INFACTORY : A RESTFUL API SERVER FOR EASILY CREATING INDOORGML

Hyemi Jeong, Hyung-gyu Ryoo, Ki-Joune Li*

Dept. of Computer Science&Engineering, Pusan National University, Kumjeong-Gu, 46241, Pusan, South Korea
- (hyemi.jeong, hgryoo, lik)@pnu.kr

Commission IV, WG IV/4

KEY WORDS: Data construction, Indoor space, OGC, IndoorGML, RESTful API, PostGIS

ABSTRACT:

Recently, services and systems that deal with indoor spatial information are increasing. Each service or system adopts a data model that can store necessary indoor space data according to its purpose. However, since the content of indoor spatial information that can be expressed by each data model is differ and limited, it is necessary to exchange information between the systems in order to use rich indoor spatial data. OGC has published IndoorGML as the standard for exchange of indoor spatial information data between systems. To use IndoorGML as an exchange format, the software which supports IndoorGML construction is fundamental. But there are several limitations in the previous IndoorGML data editing tools. There is no editing tool that can generate all the features which are defined by IndoorGML. If users want to generate IndoorGML data, they need to consider the requirements of the IndoorGML. In this study, we implemented InFactory, which is a IndoorGML generation tool based on RESTful API supporting users to easily construct IndoorGML data. Users can easily create IndoorGML without knowledge on the schema and requirements of IndoorGML using InFactory. In addition, developers on IndoorGML data construction tools such as GUI editors do not have to implement duplicated IndoorGML generation program for their systems. Using Java API that supports CRUD on IndoorGML data, users can also deal with IndoorGML data in their applications.

1. INTRODUCTION

Nowadays, there is an increasing number of services dealing with indoor space. For example, indoor space services include indoor maps from Google Maps[1], and indoor maps from OpenStreetMap (OSM)[2]. The indoor spatial information service adopts the data model for the representation and storage of the necessary data according to the purpose of the service. In this situation, an exchange format is required for smooth interface between various indoor spatial information services.

The Open Geospatial Consortium (OGC)(Open Geospatial Consortium, 1994) has published IndoorGML(Open Geospatial Consortium, 2014) as the standard exchange format for indoor spatial information exchange. IndoorGML is designed to represent and exchange indoor spatial information among indoor space data services. The existing indoor spatial information standards such as CityGML(Open Geospatial Consortium, 2008), KML(Open Geospatial Consortium, n.d.), and IFC(Liebich, 2013) can represent indoor space as the building construction elements, but can not express the information necessary for route guidance(Li et al., 2015). It is the reason why we need IndoorGML. However, the environment supporting IndoorGML is still not enough.

In particular, this paper discusses in detail the environments that support the generation of IndoorGML data. The generation of IndoorGML is difficult for the following reasons. First, IndoorGML is expressed in Extensible Markup Language(XML), and we need encoding or decoding according to XML schema. Second, it is not easy to express relationships such as associations and hierarchical structures that IndoorGML includes. When creating the IndoorGML data, it is necessary to consider the schema or requirements of the IndoorGML. To solve these difficulties, users need a tool to easily generate IndoorGML data.

In this background, we implemented InFactory[3] to handle IndoorGML data. InFactory is a server that supports the generation of IndoorGML data. It provides the following functions.

- The indoor spatial information of the users or the data created by the editing tool can be converted into IndoorGML and stored in the IndoorGML XML format.
- Provides a Java library that generates and edits IndoorGML data so that user can use the Java library to construct their own indoor spatial services.
- Users avoid repeat implementations such as a conversion tool of IndoorGML for each service by using InFactory as shown in Figure 1.

Before InFactory was implemented, each system needs to make its own IndoorGML generator or converter which fits on their program. However, if users apply InFactory, they can easily generate IndoorGML data by implementing only the client program that communicates with the Representational State Transfer(RESTful)(Richardson, Leonard and Ruby, Sam, 2008) server of InFactory.

This paper has the following structure. Section 2 introduces related research. Section 3 explains the basic concepts of IndoorGML to understand this paper. Section 4 describes the functions which are provided by InFactory. Section 5 describes the structure of InFactory. Section 6 describes how IndoorGML features are represented in InFactory. Section 7 explains the design of the schema of PostGIS which is used in InFactory. Section 8

*Corresponding author
[1]https://www.google.com/maps/about/partners/indoormaps/
[2]https://www.openstreetmap.org/
[3]https://github.com/STEMLab/InFactory

