Free and Open Source Software for Geospatial (FOSS4G) Conference Proceedings

Volume 17 *Boston, USA* Article 6

2017

Development of an extension of GeoServer for handling 3D spatial data

Hyung-Gyu Ryoo
Pusan National University

Soojin Kim
Pusan National University

Joon-Seok Kim
Pusan National University

Ki-Joune Li
Pusan National University

Development of an extension of GeoServer for handling 3D spatial data

Optional Cover Page Acknowledgements
This research was supported by a grant (14NSIP-B080144-01) from National Land Space Information Research Program funded by Ministry of Land, Infrastructure and Transport of Korean government and BK21PLUS, Creative Human Resource Development Program for IT Convergence.

Development of an extension of GeoServer for handling 3D spatial data

Hyung-Gyu Ryoo[a,*], Soojin Kim[a], Joon-Seok Kim[a], Ki-Joune Li[a]

[a] *Department of Computer Science and Engineering, Pusan National University*

Abstract: Recently, several open source software tools such as CesiumJS and iTowns have been developed for dealing with 3-dimensional spatial data. These tools mainly focus on visualization of 3D spatial data based on WebGL. An open-sourced server capable of storing, sharing and querying 3D spatial data has not yet been developed. GeoServer, one of the representative open source spatial data servers, provides many powerful features. In particular, it supports connecting to and publishing spatial data from a variety of data sources. GeoServer also supports Web Feature Service (WFS), which is a standard protocol established by the Open Geospatial Consortium to request geospatial feature data. However, GeoServer provides functions only for two-dimensional geometry, so it provides few functions for handling 3D spatial data. Because JTS Topology Suite, which is an important component of GeoServer, does not support 3D spatial operations, it also does not support solid geometries. In this paper, we discuss extension modules of GeoServer that we have implemented to handle 3D spatial data. First, instead of JTS, our modules adopted a geometry model based on the ISO 19107 standard and support 3D spatial operations from the Simple Feature CGAL library. Based on this geometry model, we have implemented new internal data structures that represent spatial information from the Feature interface in GeoServer. Second, we also extended the DataStore module to handle and store 3D spatial information for several data sources such as GeoJSON, GML and PostGIS. Finally, we extended the WFS module to share 3D spatial data via GeoServer.

*Corresponding author
Email address: hgryoo@pnu.edu (Hyung-Gyu Ryoo)

1. Introduction

Recently, several open source software tools dealing with 3D spatial data such as CesiumJS (AGI 2017) and iTown (Oslandia 2016) have been actively developed. Most of them focus mainly on visualizing 3D spatial data in a client environment. However, a universal spatial data server for providing 3D spatial data to such a visualization tool has not yet been developed, and therefore it is hard for the client to service 3D spatial information. In order to provide geospatial web services to the users, it is necessary to develop a spatial data server able to share and query 3D spatial data.

GeoServer (GeoServer 2001) is a representative open source spatial data servers that provide a variety of spatial data sources that conform to open standards. It supports file formats for handling spatial data such as a GeoJSON (Butler et al. 2008) and a shapefile (ESRI 1998) and geospatial database management systems (DBMS) such as PostGIS (PostGIS 2001) and Oracle Spatial and Graph (Oracle 2016). GeoServer allows users to add new data stores as a form of plugin. Based on these various spatial data sources, GeoServer is an important component in geospatial web services that provides functions to share and query spatial data by implementing open geospatial web services established by Open Geospatial Consortium (OGC) such as Web Map Service (Service 2006), Web Feature Service (WFS/FES 2005) and Web Processing Service (Service 2015).

Although it is powerful as a spatial data server for general purposes, GeoServer has limitations in handling 3D spatial data. For instance, it is impossible to store spatial information with solids, which are bounded by closed surfaces, into data stores as well as to load that from data sources. It is possible for GeoServer to handle and store points, curves, and surfaces with three-dimensional coordinates. However, GeoServer barely supports 3D spatial queries because it only performs spatial query processing that ignores z coordinates.

The limitations originate from data structures representing spatial information in GeoServer. JTS Topology Suite (Java Topology Suite), a fundamental library written in Java, (Tech 2000) is a core component of GeoServer for geometries. On the top of JTS, GeoServer provides a number of spatial operations and utility functions for various 2D geometries. In the most implementations, it is assumed that geometry attributes of a feature are represented as a geometry of JTS library. In other words, GeoServer is deeply coupled with JTS. Therefore, it is necessary to replace JTS for GeoServer to provide fully functional features in 3D.

