Developing a database for the LADM-IndoorGML model

Abdullah ALATTAS and Peter VAN OOSTEROM, The Netherlands,
Sisi ZLATANOVA, Abdoulaye DIAKITÉ and Jinjin YAN, Australia

Key words: indoor; navigation; LADM; database; PostgreSQL

SUMMARY

This paper presents the development of database for the conceptual model of LADM-IndoorGML. The aim of this work is to investigate all issues that related to generating the database and visualizing the content of the database. Based on the result of the transformation from conceptual model to the technical model that has been proposed by (Alattas et al., 2018), we have selected some classes of the conceptual model of LADM-IndoorGML to create a database in PostgreSQL with the extension of PostGIS. By converting those classes from class diagram to SQL DDL, a database has been generated and stored different type of data. A visualization tool has been used to visualize indoor spaces based on RRRs for the users.

Developing a database for the LADM-IndoorGML model

Abdullah ALATTAS and Peter VAN OOSTEROM, The Netherlands,
Sisi ZLATANOVA, Abdoulaye DIAKITÉ and Jinjin YAN, Australia

1. INTRODUCITON

The conceptual model of LADM-IndoorGML has been converted into a technical model by (Alattas et al., 2018) using the Enterprise Architect software, and during the conversion of the conceptual model there were several issues encountered such as correct handling of primary keys, foreign keys, the association multiplicity, the attributes multiplicity, data type, spatial data type, index, spatial index, constraints, and inheritance. The conceptual model of IndoorGML and Land Administration Domain Model (LADM) has been proposed by (Alattas et al. 2017), and it introduced an approach to determine the accessibility of the indoor spaces based on the access rights. IndoorGML is an Open Geospatial Consortium (OGC) standard that offers a structure for an indoor navigation system which describes the indoor space and Geography Markup Language (GML) syntax for encoding geoinformation (Zlatanova et al. 2016). Land Administration Domain Model (LADM) is an ISO standard (19152:2012) that focuses on rights, responsibilities, and restrictions (RRR) that affect lands and space components (Lemmen et al. 2015). The LADM-IndoorGML conceptual model determines the space access rights over time for the user of the indoor environment to provide efficient indoor navigation. The LADM-IndoorGML conceptual model utilizes LADM to create a link between the indoor spaces and the users, by indicating rights, restrictions, and responsibilities to each indoor space to define the available spaces for each type of user. In this paper we address a DBMS implementation, i.e. PostgreSQL with PostGIS spatial extension. We create technical implementation, derive the SQL code, load 3D spatial data, party data, and access rights and restriction data into the database, and perform a set of queries to represent spaces based on the RRRs. We visualize the results in in different visualization software. The database is be populated with real data for an educational building. All issues that have been addressed by (Alattas et al., 2018) are taken into consideration for efficiently populating the database. Furthermore, this paper presents a number of issues related to the data conversion and loading to the database. The Enterprise Architect software has been used to convert UML diagrams (conceptual model) into SQL code (technical model). PostgreSQL with PostGIS spatial extension has been utilized for storing the geometries and the attributes data of the model.

2. PREVIOUS WORK: FROM CONCEPTUAL TO TECHNICAL MODEL

This section shows the main issues of the transformation of the conceptual model to technical model that has been done by (Alattas et al., 2018). There were three steps to convert class diagram to SQL DDL. First step is to prepare the conceptual model and that was done by checking all classes of both standards to determine whether there any missing attributes and their values. This step has shown that the UML class diagram of IndoorGML standard is not complete and there were missing attributes, code lists, datatype, and multiplicities. Java classes have been used to generate the UML classes and then additional corrections have been

done manually to a eliminate new classes and attributes that are not related to the class diagram but to the java implementation as shown in Figure 1.

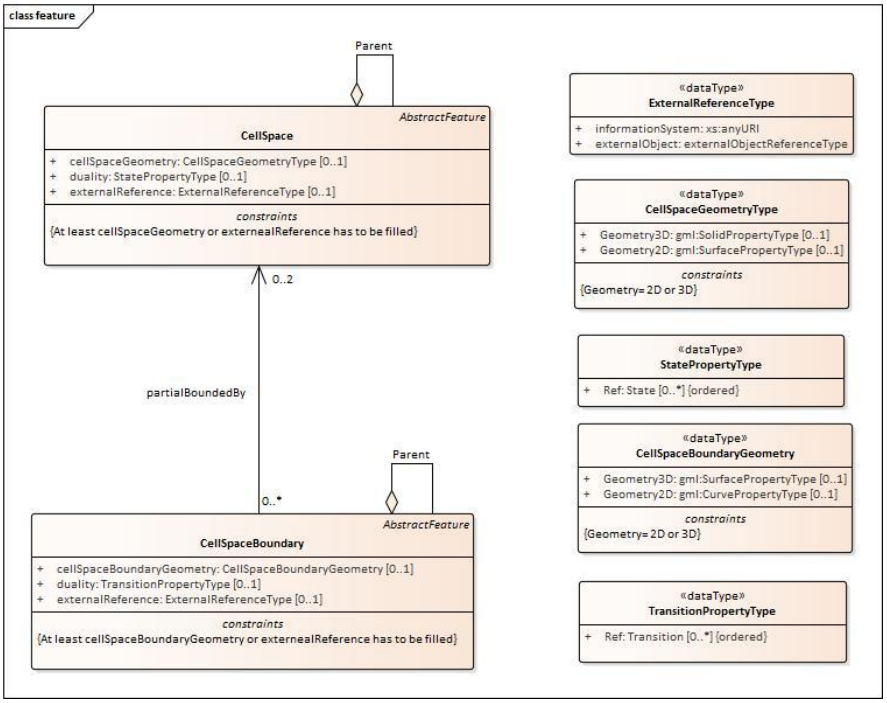

Figure 1. The corrected attributes for CellSpace class and cellSpaceBoundary (Alattas et al., 2018)

The class diagram of LADM has been modified by adding new classes from the LADM-IndoorGML conceptual model to the Party package. The LA_Employee and LA_GroupEmployee classes have been added to party package and their code lists.

After preparing each standard, a new class diagram has been created for the conceptual model of LADM-IndoorGML that contains all related classes form the two standards. The transformation tool in Enterprise Architect (EA) has been used to convert the class diagram to table diagram as shown in Figure 2. The result of the transformation had some issues related to the primary key and a foreign key, constraints, data type, spatial data type, code list classes and indexing.

- The transformation tool has generated a new primary key even if the class diagram had one, and the only solution for this issue was to remove the new PK and all associations with other classes to be able to assign the correct Id as PK as shown in Figure 3.
- The transformation tool could not include the constraints of the class diagram to the result of the transformation. The table diagram did not have constraints; for example, the LA_GroupParty class has a constraint about LA_PartyMember association class; however, the table diagram of the same class did not include that constrains as shown in Figure 3.

