An XML-based architecture for geo-referenced data integration

LÚCIO ROGÉRIO BOTÊLHO[1], JÚLIA CÉLIA MERCEDES STRAUCH[2], JANO MOREIRA DE SOUZA[1, 3]
[1]COPPE/UFRJ – Systems Engineering and Computer Science Program
Federal University of Rio de Janeiro
P.O. BOX: 68511, ZIP: 21945-970, Rio de Janeiro, RJ, Brazil.
{lucio, jano}@cos.ufrj.br
[2] ENCE /FIBGE – National College of Statistics Sciences, R. André Cavalcanti, 106 s. 501
ZIP: 20231-050, Rio de Janeiro, RJ, Brazil.
juliast@ibge.gov.br
[3]IM/UFRJ – Mathematics Institute, Federal University of Rio de Janeiro
P.O. BOX: 68511, ZIP: 21945-970, Rio de Janeiro, RJ, Brazil.

Abstract. The increasing use of geo-referenced data, that are useful to a large number of users, in a first moment led to the development of patterns for information sharing, sending and exchange among systems and data sources. This solution makes data available in a simple and uniform way, but it presents several semantic conflicts in the data integration process. This paper presents the WISE (**W**eb data **I**ntegration **S**yst**E**m) Architecture and Geo-XQuery language, an XQuery extension, to solve some geo-referenced data integration problems. WISE uses XML as a canonic data model to access several heterogeneous and distributed data over the Web.

1 Introduction

Nowadays, with the globalization paradigm, organizations are distributed on several places, and there is a need to accomplish global operations without information centralization or loss of local systems autonomy. The data sharing tendency among GIS (Geographical Information Systems) has been to substitute the closed environment, with product-oriented solutions, for an open environment characterized by different platforms, applications and databases that have real time interoperability.

These solutions are given several denominations in Database Technology literature: multidatabase (MDB), mediators, federated database and interoperability systems [23]. Their goal is to preserve existing systems, maintaining their autonomies, through a system that possesses means for the access and sharing of the data stored by the local database.

Besides the heterogeneities of operating systems, applications and communication protocols, there are semantic and syntactic conflicts because of the differences among the data models and of the modeling process.

Therefore, researchers have been taking to the development of methodologies to capture the semantics of database schemas, to understand differences among them, consistency requirements and integrity restrictions and to apply the techniques developed to solve the conflicts that exist in a way they can be integrated.

According to Pinto[18], with the appearance of World Wide Web (Web), data integration evolved from a traditional architecture of multiple databases into a new framework capable of manipulating a variety of information in several formats and structures. The Internet created new discussion themes for data integration that are more difficult than the existing ones for multi database systems. First, the number of data sources can be enormous, turning conflict resolution and view integration into a difficult problem. Second, the group of data sources is very dynamic, the inclusion or the exclusion of a data source should be accomplished with the minimum impact on view integration. Third, the data sources have different processing capabilities, varying from complete database management systems to simple file systems. Fourth, data sources can be unstructured or semi-structured, and not supply any information for the view integration. Finally, understanding of the applications semantics requires an appropriate specification of the metadata which are very dependent of the application domain.

This demand also exists in the management sectors of institutions that manipulate environmental data. An example is mentioned in Strauch[23] that shows the need for sharing environmental data and describes the means used for data exchange and the efforts of the competent groups, like Open GIS Consortium [17], to establish a specification. The WISE architecture will contemplate a subset of the previously presented conflicts for the data interoperability.

Nowadays, there are several systems for data publication and integration both for conventional data and for geo-referenced data [1, 4, 11, 19, 25, 27, 31].

We are proposing a system for integration and publication of conventional and geo-referenced data in the Web, called WISE (**W**eb data **I**ntegration **S**yst**E**m), that target semantic data integration using application domain ontology [7, 8] and a group of heuristics. This system is very flexible and could be extended, since it provides a range of existing data formats, so as to reach a great number of users from the environmental area.

WISE is part of the SPeCS project, whose goal is to supply a common, flexible and easy environment for cooperative work, where group members can be spread geographically in several heterogeneous environments and even so they are capable to interact during a decision making process. It is a framework to support collaborative spatial decision making [15, 14]. In the SPeCS project, the following research actions are implemented: Database Integration, Cooperative Work, Workflow Tools and Techniques, Mathematical Modeling for Decision Support, and information access and manipulation on the Internet.

