SPATIAL DMBS ARCHITECTURE FOR A FREE AND OPEN SOURCE BIM

S. Logothetis[a], E. Valari[a], E. Karachaliou[a], E. Stylianidis[a]

[a] School of Spatial Planning and Development, Faculty of Engineering, Aristotle University, Thessaloniki, 54124, Greece
slogothet@auth.gr; evalarig@auth.gr; ekaracha@auth.gr; sstyl@auth.gr

Commission II

KEY WORDS: BIM, IFC, FOSS, DBMS

ABSTRACT:

Recent research on the field of Building Information Modelling (BIM) technology, revealed that except of a few, accessible and free BIM viewers there is a lack of Free & Open Source Software (FOSS) BIM software for the complete BIM process. With this in mind and considering BIM as the technological advancement of Computer-Aided Design (CAD) systems, the current work proposes the use of a FOSS CAD software in order to extend its capabilities and transform it gradually into a FOSS BIM platform. Towards this undertaking, a first approach on developing a spatial Database Management System (DBMS) able to store, organize and manage the overall amount of information within a single application, is presented.

1. INTRODUCTION

1.1 An overview of BIM field, technology and market

In the late 1970's and early 1980's, building modelling based on 3D solid modelling appeared and the available CAD systems of that period started to extend their capabilities. The concept of object-based parametric modelling came into light, by representing objects through parameters and rules which determined the geometry along with non-geometric properties and features (Eastman et al., 2008).

Nowadays, the parametric modelling in the field of AEC (Architecture, Engineering, Construction), is mostly being represented by the so called "BIM models". BIM models face the object as an integrated entity allowing different level of information (geometric and non-geometric) to be included as well as pointing the relations among all the items of the model (Czmocha and Pekala, 2014). BIM is considered as an advanced approach which extends the capability of the traditional CAD design methodology by applying and defining intelligent relationships between the elements in the designed model (Garagnani and Manferdini, 2013).

BIM can be defined as a five dimensions (5D: width, height, depth, time and cost) digital representation of a structure with its physical and functional characteristics including parametric rules and data attributes for each element (Hergunsel, 2011), but also as "a methodology to manage the essential building design and project data in digital format throughout the building's life-cycle" (Penttilä, 2006).

Among the most important features and capabilities of BIM technology are considered (Logothetis et al., 2015), (Popov et al., 2006):

- Its ability to manage, store, share and exchange 3D geometries.
- Its capacity to carry all the information related to each object and element (geometric, descriptive, and functional).
- The combination of the graphical interface with the information flows and process descriptions.
- The cover of all stages of a project lifecycle, from the design to the implementation, allowing and facilitating the cooperation of the various stakeholders involved in the project faster, effective and with lower cost.

Regarding BIM's activities areas, these can be distinguished into technology, process and policy field (Succar, 2009) with BIM models to be predominantly used in any kind of AEC application and lifecycle management of a building or other structure (Azhar, 2011). In addition, the past decade, a remarkable development and use of BIM technology in the field of cultural heritage has been occurred (Heritage BIM - HBIM). More specifically, heritage elements collected, e.g. using a terrestrial laser scanner, and produced photogrammetric survey data are converted into parametric objects (Dore and Murphy, 2012) with specific standards and protocols that can be exchanged and processed from different professionals (Osello and Rinaudo, 2016) in order to document any size or complex form of cultural heritage (Singh et al., 2011) and manage various phases of its uses and conservation.

In terms of market, it is estimated that the BIM market potential is expected to grow from USD 3.16 Billion in 2016 to USD 7.64 Billion by 2022 (MarketandMarkets Report, 2017, Web1). As drivers of BIM's market growth are presumed to be the communication and coordination benefits that they offer throughout the management process. On the other hand, high costs of BIM (commercial) software and long training are considered as the main barriers that inhibit its uptake. In fact, recent research (Logothetis & Stylianidis, 2016) revealed that except of a few, accessible and free BIM viewers there is no available, a valuable and comprehensive FOSS BIM software to be used for the overall BIM process.