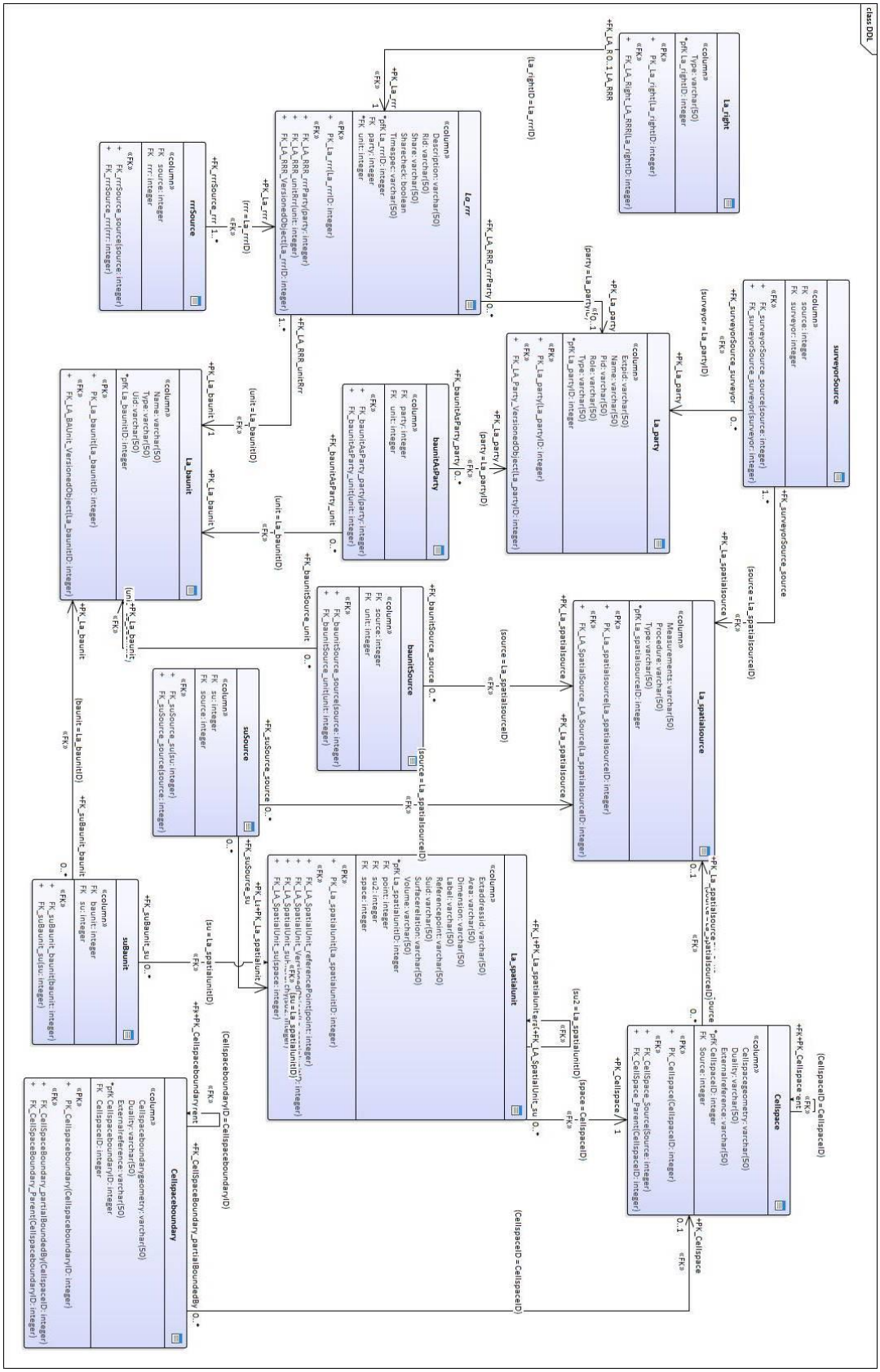

Figure 2. UML table diagram for LADM-IndoorGML UML model (Alattas et al., 2018)

- The data types that the conceptual model of LADM and IndoorGML includes have not been converted during the transformation. The table diagram does not contain the same data type such As Oid and Fraction. The only solution for this issue was to modify the SQL code manually. The spatial data type also had an issue during the transformation.
- The conceptual model of LADM-IndoorGML has already defined some spatial data such as GM_Point, GM_MultiSurface, and GM_MultiCurve, however, the table diagram did not include these types of spatial data and replaced them with "varchar." EA offers some types of spatial data in the table diagram that have to be selected manually for each attribute such as geometry, geometry collection, linestring, multilinestring, point, multipoint, polygon, and multipolygon.
- The transformation did not provide spatial index (R-tree) for spatial data as shown in Figure 4. The software supports spatial index only for ArcGIS toolbox.

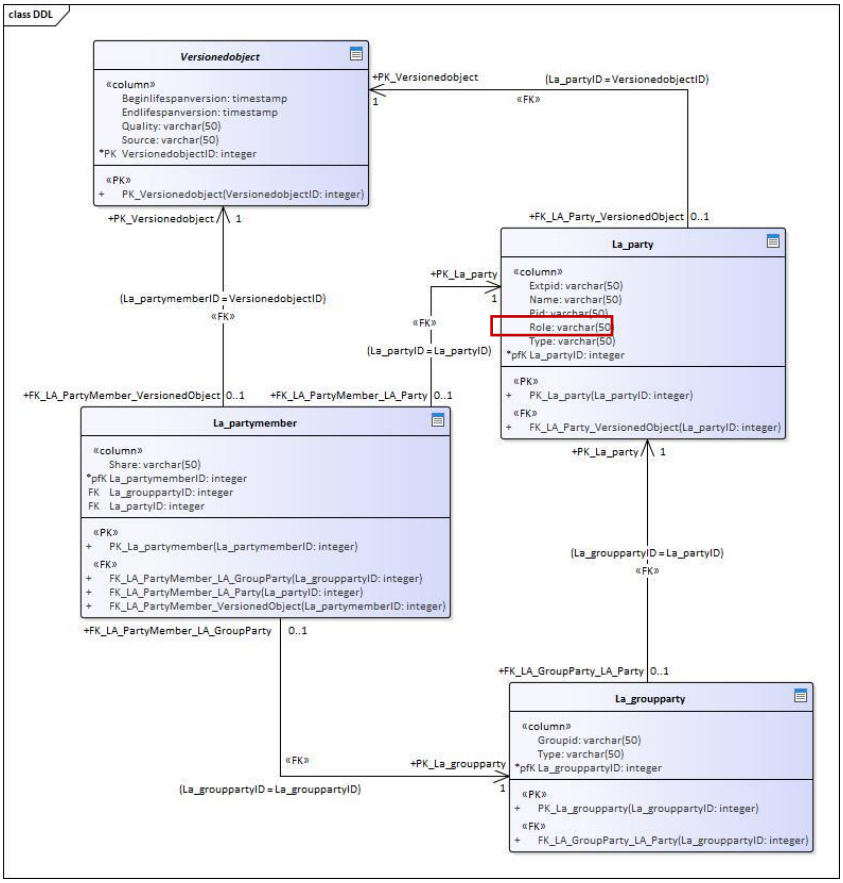

Figure 3. Additional Primary key has been created for each table (Alattas et al., 2018)

3. GENERATING & ASSESSING DATABASE FOR THE CONCEPTUAL MODEL

In this section we generate and assess a database for the conceptual model of LADM-IndoorGML. The aim of this work is to investigate all issues that related to generating the database and visualizing the content of the database.

