AN OBJECT-RELATIONAL IFC STORAGE MODEL BASED ON ORACLE DATABASE

Hang Li*, Hua Liu, Yong Liu, Yuan Wang

School of Urban Design, Wuhan University, Wuchang District, Wuhan, Hubei Province, P.R.C – (2014202090009, liu.hua, liuyong, wangyuan0827)@whu.edu.cn

Commission II, ThS14 - Recent Developments in Open Data

KEY WORDS: IFC, Object-Relational, ORDBMS, Oracle, Storage

ABSTRACT:

With the building models are getting increasingly complicated, the levels of collaboration across professionals attract more attention in the architecture, engineering and construction (AEC) industry. In order to adapt the change, buildingSMART developed Industry Foundation Classes (IFC) to facilitate the interoperability between software platforms. However, IFC data are currently shared in the form of text file, which is defective. In this paper, considering the object-based inheritance hierarchy of IFC and the storage features of different database management systems (DBMS), we propose a novel object-relational storage model that uses Oracle database to store IFC data. Firstly, establish the mapping rules between data types in IFC specification and Oracle database. Secondly, design the IFC database according to the relationships among IFC entities. Thirdly, parse the IFC file and extract IFC data. And lastly, store IFC data into corresponding tables in IFC database. In experiment, three different building models are selected to demonstrate the effectiveness of our storage model. The comparison of experimental statistics proves that IFC data are lossless during data exchange.

1. INTRODUCTION

Construction projects are comprehensive and multistage activities which involve design, construction, operation management, supervision and even demolition. Participants in different stages may use different professional applications to facilitate their work. Yet the seamless share of different models is a problem that has long vexed the architects. It is estimated that inefficient interoperability results in a total cost of $15.8 billion per year in US alone (Gallaher, 2004). To solve this problem, buildingSMART alliance® (formerly IAI, Industry Alliance for Interoperability), an international non-profit organization, developed IFC data model to facilitate the exchange of information among software applications used in the construction industry. IFC is a platform neutral and open file format specification. It defines an EXPRESS based entity-relationship model consisting of hundreds of entities organized into an object-based inheritance hierarchy. These entities cover information of every aspect of a building lifecycle. Examples of the entities include building elements such as IfcColumn, geometry such as IfcBoundingBox, and basic constructs such as IfcCartesianPoint.

Currently, IFC data are managed by file system, including ifc (Zhao Y, 2008) and ifcXML (Nour M, 2009) format files. On one hand, although IFC specification covers all the information required in project lifecycle, the architectural professional software are specialized in just one construction stage, loss of information may be occurred during the format conversion among different applications (A. Kiviniemi, 2005). On the other hand, file[1] system does not support the creating, reading, updating and deleting (CRUD) operations on IFC data. Therefore, the current file-based management method weakens the exchangeability of IFC to some extent.

In recent years, studies on database-based storage of IFC data have been continuously carried out. Generally, the researchers take advantage of relational database management systems (RDBMS) to overcome the deficiencies in file-based storage and have made some achievements. But since RDBMS doesn't support the storage of object type data, the conflicts between RDBMS and OO features in IFC are still not resolved (see section 2).

The widespread use of multimedia data and other complex structure data forces DBMS to support features associated with object-orientation, which developed into object-relational database management system (ORDBMS). It is able to store objects as columns of a relation table. Typically, Oracle database had been able to support object-oriented development since 1997. The Procedural Language/Structured Query Language (PL/SQL) acting as an extension of SQL enables developers to declare variable, write loop statement and create procedures, functions, and types. The core concepts of OOP, including inheritance, overloading and polymorphism are also supported by Oracle, which is suitable to reserve the inheritance and containment relationships among IFC entities. Therefore, in this paper, we propose an object-relational IFC storage model based on Oracle database.

In object-oriented programming (OOP) like C++, an object of a subclass type is permitted to be treated as an object of any superclass type, which is called upcasting. It enables developers to build complicated programs using simple syntax. Similarly, in ORDBMS, a column of base type is able to store the instances of all its derived types. In this way, by designing the database, we can just use a few tables to store all IFC instances and no more need to create a table for each entity in IFC, which not only significantly reduces the complexity of the database, but also improves the data management efficiency.