1.2 The FOSS BIM concept

As Steiniger and Hunter (2012) outline, the specific benefits of using FOSS are its freedom to be run it for any purpose, to be

adapted for the users' own needs, to be improved and have such improvements released to the public.

Considering BIM as the next generation of CAD systems and taking into account that amongst the BIM software that have been developed so far, there are no open source integrated platforms able to cover all stages of a BIM process, the current work aims to use a FOSS CAD software in order to extend its capabilities and develop a FOSS BIM plugins' ecosystem.

In order the current research to meet its objectives, the FOSS parametric FreeCAD (Web 2) software was selected to be used as the primary infrastructure for the FOSS BIM development. Its features:

- Open source and not freeware thus allowing access to the source code and extend its functionalities.
- Capable of accepting extensions, with the preferable Application Programming Interface (API) to be in Python programming language.
- Compatible with all types of operating systems (UNIX, Windows, Mac).
- Commonly used and accepted by the users.

In fact, these features render FreeCAD as selectable option for our overall approach towards an entire BIM ecosystem around FreeCAD as illustrated in Figure 1 (Logothetis et al, 2017).

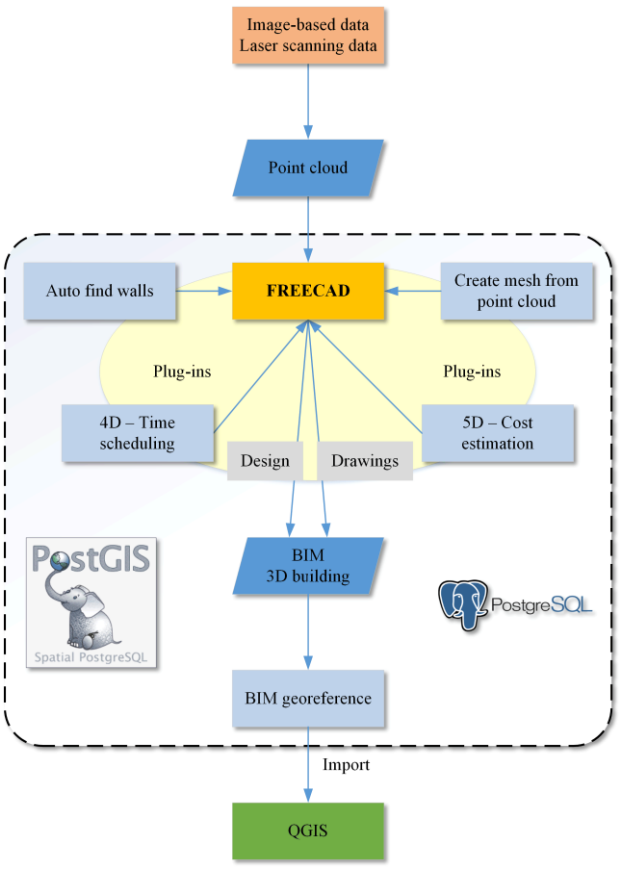

Figure 1: BIM ecosystem around FreeCAD

Towards the development of the FOSS BIM into the FreeCAD environment the first and most important step is to ensure that data can be categorized, structured and managed to meet the needs of the users to whom it is addressed. Therefore, it is necessary to develop a DBMS which will be able to store, organize and manage an enormous amount of information within a single application.
Within this concept, this paper is presenting the integration approach for an open source spatial DBMS into an open source CAD, namely FreeCAD, which we envision to become an open source BIM.

2. IFC DATA MODEL

Industry Foundation Classes (IFC) data model is an open file format developed by buildingSMART; a worldwide authority driving transformation of the built environment through creation and adoption of open, international standards (Web 3). It uses an International Organization for Standardization (ISO) standard that defines and describes all the elements of a building in a project, therefore allowing interoperability among stakeholders so as to share information encoded in a unique and mutual understandable way.

In particular, IFC model is an entity-relationship model which uses the "EXPRESS" international standard data definition language and consists of a set of schemas each of them corresponds to one IFC layer. Every IFC layer includes several classes.