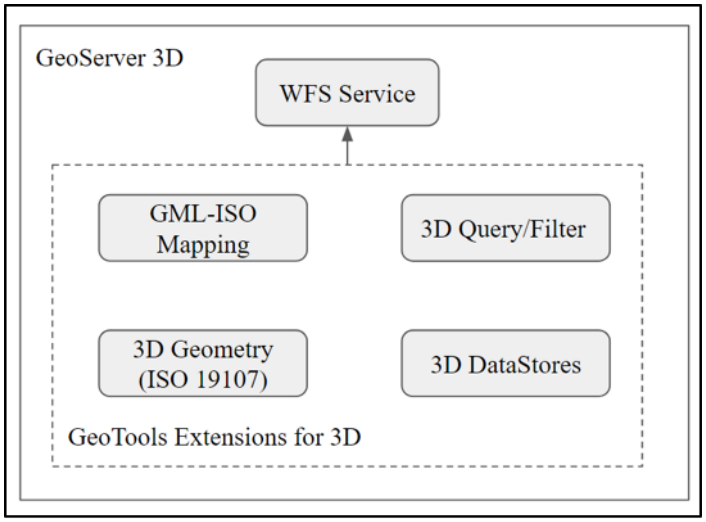

Figure 1: Overview of our extension of GeoServer to support 3D spatial data

In this paper, we introduce our extension of GeoServer to support 3D spatial data as shown in Figure 1 (STEMLab 2015, STEMLab 2016). Our contribution to GeoServer and open source communities is summarized as follows:

- We have taken the initiative to implement a fully functional open-sourced data server for 3D spatial data.
- We carefully examined GeoServer from bottom to top to determine how to add a 3D extension.
- We identified requirements and issues for a 3D spatial data server based on GeoServer.
- We built a feasible full stack composed of open source software.

This paper is organized as follows. We survey related work and present requirements for a 3D spatial data server in Section 2. In Section 3, we introduce implementations of data structures by using geometric models based on ISO 19107 spatial schema (ISO/TC211 2003) to store 3D geometries in GeoServer. Section 4 describes the data store that can store and query 3D geometries based on the data structure implemented in Section 3. In Section 5, we describe support for the Web Feature Service protocol to share 3D spatial information. In Section 6, we conclude the paper and discuss current limitations and future work.

2. Related Work

There are several DBMSs that can store 3D spatial data and perform query processing on spatial information. Oracle Spatial and Graph, a representative commercial database, supports most spatial operations and 3D spatial data types, including a solid geometry. PostGIS, a widely used open source spatial database, partially supports 3D spatial data types and operations. Although spatial DBMSs are fundamental to manipulate spatial data, common interfaces and tools have been demanded to satisfy various geospatial applications. For instance, users may need to overlay and manipulate multiple layers from various data sources without concerns about different implementations on each data store.

GeoTools (GeoTools 1998) is an open source library that meets these requirements. It provides excellent abstraction for multiple data sources. GeoServer is an implementation of a geospatial data server based on GeoTools. It allows users to simply load/store layers from/to various databases and file-based data sources, hiding details of how to handle them. A readable/writable source is called a data store. Thanks to many contributors, GeoServer supports a variety of data stores. Also, it lets users to add data stores via the Web. The reason is that GeoServer has a common interface, called Feature, between data stores and a module that provides the Web Feature Service. Thus, it acts as a bridge between heterogeneous data stores and the user interface, to exchange the spatial information.

A Feature instance can have spatial and non-spatial properties. For geometric properties, GeoServer employs JTS, a widely used open source library written in Java. Since it is robust, there exist ports of JTS in different languages such as Geometry Engine, Open Source (GEOS 2003). JTS implements the OpenGIS Simple Features Specification, which is designed to handle 2D geometry types such as points, curves and surfaces and operations with them. Unfortunately, most GeoServer modules including Feature are deeply coupled with the JTS. In order to provide 3D spatial operations in GeoServer, it is necessary to replace the geometry properties of Feature and relevant modules. If GeoServer itself has 3D spatial operation capabilities, clients can store and query 3D spatial data to any data stores.

There exists a 3D geometry model in GeoTools as an unsupported module, gt-geometry,

and it is based on ISO 19107 spatial schema. Due to the complexity of 3D spatial operations, it has had rare application. Therefore, there remains a problem of how to provide 3D spatial operations when using the ISO 19017-based geometric model as the value of the geometry property of Feature.

For implementing 3D spatial operations, there may be two options: 1) to develop or port every function from scratch and 2) to reuse or utilize existing open source libraries. Although it could be optimized for GeoServer, the first option requires a lot of effort to make the functions robust and efficient and the operations should be maintained separately. In this paper, we focus on the second method to take advantage of open source libraries that provide enough functionality.

