ABSTRACT: The heterogeneous nature of geospatial data makes the sharing of it across organizations a complex issue. As a standard means of geospatial data encoding, Geography Markup Language (GML) has been proposed by Open Geospatial Consortium (OGC), which facilitates sharing and exchange of geospatial information. With the proliferation of GML in many geospatial applications and storage support from the standard DBMS, the need arises for the efficient storage mechanism of GML data. This paper analyses approaches for XML data storage in traditional DBMS and proposes alternative solutions for GML. Every GML document structure is defined in an application schema. We propose an approach for GML storage based on the analysis of the underlying application schema. A semi-automatic mapping mechanism has been proposed for the application schema to relational schemas of standard database with the utilization of the domain ontology. A set of relational schema is presumed to exist for different geospatial features. The proposed ontology based schema-matching approach for GML storage enhances the mapping process, thus facilitates sharing of heterogeneous data repositories.

Categories and Subject Descriptors
I.7.2 [Document Preparation] Markup languages; H.2.8 [Database Applications]: Spatial databases and GIS

General Terms
Geography Markup Language, Data mapping, XML

Keywords: GML storage, Schema Mapping, Spatial Data Interoperability, OGC Standards, Ontology

Received 2 Aug. 2006; Revised and accepted 12 June 2007

1. Introduction

Spatial information (more specifically, geospatial information) is increasingly becoming essential almost in all areas of decision support applications, such as organization planning, land management, weather forecasting etc. It is also important an information resource in applications like crisis management system. Geospatial data is being collected in huge amount but in isolation as per the need of the data collector. They generally uses proprietary data formats, storage/access mechanisms. The growing increase in the demand for the spatial data and the availability of the same in diverse formats has raised several issues on interoperability for geospatial data sharing. Although it is found most of the times that the required geospatial data is available, the heterogeneity in the underlying data formats becomes a major hindrance towards successful sharing of the data. The problems that might arise due to heterogeneity of the data include structural heterogeneity (schematic/syntactic heterogeneity), and semantic heterogeneity [14, 26, 27].

The need of integrated access to geospatial data is felt by various nations and there are efforts towards the building of geospatial data infrastructure and portals at the regional as well as at the national level. Nevertheless, efforts for building a Global Spatial Data Infrastructure (GSDI) are also on the rise. The National Spatial Data Infrastructure (NSDI) [21], an attempt towards forming a national level geospatial data infrastructure, faces several problems arising due to the heterogeneity in geospatial data. There are proposals for building a central repository of geospatial information like the proposition of building a geospatial datawarehouse as reported by Labio et al. [25]. Datawarehouse integrates and stores necessary data in a central repository in advance and pre-computed results are readily available at query time. The disadvantages of this method are the storage of possibly outdated information and less flexibility. Carton et al. [26], on the other hand, proposes the concept of Federated Databases. Federated systems rely on the data storage and maintenance at the data sources. Unless a standard data-encoding scheme is put into practice, heterogeneity in data formats become severe for data integration.

Open Geospatial Consortium (OGC) has specified several standards for integrating geospatial data and geoprocessing resources into mainstream computing for enhancing interoperable geospatial information sharing. This leads to the development of standardized protocols for accessing geospatial information. Geography Markup Language (GML), an XML encoding standard for geospatial data, has been proposed for the representation and exchange of geospatial information, including the geometry as well as the properties of the features [1]. Thus instead of sharing data in proprietary format, it has to be converted to GML before being transmitted. The mechanisms and syntax that GML uses to encode spatial information and the core schemas it should adhere to, are defined in the GML implementation specification of OGC [1]. OGC manages consensus processes that result in interoperability between diverse geoprocessing systems.

As came out from OGC tested experiments, a data provider can deliver geospatial information as distinct features in GML, and then it is at the discretion of the data consumer how they are displayed. Thus data interoperability is ensured irrespective of the underlying heterogeneity in the data formats and storage/query methods. GML emerges as a common language through which geospatial data from several repositories can be shared. With increasing popularity of GML, more and more data are being generated in GML format. Thus, the need for an efficient storage mechanism for GML has become an important issue. Although GML is based on XML, and XML storage has been extensively studied [2-4], XML storage techniques can’t be straightforwardly transplanted to GML storage because of several reasons. First, traditional XML documents contain only non-spatial data i.e. they are mostly text-rich, while GML contains spatial elements with geometric properties. Second, GML documents
contain topological and temporal information, which is not present in traditional XML documents. Thirdly, GML querying involves complex spatial operations and analysis.