The three fundamental of them are (Wix, 2007):

- Object classes that record object characteristics and types (IfcObject).
- Relations classes which are defining the relationship among the objects (IfcRelationship).
- Properties classes that refer to the different types of property that follows an object (IfcPropertyDefinition).

BIM systems are using IFC data model as a collaboration data format (Eynon, 2016) aiming to define how information should be provided/stored in all stages of a project lifecycle, while maintaining data for geometry, cost, time management etc. (Wix, 2007).

3. AN OPEN SOURCE SPATIAL DBMS FOR EXTENDING A CAD SYSTEM INTO BIM

The main goal of this work is to design and develop an open source spatial DBMS in order to extend an open source CAD system, namely FreeCAD, into BIM. A spatial DBMS is not supported by the FreeCAD software as in the case of other quite similar software. FreeCAD is using files in order to handle and store the data. There is a standard zip file format called "FCStd" which holds files in a certain structure (e.g. document.xml file which stores all geometric and parametric objects definitions etc.)

However, the file management is a very consuming procedure and has many disadvantages especially with large files as in FreeCAD. A DBMS is an alternative and more efficient way for data management. In the file system approach each application has its own private files. These files cannot be shared between multiple applications. Such an approach may reflect to extensive redundancy which leads in waste of storage space. By having all the data stored in a centralized database, most of this can be avoided, while it is not possible that all redundancy should be eliminated. By using a DBMS many problems can be avoided as well as benefiting from its use. In particular, backup and data recovery is supported as well as data sharing and integration. On

the other hand, problems like data inconsistency can be avoided (Silberschatz et al., 2011).

3.1 DBMS and their overall architecture

A typical architecture for a DBMS is organized into layers, as illustrated in Figure 2 which gives one of the possible architectures. Each system could have its own variation. The main layers which are supported by a DBMS are (Date, 2013):

- disk space management
- buffer management
- file and access methods
- relational operations
- query optimization and execution layer
- concurrency control
- recovery

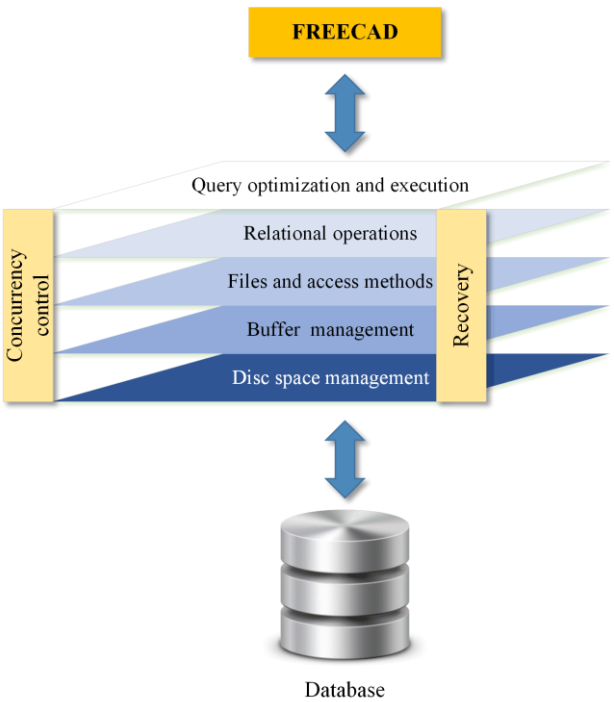

Figure 2: DBMS and the layered architecture

3.2 PostgreSQL for FreeCAD

The most suitable open source spatial DBMS system in order to be connected with the FreeCAD software is the PostgreSQL with PostGIS plugin.

The reasons which are leading to this selection are described below:

- PostgreSQL is free and open source software while the selected FreeCAD is also open source software. As a result this integrated approach will be a totally free and open source as well, having all the privileges of these platforms.
- PostgreSQL is widely used in large systems where read and write speeds are crucial and data needs to be validated. Furthermore, PostgreSQL performance is utilized best in systems requiring execution of complex queries. The objects which are produced from the FreeCAD software are usually very large and complex objects.
- PostGIS is an open source software which handles geographic objects for the PostgreSQL object-relational database. Our ultimate goal is to connect the FreeCAD software with a DBMS system which can support spatial information, such as the one we are suggesting.