3.1 Transformation to technical model

Based on the result of the transformation from the conceptual model to a technical model that has been proposed by (Alattas et al., 2018), we have selected some classes of the conceptual model of LADM-IndoorGML to create a database in PostgreSQL/GIS as shown in Figure 5. The transformation tool in EA has been used to convert the class diagram into a table diagram. All issues that have been discussed by (Alattas et al., 2018) have been considered and fixed. Figure 6 shows the table diagram of LADM-IndoorGML technical model. The data type and the spatial data type has been modified manually in this stage. EA does not support geometry type "multipolygon z,". Therefore this need to be corrected manually in the SQL code.

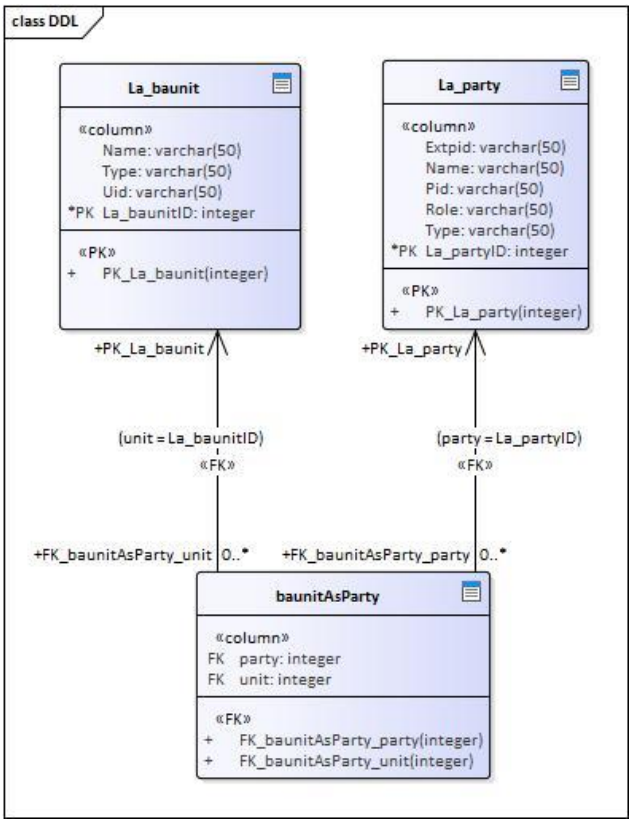

Figure 4. The index that the transformation provides in automated way (Alattas et al., 2018)