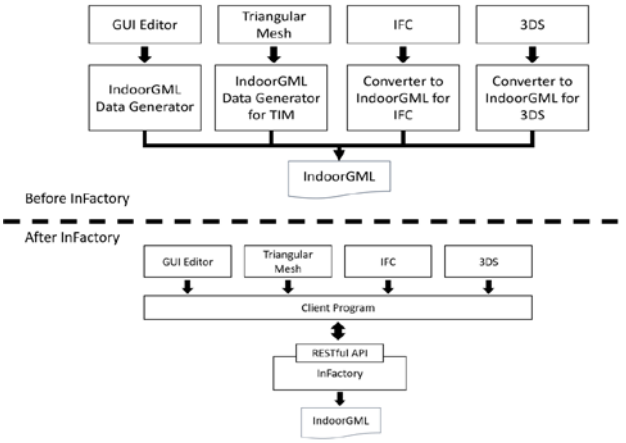

Figure 1. The systems which can use InFactory

introduces the use cases of InFactory and Section 9 concludes the paper with future works of this paper.

2. RELATED STUDIES

In order for users to take full adventage of IndoorGML, the Eco-System for IndoorGML should be implemented as shown in Figure 2. The eco-system of IndoorGML consists of several steps; collecting indoor spatial raw data, generating IndoorGML data, validating the generated data, sharing the data, and building an application or service using IndoorGML data. In this paper, we will cover the step to construct IndoorGML.

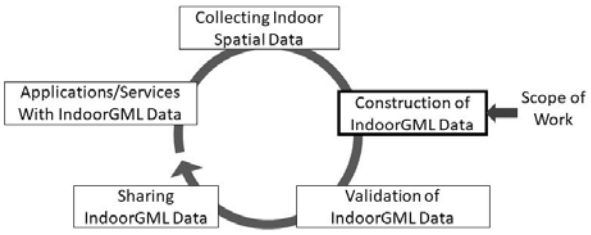

Figure 2. The Eco-System of IndoorGML

2.1 Previous researches

Before this study, there are i-locate[4] and JInEditor[5] which are editing tools that can generate IndoorGML. i-locate is an European project to develop an open geoportal that generates and provides indoor spatial information based on Location Based Services (LBS). i-locate supports the ability to edit IndoorGML's navigation network on OpenStreetMap (OSM). However, i-locate can only create network information and can not edit the geometry of interior space which can be expressed by IndoorGML.

JinEdit is a Java-based graphical editor for building IndoorGML. The user constructs the IndoorGML information by directly drawing the geometric data of IndoorGML, the positional relationship between the spaces, and the guidance network through the graphic UI. However, it is difficult to handle due to the lack of user-friendly interface and it only supports a subset of the features defined in IndoorGML.Table 1 shows a summary on the previous IndoorGML editing tools.

[4]http://www.i-locate.eu/
[5]https://github.com/STEMLab/JIneditor

Type of functionality	The kinds of editor		
	i-locate	JInEditor	InFactory
Cellular Space	x	o	o
Topology	o	o	o
Geometry	x	o	o
Multi-Layered Space Model	x	x	o

Table 1. The comparison between IndoorGML editing tools

2.2 Motivation of this study

The editing tools mentioned above are limited to express some parts of IndoorGML elements or are difficult to use due to badly designed user-interface. In addition, since the constraints of IndoorGML and the encoding of XML schema are difficult, it is difficult for user to generate IndoorGML by themselves. Therefore, we need a tool that users can use to generate IndoorGML more easily.

Therefore, this study tries to support the following factors through InFactory. First, development of a tool that includes all the elements of IndoorGML supporting the latest schema of IndoorGML. Second, we develop a tool that allows users to easily generate IndoorGML data without knowing the technical details related to IndoorGML. Third, we want to provide IndoorGML creation and management functions in the form of RESTful Server which can be flexibly used in various IndoorGML building environments. Table 1 compares the capabilities that InFactory can support with other editing tools. i-locate and JInEditor support only some of the functionality needed to generate IndoorGML data, while InFactory supports all of them.

3. INTRODUCTION OF INDOORGML

In order to understand the work presented in this paper, we must understand the concept of IndoorGML(Kang and Li, 2017). This chapter briefly describes the concept of IndoorGML. The core concepts of IndoorGML are based on cellular space model, topology, geometry and multi-layered space model. Next we also introduce the data model that deals the main concepts of IndoorGML. Finally we explain the XML schema of IndoorGML.

3.1 The basic concepts of IndoorGML

- Cellular Space Model
  IndoorGML expresses indoor space as a set of unit spaces. This approach is called the cellular space model. In this model, unit space can be determined according to the role or meaning of space. For example, in the interior space, the unit space can be defined according to the usage such as the meeting room, the hall, and the porch.
- Topology
  In IndoorGML, the topological relation among unit spaces is represented by a graph using Poincare duality. A unit space corresponds to a node of a graph, and a connection or adjacency unit spaces corresponds to an edge of the graph. For example, a space such as a door or a corridor that connects between shared surfaces or unit spaces can be expressed as an edge.