WISE aims to substitute X-Arc, a data publisher system of SPeCS. WISE will be a super set of the X-Arc, and it will accomplish the task of integrating and publishing data.

The paper is divided as follows: section 2 presents a classification of the conflicts more frequently found in the heterogeneous data source integration process; section 3 presents the Common Data Model (CDM); section 4 presents the extension of the query language XQuery, called Geo-XQuery, for geo-referenced data; section 5 describes the WISE architecture and its layers, illustrating the data source catalogue processes and the query process. Finally, in section 6 the final considerations of this work are presented.

2 Conflicts in the integration of geo-referenced data

The task of distributed and heterogeneous data source integration is by nature arduous, since it must treat several data sources with different data models, syntax and semantics and conventions, among other things.

These data source schemas present differences in representation of the same real world concept. These differences, denominated conflicts between schema or semantic heterogeneities, happen because: i) designs model the same perspective of the real world in different ways; ii) the designers use different data models or different builders to model the same aspect of the reality; and iii) the differences in the project specification of databases, considering the employed denominations, the specification of types and the integrity restrictions. To integrate the schema, these heterogeneities need to be understood with depth, in order to detect and to solve them.

Geo-referenced data source integration inherits all semantic and structural conflicts of the conventional, heterogeneous and distributed databases. Besides add to these the current conflicts of the graphic components of the geo-referenced data. These conflicts treat modeling aspects regarding the semantics representation, the similarity between the applications and the space concepts associated with each information community.

To facilitate the identification of these heterogeneities; Strauch[23] classifies the conflicts in four great groups, presented to follow:

2.1 Group I: Spatial context definition conflict

Upon implementation, a database adopts cartographic parameters that support for the database representation on an appropriate cartography. Among these parameters are: Vertical and Horizontal Data, coordinates system, projection system and scale.

These parameters are selected depending of the project needs, of the data acquisition cost, of the intended database quality [13] and of the readiness of the material to be digitalized. They are implicit in each geographical database schema.

2.2 Group II: Semantic conflicts

Semantic conflicts occur when the concepts regarding a same reality are described in different ways, given the databases present semantic divergences in the definition of the schema elements. This conflict type spreads through the data models of the organization, the applications, the procedures and integrity restrictions.

These conflicts cover a wide range, including the denominations employed, the abstractions used, properties that describe the geographic objects and the geometric conflicts:

- **Denomination conflicts** - happen because schemas incorporate denominations for the represented objects. Could be synonyms or homonyms. The synonyms occur when semantically identical entities or properties are denominated differently; and the homonyms occur when semantically different entities and properties share the same denomination;
- **Abstraction conflicts** - happen due to the different abstraction levels in reality concept representation, for instance, an element of some type (object class, attribute, relationship) corresponds to another type in another schema; and
- **Property conflicts** - occur when the properties that describe the objects are not complete, or are implicit in other properties.

2.3 Group III: Structural conflicts

These conflicts occur when two concepts even though they are the same and the database uses the same data model, have different specifications, restrictions and domains, causing common reality to be represented in a different way in the database schemas to be integrated. These conflicts are classified as:

- **Type conflicts** - occur when two schema's semantically equivalent properties present incompatible type definitions;
- **Format conflicts** - occur when the same property uses different formats in the databases to be integrated;
- **Unit conflicts** - occur when the property units are different for the same concept because of the users' operational needs;

- **Domain conflicts** - occur when objects of the real world are described in the same abstraction level and classified according to the same criterion, but the theme property assumes different groups of domain values. This arises from the fact that the values of some properties are not intrinsic to the objects, but depend on the application proposal;
- **Restriction conflicts** - happen when some property possesses a logical restriction to the values that it can assume in a schema;
- **Code conflicts** - code conflicts are used for several reasons, such as to facilitate the access to the information and to optimize space. They are not usually uniform among databases, once they assist users' specific needs. So code use introduces difference of values; and
- **Conflict among keys** - occur when the primary keys in the several schemas are different, or different keys are used for the same concept.

