

EXTENDING 2D INTEROPERABILITY FRAMEWORKS TO 3D

H. Müller, E. Curtis

Snowflake Software Ltd, East Gate House, Town Quay, Southampton, SO14 2NY, United Kingdom
(hardo.mueller, eddie.curtis)@snowflakesoftware.co.uk

KEY WORDS: WFS, GML, Databases, Three-dimensional, OGC, ISO

ABSTRACT:

The Open Geospatial Consortium (OGC) has developed a mature, standards based architecture for interoperability for 2D geographical information. This paper examines the issues in extending that architecture to the dissemination of 3D data. We will focus on two interfaces in particular: the interface between the data store (typically a relational database) and the Web Feature Server (WFS) and the interface between the WFS and the client applications. For 2D applications the interface between the data store and the WFS may be governed by OGC or de facto standards. However, there is currently no standard encoding, either public or de facto, for encoding 3D geometry in the data store. Several 3D database models are still proposed or already in use, so there is a need for applications being able to map different 3D database models to a WFS interface. We give a guideline for setting up 3D databases by applying features, complex properties, and complex 3D geometries in the relational model and simpler geometries in the DBMS build-in spatial model. Furthermore a software architecture is presented, which is based on ISO 19107 feature geometry standard, to minimize the cost of software adaptation.

1. INTRODUCTION

The Open Geospatial Consortium (OGC) has developed a mature, standards based architecture for interoperability for 2D geographical information. The most common components of this architecture are the Web Map Service (WMS) for providing maps and the Web Feature Service (WFS) for providing geographic information and related geometries as vector data (OGC, 2004b; OGC, 2002a; OGC, 2002b). In the mean time a variety of products and applications are available to support this architecture.

Recent developments resulted in updated specifications for this interoperability architecture, which allow dealing with 3D data (OGC 2004a; OGC, 2005a). This paper examines the issues in extending that architecture to the dissemination of 3D data using these updated specifications.

After giving an overview about interoperability frameworks in two dimensions, we will discuss their extension to 3D. Several proposed 3D database models will be mentioned and a guideline to store 3D data in databases is suggested. We finally discuss how a software architecture for a 3D interoperability framework can be realized and how it benefits from interoperability standards.

2. INTEROPERABILITY FRAMEWORK FOR 2D

The OGC architecture defines a number of interfaces that allow applications and data stores to exchange geographical information. These specifications include both data structure specifications for spatial data and services interfaces to provide access to and processing of that data.

This paper will focus on two interfaces in particular: the interface between the data store (typically a relational database) and the Web Feature Server (WFS) (OGC, 2002b), and the interface between the WFS and the client applications.

2.1 Architecture

We will consider three components of the architecture: The data store, the WFS and the client application (cf. Figure 1).

The geographical data resides in the data store which provides facilities for the secure management of the data. The data store provides functionality such as transaction management, backup and recovery etc. i.e. the data is managed in a (typically relational) database management system.

The WFS is an application which offers access to geographical information via a web service. The purpose of the WFS is to receive requests for data from client application and respond by returning the appropriate selection of data from the data store.

The client applications represent a wide variety of applications which require geographical data for some purpose. The WFS interface is designed to provide open access to a large number of varied clients.

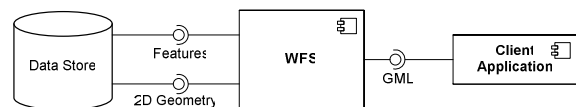


Figure 1 Architecture for a 2D WFS

2.2 Existing (De facto-) Standards

The interface between the data store and the WFS may be governed by OGC or de facto standards. The OGC Simple Features for SQL specification (OGC, 1999) provides two data structures for 2D geometry: Well Known Binary (WKB) and Well Known Text (WKT). As an example of a de facto standard, the Oracle database has a built in data type for 2D geometry which is a de facto standard amongst Oracle applications.

The interface between the WFS and the client application is governed chiefly by two OGC specifications: Web Feature Server (OGC, 2002b) and Geography Markup Language (OGC, 2002a). The WFS specification provides protocols for the client to request and receive data. GML is an extension of XML for

geography and provides the data structures for encoding geometry.

All of the geometry encodings (WKB, WKT, Oracle SDO geometry, and GML) are based on the same abstract geometry model: Feature Geometry, which is also the ISO standard ISO19107 – Spatial Schema (OGC, 2001). As a consequence the translation between these different encodings can be carried out reliably and repeatedly without loss of information.