Figure 3 presents the connection of the PostgreSQL with PostGIS plugin.

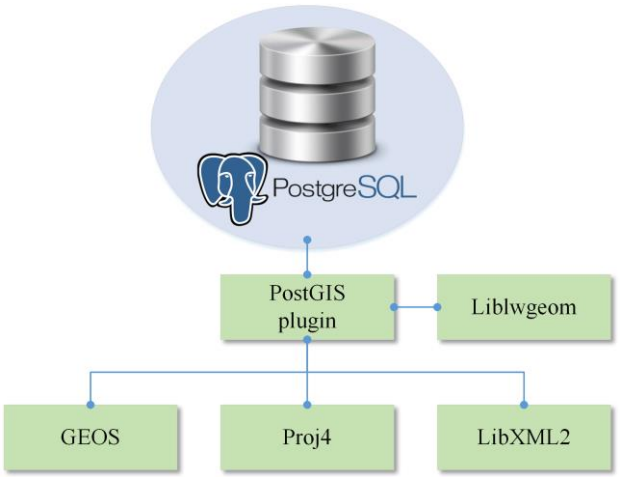

Figure 3: PostGIS architecture

4. APPLICATION

Our proposal aims at using PostgreSQL DBMS as a solution to store IFC structures (including data) in a database.

4.1 PostgreSQL from IFC

After an extensive research, the DBMS which will be used in order to be stored the IFC data is PostgreSQL. In the following we describe the structure a database that should have in order to store the entities and attributes.

The IFC specification is defined with EXPRESS data modelling language. Data types contain: “enumeration”, “select” and “primitive” type. Enumeration allows an attribute value to be one of multiple enumeration values identified by name. As the enumeration values in IFC are predefined strings, we use VARCHAR (Figure 4) in PostgreSQL to substitute enumeration.

```
CREATE TABLE “EntityName”
(
    “ID_EntityName” serial PRIMARY KEY,
    “Attribute1”, VARCHAR (150),
    “Attribute2”, VARCHAR (150),
    “Attribute3”, VARCHAR (150),
    “Attribute4”, VARCHAR (150);
)
```

Figure 4: SQL: CREATE TABLE statement

Primitive (String, Real, Integer, Logical, Boolean, or Binary as defined in ISO 10303-11) data type are simple and atomic, while the composite types can be recursively constructed starting from primitive type. The Select is the enumeration of entities. This means that the types of select instances can be any one of

candidate entities. It is similar to the concept of multiple inheritance in Python. So, the Select type is essentially entity type. IFC entity describes only data structure. In PostgreSQL is redefined with SQL according the mapping rules. Therefore, if the entity A contains an attribute, whose type is the entity B, then entity B has to be defined in advance. Figure 5 illustrates the mapping e.g. of the IfcPersonAndOrganization from IFC documentation to PostgreSQL (Li et al., 2016).

IfcPersonAndOrganization in EXPRESS specification

```
ENTITY
IfcPersonAndOrganization;

ENTITY
IfcPersonAndOrganization;

The Person      : IfcPerson;
TheOrganization: IfcTheOrganization;
Roles           : OPTIONAL LIST [1:?] OF IfcActorRole;
END_ENTITY;
```

IfcPersonAndOrganization in PostgreSQL

```
CREATE TABLE "IfcPersonAndOrganization"
(
    "ID_IfcPersonAndOrganization" serial PRIMARY KEY,
    "ThePerson" VARCHAR (150),
    "TheOrganization" VARCHAR (150),
    "Roles" VARCHAR (150),
)
```

Figure 5: Mapping of IfcPersonAndOrganization from IFC to PostgreSQL

Figure 6 illustrates the open source DBMS ecosystem for FreeCAD. Firstly, the system parsing the computer interpretable IFC official release to identity the type definitions with the help of the IFCSchemaReader, a Python open source script that respectively reading the IFC2x3 or IFC4 specifications. Then, based on mapping rules, rewriting the type definition with SQL and generating the types in PostgreSQL by executing the script (Web4). In addition, using a CAD-BIM software (e.g. FreeCAD), the users export the object to an IFC file. Lastly, running the IFCFileReader the data are parsing from IFC file and stored into database in the corresponding tables of entities.