The paper proposes an approach for GML storage on the basis of the schema mapping between GML application schemas (the metadata component for GML features) and the relational schemas of standard relational database [28]. It is presumed that relational schemas of spatial features are available as libraries of schemas for individual features and we need to find a suitable relational schema for GML storage (it is indeed the case, as relational structure in some form is readily available along with the actual data). The problem then boils down to the mapping of application schemas of the proprietary data repositories to those of the relational schemas. Since the proprietary data are converted to GML in adherence to standard GML application schema for the source, we need to search for the relational schemas, which can best fit a given application schema. While mapping, another important heterogeneity problem arises – the heterogeneity in data semantics. The use of ontology for resolving the geospatial semantics [22] suggests that ontology-based mapping of schema could be more efficient than direct text-based mapping. We adopt this concept and propose an approach of ontology-based matchmaking of the GML application schema tree to the relational structure with the help ontological description of the concepts of the domain.

The organization of the paper is as follows. In this section a general background and motivation of the research have been provided. Some of the related works have been discussed in section 2. Section 3 provides a detailed analysis of the GML application schema. The usability of ontology for semantic mapping of schemas has been discussed in section 4. Section 5 gives the detailed methodology for GML storage. Experimental results along with the efficiency analysis have been studied in section 6. Finally, we conclude in section 7.

2. Related Work

In the past decade, a significant amount of research has been directed towards designing interoperable geographic information systems [14-16]. On the other hand, many approaches for storing and retrieving XML documents have been discussed in the literature [2-4]. When XML documents are stored in off-the-shelf database management systems, the problem of storage model design for storing XML data becomes a database schema design problem. In [7], the authors categorize such approaches into two categories: structure-mapping approach and model-mapping approach. In the former the design of the database schema is based on the understanding of DTD (Document Type Descriptor) or XML Schema that describes the structure of XML documents. More details are available in [8]. In the latter methodology a fixed database schema is used to store any XML documents without the assistance of XML Schema (DTD), such as the approach proposed in [9]. The model-mapping approach is suitable for storing a large number of XML documents that conform to a limited number of document structures, and the document structures are basically static.

Although GML is gaining more and more attention of research community, GML storage has not been studied much. But, as mentioned in section 1, it is an important requirement for the realization of NSDI as well as the GSDI. Ref. [5] addresses this problem with the comparison of three XML storage approaches: LegoDB, a structure-based approach, and two model-based approaches, Monet over relational database and XParent. It has been reported [5] that the approaches give good results with alphanumeric operators but not with spatial operators. The spatial component (geometry, topology etc) contained in the data required to be considered differently. It also proposes a spatial query language over GML using extended SQL syntax [6].

Ref. [13] proposes an approach based on the analysis of the schema tree generated from the GML application schema. A methodology has been proposed for mapping the tree structure of application schema into the relational schema. Although the method is successful for GML storage (with some considered approximation), it corresponds to one-to-one generation of relational schema from application schema. This approach may lead to the generation of different relational schema for the same but semantically different GML features. Even for the same feature schema the schema elements of different repositories may differ semantically. So, the syntactic mapping proposed in [13] may not able to handle the huge GIS data domain with various semantics for individual application schemas.

There are several existing databases with the support for spatial data (known as Spatial DBMS (SDBMS)), like Oracle Spatial, PostGIS. Any of these SDBMSs can be used for GML storage as it allows us to store spatial objects. In the present work, the Oracle Spatial 10g have been used for carrying out the experimentation. The usability of ontology for semantic-based schema mapping has been studied. The proposed methodology uses model-mapping approach for GML storage. A set of standard relational schema for different geospatial features have been used as available relational schema which can be used for GML data storage. The proposed methodology finds semantic mapping of a GML feature to the relational features and selects the relational schema that can most efficiently store the feature under consideration. The semantic mapping uses a combination of Element Level mapping and Feature Level mapping. While the Element Level mapping assesses the linguistic similarity between the elements of two schemas, Feature Level mapping takes care of the structural mapping of the two schemas.