The rest part of this paper is organized as follows. Firstly, the previous work is briefly introduced in Section 2. Secondly, the details of object-relational IFC storage model are discussed in section 3. In section 4, an experiment is presented to verify the feasibility of the model. And in section 5, we extend the storage

* Corresponding author

prototype in practical application situations. Finally, we conclude this work in section 6.

2. RELATED WORK

Researches on improving the storage performance of IFC data had been carried out a long time before. Generally, these studies proposed using databases to substitute the current file-based data management. According to the underlying databases they applied, previous researches can be basically derived into three classes: relational-based, object-based and object-relational based.

Majority of the work store IFC data into traditional RDBMS. Early exploration was carried out by researchers in Technical Research Centre of Finland (VTT, 2002). They developed an IFC-based server called IFC Model Server (IMSVR), which enables IFC compatible applications communicate mutually through the internet. It utilizes EXPRESS XML Schema Converter (EXC) to automatically convert IFC EXPRESS schema to Database Schema (SQL) and thereby store IFC model data into RDB. You et al (You, 2004) use GTCIS2SQL, a relational database implementation of CIS/2, to translate EXPRESS into RDB SQL. Similarly, Eurostep Model Server (EMS) developed by Eurostep (Karstila, 2002) uses Product Model Query Language (PMQL) as a substitute for the XML Parser to convert an IFC file into SQL code.

Although RDBMS has advantage in performance, stability and usability, it has limited supports for object oriented features. First, tremendous efforts are needed to map every entity in IFC into a relational table in RDB. Second, a database with hundreds of tables is unmaintainable and inefficient in data management. So, there is a huge barrier between the Object Relational Mapping (ORM).

To avoid ORM, researchers start trying to store IFC data in object-oriented databases management system (OODBMS). Tanyer and Aouad proposed a 4D CAD system (A.M. Tanyer, 2005), in which the lowest layer is OODB called 'EXPRESS Data Manager' (EDM). Po-Han Chen et al implement an IFC-based information server to facilitate the interoperability among multidisciplinary AEC software applications. The building components are encapsulated in Java Beans (Chen, 2005).

Data in OODB are stored in the form of objects as used in OOP. So it has a natural advantage to express entities in IFC. However, OOBD is far less prevalent than RDB. The application of OODB is just in limited areas, typically geographic information system (GIS). Besides, the standards (e.g. Object Database Standard ODM, Object Query Language) and tools (e.g. DB4O, MyOODB) of OODB are also less popular than those in RDB. Therefore, OODB can hardly meets the requirements of developers.

Object-relational database management system (ORDBMS) is a preferable solution for the storage of IFC data. H. Kang and G. Lee (H. Kang, 2009) develop a set of rules to map the IFC model to an object-relational database. However, no experiments are conducted to verify the feasibility of the idea. Therefore, the high performance storage of IFC data has yet to be satisfactorily solved.

3. OBJECT-RELATIONAL IFC STORAGE SCHEMA

3.1 Mapping Rules

IFC specification is defined with EXPRESS language, in which data types contain primitive, enumeration, selected and entity type. Primitive Data Type are simple and atomic, the other complicated composite types can be recursively constructed starting from primitive type. As shown in Table 1, primitive data types can be directly corresponded to built-in types in Oracle.

IFC Data Type	Oracle Data Type
REAL	NUMBER
INTEGER	INTEGER
NUMBER	INTEGER
STRING/STRING(n)	VARCHAR2(n)
BOOLEAN	NUMBER(1)
LOGICAL	NUMBER(1)
BINARY	BLOB

Table 1. Mapping of primitive data type

Enumeration is similar in concept to "enum" in common programming languages, which 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 to substitute enumeration.