- Geometry of Indoor Space
  IndoorGML represents the geometry of interior space based on the geometric model defined in ISO 19107(ISO/TC211, 2007). IndoorGML includes the following interior space geometry. First, IndoorGML expresses the geometry of each unit space in three-dimensional or two-dimensional geometry. For example, if the unit space is a room, the geometry of the room can be expressed as a two-dimensional or three-dimensional geometry. The unit space is represented by a Solid in three dimensions or a Surface in two dimensions. Second, we can express the geometry of the elements connecting two unit spaces. For example, a door connecting a room and a hallway can be an element connecting two unit spaces, and a geometry is expressed as a surface in three dimensions or a curve in two dimensions. Third, IndoorGML expresses the geometry of node and edge which have duality relation with the unit spaces and the connections by points and curves, respectively.
- Multi-Layered Space Model
  One space can be expressed differently depending on various uses in an indoor space. In IndoorGML, this concept is called the multi-layered space model. The multi-layered space model can represent one space in several layers. Each layer can be defined according to various meanings or purposes.

3.2 The data model of IndoorGML

The Figure 3 is the UML model of the IndoorGML core module. IndoorGML core module represents the main concept of IndoorGML. The unit space is represented by CellSpace. The connection elements between the unit spaces are represented by CellSpaceBoundary. The set of CellSpace and CellSpaceBoundary make up PrimalSpaceFeatures. IndoorGML expresses Multiayered Space Model as a set of SpaceLayer corresponding to one layer. The SpaceLayer represents not only the graph expressing topological relationship between the unit space, but also the information of the space which can be differ as the usage of the space in the form of graph. The nodes of the graph represent State, and the edges represent Transition. A collection of SpaceLayers gathers to create a MultiLayeredGraph. The elements mentioned are IndoorGML Complex Features.

3.3 XML Schema of IndoorGML

IndoorGML is an XML-based schema. The XML schema of IndoorGML has following characteristics. First, it has hierarchical structure. Users who read IndoorGML data will retrieve the data according to this structure. Second, it has several elements which help to increase the expressiveness of the schema. Figure 4 shows a part of the XML schema of IndoorGML. PrimalSpaceFeatures has the attribute whose name is cellspaceMembers. This attribute is used to express the list of CellSpace under PrimalSpaceFeatures in the XML schema. Third, it can express the relationship among the IndoorGML elements by inserting the data of the referenced data or using XLink. For example, CellSpace can hold the whole data of the State for expressing the duality relationship, or just hold the XLink to the State. This way gives users the various choices of expression.

4. THE FUNCTION OF INFACTORY

4.1 Support for expressing basic concept of IndoorGML

InFactory can create all IndoorGML Complex Features that represent the core concept of IndoorGML as addressed at Table 1. InFactory creates IndoorGML Complex Features that represent them in the Celluar Space, Topology, and Multilayered space models. Geometry information can be generated by extending the Simple Feature Geometry model(Open Geospatial Consortium, 2015). It also supports the validation whether the generated information meet the requirements such as the multiplicity or the characteristics of the IndoorGML schema.

4.2 CRUD(Create,Read,Update,Delete) API

To manage IndoorGML features in InFactory, you need a function to enforce CRUD for the IndoorGML Feature. InFactory implements the CRUD API, which implements CRUD functionality for IndoorGML Features. In the case of an attribute expressing a relationship in the IndoorGML Feature, the identifier of the referencing element is input as a parameter of the function. In other words, you can pass only the identifier information without parameterizing the entire data of the referencing element. This allows users to easily use the CRUD API in their services or programs without having to worry about the data types used in IndoorGML.

4.3 RESTful API

This research supports RESTful API to use InFactory regardless of user's environment. Users can create IndoorGML by writing their own indoor space information according to the predefined Javascript Object Notation (JSON) based form and transmitting it to InFactory's server using HTTP Method (GET, POST, PUT, DELETE) . Table 2 is an example of requesting information about the CellSpace among the implemented RESTful API interfaces.

HTTP verb	Path	CRUD Action	Purpose
POST	/cellspaces/:id	create	Create one cellspace object.
PUT	/cellspaces/:id	update	Modify the information of the cellspace object.
GET	/cellspaces	Read	Return the id list of the already saved cellspaces object.
GET	/cellspaces/:id	Read	Invoke a CellSpace object with that id.
DELETE	/cellspaces:/	Delete	Delete the CellSpace object with that id.

Table 2. The list of CRUD function related to CellSpace Type

We defined the JSON-based form according to the structure of the feature class. Figure 5 is an example of JSON data for entering CellSpace. In this case, we use the Well Known Text (WKT)(Open Geospatial Consortium, 2015) format as the input form of the geometric information. However, WKT does not support 3D geometry such as Solid in ISO19107 and we extended WKT to support the 3D geometry used in IndoorGML. In the figure below, we can see that the geometry of CellSpace is defined as a solid type of WKT that is extended in three dimensions.