2.4 Group IV: Value conflicts

These conflicts occur in the extensible level of the properties. They can be:

- **Conflicts of default values** - GIS and applications can provide different default values for properties with the same semantic;
- **Value conflicts** - some GIS offer means to automatically calculate geometric values, which can assume different values because of the spatial data quality, of the precision adopted for the coordinates or even of the algorithm used by GIS;
- **Updating conflicts** - occur when the properties of different databases are updated at different times; and
- **Conflicts of register mistakes** - can be typographic mistakes or variations in the measurement processes, or data acquisition mistakes, namely digitizer/digitalize processes.

The WISE architecture will contemplate a subset of the conflicts presented previously.

3 Common data model

Data source integration is comprised of a modeling process, whose goal is schema understanding, identification of heterogeneities, solving incompatibilities and schema overlaps and integrating heterogeneous and distributed resources, through global applications.

The data model is a Database mechanism to abstract and represent a view of reality. In general, local database schemas can employ heterogeneous data models. However, in the schema integration process it is important that the data integration manager understand local schema semantics to find similarities. Therefore, it is necessary to adopt a unique data model, called canonic data model or common data model (CDM) to facilitate this process.

According to Souza[22], the objectives of adopting this model during data integration systems development is to offer a normalized understanding of the meaning of the data stored in each schema, to facilitate schema comparison in data integration process, and to be a common mechanism to perform queries.

Thus, every data source schema in the WISE architecture is converted to CDM, in order to minimize syntactic differences resulting from different data models. This transformation from local schema to CDM is not a simple translation between models. This includes a semantic enrichment process using ontologies to capture implicit semantics in local data sources.

WISE uses XML Schema [29] as its CDM. Other systems [30, 10] can be found in literature, which employ XML in a similar way.

The XML Schema CDM is enriched with domain application semantics using ontologies [7, 8] and heuristics. It is called Semantic XML Schema.

4 The query language

As XML becomes the language chosen to represent and exchange information on the Web, work progresses to standardize query languages, transformation, exchange, and data publication over XML. Among XML query languages, XQuery has been accepted by W3C [26] as a XML standard for query languages [3]. While still evolving, XQuery has been implemented in several database systems, e.g., Tamino, Oracle, etc.

Even though XQuery is a powerful query language projected to query and transform semi-structured and nested XML data, the optimized query evaluation in XQuery remains a research topic. Furthermore, XQuery does not provide proper support to querying spatial elements. It simply treats spatial elements as numerical values, and not every search criteria can be expressed as search predicates in individual spatial elements.

Due to traditional relational databases only supporting simple data types, Open GIS Consortium defined a SQL data model extension [16] to represent and query geometric data. This extension includes a set of spatial data types and spatial predicate functions to simplify queries in spatial fields. Thus, we brought this idea to our query language project, which is an XQuery extension. Our query language, Geo-XQuery, will implement part of these specifications.

4.1 XQuery language syntax

XQuery is a functional language where a query is represented through an expression. Due to space restrictions, below we describe only one XQuery expression among existent ones. We will also be concise when describing Geo-XQuery extensions in the next subsection.

A FLWR expression (Figure 1), pronounced *flower*, is composed by FOR, LET, WHERE and RETURN clauses, which appear in a given order. A FLWR expression assigns one or more values to variables and uses these to build results.

```
FLWRExpr ::= (ForClause | LetClause)+ WhereClause? "return" Expr
ForClause ::= "for" Variable "in" Expr ("," Variable "in" Expr)*
LetClause ::= "let" Variable ":=" Expr ("," Variable ":=" Expr)*
WhereClause ::= "where" Expr
```

Figure 1 – XQuery FLWR expression syntax

The first part of a FLWR expression is composed of a FOR or LET clause that associates values to variables. Values associated to variables are represented by expressions. Differences between these two clauses are illustrated by the following example: A "FOR $x IN /library/book" clause results in several associations, where each one associates a $x variable to a library book. Otherwise, a "LET $x := /library/book" clause results in a simple association where a $x variable has a sequence with all library books.

The next step is to filter results from FOR and LET clauses. This is done using a WHERE clause. This clause can contain several predicates, connected by AND and OR. These predicates generally contain references to variables that were filled through FOR and LET clauses.

Finally, RETURN clauses generate a FLWR expression output, which can be a node sequence or primitive values.