Computational Geometry ALgorithms (CGAL) is an open source library that provides a variety of robust computational geometry algorithms in 3D spaces as well as 2D (Fabri and Pion 2009). Since it is designed for general purposes, the geometry model and operations in CGAL are different from those in GIS domains. There is an existing open source project to that fills the gap called Simple Feature CGAL (Oslandia1 2016). This library is implemented using CGAL and Boost. Simple Feature CGAL (SFCGAL) implements a geometric model and application programming interfaces (APIs) defined in the ISO 19107 spatial schema and the OGC Simple Features Access 1.2 standard (Features 2011). SFCGAL is robust and yields highly precise results. There is a trade-off between precision and performance when using CGAL.

From the survey of existing open source projects, we identify requirements of a 3D spatial data server as follows:

- Support data structures to store 3D geometries
- Support connections to data stores able to store 3D spatial information
- Support functions to process queries on 3D objects
- Provide standard web services for sharing 3D spatial information

3. Implementation of Data Structure for 3D Geometry Model

GeoServer abstracts all spatial information into a class called Feature and the geometry attributes of spatial information are represented as GeometryAttribute. The GeometryAttribute interface has the actual geometry object implemented in Java. In most implementations, it is assumed that geometry attributes of a feature are represented as a geometry of the JTS library. The problem is that JTS geometry objects are represented not as a Java interface but as a Java class. This prohibits other implementation, making it difficult to extend the functionality. Therefore, it is necessary to replace the JTS and relevant modules with other libraries.

First, an interface-based structure is required for extensibility. Currently, an unsupported module in GeoServer defines and partially implements an ISO 19107 spatial schema-based interface. It has the Solid interface but no class implementing it. Also, the module does not support 3D spatial operations such as Contains and Intersects for two geometries with Z coordinates. Few functions such as finding the distance between geometries are implemented. Therefore, we added Shell class and Solid class to be compatible with ISO 19107.

In order to support 3D spatial operations, we have connected the SFCGAL library to the existing interface of the ISO 19107 spatial schema. Figure 2 shows its structure. Since SFCGAL is a library written in C ++, a bridge between native C++ and Java is required to invoke

SFCGAL functions. Among the libraries that support this, we adopted JavaCPP, which is an open source tool and relatively easy to interface. JavaCPP can generate wrapper classes corresponding to SFCGAL. The structure of SFCGAL geometry is the same as Simple Feature Geometry.

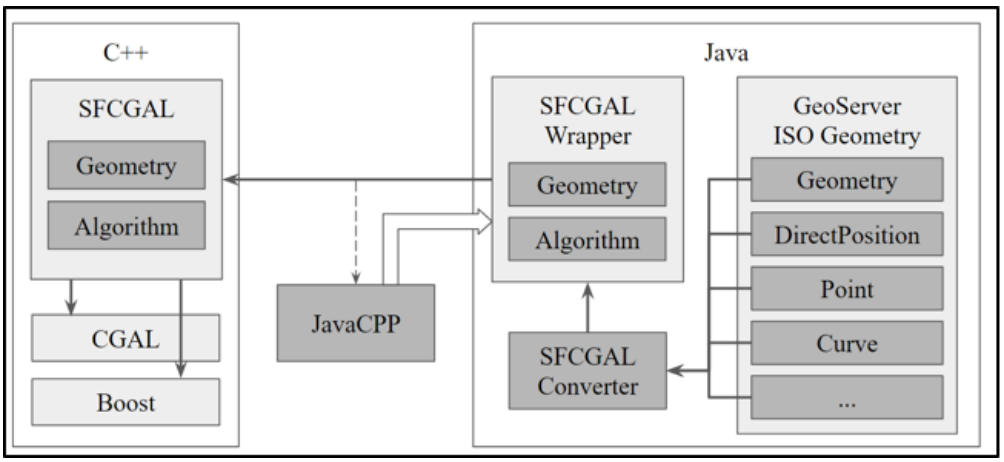

Figure 2: Connecting SFCGAL to support 3D operations

In order for communication between SFCGAL and GeoServer, conversion process is necessary because of a gap between the SFCGAL Wrapper classes and the GeoServer ISO Geometry interface (see Figure 2). In our extension, we implemented a converter that converts from ISO geometry to SFCGAL and vice versa. Then, SFCGAL Wrapper objects invoke the corresponding native methods of SFCGAL when methods of ISO geometry are called by GeoServer. Once processing in SFCGAL is completed, the results are returned as wrapper objects. The mapping relation between Simple Feature Access, SFCGAL and ISO geometry is defined as Table 1.