4.2 IFC schema reader

As a first step, we have the creation of a PostgreSQL database with tables and columns that inserted from the EXPRESS longform schema definition of the IFC2x3 and IFC4 files (Web 5). For this reason, the appropriate script is created with Python programming language. Inside the python code executed SQL code (Figure 7) is used to create the tables (entities) and corresponding columns (attribute definitions).

```
CREATE TABLE "IFCEXTRUDEDAREASOLID"
(
    "ID_fcExtrudedAreaSolid" serial PRIMARY KEY,
    "SweptArea", VARCHAR (150),
    "Position", VARCHAR (150),
    "ExtrudedDirection", VARCHAR (150),
    "Depth", VARCHAR (150);
)
```

Figure 7: SQL that creates an indicative table of an entity

Algorithm 1 GET ATTRIBUTES DATA FROM IFC SCHEMA

Input: IFC shema
Output: Database schema

```
1:  def getAttributes(self, name):
2:      # Get all attributes f an entity, including supertypes
3:      ent = self.entities[name]
4:      attrs = []
5:      while ent != None:
6:          this_ent_attrs = copy.copy(ent["attributes"])
7:          this_ent_attrs.reverse()
8:          attrs.extend(this_ent_attrs)
9:          ent = self.entities.get(ent["supertype"], None)
10:     end while
11:     attrs.reverse()
12: return;
```

Figure 8: Get attributes data from IFC schema

The functions that pick up the entities and attributes respectively are the getEntities and getAttributes. The function getEntities is

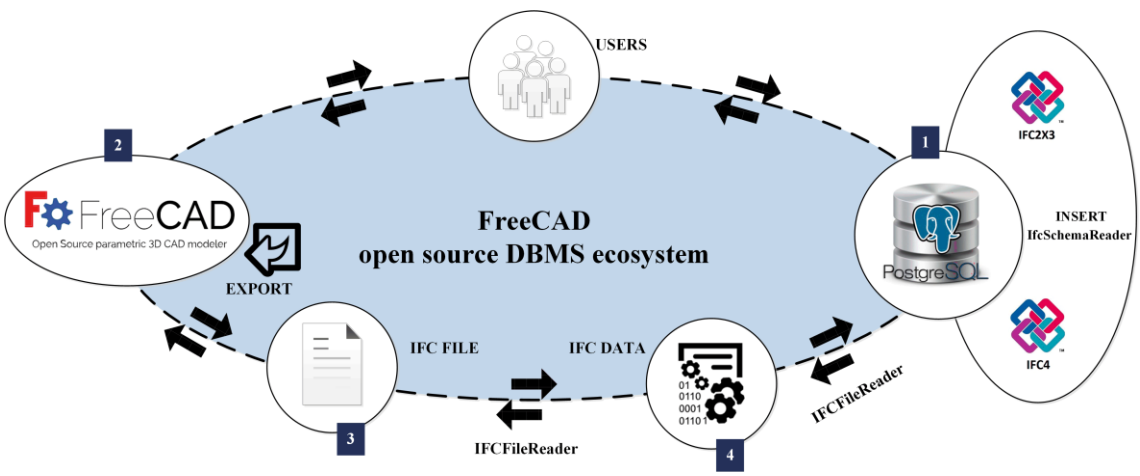

Figure 6: Workflow chart for FreeCAD open source DBMS ecosystem

responsible for parsing all the possible entities from the schema and returns a dictionary of the form, while getAttributes (Figure 8) is responsible to obtain all attributes of an entity, including supertypes.