4.2 Geo-XQuery: extending XQuery

To perform queries in spatial elements, which can be as simple as a point or as complex as a set of polygons, the Geo-XQuery syntax provides a set of spatial predicates to facilitate queries involving such elements. This predicates are used in WHERE clauses. Some of this spatial predicates are:

- **equal()** - The predicate returns true when a geometry object has a location identical to an other geometry object, false otherwise;
- **cover()** - The predicate returns true when a geometry object has a location containing an other geometry object, false otherwise;

- **coveredby()** - The predicate returns true when a geometry object has a location completely enclosed by an other geometry object, false otherwise;
- **overlap()** - The predicate returns true when a geometry object has a location overlapping an other geometry object, false otherwise;
- **meet()** - The predicate returns true when a geometry object has a location adjacent to an other geometry object, false otherwise;
- **disjoint()** - The predicate returns true when a geometry object has a location disjoint with an other geometry object, false otherwise;

Sample: The Guandu river can supply water to cities up to 250 Km away. List cities that can use water from Guandu. This query illustrates buffer analysis and spatial join operations (Figure 2).

```
FOR $r IN document("River")//river[Name="Guandu"],
    $c IN document("City")//city
WHERE overlap(buffer($r/Shape,250), $c/Shape) == 1
RETURN <CityName>
            <cname> $c/Name</cname>
       </CityName>
```

Figure 2 – Geo-XQuery sample

5 WISE Architecture

The Web data Integration SystEm architecture (WISE) solves semantic problems that arise during conventional and unconventional (geo-referenced) data integration in the World Wide Web. The main goal is to provide integration and sharing of environmental data among several groups.

Instead of other geo-referenced data integration and publication systems [30, 21], which present query results using raster image files or applets, and generally do not allow manipulation (or allow only restricted manipulation, appropriate to the needs of target users); the WISE architecture intends to make available the integrated data in several file formats allowing the process and manipulation of them by the users (Figure 3).

The WISE architecture is composed of four layers: Client Layer, Data Integration Layer, Wrapper Layer and Data Access Layer as quickly described bellow:

- **Data Access Layer** – It is defined by a set of Data Providers, which are computational resources capable of store one or more Data Sources (heterogeneous or not) and provide at least one to WISE.

  A Data Source is a homogeneous data repository, stored in a Data Provider. Wrappers access Data Sources using a URL (Uniform Resource Locator) and some parameters, which are stored in WISE Catalog.

  Each Data Source belongs to a given Data Source Type, which specifies data as conventional or geo-referenced, and data type (Relational, XML, Object-Relational, Text, etc.).

  A Wrapper created for a given Data Source Type can be used by any Data Source available in WISE to that type. Descriptions of Data Sources Types, as well as wrappers descriptions are stored in WISE Catalog.

- **Wrapper Layer** – Wrappers provide services to access data from Data Sources. Using Wrappers, WISE access data from heterogeneous and distributed data sources in a uniform way. Wrappers are created according to new Data Source Types demand, and establish communication with WISE through ECAPI.

  Generic Wrapper specifies the most generic class from the wrapper class hierarchy and has fundamental features to access data. Specific Wrapper specifies an extended wrapper from Generic Wrapper class, and adds functionalities according to the target Data Source Type. As fundamental features to access data were inherited from Generic Wrapper, and only these features are necessary to access data from Data Integration Kernel, this one gets access to data in a uniform way, independent of the Data Source Type used.

Data Integration Kernel does not care about specific wrappers implementation, because it uses a standard interface to communicate with Wrappers. Thus, when new Wrappers are aggregated to WISE, Data Integration Kernel does not need to change.

- **Data Integration Layer** – Data Integration Kernel and External Communication Application Programming Interface (ECAPI) compose this layer.

  ECAPI is the Application Programming Interface (API) for communication between WISE and external software applications. It is used by client applications and to communicate with Data Sources using Wrappers.

  Data Integration Kernel is the most important part of WISE. It process external queries, get the results from Data Sources, integrate results, convert to predefined data formats and return the results to WISE clients. It also stores Semantic XML Schemas, which are representations of Data Sources schemas, as well as an Ontology Repository to perform semantic data integration.

  This kernel maintains Data Sources related information stored in a Catalog. Catalog stores information from where Data Sources are stored, its owner, Data Sources available parts, access control and other informations. Both Catalog and Semantic XML Schemas are stored in WISE Data Repository. Data Source Cataloger graphical interface sets Catalog information.