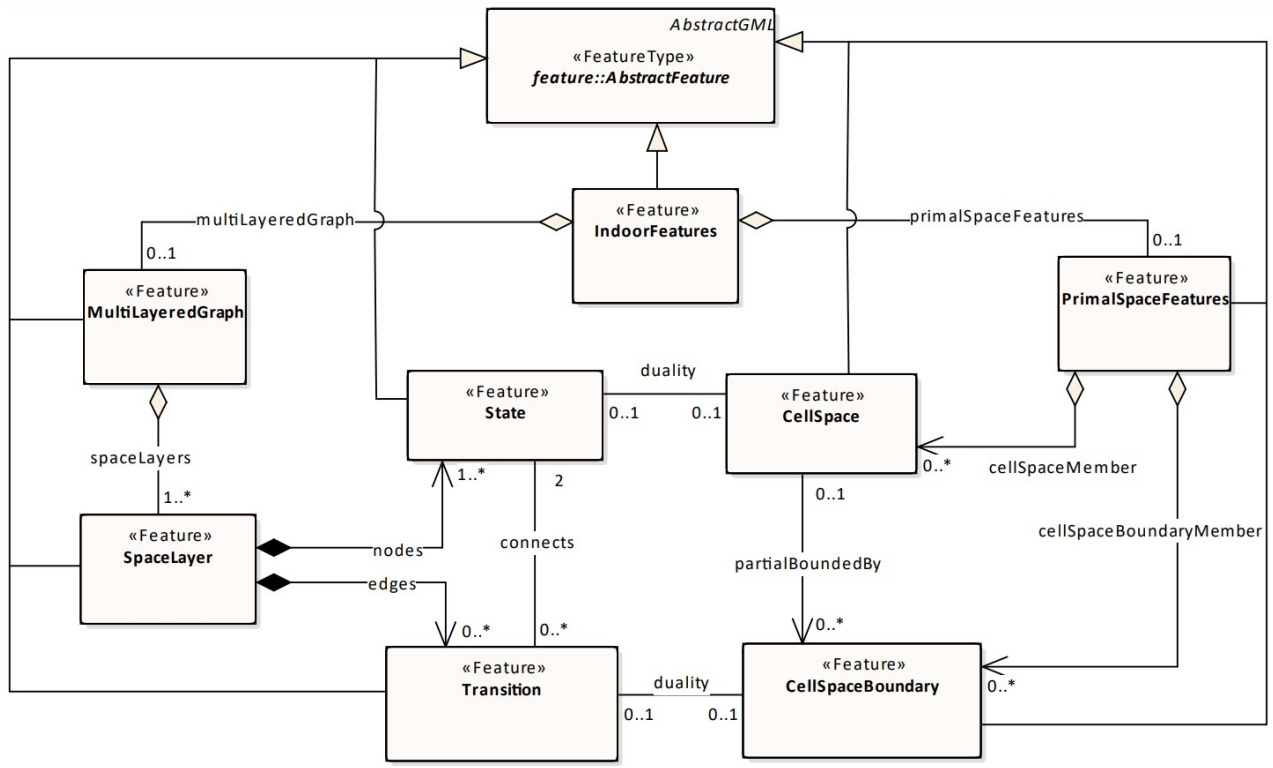

Figure 3. UML of the IndoorGML data model

```
<!-- ================================================================== -->
<xs:complexType name="PrimalSpaceFeaturesType">
    <xs:complexContent>
        <xs:extension base="gml:AbstractFeatureType">
            <xs:sequence>
                <xs:element name="cellSpaceMember" type="CellSpaceMemberType" mi
                <xs:element name="cellSpaceBoundaryMember" type="CellSpaceBounda
            </xs:sequence>
            <xs:attributeGroup ref="gml:AggregationAttributeGroup"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<!-- ================================================================== -->
<xs:complexType name="CellSpaceMemberType">
    <xs:complexContent>
        <xs:extension base="gml:AbstractFeatureMemberType">
            <xs:sequence minOccurs="0">
                <xs:element ref="CellSpace"/>
            </xs:sequence>
```