Simple Feature Access	**SFCGAL**	**ISO 19107**
Coordinate	Coordinate	DirectPosition
Point	Point	Point
LineString	LineString	Curve
Line		
LinearRing		Ring
Polygon	Polygon	Surface
Triangle	Triangle	
PolyhedralSurface	PolyhedralSurface	PolyhedralSurface
	TriangulatedSurface	TriangulatedSurface
TIN	TIN	TIN
-	Solid	Solid
MultiPoint	MultiPoint	MultiPoint
MultiLineString	MultiLineString	MultiCurve
MultiPolygon	MultiPolygon	MultiSurface
-	MultiSolid	MultiSolid
GeometryCollection	GeometryCollection	MultiPrimitive

Table 1: Mapping relation between Simple Feature Access, SFCGAL and ISO geometry

4. A Data Store for 3D Spatial Information

In order to support a general purpose 3D spatial information data server, it is necessary to provide various data stores capable of storing 3D spatial information. In GeoServer, data stores can be divided into two types: 1) a file data store and 2) a database data store. There are file data stores supporting file formats such as Shapefile, GeoJSON, and database data stores manipulating databases such as Oracle, PostGIS, and MySQL.

This section describes two data stores for GeoJSON, a spatial file format with a relatively simple structure, and PostGIS, a widely used open source spatial database. For PostGIS, if you use the SFCGAL extension, you can use geometry with Z coordinates and 3D operations. In GeoServer, each concrete store can be extended to the plug-in by implementing the necessary interfaces to the store. Likewise, new stores can be extended based on the methods described in this section.

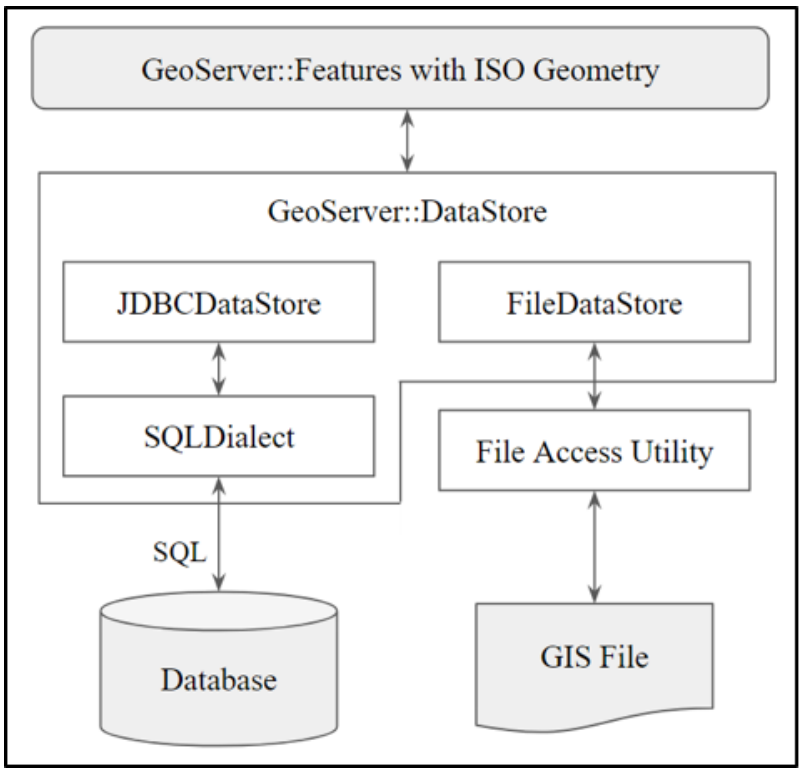

Figure 3: DataStore architecture in GeoServer

GeoServer provides a function that supports data store by implementing an interface called DataStore as shown in Figure 3. In GeoServer, DataStore is an abstraction of persistent object stores containing instances of Feature. Any implementations of DataStore are responsible for importing/exporting features from/to their physical stores. File data stores should be an implementation of the FileDataStore interface to support their file formats. Although they have different capabilities, DBMSs with a Java Database Connectivity (JDBC) driver have common functionality. For the common parts, GeoServer provides an abstract class called JDBCDataStore. Each database data store needs to extend SQLDialect for its own spatial types and functions.