  Query Processor (QP) gets external query, analyze and decompose it according to Data Repository schemas distribution; sends sub-queries and receive query results from Wrappers and finally pass results to Data and Map Integrator.

  Integrator is divided in two logical integrators, Data Integrator and Map Integrator. The former integrates conventional data and the later integrates geo-referenced data. These integrators get information from Data Repository, which maintain metadata information about Data Sources registered in WISE.

Information sent to the user is converted by the Format Converter (FC) to the format chosen by the user.

A Semantic XML Schema Generator is capable of capture schema information from Data Sources and generates an Initial XML Schema, where it adds application domain semantics through ontologies and heuristics, thus creating a data source Semantic XML Schema. This is stored in WISE Data Repository, and will be used afterwards in the Semantic XML Schemas integration process. Semantic XML Schemas can be integrated (with human interaction) in other schemas called Semantic XML Schemas Integrated.

- **Client Layer** – is the interface layer between WISE and client applications.

  WISE has a standard graphical query interface called WISE Query Interface (WISEQI), which is the standard client application to WISE Data Users (DU) perform queries. This application is implemented using Java programming language, and thus is available to several platforms. WISEQI has three components:

  - Metamodel Viewer (MV), which reflects the current metadata state of Data Sources registered in WISE. Information in the metamodel is available to users. This allows users to understand semantic, distribution, quality, and other informations about data sources. The data distribution is transparent to the user, in other words, the user do not need to know where the data was stored physically neither the information about the Data Providers, where the Data Sources are stored. MV presents schema information in Semantic XML Schema format.
  - Query Editor (QE) to perform queries in WISE using Geo-XQuery language.
  - Data Publisher (DP) to receive query results from WISE and make information available to users. Query results can be in several formats (XML[28, 2], GML[9], SVG[24], ESRI Shape [20], or some raster image format) chosen by the Data User.