3. EXTENSION TO 3D

3.1 Missing (De facto-) Standards

The geometry model of ISO19107 provides a model for 3D geometry and GML provides encodings for 3D geometry. However, there is currently no standard encoding, either public or de facto, for encoding 3D geometry in the data store. Whilst in principle WFSs can serve data with 3D geometry, in practice this process is hampered by the lack of an off-the-shelf 3D model in the data stores.

3.2 Proposed 3D Database Models

Stoter & Zlatanova (2003) show two simple approaches to represent 3D models in DBMS. The first method uses a table consisting of the polyhedron faces, which can be stored as single polygons in an Oracle DBMS. Polyhedrons are defined in another table, which is joined to the face table. The second method uses a MultiPolygon object to store the geometry. The advantage of this method is that a lot of existing CAD/GIS applications can access this structure of data without the need of software adaptation. Disadvantageous is the missing topology and the restricted potentials to model various 3D-objects, e.g. volumes containing holes.

How to store a 3D polyhedron as an Oracle Spatial 9i geometry object is described by Arens et al. (2003). Here the geometry field is used to store point coordinates of the polyhedron as well as sequences of local point references to model the faces of the object. Additional information about the 3D semantics is stored in the meta information part of the geometry field.

Gröger et al.(2004) propose a database model for 3D City Models. In this model topology is represented explicitly and it is based on OpenGIS and ISO standards. Some links between model elements are represented by nested tables. For spatial data types provided by the DBMS, a redundant storage is proposed for an efficient geometric access. Besides geometry an approach to store texture and colour properties is suggested.

3.3 Recommendation for 3D Database Modelling

Spatial DBMS like Oracle 9i Spatial support currently a set of basic geometric data types, which are equivalent to those specified by the OpenGIS Simple Features for SQL specification (OGC, 1999). For an interoperability framework, which is able to exchange 3D-models, a much wider range of geometric data types is needed. The latest WFS Specification (OGC, 2005a) provides a standard suitable to exchange 3D models by web services. This standard is based on ISO 19107 and GML (OGC, 2004), which allows exchanging an extensive range of different geometric data types. In order to integrate the spatial database into the 3D interoperability framework a mapping is required between the built in spatial data types of the database and the 3D geometry types of the framework.

Depending on the context the data structure of one geometric data type in the database can be mapped to a variety of data types in ISO 19107 and GML. E. g. a POLYGON object in a database can be mapped to a Polygon or a Surface containing one SurfacePatch or an Envelope if it is rectangular. Table 1 shows a list of possible mappings between database geometry types and ISO 19107 and the respective GML types. The decision of which mapping to choose is often context dependent and can be made when the database to GML mapping is configured. This can be aided by interactive tools and needs no programming skills.

This process imposes alternative semantics on the database data structure when it is imported. For example, a multi-polygon in the database can be treated as a surface by treating each polygon in the multi-polygon as a surface patch in the surface. However, because the semantics of surface patches are different from multi-polygon the built in validation rules of the database data type will not apply the extra constraints which apply to surface patches.

| Database Type | ISO 19107 Type | GML Type |
|---------------------|------------------------|---------------------|
| POINT | GM_Point | Point |
| | DirectPosition | pos |
| | DirectPosition | vector |
| LINE-STRING | GM_Curve | LineString |
| | GM_LineString | LineStringSegment |
| | GM_Curve | Curve |
| | GM_OrientableCurve | OrientableCurve |
| POLYGON | GM_Surface | Polygon |
| | GM_Ring | LinearRing |
| | GM_Surface | Surface |
| | GM_Polygon | PolygonPatch |
| | GM_Triangle | Triangle |
| | GM_Polygon | Rectangle |
| | GM_Ring | Ring |
| | GM_OrientableSurface | OrientableSurface |
| | GM_PolyhedralSurface | PolyhedralSurface |
| | GM_TriangulatedSurface | TriangulatedSurface |
| | GM_Envelope | Envelope |
| MULTI-POINT | GM_MultiPoint | MultiPoint |
| | GM_PointArray | posList |
| | GM_Tin | Tin |
| MULTI-LINE-STRING | GM_MultiCurve | MultiCurve |
| | GM_CompositeCurve | CompositeCurve |
| MULTI-POLYGON | GM_MultiSurface | MultiSurface |
| | GM_Surface | Surface |
| | GM_OrientableSurface | OrientableSurface |
| | GM_PolyhedralSurface | PolyhedralSurface |
| | GM_TriangulatedSurface | TriangulatedSurface |
| | GM_Solid | Solid |
| GM_CompositeSurface | CompositeSurface | |

Table 1. Mapping of Database Geometry Components to GML Elements.