3. GML Application Schema Analysis

As already mentioned, the proposed approach for GML storage finds the relational schema that best fits the given GML feature schema both syntactically and semantically. So, a mapping mechanism is necessary between GML schema and relational database schema. This can be achieved through the mapping between GML elements/attributes and relational attributes and the mapping between the structures of application schema with that of the relational schema. This section gives an overview of GML and application schema and related technologies for parsing GML data.

3.1 Geography Markup Language

Harmonization of geospatial data can be enforced through standard encoding rules for geospatial data. As part of their interoperability specification, OGC have proposed Geography Markup Language (GML) as a standard means for encoding, sharing and transferring geospatial data: “GML is an XML encoding for the transport and storage of geographic information, including both the spatial and non-spatial properties of geographic features” [10]. OGC Abstract Specification for GML defines geospatial objects as features. Each feature can have a set of properties, and each property
is defined by a triple \{name, type, value\}. Properties can have geometric values (Point, LineString, Polygon) as well. GML can be used for achieving interoperability in two ways:
- Transform the dataset of a repository entirely into GML so that other can import the data in GML format. This method is similar to the method of sharing data using external data exchange format except that the available data is now in standard format.
- Real-time conversion of the data into GML based on the request for data access. Users may acquire the object data in real-time, which results in the real-time online data sharing and interoperability.

GML defines both features distinct from geometry objects. A feature is an application object that represents a physical entity, e.g. a building, river, or a person. A feature may or may not have geometric aspects. A geometry object defines a location or region instead of a physical entity, and hence is different from a feature. The distinction between features and geometry objects in GML contrasts with models used in other geographic information systems that treat these them as single entity. Since GML maintains them as separate entity types, the storage of GML in relational database could be easy.

GML also provides the ability for features to share a geometry property with one another by using a remote property reference on the shared geometry property. Remote properties are general features of GML. An \texttt{xlink:href} attribute on a GML geometry property means that the value of the property is the resource referenced in the link. For example, a \texttt{Building} feature in a particular application might have a position given by the primitive GML geometry object type \texttt{Point}. However, the \texttt{Building} is a separate entity from the \texttt{Point} that defines its position. An example GML is shown in the figure 1. It is a GML scrap for a \texttt{City} (Synonymous to \texttt{Metropolitan or Town}). It illustrates the distinction between features and geometry objects. Each sub-feature of this feature have is treated distinctly in GML as individual sub-features. The geometry is completely isolated from the other attributes and thus may be ignored if not required. The sharing of feature structure among geospatial objects makes them reusable. It also helps in establishing relationships among GML features.

---

GML also provides the ability for features to share a geometry property with one another by using a remote property reference on the shared geometry property. Remote properties are general features of GML. An \texttt{xlink:href} attribute on a GML geometry property means that the value of the property is the resource referenced in the link. For example, a \texttt{Building} feature in a particular application might have a position given by the primitive GML geometry object type \texttt{Point}. However, the \texttt{Building} is a separate entity from the \texttt{Point} that defines its position. An example GML is shown in the figure 1. It is a GML scrap for a \texttt{City} (Synonymous to \texttt{Metropolitan or Town}). It illustrates the distinction between features and geometry objects. Each sub-feature of this feature have is treated distinctly in GML as individual sub-features. The geometry is completely isolated from the other attributes and thus may be ignored if not required. The sharing of feature structure among geospatial objects makes them reusable. It also helps in establishing relationships among GML features.