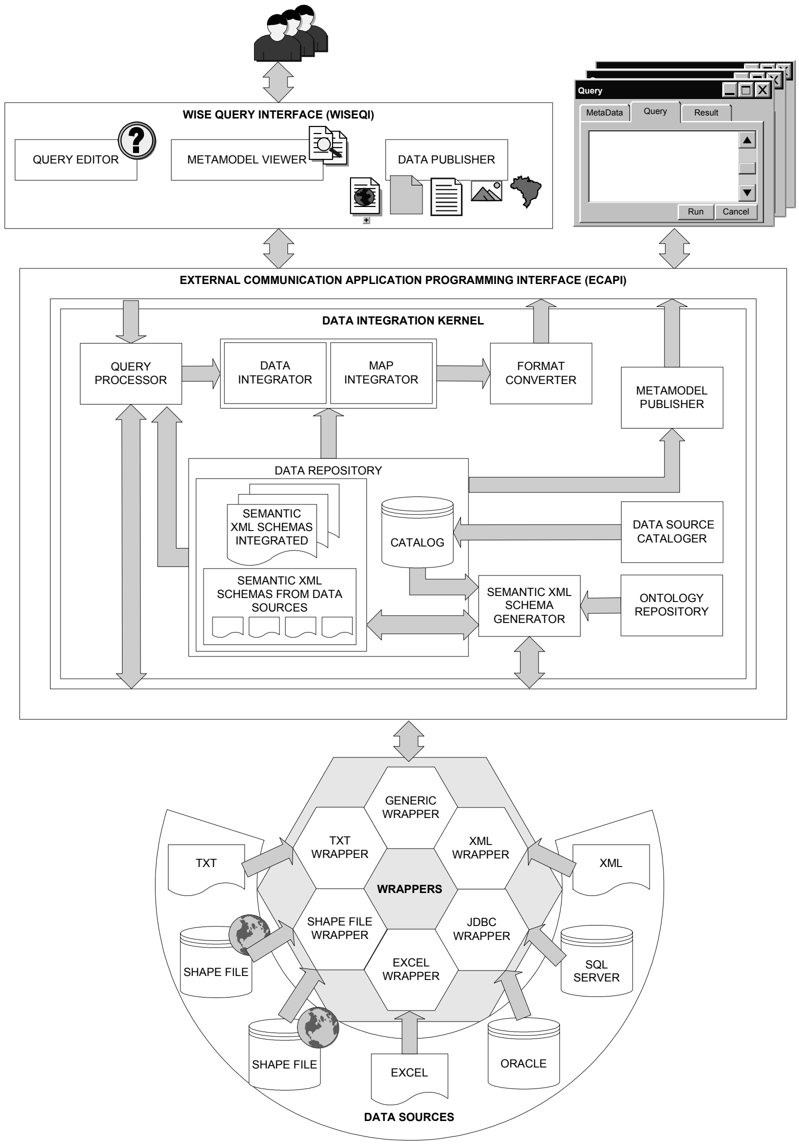

Figure 3 – WISE architecture

This architecture is in development, using a Java platform [12] with CORBA [5] to support communication with external applications, and a FireBird database [6] to store data.

To illustrate WISE proposal architecture, follows is showed a data source cataloguing and query processes.

5.1 WISE data source cataloguing process

This process occurs when a person or institution (Local Data Manager, LDM) wants to register a data source in WISE, and needs Data Integration Manager (DIM) help.

1- The Local Data Manager (LDM) through WISE Data Source Cataloger (DSC), which is the interface to register Data Sources, provides related data source information, such as data source type, location, cataloger, access privileges, etc. This information is stored in a Catalog, for posterior use by other WISE components, such as Semantic XML Schema Generator, and Query Processor.

2- After registering a Data Source, the system verifies if there exists a Specific Wrapper for the Data Source Type cataloged. If such wrapper exists, it will be provided for use. Otherwise, it is necessary to create one (by the LDM or DIM, or both) for the new Data Source Type. Then, any Data Source of his Data Source Type can use the new Specific Wrapper, it becomes available in WISE to other Data Providers.

3- With a wrapper, it is possible to execute the Semantic XML Schema Generator. This one first extracts local Data Source schema using the wrapper, and generates an Initial XML Schema, where it adds application domain semantics through ontologies and heuristics, thus creating a data source Semantic XML Schema. This is stored in WISE Data Repository, and will be used afterwards in the Semantic XML Schemas integration process.

5.2 Query process by Data User through WISEQI

This process describes a query by a person using a standard WISE query interface called WISEQI (Figures 4 and 5).

1- The Data User (DU), through WISEQI – which has three components: Query Editor (QE), Metamodel Viewer (MV) and Data Publisher (DP) – performs a meta-model analysis of the necessary information using MV, and edits his query using the Query Editor, which uses Geo-

XQuery. This editor has two interfaces: the former abstracts Geo-XQuery details because it uses visual components perform data selection, where the user can select what to query; in the other interface, Data User uses a Geo-XQuery textual editor to enter a query. Thus, both naïve and expert user can perform queries in WISE.

Queries written in the Query Editor are passed to the Query Processor (QP) through the External Communication Application Programming Interface (ECAPI).

2- Query Processor receives an external query and analyzes if it is encoded as Integrated Semantic XML Schemas or as individual Semantic XML Schemas, from information acquired in Data Repository about how data are distributed. If the query is an Integrated Semantic XML Schema, then QP divides the original query in subqueries. It finally passes each query processed to the Wrapper responsible by the Data Source informed in each query.

3- Wrappers are capable of receiving a given query and returning a result from data sources associated with them.

Wrappers receive queries in Geo-XQuery and interpret them according to related Data Source Types. After interpreting a query, a wrapper queries the target data source, and this returns a result (or error) to the wrapper that launched the query. Finally, the wrapper returns the results to the Query Processor.

Results returned to the Query Processor can be in XML/GML if it came from a conventional data source, or in ESRI Shape format [20] if it came from a geo-referenced data source.

4- The Query Processor receives query results from each wrapper used in the query process, and sends them to the Data and Map Integrator, which performs integration of these formats using conventional and geo-referenced semantic data integration techniques, with the aid of ontologies and heuristics.

5- The integration process results are then sent to the Format Converter (FC), which converts them to the format chosen by the Data User. This format can be XML[28, 2], GML[9], SVG[24], ESRI Shape [20], or some raster image format.

6- The Format Converter returns results that are passed to the Data Publisher that makes them available for Data User visualization (according to the output format) and provides a means to store results for posterior use.