Figure 4. part of the XML schema of IndoorGML

```
{
  "docId":"doc1",
  "parentId":"psf1",
  "id":"c1",
  "geometry": {
    "type" : "Solid",
    "coordinates" : "SOLID (( ((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)), ((0 0 0, 0 1 0
      , 0 1 1, 0 0 1, 0 0 0)), ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)), ((1 1 1, 1 0
      1, 0 0 1, 0 1 1, 1 1 1)), ((1 1 1, 1 0 1, 1 0 0, 1 1 0, 1 1 1)), ((1 1 1, 1 1
      0, 0 1 0, 0 1 1, 1 1 1)) ))",
    "properties" : {
      "id" : "c1g",
      "type" : "wkt"
    }
  },
  "properties" : {
      "duality":"s1",
    "partialboundedBy" : ["B1"]
  }
}
```

Figure 5. The JSON format for representing CellSpace

4.4 Unordered creation of IndoorGML Complex Feature

When the user enters the data related IndoorGML to InFactory, it is not necessary to follow the order of the hierarchical structure of IndoorGML. InFactory accepts the data that makes up the IndoorGML in any order. And InFactory loads the necessary IndoorGML element data in order of hierarchy when it needs to export IndoorGML as XML data. For example, a user sends a request to InFactory to generate data representing that a transition connects two states. If you follow the IndoorGML schema, you must first transfer the state data and transfer the transition data. However, you do not need to follow the order when you send data to InFactory. Thus, using InFactory allows users to generate IndoorGML documents without deep understanding of IndoorGML.

5. THE STRUCTURE OF INFACTORY

InFactory has the following structure which is described in Figure 6;RESTful API Servlet,CRUD Java API, IndoorGML Complex Features, PostGIS(PostGIS Development Team, 2018), Binder, IndoorJSON Convector, and JAXB Converter.

1. HTTP Request from User : The client program of user sends the HTTP request to InFactory.
2. RESTful API : The RESTful API handles Http requests sent by users.
3. CRUD API : This generates, retrieves, modifies, and removes the IndoorGML complex feature data. The RESTful API calls the appropriate CRUD API according to the type of request sent by the client program.
4. IndoorGML Complex Feature : InFactory saves the IndoorGML Complex Feature data in Java Class format.
5. PostGIS : To deal with the big sized data, InFactory stores the IndoorGML Complex Feature data in PostGIS.

6. Binder : Binder converts the IndoorGML data stored in the Java class to the Java Architecture for XML Binding(JAXB)(Ort and Mehta, 2003) class. Binder checks the requirements of the IndoorGML data whether the element referenced by the Java class of the IndoorGML Complex Feature actually exists, multiplicity is satisfied,hierarchical structure can be made by the elements, and so on. This ensures that the input IndoorGML data meets the requirements of the IndoorGML. InFactory only confirms those things at once when exporting the IndoorGML data as XML. This prevents the program from repeatedly running to check the requirements.

7. IndoorJSON Converter : If the user requests the IndoorGML Complex Feature element data by GET instead of the IndoorGML document, IndoorJSON Converter converts it to JSON format and outputs it.

8. JAXB Converter : The Converter converts the JAXB class to XML. JAXB supports encoding and decoding between Java class instances and XML Schema components. It was implemented using open-source ogc-schemas[6].

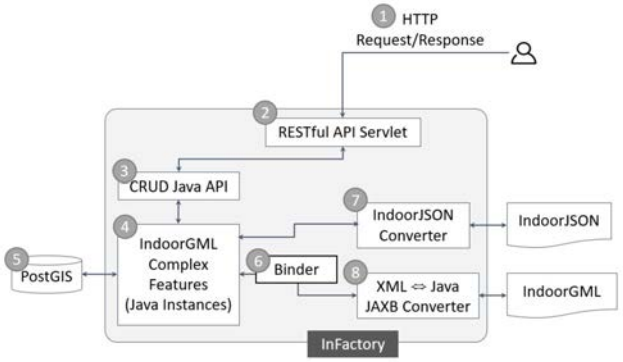

Figure 6. Overall Structure of InFactory

6. REPRESENTATION OF INDOORGML FEATURE

6.1 Geometry

IndoorGML expresses the geometry data as a geometric model of OGC GML 3.2.1. Because GML is a complex geometry model, it is difficult and complex to implement all the geometry models in GML. So we first used the geometry model defined in OGC's Simple Feature Geometry (SFG)(Open Geospatial Consortium, 2015). SFG provides the geometry model as Figure 7.

However, since the above geometric model is limited to two dimensions, there are the following problems in expressing the three-dimensional geometry required by the IndoorGML. First, the SFG uses a two-dimensional (x, y) coordinate system. Second, 3D geometry objects - especially Solid - are not included. To solve these two problems, the coordinates are extended to three dimensional (x, y, z) coordinates, and Solid is expressed as closed and oriented given MultiPolygon. In Figure 7, the red box is extended part for Solid.

Although we used SFG as a simple geometric model in this development, IndoorGML will eventually adopt the geometric model defined in ISO 19107. In IndoorGML, however, the use of a wide variety of 3-dimensional geometric elements, such as the Curved Surface, as defined in ISO 19107, is not useful and is complex to implement. SFG was used in this development because 3D geometry elements used in IndoorGML can be expressed sufficiently by SFG and its extension.