---

```xml
<?xml version="1.0" encoding="UTF-8"?>
<CityModel xmlns="http://www.opengis.net/examples"
xmlns:gml="http://www.opengis.net/gml"
xsi:schemaLocation="http://www.opengis.net/examples city.xsd">
<gml:name>Cambridge</gml:name>
<gml:boundedBy>
<gml:Box srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
<gml:coord><gml:X>0.0</gml:X><gml:Y>0.0</gml:Y></gml:coord>
<gml:coord><gml:X>100.0</gml:X><gml:Y>100.0</gml:Y></gml:coord>
</gml:Box>
</gml:boundedBy>
<cityMember>
<River>
<gml:description>The river that runs through Cambridge.</gml:description>
<gml:name>Cam</gml:name>
<gml:centerLineOf>
<gml:LineString srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
<gml:coord><gml:X>0</gml:X><gml:Y>50</gml:Y></gml:coord>
<gml:coord><gml:X>70</gml:X><gml:Y>60</gml:Y></gml:coord>
</gml:LineString>
</gml:centerLineOf>
</River>
<cityMember>
<Road>
<gml:name>M11</gml:name>
<linearGeometry>
<gml:LineString srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
<gml:coord><gml:X>0</gml:X><gml:Y>5.0</gml:Y></gml:coord>
<gml:coord><gml:X>80.5</gml:X><gml:Y>60.9</gml:Y></gml:coord>
</gml:LineString>
</linearGeometry>
<classification>motorway</classification>
<number>11</number>
</Road>
<cityMember>
</CityModel>
```

Figure 1. Sample GML Feature Encoding
3.2 GML Application Schema

GML is a markup language, which means that GML document have to follow certain rules in order to be a valid GML document. This set of rules is defined in a schema document. The documents should confrom to the requirements in the GML specification. GML version 1.0 uses the Document Type Descriptors (DTDs) for defining the structure, the elements and the associated attributes for a feature. GML version 2.0 and 3.0 use XML schema instead of DTD.

GML application schema is an extension of XML Schema. It provides a set of type definitions and element declarations that can be used to check the validity of well-formed GML documents [15]. An example schema is shown in figure 2. The UML model for the schema is also shown in figure 3. GML defines various entities such as features, geometries and topologies through a hierarchy of GML objects. It also provides a series of core schemas for describing geographic data. These include feature, geometry, topology, value, coordinate reference system, and style-descriptor etc [8]. Most applications make use of only a subset of these core schemas as per the requirement of the domain. GML also help linking one feature with another. This way, relationships among geospatial objects can be captured in GML. For converting a propriatory spatial data into GML, a necessary schema in conformance of the source schema needs to be formed.

3.2 GML Parser

The analysis of GML document structure can be done with the help of available parsers. There are two standard APIs, which are used in software applications for this purpose: DOM (Document Object Model) and SAX (Simple API for XML) [12]. They basically differ in the way the GML documents are processed. DOM constructs a tree structure of the GML data for processing. This structure would consume a large amount of memory for spatial databases. Therefore, DOM seems to be unsuitable for GML documents due to the huge size of GML documents. A SAX parser parses the documents sequentially, treating the document as a data stream. However, SAX parser has to process the entire document before the processed data is available to the application. Various studies have been performed to compare the performance of GML parsers [12]. SAX is suitable for storing and processing data in server side; in contrast, DOM is suitable for storing...
documents at the client side since it requires more resources. Since for GML storage purpose parsing is done at the server side, SAX parser has been used for parsing of the GML documents in the present work.

4. Ontology-based GML Storage

The proposed methodology for determining suitable relational schema for a given GML application schema explores the structure and properties of the schema both at the syntactic and semantic level with the use of ontology. Keeping in mind the static nature of the feature structure of geographical objects, we presume a set of relational schemas libraries to be existing a priori for spatial data storage. Geospatial feature properties/structures are more or less fixed. This is true if consider the data produced by a particular organization or the surveying data of any particular region. The structure of the legacy datasets is defined and accepted as standard. This is to say, the properties geospatial features like road can be standardized and corresponding relational schema can be established. Other feature that relates road can be a subclass or super-class of road. This way, the same relational schema for road can be used for storing other features of which road feature is a part. Thus we can utilize the available metadata of the geospatial repositories for similarity assessment.