Entity is similar to the term "class" in common programming languages, which most embodies the OO features of IFC. The difference is that entity describes data structure only, but not behaviour such as methods. Every entity in IFC is redefined in Oracle with PL/SQL according the mapping rules. The mapping order of entity matters. If the entity A contains an attribute whose type is entity B, then entity B has to be defined in advance. Table 2 displays the mapping of IfcRoot from IFC to Oracle.

Select is the enumeration of entities, which means that the types of Select instances can be any one of candidate entities. It is similar to the concept of multiple inheritance in C++. Therefore, Select type is essentially entity type.

Beside, reference is widely used in IFC files. Many instances may refer to the same instance, which has only single copy in file. Reference not only reduces the redundancy, but also improves data consistency. To support this property, as shown below, we define a new class 'Reference' to simulate the function of reference in OOP. Just as memory physical address in computer, with table name and ID, we can locate any instance in database.

IfcRoot in IFC Specification	IfcRoot in Oracle database
`ENTITY IfcRoot`	`CREATE OR REPLACE TYPE IfcRoot AS OBJECT`
`   ENTITY IfcRoot`	`(`
`     GlobalId      : IfcGloballyUniqueId;`	`   GlobalId     VARCHAR2(100),`
`     OwnerHistory  : OPTIONAL IfcOwnerHistory;`	`   OwnerHistory IfcOwnerHistory,`
`     Name          : OPTIONAL IfcLabel;`	`   Name         IfcLabel,`
`     Description   : OPTIONAL IfcText;`	`   Description  VARCHAR2(40000)`
`   END_ENTITY;`	`);`

Table 2. Mapping of IfcRoot from IFC to Oracle ORDBMS

```
CREATE OR REPACLACE TYPE Reference AS OBJECT
(
    TableName VARCHAR2(100),
    ID INTEGER
)
```

According to the above mapping rules, types defined in IFC specification can be redefined in Oracle database. First, parsing the computer interpretable IFC official release to identity the type definitions. Here, we finish this work with the help of IfcGearExtender an open source implementation of IFC standard. Second, based on mapping rules, rewriting the type definition with PL/SQL. Finally, generating the types in Oracle by executing the script.

3.2 IFC Database Design

The design of IFC database is supposed to base on relationships of IFC entities, so the amount of the tables and data access efficiency can be balanced. The data schema architecture of IFC defines four conceptual layers, respectively are Resource layer, Core layer, Interoperability layer and Domain layer. Entities defined in the last three layers are all derived from ‘IfcRoot’, which is the most abstract and root class for all IFC entity definitions. The rest entities belonging to Resource layer are not the subtypes of ‘IfcRoot’. They cannot exist independently, but can only exist if referenced (directly or indirectly) by one or more entities deriving from “IfcRoot”. Figure 1 shows how a building is described by IFC entities.

As show in the figure, the first level of IFC tree structure contains three nodes: IfcObjectDefinition, IfcRelationship and IfcPropertyDefinition. They define the most essential three concepts in IFC schema. IfcObjectDefinition is the generalization of any semantically treated thing or process. All the physical products (e.g. wall, beam and furniture) that we can see and touch can be classified as IfcObjectDefinition. IfcRelationship abstracts all the relationships among objects. For instance, building are “aggregated” by each storey. IfcPropertyDefinition defines of all characteristics that may be assigned to objects. Important information like size and material are expressed by this type.
Therefore, tables should be created for the three entities to store the core information in IFC. Object Table can created as follows.

```
CREATE OR REPLACE TABLE OBJECT
(
    OID INTEGER PRIMARY KEY,
    Instance IfcObjectDefinition NOT NULL
    EntityName VARCHAR2(100) NOT NULL
)
```

Notice that the type of column ‘Instance’ is a base class, so all the instances of all its subtypes. Similarly, Relationship Table and Property Table have the same table structure with Object Table. The three tables can store all the instances derived from IfcRoot. However, the rest entities in Resources layer have no common superclass, so their instances can’t be stored in the one table. Here, we create a new class ‘IfcResource’ as the common base class of entities in Resource layer. And then, the Resource Table can be created like the former three ones to store the rest non-IfcRoot instances.