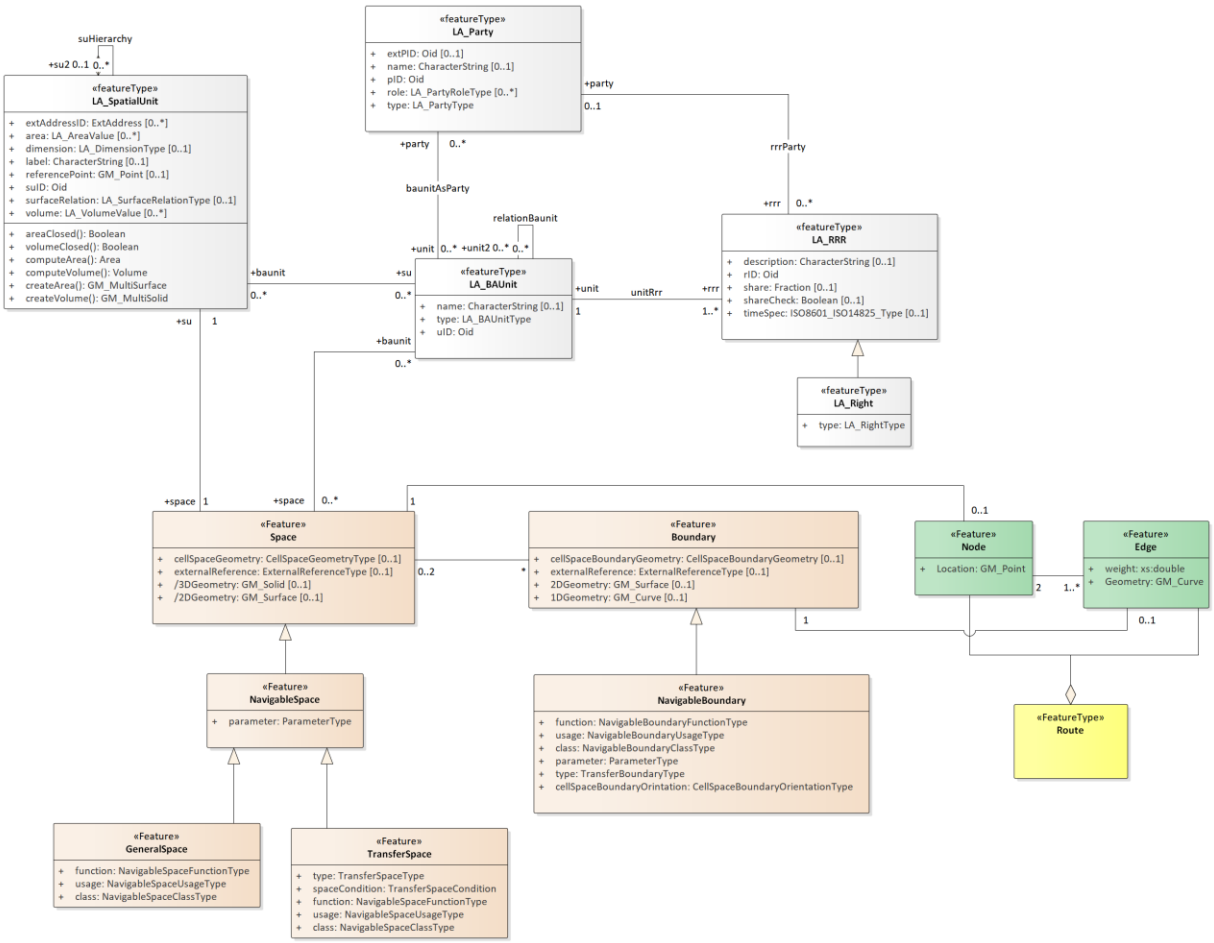

Figure 5. The selected classes from the conceptual model of LADM-IndoorGML

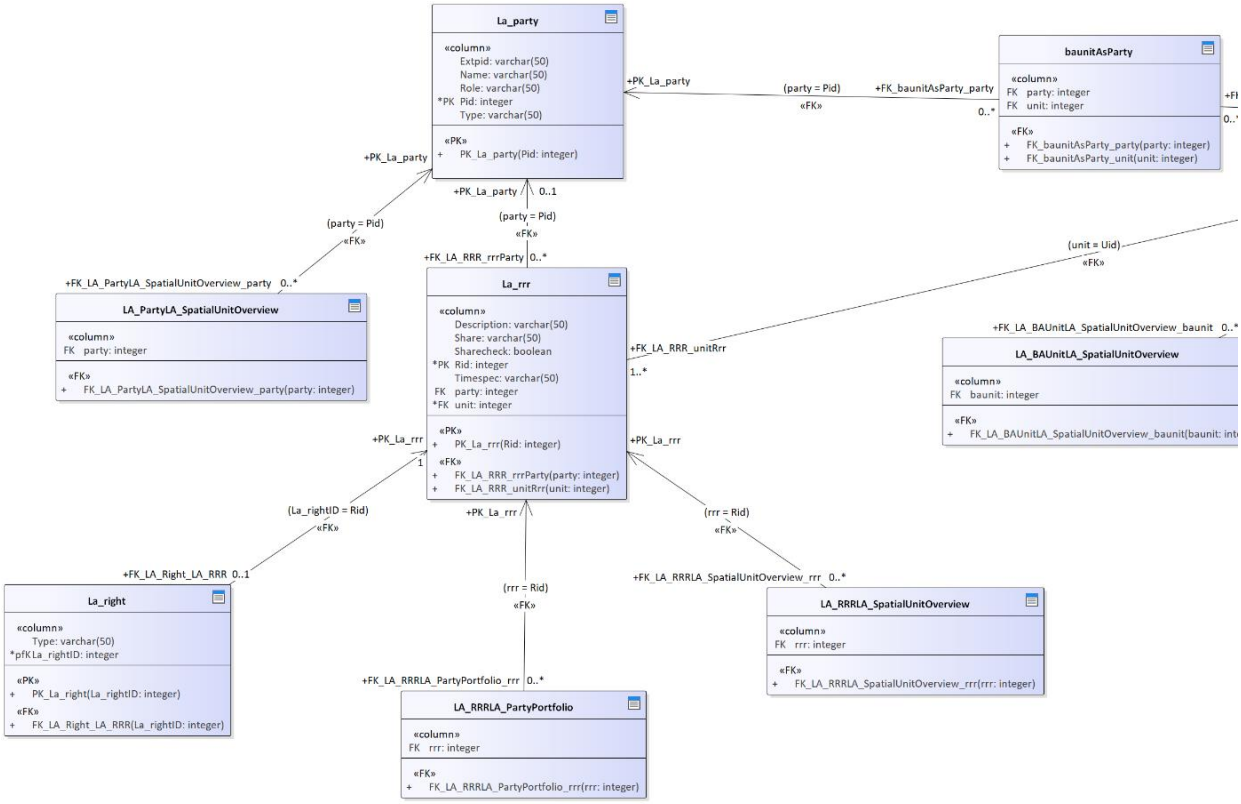

Figure 6. Part of the table diagram

A SQL DDL has been generated by using the code engineering tool in EA. The code engineering has converted the entire table diagram into SQL DDL. Figure 7 shows the SQL code for LA_RRR and LA_SpatialUnit classes (For the complete SQL Code see appendix A).

```
CREATE TABLE La_rrr(
Description varchar(50) NULL,
Share varchar(50) NULL,
Sharecheck boolean NULL,
Rid integer NOT NULL,
Timespec varchar(50) NULL,
party integer NULL,
unit integer NOT NULL
);

CREATE TABLE La_spatialunit(
Extaddressid varchar(50) NULL,
Area varchar(50) NULL,
Dimension varchar(50) NULL,
Label varchar(50) NULL,
Referencepoint geometry(point) NULL,
Surfacerelation varchar(50) NULL,
Suid integer NOT NULL,
Volume varchar(50) NULL,
su2 integer NULL,
space integer NOT NULL
);
```

Figure 7. SQL code for creating LA_RRR and LA_SpatialUnit tables

3.2 Generating Database and visualizing the data

By having the SQL code, a database could be generated in PostgreSQL as shown in Figure 8. The 3D geometry of the building has been created in Revit software and have been importred into the database by in-house software using Open Database Connectivity (ODBC). The semantic information and schedules of the building have been exported directly from Revit. The 3D geometry has been lnked to the semantic information, schedules, and all other properties in the database by the uniqie geometry ID. Afterwards, the data has been re-organised according to the spatial schema of LADM-IndoorGML using SQL queries. The tables related to LADM have been populated manually.