The proposed methodology aims at finding appropriate mapping between the GML schema and relational schema in two steps — at the **Element level** and at the **Structural level**. The procedure makes the GML storage a semi-automatic, in the sense that the necessary relational schemas need to be designed manually from the available metadata descriptions. Once the relational schemas are available, GML schema mapping and subsequent storage procedure is automated. Instead of trying to map each individual GML application schema into separate relational schema, we can use the same relational schema for mapping semantically similar GML schema.

4.1 Spatial Data Storage

It is obviously beneficial using a spatial database for storing the GML data due to the formers support of spatial data with predefined spatial objects. This way, we can make use of sophisticated functions available in traditional Spatial DMBS, such as concurrency control, crash recovery, scalability, and highly optimized query processors. A spatial database system has to provide various functionalities, including input, storage, retrieval, selection, analysis, and display of the information [11].

Currently, there are many spatial databases, such as ESRI’s ArcSDE, Oracle Spatial and DB2 Spatial, open source software...
PostGIS / PostgreSQL. In the present work, Oracle Spatial 10g has been used, mainly due to its support for spatial data storage. According to our proposed methodology for geospatial data storage, we store non-spatial geographic information in relational tables, but the spatial information in GML is mapped to the column with geometry type. Oracle Spatial database has a special data type called MDSYS.SDO_GEOMETRY, which is used to store geometries of geospatial features like point, line and polygon.

4.2 Application Ontology
Onontology has been introduced in artificial intelligence (AI) as an explicit specification of a conceptualization; therefore it can be used to describe the semantics of the information sources and to make the content explicit with respect to the integration task. It can be defined in many ways that suits the need of its purpose [19]. For our purpose, we can define it as a schema and metadata capturing the semantic relationships of the concepts for a domain. An ontology "O" is a tuple consisting of the following: the concepts "C" of the schema are arranged in a subsumption hierarchy H_c, relations R_c exist between single concepts, relations can also be arranged in a hierarchy H_R.

\[ O := (C, H_c, R_c, H_R) \]

An ontology structure for Waterbody domain is shown in figure 4. The different concepts of the domain are represented in ontological structure for preparing a conceptual model of the domain. The concepts for this domain are Pond, River or Lake. The subsumption hierarchy shows that a River IsA WaterBody or Lake IsA Pond. Relationships may exist among the concepts e.g. River Meets Lake. This way the concepts for a domain can be modeled as an ontology.

In general, there are three different possible ways of how ontologies are employed for conceptualizing a domain; single ontology approaches, multiple ontologies approaches and hybrid approaches [15]. Single ontology approach uses a global ontology of shared vocabulary for semantics specifications. Multiple ontologies approaches use separate local ontology for each information source, which can simplify the integration task and supports the change, but increase the difficulties to compare different source ontologies. We adopt Hybrid ontology approaches, which is the combination of single and multiple ontology approaches, in which the semantics of each source is described by its own ontology and a global shared vocabulary is also built on the local ontologies.

Many formal languages to specify ontologies for semantic web have been proposed, such as RDF [20], OWL [27]. Though these languages differ in their terminologies and expressiveness, the ontologies that they model essentially share the same features we described above.

4.3 System Architecture
A prototype system for geospatial data (in GML format) storage in spatial database has been implemented. The overall architecture of the system is shown in figure 5. The system operates at two levels – first, it needs to find the similar relational schema for a given GML schema and secondly, the GML data has to be loaded in the database. These two steps are performed by the Schema Matcher and Data Loader components of the system respectively.

The Schema Matcher takes a GML Application schema as input and uses available Relational Schemas for semantic similarity assessment among them. The given Application Schema is matched against each of the Relational Schemas. The matching is done in two steps - Feature Mapping and Element Mapping. Feature Mapping uses the domain ontology for structural similarity assessment and if a match is found, the Element Level mapping is carried out on the same Relational Schema. It uses element to element mapping between the two input schemas. The detailed description of these two approaches is given in section 5.