4.1. Implementation of DataStore

GeoJSON is an open standard format that supports simple features, based on JavaScript Object Notation (JSON), which is mainly used for data exchange in the web environment. GeoJSON is a lightweight data format, compared to rich formats based on XML such as GML. It supports simple geometry types: Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, and GeometryCollection. In order to take advantages of GeoJSON, we define an extended GeoJSON for solids by adding Solid. Structure of Solid's coordinates is similar to MultiPolygon and Figure 4 is an example of cube in extended GeoJSON.

```
{ "type": "FeatureCollection",
  "features": [
  { "type": "Feature",
  "geometry":
    { "type": "Solid",
    "coordinates": [[
        [[0.0,0.0,0.0],[0.0,0.0,1.0],[0.0,1.0,1.0],[0.0,1.0,0.0],[0.0,0.0,0.0]],
        [[0.0,0.0,0.0],[0.0,1.0,0.0],[1.0,1.0,0.0],[1.0,0.0,0.0],[0.0,0.0,0.0]],
        [[0.0,0.0,0.0],[1.0,0.0,0.0],[1.0,0.0,1.0],[0.0,0.0,1.0],[0.0,0.0,0.0]],
        [[1.0,1.0,0.0],[1.0,1.0,1.0],[1.0,0.0,1.0],[1.0,0.0,0.0],[1.0,1.0,0.0]],
        [[0.0,1.0,0.0],[0.0,1.0,1.0],[1.0,1.0,1.0],[1.0,1.0,0.0],[0.0,1.0,0.0]],
        [[0.0,0.0,1.0],[1.0,0.0,1.0],[1.0,1.0,1.0],[0.0,1.0,1.0],[0.0,0.0,1.0]]
     ]]},
     "properties": {
      "name": "Room A",
      "floor": 1
      }]}}
```

Figure 4: Example of Solid in the extended GeoJSON

We implemented a new FileDataStore to handle the extended GeoJSON. While they parse/interpret files in specific formats, some file data stores without schema definitions including GeoJSON define types of features and instantiate them. For consistency, geometry attributes of features in the extended GeoJSON are interpreted as ISO geometry types in GeoServer according to Table 1. This enables users to import/export features with a solid geometry from/to the extended GeoJSON.

In a database data store, a feature type is created from table schema because records in a conventional DBMS such as PostGIS are shaped by schema. Because PostGIS provides geometric types based on Simple Feature Access, geometry attributes of features can be represented as ISO geometry according to Table 1.

PostGIS supports 3D geometry types and operations if and only if it has the SFCGAL extension. This extension includes additional types such as PolyhedralSurface and Solid. It also provides various operations that take into account Z coordinates. To maximize utilization of PostGIS capabilities, the data store is implemented using the SFCGAL extension. As a result, this reduces gaps between two data models so we could avoid a strained implementation by converting solids into alternative types such as MultiSurface. On the SFCGAL installation, the data store enables users to import/export features with 3D geometries from/to PostGIS.

4.2. 3D Spatial Query Processing

This section describes how to process spatial queries for each data store. GeoServer defines the query conditions based on the OGC Filter Encoding Specification (WFS-FE 2005). Therefore, queries to the DataStore are implemented to pass the filter condition and process it. Figure 5 shows the overall flow to perform query processing with 3D spatial filters.

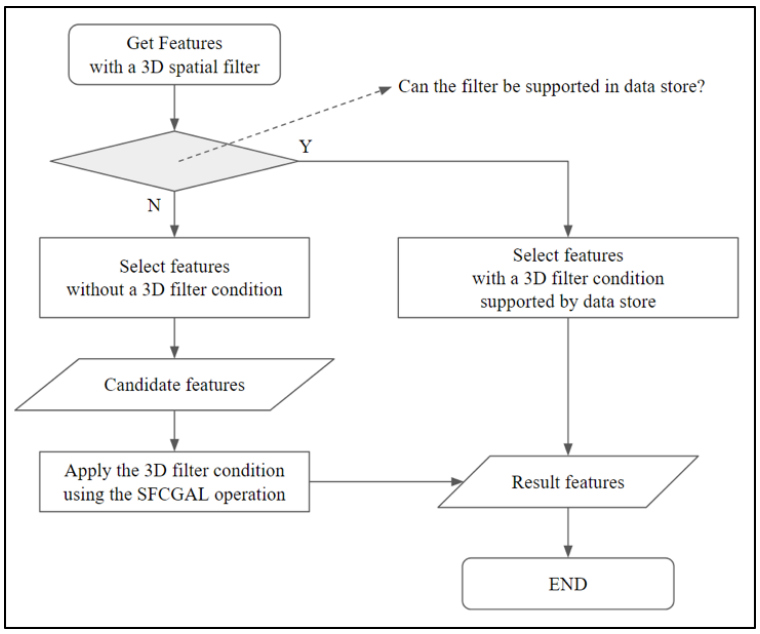

Figure 5: Flow diagram of handling a query with 3D spatial filter in data store

Basically, 3D spatial queries including 3D spatial filters can be handled without assistance of a data store in our extension of GeoServer. A spatial filter such as Contains and Intersects, however, can considerably reduce the number of candidates if a data store supports the filter. As file stores have no spatial operation capability, any 3D spatial filter will be processed using the SFCGAL. On the other hand, database data stores can apply some spatial filters, depending on their capabilities.