There are various commercial as well as open source tools available for working with IFC data. Some of the open source python modules available that used in our project are:

- IfcOpenShell: is an open source software module that helps software developers and users to work with the IFC file format. IfcOpenShell uses the Open CASCADE Community Edition to convert the geometry of IFC files into geometry that any BIM modelling software can understand (Dhillon et al., 2014).
- Psycopg: is a PostgreSQL database adapter for the Python. The main features are the thread safety and the complete implementation of the "Python DB API 2.0" specification. It was designed for heavily multi-threaded applications that create and destroy many of cursors and make a large number of concurrent "UPDATE"s or "INSERT"s (Web 6).
- IFCSchemaReader: is an open source script that detect all the possible entities and types of an IFC2x3 or IFC4 schema. Also, the script get all attributes of an entity, including the supertypes (Web 7).

4.3 IFC data extraction from IFC file

Even though there are several open source and commercial tools for working with IFC files but most of them have weaknesses like:

- None stored the data in a database.
- Most of them work with IFC2X3 which is not the latest version of IFC documentation; the latest is IFC4.
- Not all information is provided in detail.
- Some tools are outdated.

Considering these weaknesses, a project was created named IFCFileReader, which works with IFC files that created for the FreeCAD software. IFCFileReader is GUI based application to read IFC files and to export them into PostgreSQL database. The project created in Python and is capable to distinguish the data of the entities in the IFC file. The experimental test has showed that the project can handle the storage of hundreds data of the entities per file with no problem. It can parse IFC2X3 as well as IFC4 documentation files (Dhillon et al., 2014).

IFCFileReader is not thoroughly tested and is currently in pre-alpha version.

Figure 9 shows the framework of IFCFileReader in which a simple IFC file (model: 3D_wall.ifc) is created using FreeCAD. This model is exported in IFC format, which can be IFC2X3 or IFC4. IFCFileReader works on IFC file to create a PostgreSQL database file as output which contains extracted data from IFC entities.

Features of IFCFileReader:

- Dynamic EXPRESS parser (Figure 10): Quickly parsing of any IFC file schema (IFC2x3 or IFC4).
- Saves to PostgreSQL database file (Figure 11): The output of parsed model is stored in tables (the columns are the attributes). It can write up to hundreds of attributes of entities in few minutes.
- Console output: It has the capability to output the result in a Graphical User Interface (GUI) console.