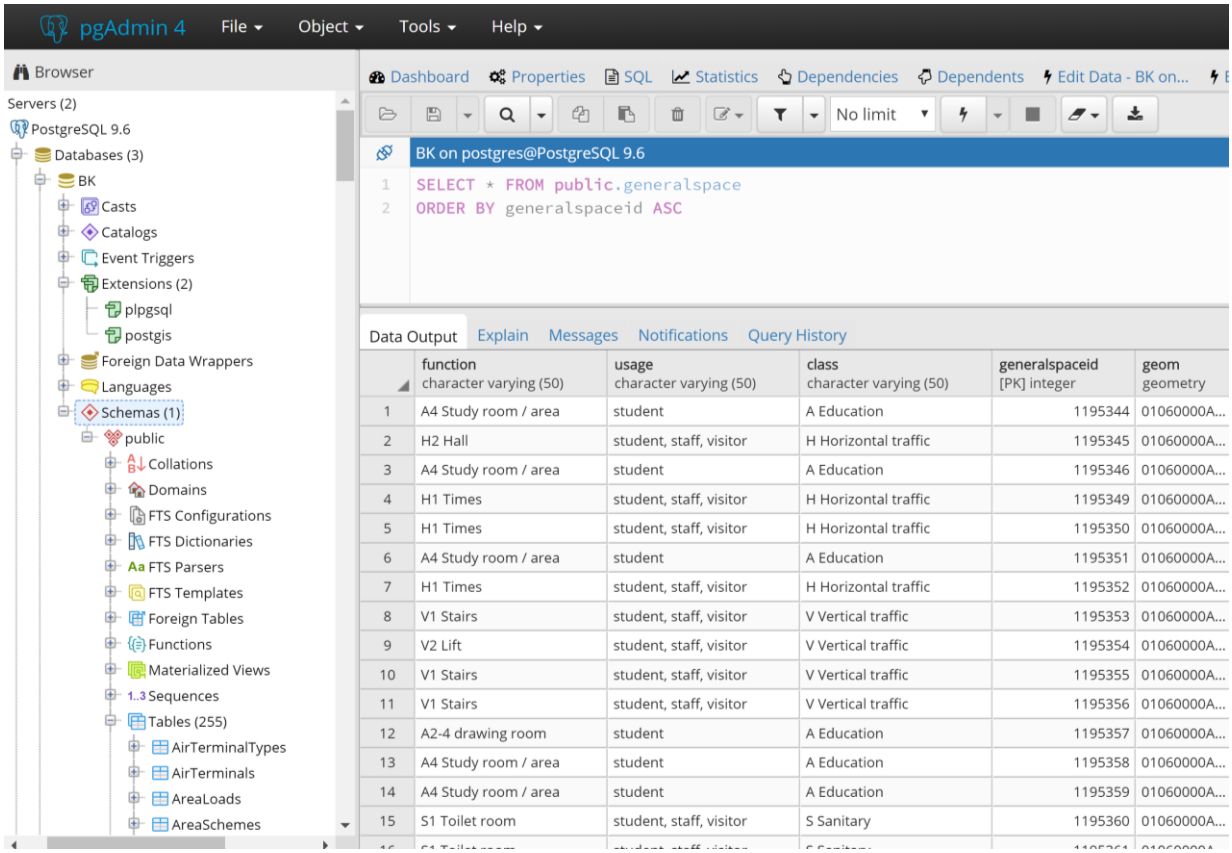

Figure 8. Part of the LADM-IndoorGML schema

A number of queries was performed to illustrate ho wthe DBMS can be used. For example, figure 9 shows the spaces that are accessible for student and staff.

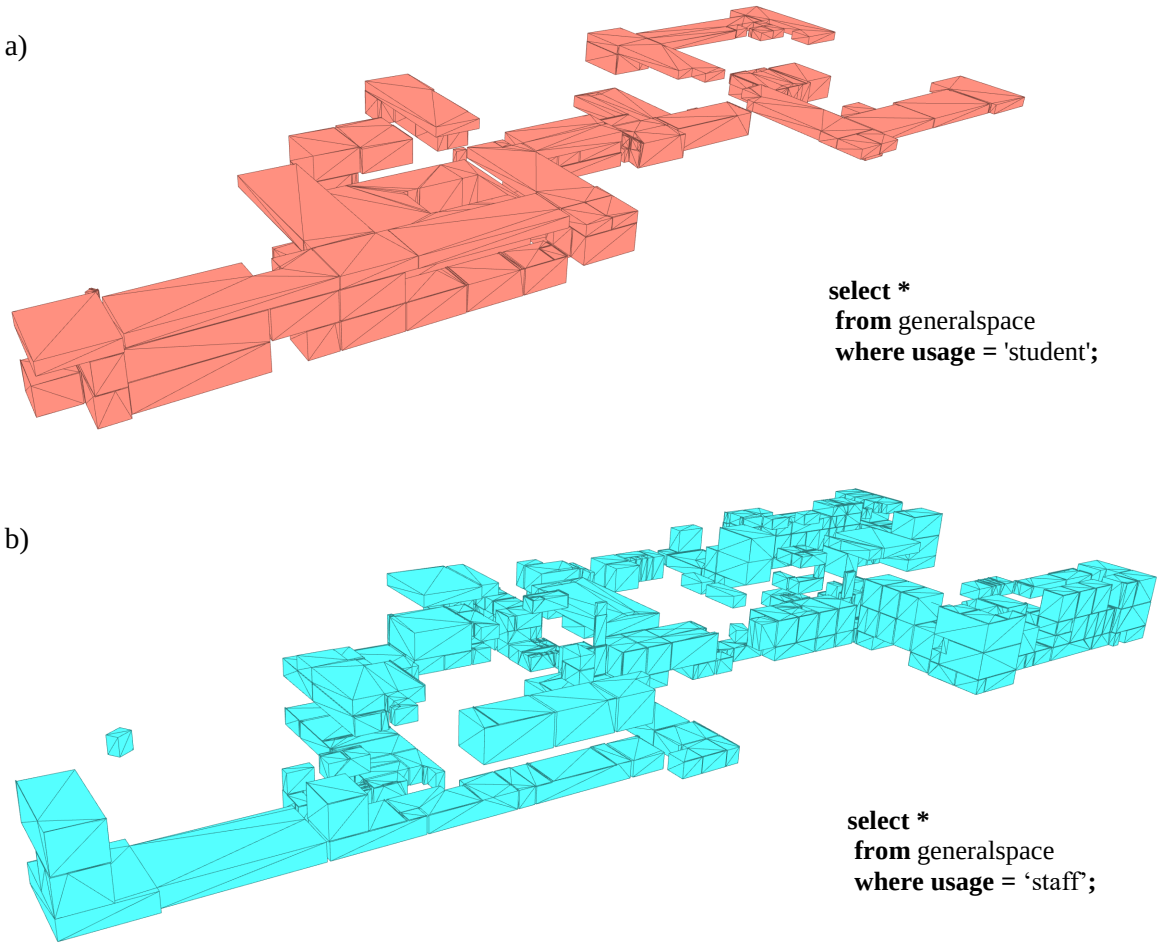

Figure 9. Accessible spaces for a) student and b) staff

4. CONCLUSION

This paper has presented the development of a database for the conceptual model of LADM-IndoorGML after converting the class diagram into table diagram and then to SQL DDL for a selected set of classes. In this paper, some critical issues discussed in (Alattas et al., 2018) have been provided a solution. The paper illustrates that the process of conceptual modelling to technical implementation can be largely automated. Very few aspects require manual intervention. This experiment has also illustrated a workflow for import of data into LADM-IndoorGML relational tables. The software package Revit is able to exports all textual data to Postgre/PostGIS, but the 3D geometry. Therefore, an ODBC importer was developed to import the 3D geometry of ifcSpace. As such most of the tables of the schema have been populated in automated way. Only a few tables of LADM have been populated manually in this experiment. However, we assume that these data will be avaible sand can be imported automatically. Having all the information stored in the database a large numbers of queries can be performed and the result can be visualized in different front-end visualization software (FME, 3DQGIS, RHINO, ArcGIS) that can read ST_Geometry data types.

Future work will focus on web-based options for visualisation of queries. We will explore WebGL to develop a web user interface to provide some interactive 3D visualisations within web browsers. WebGL is a valuable approach for creating applications that carry the desktop application experience in a web browser (Matsuda and Lea 2013, Guerrero et al, 2013). Two web applications will be considered: maintenance and navigation (on a mobile device). The web user interface will be used to explore the relationship between the indoor spaces and the users to determine the rights of use for the indoor spaces. Furthermore, the subdivision of the indoor space will be examined to assess the accessibility of the indoor spaces based on the rights, restrictions, and responsibilities (RRRs).

REFERENCES

Alattas, A., Van Oosterom, P., & Zlatanova, S. (2018). Deriving the technical model for the indoor navigation prototype based on the integration of IndoorGML and LADM Conceptual Model. 7th International FIG Workshop on the Land Administration Domain Model, 246-266.