5. Web Feature Service Extension for 3D Queries

Web Feature Service (WFS), one of OGC's standard web protocols, is an interface for providing spatial data in vector format. This allows performing data manipulation operations including retrieving, creating, deleting, and updating features across the web. In this paper, we implemented the functionality of Web Feature Service 1.1 to access, store, query, and share 3D spatial information through the web environment. WFS 1.1 supports the GML 3.1.1 used in HTTP requests from and responses to the client. Since its geometry model is an implementation of ISO 19107 in XML, the GML 3.1.1 contains solids.

However, GeoServer does not give a correct response when the WFS client makes a request to the store containing features with solids. The reason is that a GML geometry is interpreted as a geometry of JTS while parsing GML instances. Therefore, in order to work properly, we implemented a module mapping each GML geometry not to a geometry of JTS but to a geometry defined in ISO 19107.

WFS requests are handled according to the flow as shown in Figure 6. WFS requests sent by a WFS client to GeoServer include an XML-formatted Filter and GML. The WFS function of GeoServer converts GML, the geometry information described in the request statement, to the geometry type used by GeoServer. For example, if you have a 3D region as a query condition, the 3D region in GML is represented by gml:Solid, which is converted to the solid type of the ISO geometry. Also, the features that result from the query performed in the data store will contain a 3D geometry type. This 3D geometry type is expressed in ISO geometry. Therefore, we converted the ISO geometry of each feature to GML to create a WFS response.

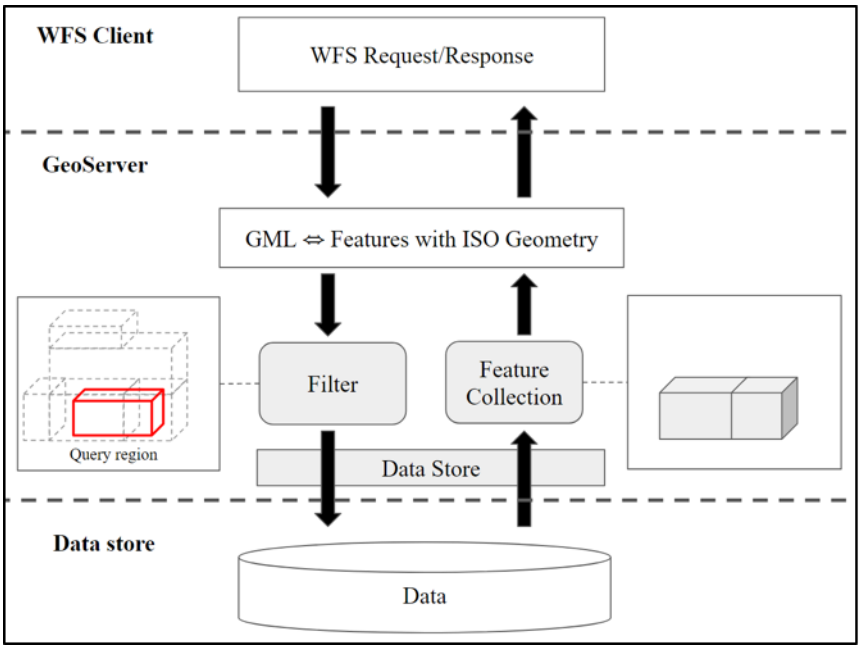

Figure 6: Flow of converting between GML and Feature with ISO Geometry

As a result, Figure 7 visualizes the result of a GetFeature query by implementing a simple WFS client using CesiumJS. The example data of 3D objects is composed of rooms and corridors in a shopping mall in Seoul. The number of rooms is 4893 and each room is made of solid geometry. In Figure 8, the client sends a query using the WFS 1.1 GetFeature to find the overlapping room by specifying a query area with a 3D space represented by the red portion. As a result, the request is executed in the 3D store where the 3D objects are stored, and each room corresponding to the query result is returned as the Figure 9, and the corresponding geometry is displayed in yellow in the WFS client.