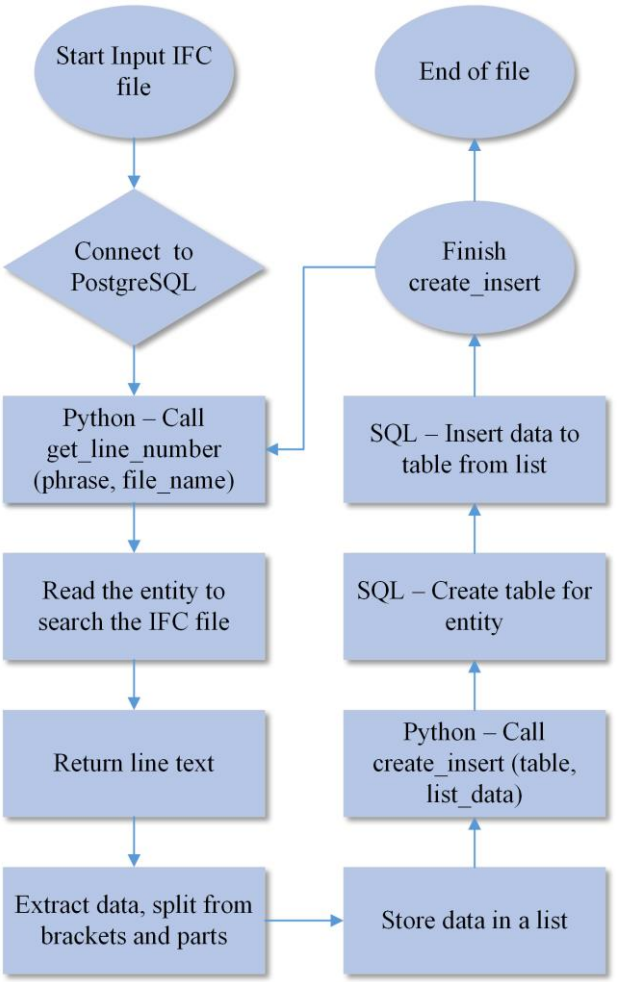

Figure 9: Diagram that shows the store method of IFC data in PostgreSQL

Algorithm 2 EXTRACT ATTRIBUTES FROM ENTITIES

Input: IFC entities
Output: Entities' attributes

```
    # Search for a phrase and return the content of the line
    # being displayed
1:  def getAttributes(self, name):
2:      with open(file_name) as f :
3:      for i, line in enumerate(f, 1) :
4:          if phrase in line:
5:          return line

    # Processing text into parentheses
6:      get_line_number ('IFCEXTRUDEDAREASOLID',)
        'wall.ifc'
7:      n = s[s.find("(") + 1 : s.find(")")]
8:      print n

    # Splits words using comma as separator
9:      mylist = n.split(',')
10:     print mylist
```

Figure 10: Extract attributes from entities

Algorithm 3 SAVE IFC ENTITY TO THE DATABASE

Input: IFC entity
Output: Create an entity table in the database

```
conn = psycopg2.connect("dbname =' wall_db2',
user =' postgres', password =' 123'")
    cur = conn.cursor()
    query = INSERT INTO
    "IFCEXTRUDEDAREASOLID"("SweptArea","Position",
     "ExtrudedDirection","Depth")
    VALUES(%s,%s,%s,%s);
        data = (mylist[0], mylist[1], mylist[2], mylist[3])
        this_entattrs.reverse()
        cur.execute(query, data)
conn.commit()
```

Figure 11: Save IFC entity to the database

4.4 IFC data access in PostgreSQL

Data from the IFC file (3D_wall.ifc) is stored into the PostgreSQL database, in tables of the corresponding entities as shown in Figure 12. With the SQL SELECT (Select * from "IFCEXTRUDEDAREASOLID";) statement the 3D view of the object is presented in the FreeCAD, using the appropriate API (FreeCADGui API or Base API). The FreeCADGui API contains everything related to the user interface and the 3D views while the Base API contains constructors for different types of objects heavily used in FreeCAD. Also, an appropriate interface (GUI) for the software is under development, which will present the IFC data like a table-view form. The stage of the script development is Pre-Alpha (Python script before formal testing).

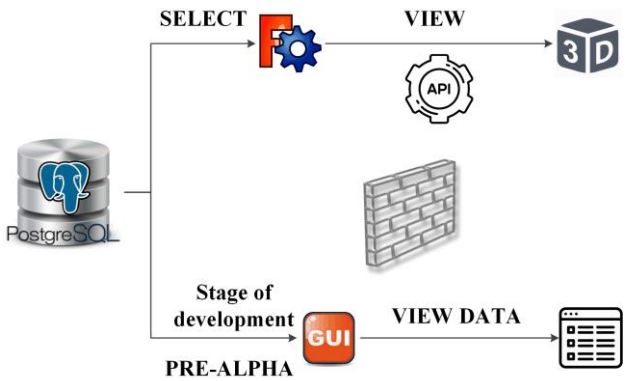

Figure 12: How to access IFC data in PostgreSQL

5. CONCLUSIONS & FUTURE WORK

The current work introduces the idea of using a FOSS CAD environment in order to develop an ecosystem of BIM plugins that will facilitate a comprehensive and integrated BIM model analysis through a FOSS solution. The first step towards this attempt concerns the development of an open source spatial DBMS (PostgreSQL) to support data management.

By selecting as the referred FOSS CAD software to be the "FreeCAD" we proceed to the development of the PostgreSQL spatial DBMS inside software's environment.

The next steps in our research will focus on storing issues -create an IFC server- as well as on GUI development so as to display the IFC data into FreeCAD's environment.

Moreover, we will continue on developing the FreeCAD DBMS for IFC files aiming to run a full test in a pilot project.

Finally, we work on optimizing the current DBMS for fast data access efficiency.

REFERENCES

Azhar, S. 2011. Building Information Modeling (BIM): Trends, Benefits, Risks, and Challenges for the AEC Industry. *Leadership and Management in Engineering*, Vol. 11(3), pp. 241 – 252.

Czmocha, I. and Pekala, A., 2014. Traditional Design versus BIM Based Design. *Procedia Engineering*, Vol. 91, pp. 210-215.