Admittedly not all data types required by the framework can be mapped from one spatial data type in the database. In particular, 3D-data types like Solid or CompositeSolid cannot be represented by one of the build-in database geometries. Fortunately these data types are always composed of simpler data types which can be derived from a database geometry type. E.g. the Solid data type contains ‘exterior’ and ‘interior’ properties which are Surface objects in the ISO model. The

Surface again contains SurfacePatch objects, which can be represented by POLYGON objects in the database.

For the more complex geometry types, rather than storing geometry in a single column, we propose to model these geometries as tables which aggregate other geometries via table joins. The simpler geometries aggregated in this way can be represented either as build-in database types or further aggregations through table joins. Table 2 shows how database elements can be mapped to GML components using this proposed approach.

| Database Element | GML Component | Conditions |
|----------------------|----------------------------|--|
| Table | Feature | |
| | Complex Property | |
| | Simple Property | maxOccurs > 1 |
| | Reference | maxOccurs >1 |
| | Geometry Element (complex) | Not supported by Simple Features for SQL |
| Column (non spatial) | Simple Property | maxOccurs = 1 |
| | Reference | maxOccurs = 1 |
| | ID | |
| Spatial Column | Geometry Element (simple) | Supported by Simple Features for SQL |

Table 2. General Approach of Database to GML Mapping.

Using this approach, existing DBMS to WFS translation software can easily be extended to handle 3D data. Since complex 3D objects are handled in the same manner as complex feature properties there is no architectural software redesign necessary. Furthermore the details of how to map the 3D geometries to database tables are not fixed, which means it can be adapted to the requirements of the service provider. E.g. if topology is an important issue, only primitive data types like points or lines are used as DBMS build-in geometries whereas all the topology information is stored in the relational model. Otherwise if database performance is more important DBMS build-in geometries like Multi-Polygons are mainly used.

The database model of City GML data has been set up using the method described above. Table 3 shows a selection of database components from the City GML database and their XML mappings.

| Table Column(data type) | CityGML Element |
|-------------------------|---|
| BUILDING | Building |
| FID(STRING) | Building/@gml:id |
| ROOFTYPE(STRING) | Building/roofType |
| BLDG_ADDRESS | Building/address |
| CITY (STRING) | Building/address/city |
| BLDG_LD1_SOLID | Building/lod1SolidProperty/gml:Solid |
| EXTERIOR (MULTIPOLYGON) | Building/lod1SolidProperty/gml:Solid/gml:exterior/gml:Surface |
| BLDG_LD1_SOLID_INT | |
| INTERIOR (MULTIPOLYGON) | Building/lod1SolidProperty/gml:Solid/gml:interior/gml:Surface |

Table 3. Mapping of a Database Model to CityGML (excerpt)

3.4 Spatial Queries

In an Interoperability Framework spatial queries can be passed to a WFS (OGC, 2005a), which uses the OGC Filter expressions for spatial requests. Version 1.1 of the OGC Filter specification allows GML 3 to be used for geometry values and supports the operations Equals, Disjoint, Touches, Within, Overlaps, Crosses, Intersects, Contains and BBOX, which is equivalent to 'Not Disjoint' (OGC, 2005b).

These operations are translated to SQL queries by the framework. Since Oracle 9i does not have full support for 3D spatial queries, in some cases only approximate results can be expected (cf. Gröger et al., 2004).

Where the interpretation of the database data structure imposes new semantics on the data structure those semantics are not understood by the built-in query facilities of the database. This also places some restrictions on the ability to efficiently carry out accurate 3D queries. However, many of these potential limitations are obscured by the restrictions of query functions to 2D.

3.5 Migration to Standardized 3D Database Models

At present, to extend a database architecture to 3D requires implementers to create a bespoke implementation of 3D geometry in the data store and a bespoke interface for 3D between the data store and the WFS. However, by translating from this bespoke interface to GML the WFS can then provide a standard interface to the client applications since the GML encoding does support 3D. This architecture therefore insulates the client applications from the non-standard interfaces of the data store.

If in future off-the-shelf models for 3D geometry become available for data stores then only early adopters of this architecture will need to bear the cost of developing bespoke 3D models. If the (de facto-) standards are extended to 3D then late adopters will be able to acquire 3D geometry support in the data store.