Alattas, A., Zlatanova, S., Van Oosterom, P., Chatzinikolaou, E., Lemmen, C., & Li, K.-J. (2017). Supporting Indoor Navigation Using Access Rights to Spaces Based on Combined Use of IndoorGML and LADM Models. ISPRS International Journal of Geo-Information, 6(12), 384. doi:10.3390/ijgi6120384.

Guerrero, J., S. Zlatanova, and M. Meijers, 2013, 3D visualisation of underground pipelines: best strategy for 3D scene creation, ISPRS Annals - Volume II-2/W1, 2013, WG II/2, ISPRS 8th 3D GeoInfo Conference & ISRS WG II/2 Workshop, 139 - 145, November 2013, Istanbul, Turkey.

Lemmen, C.H.J.; van Oosterom, P.J.M.; Bennett, R. The Land Administration Domain Model. *Land Use Policy*; **2015**, *49,* 535–545.

Matsuda, K., & Lea, R. (2013). WebGL programming guide: interactive 3D graphics programming with WebGL. Addison-Wesley.

Zlatanova, S.; Van Oosterom, P.J.M.; Lee, J.; Li, K.-J.; Lemmen, C.H.J. LADM and IndoorGML for Support of Indoor Space Identification. In Proceedings of the 11th 3D Geoinfo Conference on ISPRS Annals of the photogrammetry, Remote Sensing and Spatial Information Science, Athens, Greece, 20–21 October 2016.

BIOGRAPHICAL NOTES

Abdullah Alattas is a PhD candidate at the department of Research for the built environment (OTB), Faculty of Architecture and the Built Environment, Delft University of Technology (TU Delft). He is a lecturer at the Geomatics department at the Faculty of Environmental Design, King Abdulaziz University in Jeddah, Saudi Arabia. In 2014, he obtained a master's degree in Cartography from the international Master program that is a cooperation of: Technische Universität München (TUM), Department of Cartography, Technische Universität Wien (TU Vienna), Research Group Cartography, and Technische Universität Dresden (TU Dresden), Institute for Cartography. In 2008, he received a bachelor's degree in architecture from Faculty of Environmental Design, King Abdulaziz University in Jeddah, Saudi Arabia.

Peter van Oosterom obtained an MSc in Technical Computer Science in 1985 from Delft University of Technology, The Netherlands. In 1990 he received a PhD from Leiden University. From 1985 until 1995 he worked at the TNO-FEL laboratory in The Hague. From 1995 until 2000 he was senior information manager at the Dutch Cadastre, where he was involved in the renewal of the Cadastral (Geographic) database. Since 2000, he is professor at the Delft University of Technology (OTB institute) and head of the section 'GIS Technology'. He is the current chair of the FIG working group on '3D-Cadastres'.

Sisi Zlatanova obtained her MSc in Geodesy, Protogrammetry and Cartography at the University of Architecture, Civil Engineering and Geodesy, Sofia, Bulgaria in 1984 and specialised Applied Mathematics at Technical University, Sofia. She has received her PhD degree from Graz University of Technology, Austria in 2000. She worked as software developer at Bulgarian Central Cadastre (1985-1989), assistant professor at University of Architecture and Civil Engineering, Sofia (1989-1999) and associate professor at the Delft University of Technology (2000-2017. Since 2018 she is professor and head of GRID, at the University of New South Wales, Faculty of Built Environment, Sydney, Australia. She is the current president of ISPRS Technical Commission IV 'Spatial Information Science'.

Abdoulaye A. Diakité is a postdoctoral fellow at the Faculty of Built Environment, UNSW, Sydney, Australia and member of Geospatial, Research, Innovation and Development (GRID) lab. He was graduated at University of Burgundy, France and completed his PhD in computational geometry at the LIRIS lab (University of Lyon 1, France) in 2015. After this, he joined the 3D Geoinformation of Delft University of Technology (2015-2018), where he first worked as a post-doctoral researcher on the SIMs3D project (smart space subdivision of building model for indoor navigation purpose in emergency situations; www.sims3d.net). He is now in charge of the BIM, BIM/GIS and Navigation related researches of the GRID lab.

Jinjin Yan is a PhD candidate at the Faculty of Built Environment, UNSW, Sydney, Australian and a member of the GRID Lab. His PhD research topic is Seamless Pedestrian Navigation in Indoor/Outdoor Large Spaces with No Clear Patterns for Movement. He finished his master's degree (2013-2016) and bachelor's degree (2009-2013) in the Faculty of Information Engineering, China University of Geosciences in Wuhan. After that, he started his PhD research in the 3D GeoInformation group at the Department of Urbanism, Delft University of Technology from October 2016 to February 2018.

CONTACTS

Abdullah Alattas
Delft University of Technology Section GIS-technology,
Department OTB, Faculty of Architecture and the Built Environment
P.O. Box 5030, 2600 GA Delft
THE NETHERLANDS
Tel. +31 639898691
E-mail: a.f.m.alattas@tudelft.nl

Peter van Oosterom
Delft University of Technology Section GIS-technology,
Department OTB, Faculty of Architecture and the Built Environment
P.O. Box 5030, 2600 GA Delft
THE NETHERLANDS
Tel. +31 15 2786950
E-mail: P.J.M.vanOosterom@tudelft.nl
website http://www.gdmc.nl

Sisi Zlatanova
UNSW Built Environment
Kensington Campus
Sydney, NSW 2052 Australia
Tel: +61 2 93856847
E-mail: s.zlatanova@unsw.edu.au
website http://www.be.unsw.edu.au

Abdoulaye A. Diakité
UNSW Built Environment
Kensington Campus
Sydney, NSW 2052 Australia
Tel: +61 2 9385 4780
E-mail: a.diakite@unsw.edu.au
website http://www.be.unsw.edu.au

Jinjin Yan
UNSW Built Environment
Kensington Campus
Sydney, NSW 2052 Australia
Tel: +61 0 449558983
E-mail: Jinjin.yan@student.unsw.edu.au
website http://www.be.unsw.edu.au