Date, C., 2013. Basic Database Concepts. *Relational Theory for Computer Professionals.*

Dhillon, R.K., Jethwa, M., Rai, H.S., 2014. Extracting Building Data from BIM with IFC. Int. J. Recent Trends Eng. Technol. 11, 202–210. doi:01.IJRTET.11.1.1500

Dore, C. and Murphy, M., 2012. Integration of historic building information modeling and 3D GIS for recording and managing cultural heritage sites. *18th International Conference on Virtual Systems and Multimedia: "Virtual Systems in the Information Society"*, pp. 369–376.

Eastman, C., Teicholz, P., Sacks, R. and Liston, K., 2008. BIM Tools and Parametric Modeling. *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers, and Contractors*, pp. 25-65.

Eynon, J., 2016. Construction Manager's BIM Handbook.

Garagnani, S. and Manferdini, A., 2013. Parametric Accuracy: Building Information Modeling Process Applied to the cultural heritage preservation. *3D-ARCH 2013 – 3D Virtual Reconstruction and Visualization of Complex Architectures*, pp. 87-92.

Hergunsel, M. F., 2011. Benefits of Building Information Modeling for Construction Managers and BIM-based scheduling. MSc thesis, Worcester Polytechnic Institute.

Li, H., Liu, H., Liu, Y., Wang, Y., 2016. An object-relational IFC storage model based on oracle database. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Volume XLI-B2, 625–631. doi:10.5194/isprsarchives-XLI-B2-625-2016.

Logothetis, S., Delinasiou, A. and Stylianidis, E., 2015. Building Information Modelling for Cultural Heritage: A review. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. II-5/W3, pp. 177-183.

Logothetis, S., Karachaliou, E. and Stylianidis, S., 2017. From OSS CAD to Bim for Cultural Heritage Digital Representation. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. XLII-2/W3, pp. 439-445.

Logothetis, S. and Stylianidis, E., 2016. BIM Open Source Software (OSS) for the documentation of Cultural Heritage. Proc. *8th International Congress on Archaeology, Computer Graphics, Cultural Heritage and Innovation 'ARQUEOLOGICA 2.0'*, pp. 28-35.

Osello, A. and Rinaudo, F., 2016. Cultural Heritage Management Tools: The Role of GIS and BIM. *3D Recording, Documentation and Management of Cultural Heritage.*

Penttilä, H., 2006. Describing the changes in architectural information technology to understand design complexity and free-form architectural expression. *ITCON*, Vol. 11, pp. 395–408.

Popov, V., Mikalauskas, S., Migilinskas, D. and Vainiunas, P., 2006. Complex Usage of 4D Information Modeling Concept for Building Design, Estimation, Scheduling, and Determination of Effective Variant. *Technological and Economic Development of Economy*, Vol. 12(2), pp. 91-98.

Silberschatz, A., Korth, H. and Sudarshan, S., 2011. Database System Concepts.

Singh, V., Gu, N. and Wang, X., 2011. A theoretical framework of a BIM-based multi-disciplinary collaboration platform. *Automation in Construction*, Vol. 20(2), pp. 134–144.

Steiniger, S. and Hunter, A. J., 2012. The 2012 free and open source GIS software map – A guide to facilitate research, development, and adoption. *Computers, Environment and Urban Systems*, Vol. 39, pp. 136-150.

Succar, B., 2009. Building information modelling framework: A research and delivery foundation for industry stakeholders. *Automation in Construction*, Vol. 18(3), pp. 357-375.

Wix, J., 2007. What is IFC? http://www.it.civil.aau.dk/it/education/reports/building_smart/WS3_IDM_WhatIsTheIFCModel.pdf (June 2017)

Web1: http://www.marketsandmarkets.com

Web2: http://www.freecadweb.org

Web3: http://buildingsmart.org

Web4: https://github.com/mvaerle/python-ifc

Web5: http://www.buildingsmart-tech.org

Web6: https://pypi.python.org/pypi/psycopg2

Web7: https://github.com/mvaerle/python-ifc