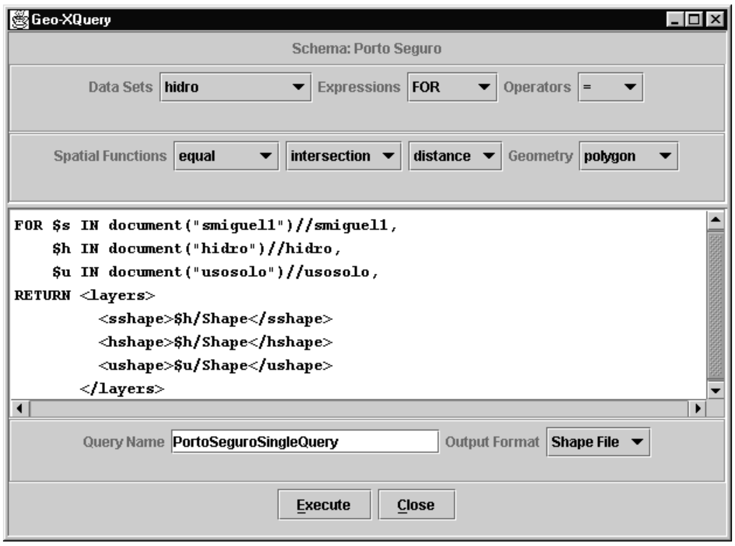

Figure 4 – A Geo-XQuery in a WISEQI interface

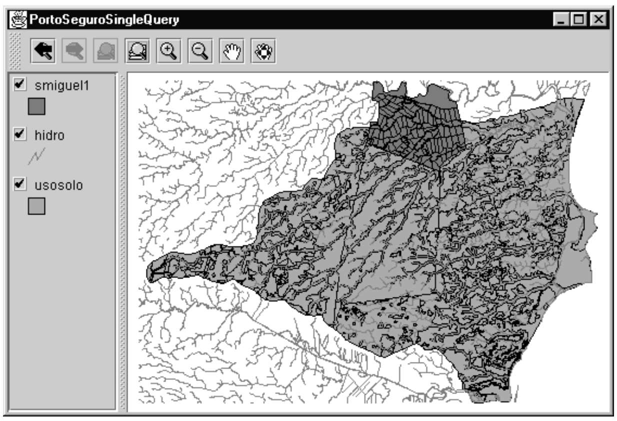

Figure 5 – Visualizing an ESRI Shape format result from query in Figure 4

6 Final considerations

WISE is a geo-referenced semantic data integration system in the Web environment. Semantic integration is found in CDM elaboration with the use of XML Schema and domain application ontologies, as well as in the query results integration.

Differently from others systems, WISE besides permitting access and integration of heterogeneous data sources, offers converters that provide data in several formats available in its architecture, pre-selected by users.

XQuery language is extended with predicates to include spatial data features.

WISE is in its first prototype version with wrappers, data cataloger and query processor implemented.

Acknowledgements: This work was partially financed by CAPES and CNPq Agencies.

References

[1] C. Baru, A. Gupta, B. Ludascher, R. Marciano, Y. Papakonstantinou, and P. Velikhov, *XML-Based Information Mediation With MIX*. Procedings of ACM SIGMOD Conf. on Management of Data, 1999, at Philadelphia, PA, USA.

[2] L. R. Botêlho and J. M. Souza, "Pequeno Glossário De Termos De Linguagens De Marcação", *Developers' Magazine*, 2003, 7, no. 77, 22-25.

[3] D. Chamberlin, "XQuery: An XML Query Language", *IBM Systems Journal* 41, no. 4 (2002): 597-615.

[4] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom, *The TSIMMIS Project: Integration of Heterogeneous Information Sources*. Information Processing Society of Japan, Proceedings of the 10th Anniversary Meeting, 1994, at Tokyo, Japan.

[5] CORBA, "CORBA", 2003. Available from www.corba.org.

[6] Firebird, "Firebird", 2003. Available from firebird.sourceforge.net.

[7] F. Fonseca, M. Egenhofer, and K. A. V. Borges, *Ontologias e Interoperabilidade Semântica Entre SIGs*. Anais do II Workshop Brasileiro de GeoInformática - GeoInfo 2000, at São Paulo - São Paulo, Brasil, 2000.

[8] F. T. Fonseca, M. J. Egenhofer, C. A. Davis, and K. A. V. Borges, "Ontologies and Knowledge Sharing in Urban GIS", *CEUS - Computer, Environment and Urban Systems*, 2000, 24, 3, 232-251.