A pre-processing is done on the Application Schema for transforming it in accordance to the matched Relational Schema. Finally, it is the Data Loader that takes an instance of the GML application schema, converts it into relational form and loads the data into the relational table corresponding to the matched schema.
5. Schema Similarity Mapping

Many existing spatial data spread over the Internet in conventional GIS data format (usually stored in files, e.g. ArcInfo shapefile), and non-spatial data is stored in relational database or database files. For ensuring interoperability among these disparate sources, we need to convert them in GML-encoded format. Once a domain of interest is chosen, the semantics of each source is described by a local ontology. The local ontology models only the concepts which are in the scope of the local data repository. For example, it models the data it may have for road network, water body etc. A global ontology of shared concepts is also built upon the local ontologies, which contains basic terms or primitives of the domain and captures the inter-relation among different features. Ontologies are used to describe more complex semantics for the efficient determination of the correspondence of GML documents to a relational table.

With general observation, it can be identified that each feature member in GML schema is collection of features (known as sub-feature), where a feature is a real world entity. The featureCollection object contains a set of features. As an example, we can take the feature City, which is a collection of houses, roads, schools and many other features. The major task of the proposed approach is to match the semantically similar features and corresponding attributes and structure of the GML feature to those of the relational feature. The semantic matching of all the attributes increases the recall of the search method. The global ontology is used for semantic mapping of the application schema features with the relational schema features. This will help in finding similar relational features (say, TransportNetwork, CommunicationNetwork) corresponding to a GML feature (say, RoadNetwork) by comparing the structure of the respective features. The local ontology contains the detailed structure for the feature and hence used for more detailed analysis. This is to say, the structure of the feature in the application schema is mapped to that of the relational schema with the help of local ontology. Although the procedure bears similarity to the model-mapping approach [9] for XML storing, the semantic issue has not been addressed in model-mapping approach. The procedure for semantic based GML storage can be divided into the following two steps:

- **Feature Mapping** step for mapping features and sub-features from the relational schema with those of the GML application schema, and
- **Element Mapping** step for mapping features elements in application schema with those of the relational schema.

The **Matchmaker** module (figure 5) does the mapping between the schemas as follows:

**Input:**

- \( \text{AS} = \text{Set of all elements in Application Schema} \ A \),
- \( \text{AS} = \{ \text{AS}_1, \text{AS}_2, \text{AS}_3, ..., \text{AS}_n \} \) and
- \( \text{RS} = \text{Set of all elements in Relational Schema} \ R \),
- \( \text{RS} = \{ \text{RS}_1, \text{RS}_2, \text{RS}_3, ..., \text{RS}_n \} \)

**Output:**

- \( \text{MP} = \text{Mapping between AS, RS, with matching score} \ M, \text{where AS} \in \text{AS} \text{ and RS} \in \text{RS \ and M} \in [0, 1] \)

The mapping procedure gives more precedence to the **Feature Mapping** over **Element Mapping**. This is determined on the basis of weights assigned for each mapping method. The **Matching score** \( M \) is a weighted average of \( W_f \) and \( W_e \) weights for the two mapping methods respectively. The weights are given between 0 and 1. If two schemas have a matching feature structure then more weightage should be given to \( W_f \). If a feature does not have any structural matching with a relational schema then the element mapping is not considered i.e. \( W_e \) is also set to 0. This is due to the simple reason that if the structure of the respective schema is not similar, they can not mapped anyway. As an example, if we find the structure of the feature City in GML schema is different from that of the relational schema, there is no point in making further analysis with the elements of these features. This can be set as the pruning condition of our proposed methodology semantic-based schema similarity assessment. Thus **Mapping Score** \( M \) can be set as

\[
M = \frac{W_f \ast \text{Feature Mapping} + W_e \ast \text{Element Mapping}}{W_f + W_e}
\]

5.1 Feature Mapping

The **Feature Mapping** step uses global ontology for the mapping of the GML schema features into corresponding relational features. To integrate data from disparate ontologies, we must know the semantic correspondences between their elements. The **Feature Mapping** process matches the structural similarity of a feature in terms of its sub-feature to the structure of the relational schema. Thus we need to match the sub-concepts under the feature concepts as well. For example, a Road feature may have Highways, Bridges as its sub-concepts. It is required to map the sub-concepts of the GML schema to the relational schema for better storage purpose. The mapping procedure exploits both semantic and syntactic mapping for this purpose. The sub-feature similarity \( S_1 \) is the average match score of each individual property of the feature. If there are \( n \) sub-features for the feature under consideration then \( S_1 \) is given by equation 1. On the other hand, sub-feature match \( S_2 \) is defined as the fraction of the total number of properties of a feature that are matched and is computed as shown in equation 2.