Theoretically, these four tables are sufficient to store any instances in IFC files. However, the records in Resource Table is much more than the other three tables. The reason is that the instances associated to geometry take up a greater proportion in IFC files and are stored in Resources Table. Geometric information is the most important features to a building model,

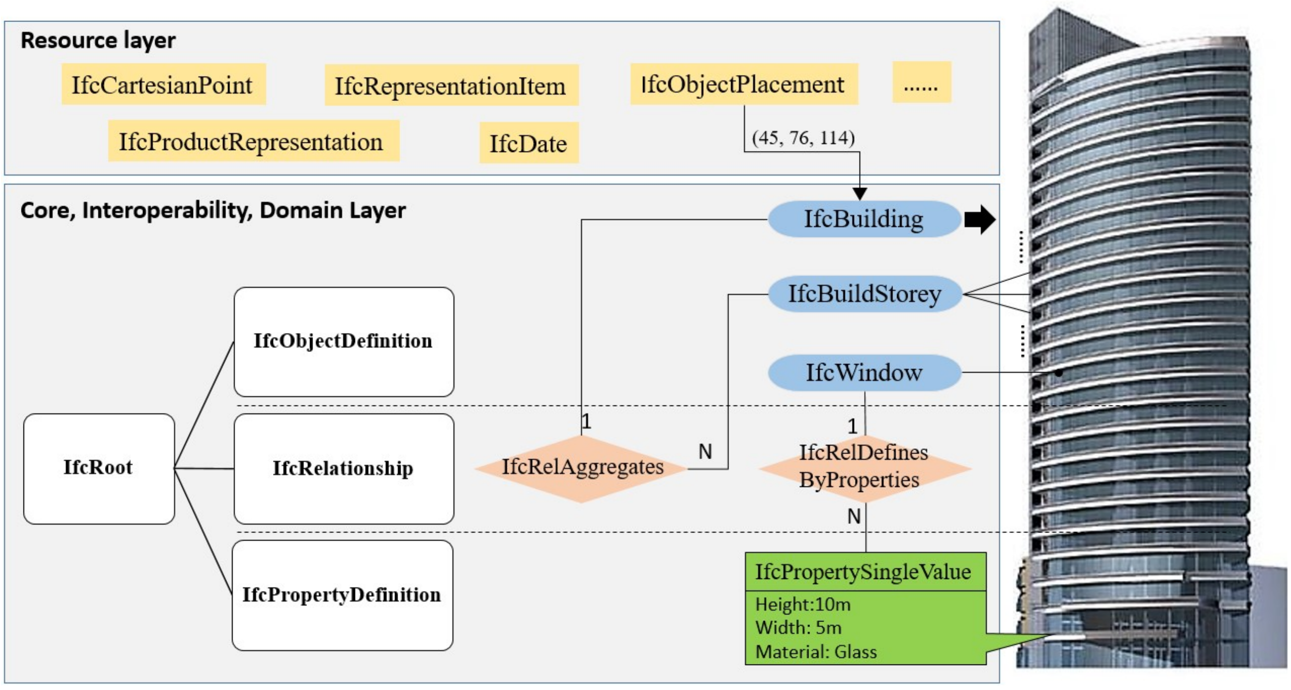
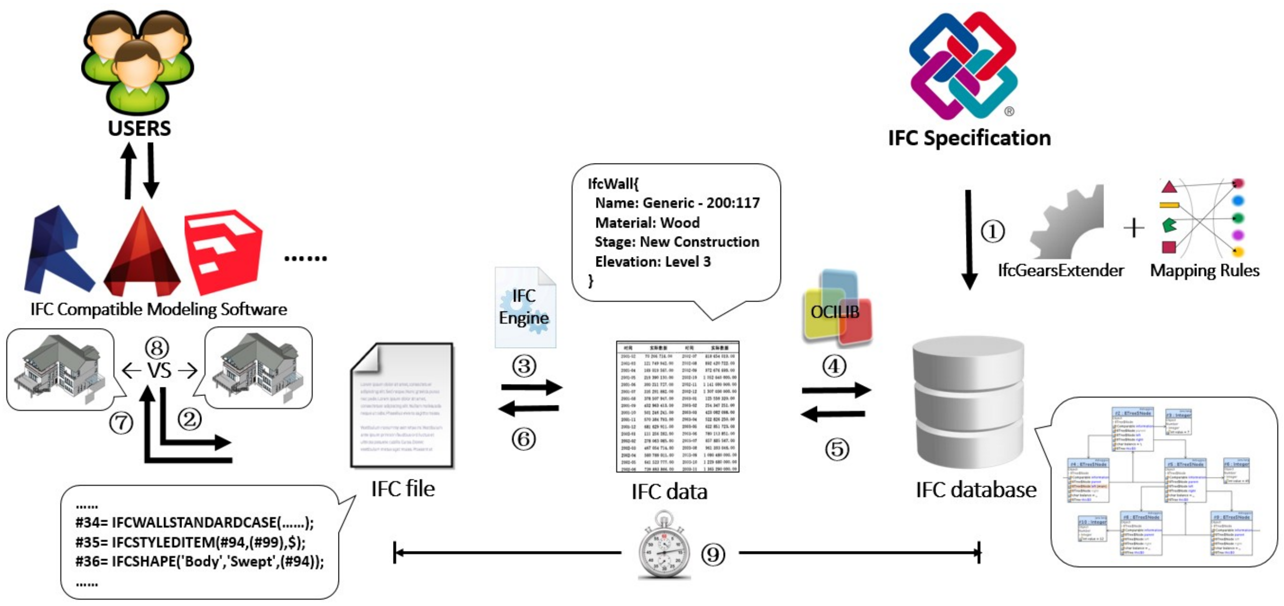