[9] GML, "Geography Markup Language", 2000. Available from www.opengis.net/gml.

[10] R. Goldman, J. McHugh, and J. Widom, *From Semistructured Data to XML:Migrating the Lore Data Model and Query Language*. 2nd International Workshop on the Web and Databases (WebDB '99), 1999, at Philadelphia, Pennsylvania, USA.

[11] A. Gupta, R. Marciano, I. Zaslavsky, and C. Baru, "Integrating GIS and Imagery Through XML-Based Information Mediation," *Lecture Notes in Computer Science* 1737 (1999).

[12] Java, "Java," 2003. Available from www.java.sun.com.

[13] C. B. Medeiros and A. C. Alencar, *Qualidade Dos Dados e Interoperabilidade Em SIG*. Anais do I Workshop Brasileiro de GeoInformática - GeoInfo 1999, at Campinas – SP – Brasil, 1999, 1, 12-19.

[14] S. P. J. Medeiros, "SPeCS - Sistema de suporte à decisão espacial colaborativa" (D.Sc.COPPE/UFRJ, 2002), Rio de Janeiro, RJ, Brasil.

[15] S. P. J. Medeiros, J. C. M. Strauch, J. M. Souza, and G. R. B. Pinto, *SPeCS - a Spatial Decision Support Collaborative System for Environment Design*. Procedings of SAC2001, 2001, at Las Vegas, USA.

[16] OGC, "Open GIS Simple Features Specification for SQL. Revision 1.1," 2001. Available from www.opengis.org/techno/specs/99-049.pdf.

[17] OGC, "Open GIS Consortium", 2003. Available from www.opengis.org.

[18] G. R. B. Pinto, "Publicação de Dados Ambientais" (M.Sc.COPPE/UFRJ, 2003), Rio de Janeiro, RJ, Brasil.

[19] G. R. B. Pinto, S. P. J. Medeiros, J. M. Souza, J. C. M. Strauch, and C. R. F. Marques, *X-Arc Spatial Data Integration in the SPeCS Collaborative Design Framework*. Proceedings of the Sixth International Conference on CSCW in Design, 2001, at London, Ontario, Canada, 56-60.

[20] Shape file, "ESRI Shape", 2003. Available from www.esri.com/library/whitepapers/pdfs/shapefile.pdf

[21] A. Sorokine and I. Merzliakova, "Interactive Map Applet for Illustrative Purpose," *AGM GIS'98* (1998): 46-51.

[22] J. M. Souza, "Software tools for Conceptual Schema integration" (Ph. D., School of Information Systems, University of East Anglia, 1986), England.

[23] J. C. M. Strauch, "Integração de Bases de Dados Geográficas Heterogêneas e Distribuídas" (D.Sc.COPPE/UFRJ, 1998), Rio de Janeiro, RJ, Brasil.

[24] SVG, "Scalable Vector Graphics (SVG)", 2003. Available from www.w3.org/Graphics/SVG.

[25] A. Tanaka, P. Valduriez, E. Simon, and M. Mattoso, "The Ecobase Project: Database and Web Technologies for Environmental Information Systems", *ACM SIGMOD Record* 30(3) (2001): 70-75.

[26] W3C, "World Wide Web Consortium", 2003. Available from www.w3.org.

[27] F. Xhumari, M. Amzal, and E. Simon, "Le Select: a Middleware System for Publishing Autonomous and Heterogeneous Information Sources", 2000. Available from www-caravel.inria.fr/LeSelect

[28] XML, "Extensible Markup Language (XML)", 2003. Available from www.w3.org/xml.

[29] XML Schema, "XML Schema", 2003. Available from www.w3c.org/XML/Schema.

[30] I. Zaslavsky, R. Marciano, A. Gupta, and C. Baru, *XML-Based Spatial Data Mediation Infrastructure for Global Interoperability*. 4th Global Spatial Data Infrastructure Conference, 2000, at Cape Town, South Africa, 2000.

[31] J. Zhang, M. S. Javed, A. Shaheen, and L. Gruenwald, *A Prototype for Wrapping and Visualizing Geo-Referenced Data in Distributed Environments Using the XML Technology*. ACMGIS 2000, 2000, at VA, USA, 27-32.