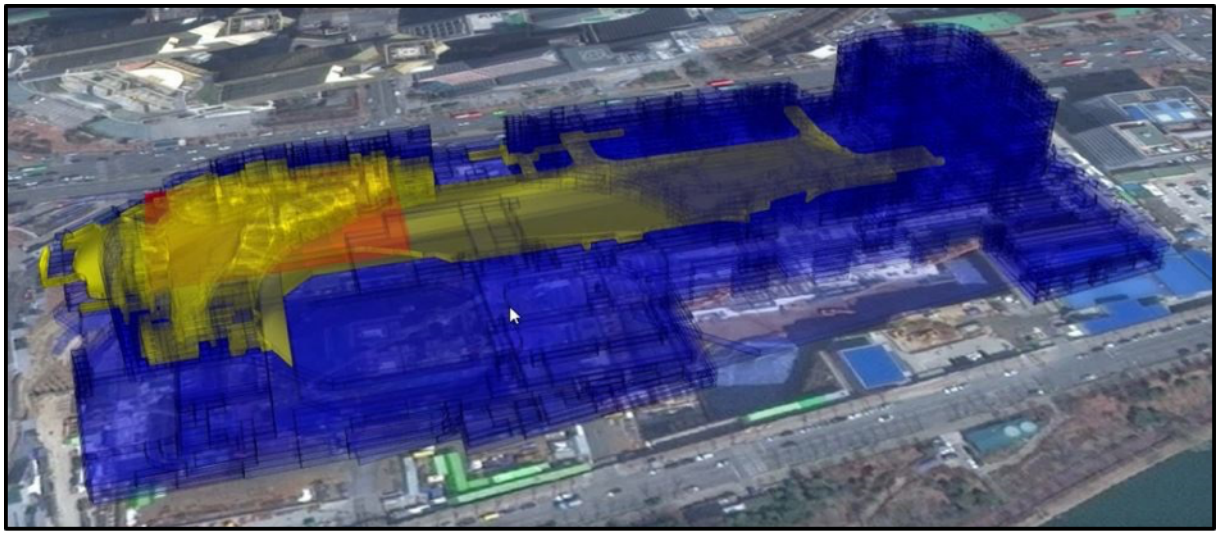

Figure 7: A result of GetFeature operation selecting features with Solid geometry

```
<?xml version="1.0" ?>
<wfs:GetFeature
        service="WFS"
        version="1.1.0"
        outputFormat="text/xml; subtype=gml/3.1.1"
        xmlns:stem="http://stemlab.pnu.edu"
        xmlns:wfs="http://www.opengis.net/wfs"
        xmlns:ogc="http://www.opengis.net/ogc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.1.0/WFS.xsd">
        <wfs:Query typeName="stem:solid3d_sfcgal">
          <ogc:Filter>
            <ogc:Intersects>
              <ogc:PropertyName>stem:geom</ogc:PropertyName>
              <gml:Envelope srsName="EPSG:4329">
                <gml:lowerCorner>127.10156397 37.46185285 -100</gml:lowerCorner>
                <gml:upperCorner>127.10226404 37.46227915 100</gml:upperCorner>
              </gml:Envelope>
            </ogc:Intersects>
          </ogc:Filter>
        </wfs:Query>
</wfs:GetFeature>
```