Figure 1. Expressing a building in IFC instances



```
IfcWall{
 Name: Generic - 200:117
 Material: Wood
 Stage: New Construction
 Elevation: Level 3
}
```

```
......
#34= IFCWALLSTANDARDCASE(......);
#35= IFCSTYLEDITEM(#94,(#99),$);
#36= IFCSHAPE('Body','Swept',(#94));
......
```

Figure 2. Generation of IFC database and IFC data access

especially for the models with rich details. Geometric elements in IFC include lines, surfaces and bodies and are finally expressed through sets of Cartesian points. To optimize database schema, we create another two tables Representation Table and Point Table to store geometric information and Cartesian points. They are defined as shown below.

```
CREATE OR REPLACE TABLE REPRESENTATION
(
    RID INTEGER PRIMARY KEY,
    Instance IfcRepresentationItem NOT NULL,
    EntityName VARCHAR2(100) NOT NULL
)
```

```
CREATE OR REPLACE TABLE POINT
(
    PID INTEGER PRIMARY KEY,
    X INTEGER,
    Y INTEGER,
    Z INTEGER
)
```

3.3 IFC Data Extraction from IFC Files

The building modellers commonly do not deal directly with IFC files. But with IFC-compatible modelling software, building models can be easily exported as IFC files and vice versa. IFC certified software[2] are listed in buildingSmart official website. The data in IFC files are organized according to IFC specification, which is not understandable to database. Thus, we must parse the IFC file and then extracted the IFC data before storage. The parsing work can be easier with the help of third-party libraries. IFC Engine DLL[3] is a powerful and stable IFC parser, with which we extract the all data contained in IFC files.

3.4 IFC Data Access in Oracle Database

In database, data is stored with Database Access Interfaces, which provide a set of standard Application Programming Interfaces (APIs) for service callers. Open Database Connectivity (ODBC) and Java Database Connectivity (JDBC) are two most frequently used database APIs. However, they do not properly support Oracle object-oriented features. To solve this problem, Oracle developed a specialized data access interface called Oracle Call Interface (OCI), which acts as an "interpreter" between applications and the low-level database network protocol. It offers a procedural API for using PL/SQL or SQL to query, access, and manipulate data. The OCI library, implemented in C-language, is fast in data access speed but complicated to use, especially for data of object type.