[6]https://github.com/highsource/ogc-schemas

6.2 IndoorGML Complex Feature

We implemented the Java Class in consideration of the characteristics of the IndoorGML Complex Feature to store the IndoorGML information in the InFactory. IndoorGML schemas such as various relationship-association, aggregation, composition- and multiplicity among IndoorGML elements have been considered. In this study, among the components of IndoorGML, the elements that inherit the AbstractFeature of GML are implemented in Java class. The attribute representing the relationship is represented by the object's Id. Other attributes are stored according to Java data type. Table 3 defines how relationships related to CellSpace are implemented as attributes of a Java class. In CellSpace, duality is an attribute that expresses the duality relationship between State and CellSpace.

Attribute	Type	Description
id	string	id of this instance
parentId	string	id of the instance which is PrimalSpaceFeature Type and refers this instance
duality	string	id of the instance of State Type which has duality relationship with this instance
geometry	string	id of the instance which hold the geometric data of this instance
patialboundedBy	string[]	the list of id of CellSpaceBoundary Type which is the boundary of this instance

Table 3. The main attributes in the Java class of CellSpace Type

7. THE SCHEMA DESIGN OF POSTGIS

In order to handle a large amount of data, it is effective to store the feature of the IndoorGML in the database rather than to use a simple file system. InFactory stores IndoorGML Features in PostGIS. In this study, the schema of PostGIS was created considering the following factors. First, the schema should be able to express various relations of IndoorGML. Second, it must be able to avoid excessive join operations that can occur depending on the expression of the relationship.

Accordingly, the database schema is designed as follows. First, we classify the relationships among the elements in the IndoorGML according to the relationship of the schema. Table 4 summarized the classification of the relationship and the implementation of the schema accordingly.

In Table 4, the some relationships are 1:1 or 1:N type so there will be few join operations. But the relationship between State and Transition is N:M relationship. So many joins occur in this relationship. Because State and Transition are components of a graph that shows the topology relation between CellSpace. We need to continue join operation to obtain connection information among Transitions and States when searching graph. This leads