Figure 8: Example of the GetFeature request with Intersects query

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:FeatureCollection
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:wfs="http://www.opengis.net/wfs"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:ows="http://www.opengis.net/ows"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:stem="http://stemlab.pnu.edu"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  numberOfFeatures="309" timeStamp="2017-05-21T13:25:58.214Z"
  xsi:schemaLocation="http://www.opengis.net/wfs
  http://127.0.0.1:8883/geoserver/schemas/wfs/1.1.0/wfs.xsd http://stemlab.pnu.edu
  http://127.0.0.1:8883/geoserver/wfs
  service=WFS&amp;version=1.1.0&amp;
  request=DescribeFeatureType&amp;typeName=stem%3Asolid3d_sfcgal"
  ...
  <gml:featureMembers>
  <stem:solid3d_sfcgal gml:id="solid3d_sfcgal.fid--4a5ecf22_15c2b2204b3_-7ff7">
  <gml:boundedBy>
        <gml:Envelope srsName="urn:x-ogc:def:crs:EPSG:4329" srsDimension="3">
                <gml:lowerCorner>127.10227883229379 37.51336579829316 0.0</gml:lowerCorner>
                <gml:upperCorner>127.10479641589909 37.514514506176944 3.0</gml:upperCorner>
  </gml:Envelope></gml:boundedBy>
  <stem:fid>C835</stem:fid>
  <stem:geom>
        <gml:Solid srsName="urn:x-ogc:def:crs:EPSG:4329" srsDimension="3">
                <gml:exterior><gml:CompositeSurface><gml:surfaceMember>
                <gml:Polygon><gml:exterior><gml:LinearRing>
                <gml:posList>
        127.10233736459824 37.51337631034256 3.0 127.10231000709786 37.51342579228809 3.0
        127.10227883229379 37.51341528023869 3.0 127.10230618979418 37.51336579829316 3.0
        127.10233736459824 37.51337631034256 3.0</gml:posList>
                </gml:LinearRing></gml:exterior></gml:Polygon>
                </gml:surfaceMember>
                        ...
                </gml:CompositeSurface>
                </gml:exterior>
        </gml:Solid>
        </stem:geom>
        </stem:solid3d_sfcgal>
        ...
</gml:featureMembers>
</wfs:FeatureCollection>
```

Figure 9: Example of the GetFeature response containing features with Solid geometry

6. Conclusion

GeoServer has been widely used as an open source spatial data server. However, GeoServer does not support 3D spatial data handling because it has been implemented based on Simple Feature Access Specification. Therefore, this study used the ISO 19107 geometric model which can support 3D spatial geometry and computation. Based on this, we extended GeoServer to support the data stores that can store 3D spatial information. We also extended GeoServer's WFS interface to share and query the spatial information stored in the 3D data store across the web. The version of GeoServer that is developed from this study can be used in fields requiring 3D spatial information and queries. For example, if there is a complicated pipe in a building, it would be possible to query the rooms where the pipe passes.

However, there are still items to improve. Currently, as a 3D spatial server, GeoServer does not provide all kinds of 3D data types and operations for general purposes. For example, the SFCGAL library does not support curve geometries like arcs or geodesics, so we cannot

support all geometries in ISO 19107. Thus, it is an interesting topic to implement a function to provide curved geometry. It will be practical if our next version can support WFS 2.0, the latest version of Web Feature Service, which provides rich functionality by supporting temporal queries and join queries through GetFeature requests. It also supports complex filter queries that can be stored in WFS server. Furthermore, 3D Portrayal IE (Portrayal 2015), a web-based interface for supporting 3D scene rendering at the server side, is being actively discussed as a standard candidate for OGC. It is necessary to provide a server capable of handling 3D spatial information to support such a standard.

References

AGI, 2017. Cesium webgl virtual globe and map engine. Online, online; accessed May 23, 2017.
URL http://cesiumjs.org

Butler, H., Daly, M., Doyle, A., Gillies, S., Schaub, T., Schmidt, C., 2008. The geojson format specification. Rapport technique 67.

ESRI, 1998. Shapefile technical description. An ESRI White Paper 7855.

Fabri, A., Pion, S., 2009. Cgal: The computational geometry algorithms library. Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems, 538–539.

Features, O. S., 2011. Opengis implementation specification for geographic information - simple feature access - part 1: Common architecture. Online, online; accessed May 23, 2017.
URL http://www.opengeospatial.org/standards/sfa

GEOS, 2003. Geos geometry engine, open source. Online, online; accessed May 23, 2017.
URL https://trac.osgeo.org/geos

GeoServer, 2001. Geoserver. Online, online; accessed May 23, 2017.
URL http://geoserver.org

GeoTools, 1998. Geotools the open source java gis toolkit. Online, online; accessed May 23, 2017.
URL http://geotools.org

ISO/TC211, 2003. Geographic information - spatial schema, iso. An ESRI White Paper 19107:2003.

Oracle, 2016. Oracle spatial and graph 12c release 2. Online, online; accessed May 23, 2017.
URL http://www.oracle.com/technetwork/database-options/spatialandgraph/overview/spatialandgraph-1707409.html

Oslandia, 2016. Oslandia itowns. Online, online; accessed May 23, 2017.
URL http://www.itowns-project.org

Oslandia1, 2016. Simple feature cgal (sfcgal). Online, online; accessed May 23, 2017.
URL http://www.sfcgal.org/

Portrayal, O. D., 2015. 3d portrayal service candidate standard. Online, online; accessed May 23, 2017.
URL http://www.opengeospatial.org/projects/initiatives/3dpie

PostGIS, 2001. Postgis. Online, online; accessed May 23, 2017.
URL http://www.postgis.net

Service, O. W. M., 2006. Opengis web map service (wms) implementation specification, 06-042. Online, online; accessed May 23, 2017.
URL http://opengeospatial.org/standards/wms

Service, O. W. P., 2015. Wps 2.0 interface standard, 04-065. Online, online; accessed May 23, 2017.
URL http://opengeospatial.org/standards/wps

STEMLab, 2015. Geotools 3d extension. Online, online; accessed May 23, 2017.
URL https://github.com/STEMLab/geotools-3d-extension/

STEMLab, 2016. Geotools 3d extension. Online, online; accessed May 23, 2017.
URL https://github.com/STEMLab/geoserver-3d-extension/

Tech, L., 2000. Jts topology suite (java topology suite). Online, online; accessed May 23, 2017.
URL https://www.locationtech.org/projects/technology.jts

WFS-FE, O., 2005. Opengis filter encoding implementation specification, 04-095. Online, online; accessed May 23, 2017.
URL http://www.opengeospatial.org/standards/filter

WFS/FES, O., 2005. Opengis web feature service (wfs) implementation specification, 04-094. Online, online; accessed May 23, 2017.
URL http://opengeospatial.org/standards/wfs