\[
S_1 = \frac{1}{n} \sum_{i=0}^{n} M(\text{subfeature})
\]

(1)

\[
S_2 = \frac{\text{matching subfeature}}{\text{total subfeature}}
\]

(2)

So, Feature Mapping can be defined as in equation 3.

\[
\text{Feature Mapping} = \sqrt{S_1 \ast S_2}
\]

(3)

The matching procedure can be as simple as mapping of a Road feature in the GML schema to the Road feature of the relational schema. On the other hand, the feature could have different names but they are similar. As an example Road and Street are semantically similar and should match in the same relational schema. This sort of matching can be said to be Exact Mapping. Let us assume the GML schema \( G \) and the matching relational schema \( R \). Then formally we can say that

\[
\text{Match}_{\text{exact}}(G, R) \Rightarrow G = R \text{ \ or \ } G \equiv R
\]
where ‘=’ exactly similar terms and ‘*’ means semantically similar terms. A more complex case arises for features which are subClassOf some other feature. As an example MainRoad can be a subClassOf Road feature. This sort of matching can be defined as Subsumption and can be defined formally as follows:

\[
\text{Match}_{\text{subsumption}}(G, R) \Rightarrow G = \text{subClassOf}(R)
\]

### 5.2 Element Mapping

Once a suitable relational schema has been identified for the GML feature schema, the next step involves Element Mapping for closer analysis of the matched features. The matching of the spatial elements, in addition to non-spatial elements, is also an important issue. Overall, there are three levels of naming geometry properties in GML:

- formal names that denote geometry properties in a manner based on the type of geometry allowed as a property value;
- descriptive names provide a set of standardized synonyms or aliases for the formal names;
- application-specific names defined in application schemas based on GML.

The formal and descriptive names for the basic geometric properties are listed in Table 1.

For matchmaking among the elements in the GML schema to those of the relational schema, we adopt schema-level matcher approach, which only considers schema information, not instance data. This basically includes the schema elements, such as names, description, data type, relationships, constraints and schema structure, which are organized in each local ontology. The procedure assesses linguistic similarity among the schema elements. It uses various element matching approaches like synonym matching, abbreviation expansion, stemming, tokenization etc. The synonym matching uses WordNet [18], whereas abbreviation matching uses a custom abbreviation dictionary. The tokenization approach first tokenizes the elements based on punctuation and capitalization. Then it removes unnecessary words from the list of tokens, using a stop-word list. The closely related terms in the application schema are discovered using a thesaurus. Thus it will be able to find an obvious matching between the elements (FeatureID and FID) as well as non-obvious ones (such as, DisasterType, NaturalHazardType). The adopted procedure is influenced from the procedure described in [17]. These include the following filtering procedures.

If the Element Mapping procedure (any one of the procedures) returns a full match, then a match score of 1 for element mapping is returned. If all the match algorithms give a match value of zero, then the element mapping of those concepts is 0. If on the other hand, any of the procedure returns a value between 0 and 1, then, x, the average of all non-zero match scores is taken. Thus we can define Element Mapping as

\[
\text{Element Mapping} = \begin{cases} 
1; & \text{if any procedure returns 1} \\
0; & \text{all procedure returns 0} \\
x; & \text{some procedure doesn't return}
\end{cases}
\]

As an example, comparing the element fid with featureid may return 1 for some (Abbreviation expansion method) and may return 0 for others (Synonym matching). Thus the Element mapping value is 0.5 for this element. As each application schema feature is compared against all the concepts from ontologies, it is necessary to find the best matching concept. An algorithm for this maintains a variable for best mapping, whose Mapping Score (M) is checked against the newly generated mapping. If the new mapping has a better M, it is assigned as the best mapping. The overview of the algorithm for best mapping is as follows

\[
\text{Input: } as_i \in AS, rs_j \in RS \text{ and } \bigcirc \\
\text{Output: } MP = \left( as_i, rs_j, M \right)
\]

The proposed methodology thus helps in determining the mapping score for a application schema feature with the set of relation schema feature of same or varying domain. The suitable schema can be chosen from the schemas having best possible score or having score above some threshold. In section 6, we carry out some studies on determining the efficiency of the proposed method.