OCILIB[4], an OCI-based library, is an open source and cross platform Oracle driver that delivers efficient access to Oracle databases. OCILIB is suitable to access IFC data in Oracle for it encapsulates OCI with C++ language and screens many details. With the provided APIs in OCILIB, object data of IFC can be easily manipulated in Oracle database.

4. EXPRIMENT

4.1 Experiment Content

In this section, we construct an IFC database based on the proposed object-relational IFC storage model and further, try to store the test objects into the database. The test objects are three building models with different size and complexity. The first model is simple and are just consisted of four walls. The other two models, selected from the built-in samples in Revit 2013, are complex. The experiment is conducted on a PC with Windows 10 operating system, 2.2GHz CPU and 4GB memory.

Specifically, as show in the figure 2, the experiment is divided into five steps. First, parse the IFC official specification and extract data types defined in IFC, and then generate the IFC database based on the mapping rules (①). Second, export the

[2] www.buildingsmart.org/compliance/certified-software
[3] http://rdf.bg/ifc-engine-dll.php?page=products

[4] https://vrogier.github.io/ocilib/

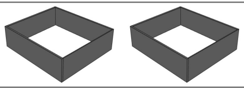
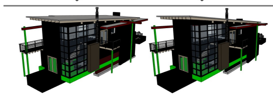
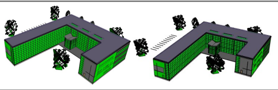
Building Models	File Size(KB)	Instance Amount	Storage Time(s)	Access Time(s)
	31/33	425/425	3’	2’
	5298/5378	102285/102285	166’	143’
	37568/37726	660952/660952	669’	602’

Table 3. Comparison of statistics of different models

building model as IFC file with IFC-compatible software (②). Third, parse the IFC file and extract building data with the APIs provides by IFC Engine DLL (③). Forth, use OCILIB to store the extracted data into IFC database (④). Fifth, after completion of storage, read IFC data from IFC database and restore the data to building model (⑤⑥⑦). These three steps are the inverse operations of previous steps. Finally, compare the generated model with the original one to verify if information errors or loss of data occurs during the data exchange (⑧). In addition, the time of data transferred between IFC file and database is recorded to measure the efficiency of our storage model(⑨).

4.2 Experiment Results

Statistics of experiment are listed in the Table 3. The first three columns contain two sub-columns. The left columns are information related to original model and the right columns are generated ones. The snapshots of two models are listed in the first column. They are identical in appearance and no differences are found even highly enlarged. However, from the second column, there is a little difference in file size. The reason is that the file exported from IFC database is generated by IFC Engine DLL, which has different file organizational rules with Revit. The amount of entities instances are completely the same as shown in the third column, because every instance in IFC file are stored into tables in IFC database. The experiment results indicate that the IFC database support lossless import and export of building models and further prove the feasibility of our proposed object-relational IFC storage model.

From the consuming time listed in last two columns, the storage performance of IFC database is not very satisfactory. That is due to two reasons. First, Oracle database is intrinsic low-efficiency in managing object data. Second, time is spent to ensure lossless storage of IFC data.

5. APPLICATION

The object-relational IFC storage model realizes the database-based management of IFC data. By extending the proposed storage prototype, users are able to develop various IFC-based applications. In this section, we design two applications based on the IFC database. The two applications, include high-level query and construction process simulation are common functional requirements in real construction project. They demonstrate the powerful potential of the new IFC storage model in architectural industry.

5.1 High-level Query

Information of a specific building component, for example the size of the door, can be easily achieved through the traditional SQL. However, architects always concern more about macroscopic information, for example the entire first storey of a building. The high-level query is used to address the issue.