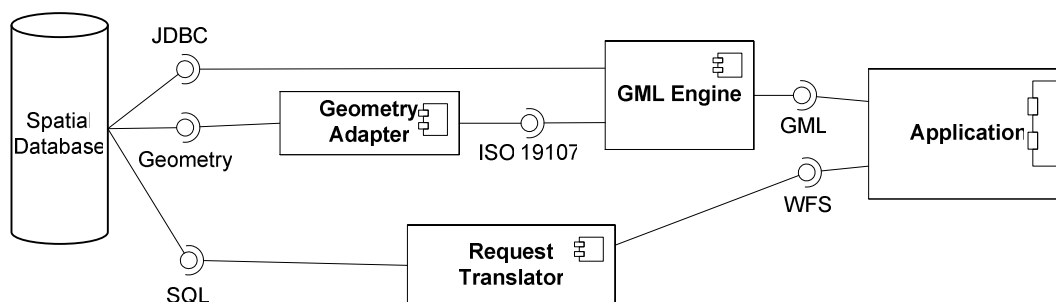


Figure 2 Architecture of a 3D WFS using exchangeable adapters to connect different kinds of data stores.

Because this architecture insulates the client applications from the data store, interface changes to the data store interface will affect only the WFS. Migration from bespoke 3D geometry implementation to an off-the-shelf implementation can therefore take place by changing the data store and WFS. No change is required in the applications. Since there are many clients to each WFS, this architecture limits the change to two of many components and so offers a seamless migration path from bespoke to off-the-shelf 3D data store models.

To minimize the costs of the migration process, the internal WFS architecture can be further optimized (cf. Figure 2). In this scenario the WFS consists of a GML Engine component and a Geometry Adapter component. The GML engine reads the non geometric data direct from the data store using e.g. standard database access interfaces like JDBC or ODBC, whereas the geometric part of the data is taken from an adapter component, which provides an ISO 19107 based API to the GML engine. Geometry components which are not supported by build-in DBMS geometries are constructed using the JDBC interface. The GML engine of the WFS will therefore be unaffected by changing the geometry model in the data store. The adapter component, which connects direct to the geometry part of the data store, is the part of the WFS which encapsulates the geometry specific interfaces of the DBMS.

4. CONCLUSIONS

ISO19107 has proved an effective aid to translating 2D geometries between different encodings. It could play the same role for 3D. New updates of OGC standards enable now 3D interoperability frameworks.

The current absence of off-the-shelf models for the storage of 3D geometry is an obstacle to the extension of the WFS architecture to 3D data. The creation of bespoke 3D geometry models for the data store adds to the cost and complexity of implementing a 3D data store. The absence of built-in 3D geometry types also restricts the ability to efficiently carry out full 3D query operations.

A database can be designed containing complex 3D geometries expressed as relational models aggregating simpler geometries. Such a database can easily be integrated into an interoperability framework.

Since this architecture insulates the client applications from the data store interface it provides a smooth migration path for the migration from bespoke 3D geometry storage to off-the-shelf models for geometry storage.

REFERENCES

- Arens, C., Stoter, J.E., & van Oosterom, P.J.M., 2003. Modelling 3D spatial objects in a GeoDBMS using a 3D primitive, *AGILE 2003*, Lyon, France.
- Gröger, G., Reuter, M. & Plümer, L., 2004. Representation of a 3-D City Model in Spatial Object-Relational Databases. In: *Intern. Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. 34, Part B4 (Proc. of the XXth ISPRS Congress, Istanbul, Turkey).
- Stoter, J.E. & Zlatanova, S., 2003. Visualising and editing of 3D objects organised in a DBMS *EUROSDR workshop: Rendering and visualisation*, Enschede, The Netherlands.
- OGC, 2005a. Web Feature Service Implementation Specification, Version 1.1.0, OGC document 04-094
- OGC, 2005b. Filter Encoding Implementation Specification, Version 1.1.0, OGC document 04-095
- OGC, 2004a. Geography Markup language (GML) Implementation Specification, Version 3.1.0, OGC document 03-105r1
- OGC, 2004b. Web Map Service, Version 1.3, OGC document 04-024
- OGC, 2002a, Geography Markup Language, OGC document 02-023r4
- OGC, 2002b, Web Feature Service Implementation Specification, Version 1.0.0, OGC document 02-058
- OGC, 2001, Abstract Specification Topic 1 – Feature Geometry (same as ISO19107), OGC document 01-101
- OGC, 1999, Simple Features - SQL, OGC document 99-049