### 6. Case Study

The proposed methodology for GML storage based on ontology has been tested over a collection of relational schemas. Various features with varied semantics have been used for finding the mapping relational schemas. The efficiency of the method has been analyzed on the basis of successful mapping. We have selected a collection of relational schemas belonging to two different domains, Agriculture and Weather domains. The Feature Mapping approach, then, takes collection of GML feature schemas belonging to different domains. Some of these belong to Agriculture domain and others to the Weather domain. The Feature Mapping approach first tries to categorize the GML features into domains of relational features. The experiment has been carried out on 25 different GML features. The mapping is done on various values for the mapping score (MP). The observed output is shown in the figure 6. The blue columns give the actual number of features of each domain. The efficiency of mapping depends on the selection of the value for MP. With MP=0.7 we obtained a significant mapping among the features.

<table>
<thead>
<tr>
<th>Formal Name</th>
<th>Descriptive Name</th>
<th>Geometry Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>boundedBy</td>
<td>-</td>
<td>Box</td>
</tr>
<tr>
<td>PointProperty</td>
<td>Location, Position,</td>
<td>Point</td>
</tr>
<tr>
<td>lineStringProperty</td>
<td>centerLineOf, edgeOf</td>
<td>LineString</td>
</tr>
<tr>
<td>polygonProperty</td>
<td>extentOf, coverage</td>
<td>Polygon</td>
</tr>
<tr>
<td>geometryProperty</td>
<td>-</td>
<td>any</td>
</tr>
<tr>
<td>multyGeometry</td>
<td>Property</td>
<td>MultiGeometry</td>
</tr>
</tbody>
</table>

Table 1. Geometry Properties of Spatial elements
Next, the Element Level mapping is applied for the relational schema belonging to Weather domain. Figure 7 depicts the outcome of the Element Mapping approach. We selected the relational schema for WeatherReport, WeatherMetric and WeatherFetcher. The GML feature schemas under consideration are AirportWeather, WindEvent, HumidityEvent, WeatherProvider and CityWeather. We observe that almost every feature matches to the corresponding relational schema with good mapping score. Only in the case of WeatherProvider schema maps more for WeatherMetric relational schema than the expected WeatherFetcher relation. This is due to the fact that the structure as well as elements of this feature substantially overlaps with both the relational schemas.

Thus it is observed, except some unexpected situation, the proposed method finds the suitable relational schemas in conformance of an application schema for storing GML data. So, the objective of finding suitable Relational Schema for Application Schema has been achieved with good accuracy of matching.

7. Conclusion

The heterogeneity in various geospatial information repositories becomes a barrier in wide-scale use of it by the large user community. The heterogeneity is both in syntactic and semantic aspects of the data. There are increasing efforts towards standardization of the geospatial data collection and storage mechanism. OGC specified standardizations are now being used widely. GML has been developed as standard encoding for geospatial information for increasing the transportability across diverse systems over Internet. With growing popularity of GML, more and more geospatial data are being generated in GML format. Although OGC has provided standard guidelines for GML, much work has still to be done in the area of parsing, querying, and storage of geospatial data.

With the increasing availability of GML, the requirement has come for efficient storage of the geospatial information in standard database management systems (DBMS). Since most of the geospatial data descriptions are available as metadata, it is possible to map the available metadata (relational schema) with the application schema. But the semantic issue puts a barrier in efficient utilization of the same. In this paper, we have proposed an approach for semantic based mapping of GML application schema to relational schema with the use of hybrid ontology. We have shown that the method efficiently identifies the relational schema from the schema libraries for GML storage purpose and subsequently loads the GML documents into the schema table of a spatial database. The broad objective of standardization for geospatial information sharing can be achieved with standard data repositories built on spatial databases and the data repositories capable of providing data in GML encoded format.

References