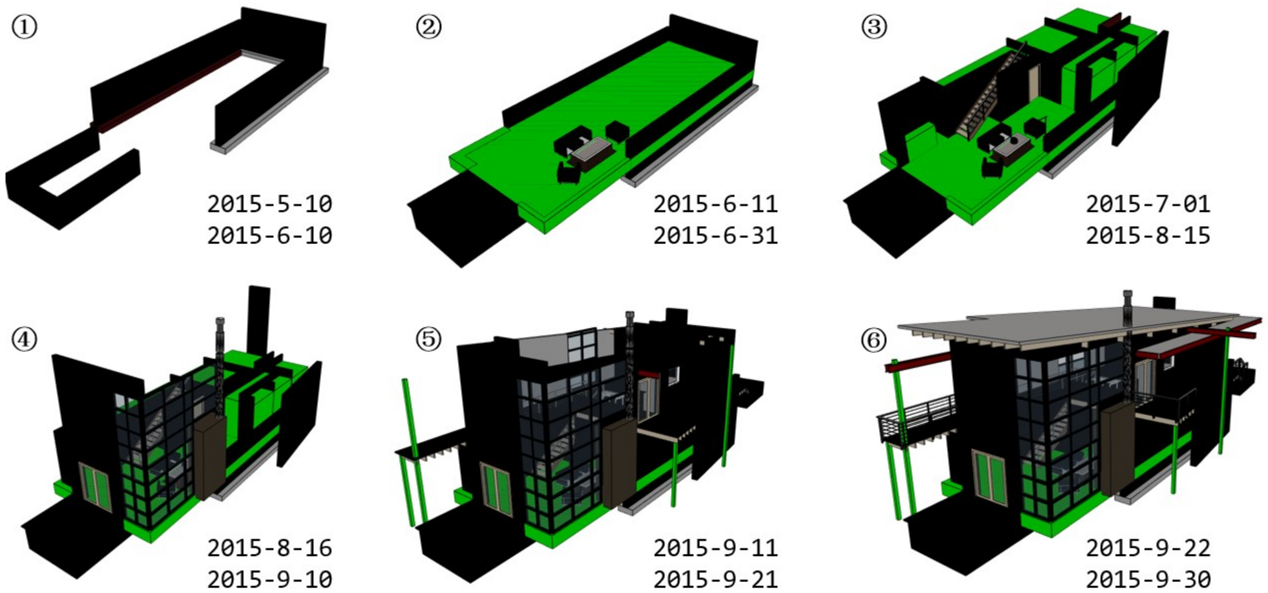


Figure 3. Construction process simulation

In IFC file, a building model is deconstructed into an entity tree, in which nodes represent building elements and links represent the relationships between them. All the relationships between building elements are stored in 'Relationship' table in IFC database. Among all kinds of relationship, decomposition relationship records the including and included implication between building elements. The high-level query to a building element means recursively query the child nodes it contained. To accelerate the speed of high-level query, a new field 'Parent' is added to 'Object' table to record the ID of father node.

For example, in the three-storeyed cottage model, the second floor is named 'Level 2'. Information of the whole floor can be achieved by querying 'Level 2' in IFC database. Figure 4 visualizes the query result of the second floor.

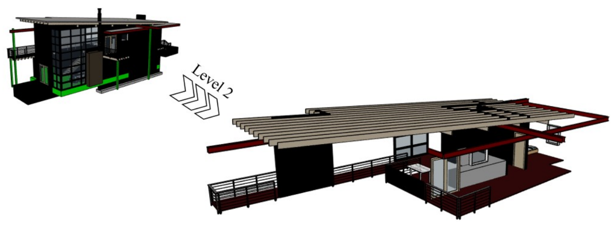


Figure 4. Query for the second floor

Besides, based on high-level query, we can easily implement multi-projects management. One IFC file corresponds to a project. Storing multiple IFC files into the IFC database, the corresponding projects data are generated. We can achieve the information of a specific project by querying the project name. Therefore, with high-level query, architects are able to check any information they wanted.

5.2 Construction Process Simulation

In many cases, the progress of construction site is delayed by unexpected mistakes, for example the collision between walls and pipes. This will result in wasting of resources, time and costs. To avoid this happens, dynamic construction simulation is applied in modern architectural project. To avoid this happens, dynamic construction simulation is applied in modern architectural project.