APPENDIX A

SQL DDL for selected classes from LADM-IndoorGML conceptual model.

```
DROP TABLE IF EXISTS baunitAsParty CASCADE;
DROP TABLE IF EXISTS Boundary CASCADE;
DROP TABLE IF EXISTS BoundarySpace CASCADE;
DROP TABLE IF EXISTS Edge CASCADE;
DROP TABLE IF EXISTS Generalspace CASCADE;
DROP TABLE IF EXISTS La_baunit CASCADE;
DROP TABLE IF EXISTS LA_BAUnitLA_PartyPortfolio CASCADE;
DROP TABLE IF EXISTS LA_BAUnitLA_SpatialUnitOverview CASCADE;
DROP TABLE IF EXISTS LA_BAUnitSpace CASCADE;
DROP TABLE IF EXISTS La_party CASCADE;
DROP TABLE IF EXISTS LA_PartyLA_SpatialUnitOverview CASCADE;
DROP TABLE IF EXISTS La_right CASCADE;
DROP TABLE IF EXISTS La_rrr CASCADE;
DROP TABLE IF EXISTS LA_RRRLA_PartyPortfolio CASCADE;
DROP TABLE IF EXISTS LA_RRRLA_SpatialUnitOverview CASCADE;
DROP TABLE IF EXISTS La_spatialunit CASCADE;
DROP TABLE IF EXISTS LA_SpatialUnitLA_BAUnit CASCADE;
DROP TABLE IF EXISTS Navigableboundary CASCADE;
DROP TABLE IF EXISTS Navigablespace CASCADE;
DROP TABLE IF EXISTS Node CASCADE;
DROP TABLE IF EXISTS NodeEdge CASCADE;
DROP TABLE IF EXISTS Route CASCADE;
DROP TABLE IF EXISTS Space CASCADE;
DROP TABLE IF EXISTS Transferspace CASCADE;

CREATE TABLE baunitAsParty(
            party integer NULL,
            unit integer NULL);
CREATE TABLE Boundary(
            Cellspaceboundarygeometry varchar(50) NULL,
            Externalreference varchar(50) NULL,
            geometry2d geometry(polygon) NULL,
            geometry1d geometry(linestring) NULL,
            BoundaryID integer NOT NULL);
CREATE TABLE BoundarySpace(
            SpaceID integer NULL,
            BoundaryID integer NULL);
CREATE TABLE Edge(
            Weight varchar(50) NULL,
            Geometry varchar(50) NULL,
            EdgeID integer NOT NULL,
            BoundaryID integer NULL);
CREATE TABLE Generalspace(
            Function varchar(50) NULL,
            Usage varchar(50) NULL,
            Class varchar(50) NULL,
            GeneralspaceID integer NOT NULL);
CREATE TABLE La_baunit(
            Name varchar(50) NULL,
            Type varchar(50) NULL,
            Uid integer NOT NULL);
CREATE TABLE LA_BAUnitLA_PartyPortfolio(
            baunit integer NULL);
CREATE TABLE LA_BAUnitLA_SpatialUnitOverview(
            baunit integer NULL);
CREATE TABLE LA_BAUnitSpace(
            space integer NULL,
            baunit integer NULL);
CREATE TABLE La_party(
```

```
            Extpid varchar(50) NULL,
            Name varchar(50) NULL,
            Role varchar(50) NULL,
            Pid integer NOT NULL,
            Type varchar(50) NULL);
CREATE TABLE LA_PartyLA_SpatialUnitOverview(
            party integer NULL);
CREATE TABLE La_right(
            Type varchar(50) NULL,
            La_rightID integer NOT NULL);
CREATE TABLE La_rrr(
            Description varchar(50) NULL,
            Share varchar(50) NULL,
            Sharecheck boolean NULL,
            Rid integer NOT NULL,
            Timespec varchar(50) NULL,
            party integer NULL,
            unit integer NOT NULL);
CREATE TABLE LA_RRRLA_PartyPortfolio(
            rrr integer NULL);
CREATE TABLE LA_RRRLA_SpatialUnitOverview(
            rrr integer NULL);
CREATE TABLE La_spatialunit(
            Extaddressid varchar(50) NULL,
            Area varchar(50) NULL,
            Dimension varchar(50) NULL,
            Label varchar(50) NULL,
            Referencepoint geometry(point) NULL,
            Surfacerelation varchar(50) NULL,
            Suid integer NOT NULL,
            Volume varchar(50) NULL,
            su2 integer NULL,
            space integer NOT NULL);
CREATE TABLE LA_SpatialUnitLA_BAUnit(
            su integer NULL,
            baunit integer NULL);
CREATE TABLE Navigableboundary(
            Function varchar(50) NULL,
            Usage varchar(50) NULL,
            Class varchar(50) NULL,
            Parameter varchar(50) NULL,
            Type varchar(50) NULL,
            Cellspaceboundaryorintation varchar(50) NULL,
            NavigableboundaryID integer NOT NULL);
CREATE TABLE Navigablespace(
            Parameter varchar(50) NULL,
            NavigablespaceID integer NOT NULL);
CREATE TABLE Node(
            Location varchar(50) NULL,
            NodeID integer NOT NULL,
            SpaceID integer NULL);
CREATE TABLE NodeEdge(
            EdgeID integer NULL,
            NodeID integer NULL);
CREATE TABLE Route(
            RouteID integer NOT NULL,
            EdgeID integer NULL,
            NodeID integer NULL);
CREATE TABLE Space(
            Cellspacegeometry varchar(50) NULL,
            Externalreference varchar(50) NULL,
            geometry3d geometry(multipolygonz) NULL,
            geometry2d geometry(polygon) NULL,
```

```
            SpaceID integer NOT NULL);
CREATE TABLE Transferspace(
            Type varchar(50) NULL,
            Spacecondition varchar(50) NULL,
            Function varchar(50) NULL,
            Usage varchar(50) NULL,
            Class varchar(50) NULL,
            TransferspaceID integer NOT NULL);

ALTER TABLE Boundary ADD CONSTRAINT PK_Boundary
            PRIMARY KEY (BoundaryID);
ALTER TABLE Edge ADD CONSTRAINT PK_Edge
            PRIMARY KEY (EdgeID);
ALTER TABLE Generalspace ADD CONSTRAINT PK_Generalspace
            PRIMARY KEY (GeneralspaceID);
ALTER TABLE La_baunit ADD CONSTRAINT PK_La_baunit
            PRIMARY KEY (Uid);
ALTER TABLE La_party ADD CONSTRAINT PK_La_party
            PRIMARY KEY (Pid);
ALTER TABLE La_right ADD CONSTRAINT PK_La_right
            PRIMARY KEY (La_rightID);
ALTER TABLE La_rrr ADD CONSTRAINT PK_La_rrr
            PRIMARY KEY (Rid);
ALTER TABLE La_spatialunit ADD CONSTRAINT PK_La_spatialunit
            PRIMARY KEY (Suid);
ALTER TABLE Navigableboundary ADD CONSTRAINT PK_Navigableboundary