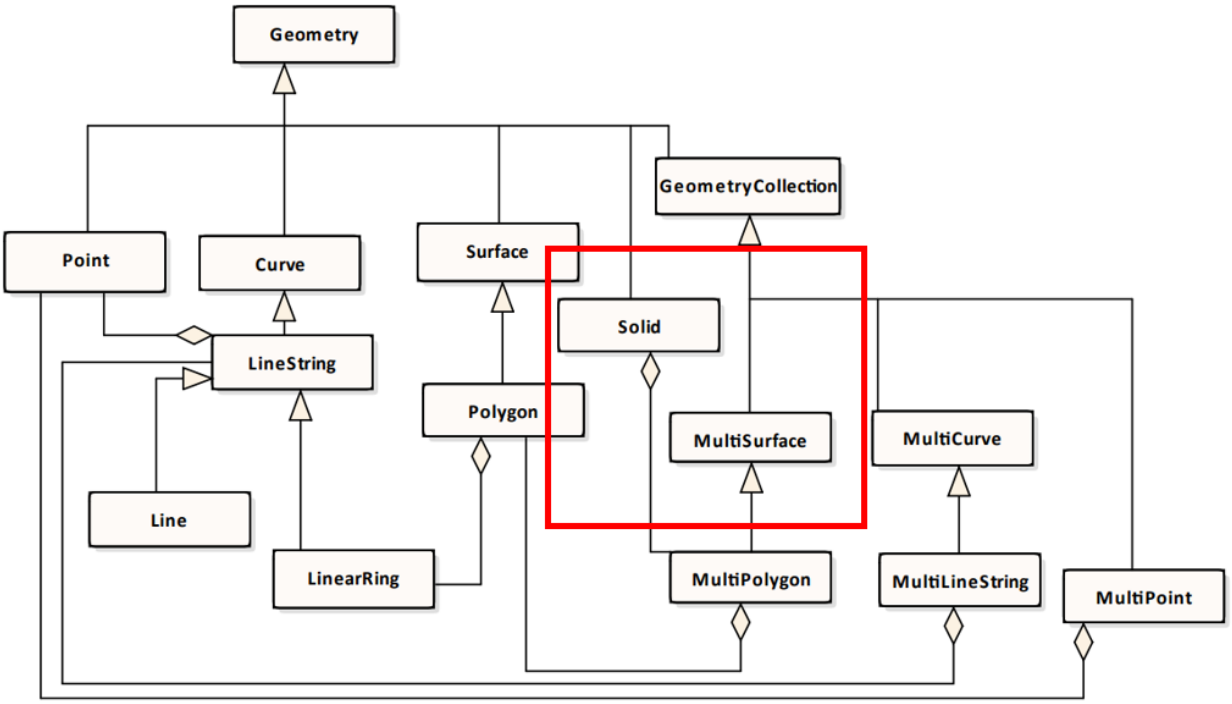

Figure 7. The extended model of Simple Feature Geometry

Relationship in IndoorGML	Type
PrimalSpaceFeatures-CellSpace	1:N
PrimalSpaceFeatures-CellSpaceBoundary	
SpaceLayer-State	
SpaceLayer-Transition	
CellSpace-CellSpaceBoundary	
CellSpace-State	1:1
CellSpaceBoundary-Transition	
State-Transition	N:M

Table 4. The categories of relationships in IndoorGML

to excessive joins and overhead. In order to avoid this, we can save the connection between state and transition in a separate join table so that the graph can be searched with only the join table. Figure 8 shows the database schema that designed the join table. The join table is named Network. Network table only contain connection information between State and Transition.

8. USE CASE

8.1 Sample program

We implemented a simple Web client program to check the server communication capabilities of InFactory. This program is JavaScript-based and sends requests that follow the Http method to InFactory. Figure 9 shows the process of creating an IndoorGML XML document and the Http method request for it. It first sends a POST request to the InFactory's RESTful server to generate a document, and then repeatedly sends Http method-based requests to CRUD on the IndoorGML Feature. After sending all the requests for document creation and sending a GET request, InFactory generates the IndoorGML document and sends it to the client program. Figure 10 shows the IndoorGML data received as a result of a GET request.

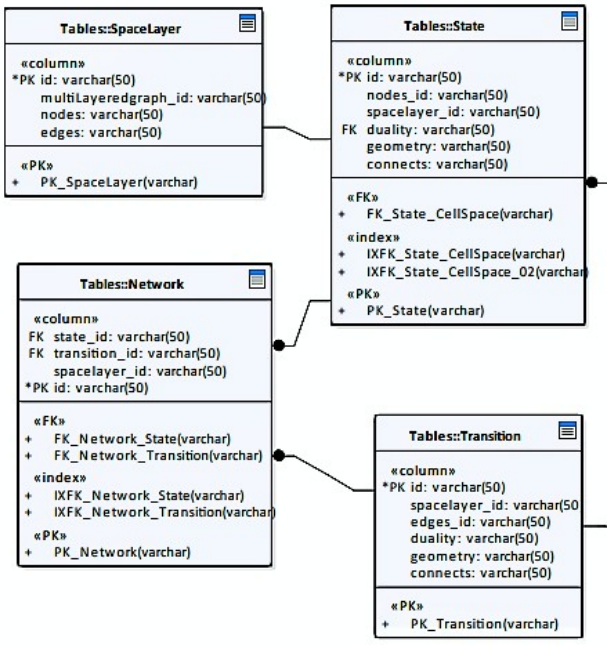

Figure 8. The part of the schema in PostGIS

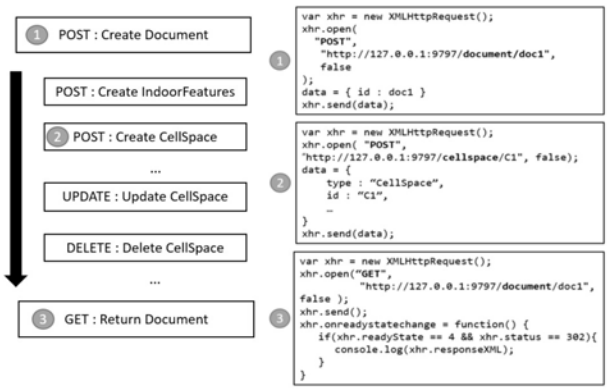

Figure 9. The process of sample program

Figure 10. The data created by sample program

8.2 Practical Usecase : InEditor

This chapter describes applications that use InFactory. Currently, InEditor[7] is an application that uses InFactory. InEditor is a web program that edits IndoorGML data graphically based on WebGL. Figure 11 is the user interface of InEditor. InEditor uses InFactory as following processes. First, InEditer edits the IndoorGML data and sends a request to the InFactory server in the defined JSON format. Next, InFactory parses the input JSON data, converts it into IndoorGML data in xml format, and sends XML data to InEditor. If the data sent by InEditor does not meet the requirements of IndoorGML, it sends an error message informing the contents of the unsatisfied requirement instead of the XML data.

If InEditor generated IndoorGML information, it would be difficult to bind the generated indoor information data to the IndoorGML XML element. And it also would be difficult for InEditor to process large amount of data. However, InEditor can easily generate IndoorGML data by simply requesting InFactory to generate data based on RESTful API.