To implement this functionality, a new table 'Schedule' is needed to store the construction schedules and a new field 'schedule' is added to 'Object' table to establish an association with 'Schedule' table. And then, traversing from start data to the completion data of the project, dynamically rendering the building elements whose finish time is later than the current date. The following pseudo code represents the implementation process.

```
Set StartDate = "2015-5-10"
Set CompletionDate = "2015-9-31";
Set TimeSpan = 1;
For Date = StartDate : CompletionDate
  Render(
     SELECT Object.Instace
     From Object, Schedule
     Where Object.Schedle = Schedule.SID And
     FinishTime > Date);
  Time.AddDays(TimeSpan);
End
```

Several scenes are intercepted from the simulation process and are shown in figure 3.

Not limited to the above two applications, developers are able to implement their own applications based on IFC database.

6. CONCLUSION

In view of the existence of problems in file-based storage of IFC data, in this paper, we propose an object-relational storage model of IFC that realizes the database-based management of IFC data without losing any information. The IFC database acts as a shared building data centre, through which models of any format can be easily exchanged among different software platforms. Therefore, it's a new solution to improve the levels of collaboration in architecture industry. Besides, developers are able to customize their own applications by extend the prototype of the storage model

In the future, we will try to optimize the current storage model for fast data access efficiency. And we also plan to develop a project management system which uses IFC database as its underlying database.

7. REFERENCES

M Z, Zhao Y. Model of next generation energy-efficient design software for buildings [J]. *Tsinghua Science and Technology*, 2008, 13(S1):298-304.

Nour M. Performance of different (BIM/IFC) exchange formats within a private collaborative workspace for collaborative work [J].*Electronic Journal of Information Technology in Construction*, 2009(14):736-752.

A. Kiviniemi, M. Fischer, V. Bazjanac. Integration of multiple product models: IFC model servers as a potential solution, in: 22nd CIB-W78 *Conference on Information Technology in Construction*, 2005.

VTT Building and Transport, SECOM CO. IFC Model Server Development Project [EB/OL]. http://cic.vtt.fi/projects/ifcsvr/.

You, S.-J., D. Yang, and C.M. Eastman. (2004) Relational DB Implementation of STEP based product model, in *CIB World Building Congress*, Toronto, Ontario, Canada

Karstila, K. and Hermio, T. "WebSTEP IFC Model Server project Official webpage", Eurostep, 2002. http://www.eurostep.com/prodserv/ems/ems.html

GALLAHER, M. P., O'CONNOR, A. C., DETTBARN, J. L. & GILDAY, L. T. (2004). Cost Analysis of Inadequate Interoperability in the *U.S. Capital Facilities Industry. NIST.*

van den Helm, Böhms, van Berlo, "IFC-based clash detection for the open-source BIMserver". In *Computing in Civil and Building Engineering*, Proceedings of the International Conference, W. TIZANI (Editor), 30 June-2 July, 2010m, Nottingham, UK, Nottingham University Press, Paper 91, p. 181, ISBN 978-1-907284-60-1

Ya-Hong Lin, Yu-Shen Liu, Ge Gao, Xiao-Guang Han, Cheng-Yuan Lai, Ming Gu. The IFC-based path planning for 3D indoor spaces, *Advanced Engineering Informatics* 27 (2) (2013) 189–205. doi:10.1016/j.aei.2012.10.001.

BIMserver, The Building Information Model server (short: BIMserver) projects. <http://bimserver.org/>, 2012.

Staub, F.S., Fischer, M.: Practical and research issues in using Industry Foundation Classes for construction cost estimating, *CIFE Working Paper* (2000).

A.M. Tanyer, G. Aouad, Moving beyond the fourth dimension with an IFC based single project database, *Automation in Construction* 14 (1) (2005) 15–32.

H. Kang and G. Lee, "Development of an Object-Relational IFC Server", *ICCEM/ICCPM* 2009, Jeju, Korea.

Chen, P.H., Cui, L., Wan, C., Yang, Q., Ting, S.K. and Tiong, R.L.K., 2005,'Implementation of IFC based web server for collaborative building design between architects and structural engineers', in *Automation in Construction*, 14(1), 115–1