            PRIMARY KEY (NavigableboundaryID);
ALTER TABLE Navigablespace ADD CONSTRAINT PK_Navigablespace
            PRIMARY KEY (NavigablespaceID);
ALTER TABLE Node ADD CONSTRAINT PK_Node
            PRIMARY KEY (NodeID);
ALTER TABLE Route ADD CONSTRAINT PK_Route
            PRIMARY KEY (RouteID);
ALTER TABLE Space ADD CONSTRAINT PK_Space
            PRIMARY KEY (SpaceID);
ALTER TABLE Transferspace ADD CONSTRAINT PK_Transferspace
            PRIMARY KEY (TransferspaceID);

ALTER TABLE baunitAsParty ADD CONSTRAINT FK_baunitAsParty_party
            FOREIGN KEY (party) REFERENCES La_party (Pid) ON DELETE No Action ON
            UPDATE No Action;
ALTER TABLE baunitAsParty ADD CONSTRAINT FK_baunitAsParty_unit
            FOREIGN KEY (unit) REFERENCES La_baunit (Uid) ON DELETE No Action ON
            UPDATE No Action;
ALTER TABLE BoundarySpace ADD CONSTRAINT FK_BoundarySpace_Space
            FOREIGN KEY (SpaceID) REFERENCES Space (SpaceID) ON DELETE No Action ON
            UPDATE No Action;
ALTER TABLE BoundarySpace ADD CONSTRAINT FK_BoundarySpace_Boundary
            FOREIGN KEY (BoundaryID) REFERENCES Boundary (BoundaryID) ON DELETE No
            Action ON UPDATE No Action;
ALTER TABLE Edge ADD CONSTRAINT FK_Edge_Boundary
            FOREIGN KEY (BoundaryID) REFERENCES Boundary (BoundaryID) ON DELETE No
            Action ON UPDATE No Action;
ALTER TABLE Generalspace ADD CONSTRAINT FK_GeneralSpace_NavigableSpace
            FOREIGN KEY (GeneralspaceID) REFERENCES Navigablespace
            (NavigablespaceID) ON DELETE No Action ON UPDATE No Action;
ALTER TABLE LA_BAUnitLA_PartyPortfolio ADD CONSTRAINT
            FK_LA_BAUnitLA_PartyPortfolio_baunit
            FOREIGN KEY (baunit) REFERENCES La_baunit (Uid) ON DELETE No Action ON
            UPDATE No Action;
ALTER TABLE LA_BAUnitLA_SpatialUnitOverview ADD CONSTRAINT
            FK_LA_BAUnitLA_SpatialUnitOverview_baunit
            FOREIGN KEY (baunit) REFERENCES La_baunit (Uid) ON DELETE No Action ON
```

```
            UPDATE No Action;
ALTER TABLE LA_BAUnitSpace ADD CONSTRAINT FK_LA_BAUnitSpace_space
            FOREIGN KEY (space) REFERENCES Space (SpaceID) ON DELETE No Action ON
            UPDATE No Action;
ALTER TABLE LA_BAUnitSpace ADD CONSTRAINT FK_LA_BAUnitSpace_baunit
            FOREIGN KEY (baunit) REFERENCES La_baunit (Uid) ON DELETE No Action ON
            UPDATE No Action;
ALTER TABLE LA_PartyLA_SpatialUnitOverview ADD CONSTRAINT
            FK_LA_PartyLA_SpatialUnitOverview_party
            FOREIGN KEY (party) REFERENCES La_party (Pid) ON DELETE No Action ON
            UPDATE No Action;
ALTER TABLE La_right ADD CONSTRAINT FK_LA_Right_LA_RRR
            FOREIGN KEY (La_rightID) REFERENCES La_rrr (Rid) ON DELETE No Action ON
            UPDATE No Action;
ALTER TABLE La_rrr ADD CONSTRAINT FK_LA_RRR_rrrParty
            FOREIGN KEY (party) REFERENCES La_party (Pid) ON DELETE No Action ON
            UPDATE No Action;
ALTER TABLE La_rrr ADD CONSTRAINT FK_LA_RRR_unitRrr
            FOREIGN KEY (unit) REFERENCES La_baunit (Uid) ON DELETE No Action ON
            UPDATE No Action;
ALTER TABLE LA_RRRLA_PartyPortfolio ADD CONSTRAINT FK_LA_RRRLA_PartyPortfolio_rrr
            FOREIGN KEY (rrr) REFERENCES La_rrr (Rid) ON DELETE No Action ON UPDATE
            No Action;
ALTER TABLE LA_RRRLA_SpatialUnitOverview ADD CONSTRAINT
            FK_LA_RRRLA_SpatialUnitOverview_rrr
            FOREIGN KEY (rrr) REFERENCES La_rrr (Rid) ON DELETE No Action ON UPDATE
            No Action;
ALTER TABLE La_spatialunit ADD CONSTRAINT FK_LA_SpatialUnit_suHierarchy
            FOREIGN KEY (su2) REFERENCES La_spatialunit (Suid) ON DELETE No Action
            ON UPDATE No Action;
ALTER TABLE La_spatialunit ADD CONSTRAINT FK_LA_SpatialUnit_su
            FOREIGN KEY (space) REFERENCES Space (SpaceID) ON DELETE No Action ON
            UPDATE No Action;
ALTER TABLE LA_SpatialUnitLA_BAUnit ADD CONSTRAINT FK_LA_SpatialUnitLA_BAUnit_su
            FOREIGN KEY (su) REFERENCES La_baunit (Uid) ON DELETE No Action ON
            UPDATE No Action;
ALTER TABLE LA_SpatialUnitLA_BAUnit ADD CONSTRAINT FK_LA_SpatialUnitLA_BAUnit_baunit
            FOREIGN KEY (baunit) REFERENCES La_spatialunit (Suid) ON DELETE No
            Action ON UPDATE No Action;
ALTER TABLE Navigableboundary ADD CONSTRAINT FK_NavigableBoundary_Boundary
            FOREIGN KEY (NavigableboundaryID) REFERENCES Boundary (BoundaryID) ON
            DELETE No Action ON UPDATE No Action;
ALTER TABLE Navigablespace ADD CONSTRAINT FK_NavigableSpace_Space
            FOREIGN KEY (NavigablespaceID) REFERENCES Space (SpaceID) ON DELETE No
            Action ON UPDATE No Action;
ALTER TABLE Node ADD CONSTRAINT FK_Node_Space
            FOREIGN KEY (SpaceID) REFERENCES Space (SpaceID) ON DELETE No Action ON
            UPDATE No Action;
ALTER TABLE NodeEdge ADD CONSTRAINT FK_NodeEdge_Edge
            FOREIGN KEY (EdgeID) REFERENCES Edge (EdgeID) ON DELETE No Action ON
            UPDATE No Action;
ALTER TABLE NodeEdge ADD CONSTRAINT FK_NodeEdge_Node
            FOREIGN KEY (NodeID) REFERENCES Node (NodeID) ON DELETE No Action ON
            UPDATE No Action;
ALTER TABLE Route ADD CONSTRAINT FK_Route_Edge
            FOREIGN KEY (EdgeID) REFERENCES Edge (EdgeID) ON DELETE No Action ON
            UPDATE No Action;
ALTER TABLE Route ADD CONSTRAINT FK_Route_Node
            FOREIGN KEY (NodeID) REFERENCES Node (NodeID) ON DELETE No Action ON
            UPDATE No Action;
ALTER TABLE Transferspace ADD CONSTRAINT FK_TransferSpace_NavigableSpace
            FOREIGN KEY (TransferspaceID) REFERENCES Navigablespace
            (NavigablespaceID) ON DELETE No Action ON UPDATE No Action;
```