CAiSE 2004, Riga, Latvia, June 7-11, 2004, Proceedings*. Springer-Verlag.

[Cox et al., 2001] Cox, S., Cuthbert, A., Lake, R., and Martell, R. (2001). Geography Markup Language (GML) 2.0. `http://www.opengis.net/gml/01-029/GML2.html`.

[Devogele et al., 1998] Devogele, T., Parent, C., and Spaccapietra, S. (1998). On Spatial Database Integration. *International Journal of Geographical Information Science*, 12(4):335–352.

[Friedman et al., 1999] Friedman, M., Levy, A. Y., and Millstein, T. D. (1999). Navigational Plans For Data Integration. In *AAAI/IAAI*, pages 67–73.

[Garcia-Molina et al., 1997] Garcia-Molina, H., Papakonstantinou, Y., Quass, D., Rajaman, A., Sagiv, Y., Ullman, J., Vassalos, V., and Widom, J. (1997). The TSIMMIS Approach to Mediation: Data Models and Languages. *Journal of Intelligent Information Systems*, 8(2):117–132.

[Kirk et al., 1995] Kirk, T., Levy, A. Y., Sagiv, Y., and Srivastava, D. (1995). The Information Manifold. In Knoblock, C. and Levy, A., editors, *Information Gathering from Heterogeneous, Distributed Environments*, Stanford University, Stanford, California.

[Lattes and Rousset, 2000] Lattes, F. G. V. and Rousset, M.-C. (2000). The Use of CARIN Language and Algorithms for Information Integration: The PICSEL System. *International Journal of Cooperative Information Systems*, 9(4):383–401.

[OpenGIS, 2002] OpenGIS (2002). Web Feature Service Implementation Specification. `http://www.opengis.org/docs/02-058.pdf`.

[Vckovski et al., 1999] Vckovski, A., Brassel, K. E., and Schek, H.-J., editors (1999). *Interoperating Geographic Information Systems, Second International Conference, INTEROP '99, Zurich, Switzerland, March 10-12, 1999, Proceedings*, volume 1580 of *Lecture Notes in Computer Science*. Springer.

[W3C, a] W3C. XML Schema Part 1: Structures. `//www.w3.org/TR/2001/REC-xmlschema-1-20010502/`.

[W3C, b] W3C. XQuery 1.0: An XML Query Language. `//www.w3.org/TR/xquery/`.