9. CONCLUSION

IndoorGML was selected as the standard in OGC for data exchange between various systems dealing with indoor spatial information. However, few applications have been able to generate IndoorGML data. In order to generate IndoorGML data directly, it is necessary to consider the schema of IndoorGML and to encode XML format data. Therefore, it is difficult to generate

[7]https://github.com/STEMLab/InEditor

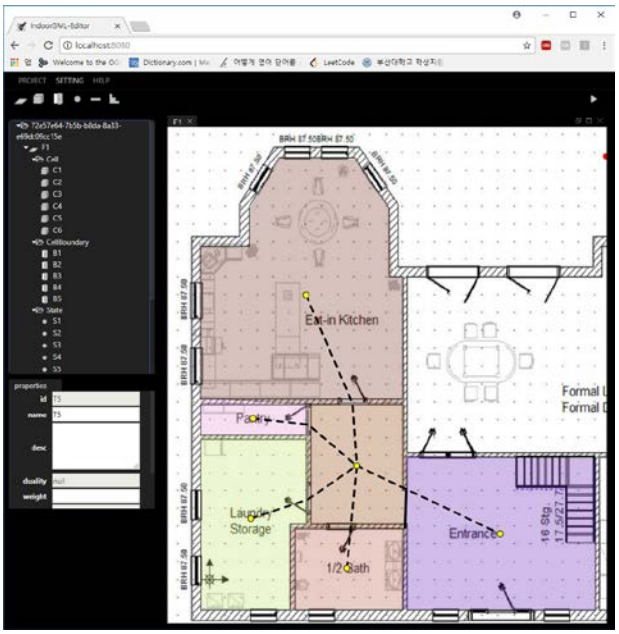

Figure 11. InEditor

IndoorGML data. Therefore, this study has implemented InFactory, a RESTful Server that generates and manages IndoorGML data. Users can easily manage indoor space information between various platforms in IndoorGML format using InFactory. This enabled InFactory to provide the support for a standard called IndoorGML.

The contribution of this study is summarized as follows.

- InFactory contributes to the step of generating IndoorGML data in the Eco-System of IndoorGML.
- Users can easily create IndoorGML without knowing the requirements of IndoorGML through InFactory.
- In the indoor spatial information service, using the InFactory's RESTful API and the CRUD API can save the effort of creating the IndoorGML generation tool.

The current development of InFactory is uploaded to the Github repository (https://github.com/STEMLab/InFactory). Users who wish to contribute to the Eco-system or IndoorGML spread of IndoorGML uses InFactory at any time via the repository or may contribute to the development of InFactory.

ACKNOWLEDGEMENTS

This work was partially supported by BK21PLUS, Creative Human Resource Development Program for IT Convergence and also supported by a grant(18NSIP-B135746-02) from National Spatial Information Research Program (NSIP) funded by Ministry of Land, Infrastructure and Transport of Korean government.

REFERENCES

ISO/TC211, 2007. ISO/TS 19139: 2007 Geographic information–Metadata–XML Schema Implementation.

Kang, H.-K. and Li, K.-J., 2017. A Standard Indoor Spatial Data ModelOGC IndoorGML and Implementation Approaches. *ISPRS International Journal of Geo-Information* 6(4), pp. 116.

Li, K.-J., Kim, T.-H., Ryu, H.-G. and Kang, H.-K., 2015. Comparison of CityGML and IndoorGML-A Use-Case Study on Indoor Spatial Information Construction at Real Sites. *Journal of Korea Spatial Information Society* 23(4), pp. 91–101.

Liebich, T., 2013. IFC4-The new buildingSMART standard. *BuildingSMART International*.

Open Geospatial Consortium, 1994. Open Geospatial Consortium. Open Geospatial Consortium http://www.opengeospatial.org/.

Open Geospatial Consortium, 2008. OpenGIS City Geography Markup Language (CityGML) Encoding Standard. *Document Number 12-019 OGC*.

Open Geospatial Consortium, 2014. OGC® IndoorGML. *Document Number 14-005r5 OGC*.

Open Geospatial Consortium, 2015. Geographic information– Well- known text representation of coordinate reference systems. *Document Number 12-063r5 OGC*.

Open Geospatial Consortium, n.d. OGC KML. *Document Number 12-007r2 OGC*.

Ort, E. and Mehta, B., 2003. Java architecture for xml binding (jaxb). *Sun Developer Network*.

PostGIS Development Team, 2018. PostGIS 2.4.4. PostGIS https://postgis.net/ (5 Apr 2018). Accessed on 2018-05-08.

Richardson, Leonard and Ruby, Sam, 2008. *RESTful web services*. O'Reilly Media, Inc.

Revised May 2018