ARCHITECTURAL INFORMATICS SOCIETY

The study of Building Information Model server focusing on IFC to RDB conversion

Zhenglu ZHU[1], Kazuya SHIDE[2]

[1] Ph.D. Candidate, Graduate School of Eng, Shibaura Institute of Technology
Inkx, Inc
[2] Prof, School of Architecture, Shibaura Institute of Technology

* na21102@shibaura-it.ac.jp

Abstract

This study defines and establishes all feasible database table design solutions for storing IFC data into RDB and evaluates the relative performance of each solution through query experiments to find a design solution that maximizes both performance and data consistency. As a result, we believe that the method of creating a table for each entity and maintaining the inheritance relationship by giving it a foreign key is the most appropriate method for constructing an IFC relational database at present. As BIM continues to develop, there is a growing momentum to share and utilize BIM data on cloud servers. We will contribute to the realization of such an ideal.

Keywords

Building Information Model, buildingSMART International, IFC Server, Relational Database, SQL, Auto-mapping

1. Background

1.1. The importance of BIM server

With the rise of smart city concept, Building information Model (BIM) [1] and Geographic Information System (GIS) technologies are becoming more and more important in acquiring, managing, and analyzing building and city information. BIM provides accurate information model of single building for city construction management, focusing on the fine representation of local single building. GIS, on the other hand, can provide a basic framework for urban construction management, focusing on the representation of outdoor information. Therefore, by leveraging the technical advantages of both systems, integrating BIM and GIS is an effective way to realize digital and smart city construction [2], [3]. Prior studies in the field of integrating BIM and GIS rely on two data standards, CityGML and IFC, and employ the data format conversion or standard extension approaches [4]. Prior research related to the data format conversion is overwhelmingly focused on the conversion from IFC to CityGML, which is currently the mainstream way to integrate BIM and GIS [5]. Due to the large differences in geometric expressions and object semantics between the two data models, only 60-70 of the approximately 900 objects defined in IFC can be directly matched to CityGML semantically [6], [7]. Therefore, the loss of semantic information is considered unavoidable in the process of IFC to CityGML transformation. As for the research on the extension of standards, it is mainly based on the extension of IFC and CityGML data schema, which leads to a new data standard that meets the needs of urban construction [8]. Compared with the former, although such methods effectively circumvent the information loss problem in data conversion, redefining the data model is not only a large and difficult workload, but also requires the development of a specific platform to support data visualization. While using data conversion or standard extension, both methods have limitations for the use of BIM data. Because essentially, they are all about extracting and transferring the required BIM data into a specific

Type: Research article

Citation: F. Author et al. "How to write a peer-reviewed paper of the Journal of Architectural Informatics Society: ver. 20210329". Journal of Architectural Informatics Society, vol. 0, no. 0, pp. a1-aXX. doi: https://doi.org/xx.xxxx/xxxx/xxxxx

Received: 15 April 2020
Revised: 29 December 2021
Accepted: 05 January 2021
Published: 10 January 2021

data format to be used at a certain time for specific needs. However, with the continuous development of the city, the corresponding needs are constantly changing. The two methods above are difficult to support the changing needs of the future city construction [9].

On the other hand, the concept of BIM server is being popularized. Unlike the traditional form of storing BIM data in one (or more) files in a discrete manner, BIM server stores all BIM data in a unified form internally and exposes the required BIM information to client applications in the form of Application Programming Interface (API)s according to users' requirements [10]. Since the BIM server is in a separate state from the client application, it would made it easier to develop new applications using BIM data. This is because developers only need to call the corresponding APIs to the server to access the relevant subset of information without the need to understand the complex data model of BIM [11], [12].

Based on the a forementioned characteristics of BIM server, it is reasonable to believe that in the process of transferring BIM information to urban information model, the way of providing data through BIM server will be more able to cope with the changing needs of future urban construction than data format conversion and standard extension [13]. When it comes to responding to new requirements, developers will only need to add new APIs in BIM server for the corresponding requirements, whereas they may have to reconfigure the whole system if data format conversion or standard extension is employed. To validate this idea, it is important to understand how to build a suitable BIM server.

Since BIM data is characterized with large data volume, various data formats and complicated data users [12], what form of BIM data to handle and how to store these data will become the key point of building BIM server. In order to ensure the interoperability of BIM models built by different applications, buildingSMART International (bSI) has created a series of international standards such as Industry Foundation Classes (IFC), Information Delivery Manual (IDM) and Model View Definition (MVD) [14]. Among them, IFC defines the BIM data structure, IDM defines the business process of data conversion, and MVD can define the data subset of IFC model according to the Exchange Requirements (ER) defined by IDM [15]. In addition, with the increasing number of cases of BIM models integration through IFC in practice [16], [17], the effectiveness of using IFC as a standard specification for BIM data has been recognized. Therefore, we believe that it is valid to build a BIM server around the IFC data format.

1.2. Research objectives

Several research institutions and companies have conducted in-depth research on how to build BIM servers with IFC. Due to the limitations of storing IFC data in the form of files, such as not supporting parallel operation, difficulty in ensuring data integrity and completeness, and containing too much redundant information [18], many IFC servers nowadays choose to store IFC data in the form of databases. The relational database (RDB) is favored by many IFC servers, such as IMSvr, CIS2SQL [19], [20], owning to its long history, stable performance, ability to maintain data consistency, ability to provide rich APIs, and mature commercial software and supporting tool software [21]. These studies show that it is feasible to build IFC servers with RDB. However, when building IFC servers, the different database table design schemes mean that it is inevitable to make trade-offs between the access efficiency and data consistency [22]. The exiting research remains silent on the impact of these trade-offs on the system, as it mostly focuses on whether RDB can be built or not. Therefore, this study defines and establishes all feasible database table design solutions for storing IFC data into RDB, and evaluates the relative performance of each solution through query experiments to find a design solution that maximizes both performance and data consistency. In addition, some studies have been conducted to compare RDB with other types of databases [23]. This paper provides an understanding of whether the database design solutions used in these studies are the most representative for RDBs. Finally, since this paper lists all the possible database design options and performs performance tests on each of them, it provides options for developers who are trying to use RDB to build IFC servers in terms of database design options.

2. Literature review

So far, many researchers have worked on the implementation of IFC model servers and the method of storing IFC data in the form of a database has received a lot of attention. Databases can be broadly classified into two types: Structured Query Language (SQL) databases, also called relation database (RDB) and Not only Structured Query Language (NoSQL) databases.

In terms of research on NoSQL databases, Watanuki used graph database to design a data model storage method independent from IFC schema version differences [23], Beetz et al.

developed a BIM server using a key-value storage Database Management System (DBMS) [24]. Although NoSQL databases are fast, efficient, simple to extend and have some advantages in high concurrency processing, since most non-relational databases do not support transaction processing, they cannot guarantee data integrity and security during frequent data updates and deletions. Moreover, most NoSQL databases do not support SQL language queries, which makes it not only costly to learn and use, but also difficult to perform more complex queries [22].

Compared with NoSQL databases, most RDBs support the four major characteristics of transactions, namely Atomicity, Consistency, Isolation and Durability, thus greatly reducing the problems of data redundancy and data inconsistency. Even in the face of very complex queries, they can be implemented using the standardized query language SQL [19]. As for the research of RDB, Adachi converts the IFC schema defined in EXPRESS language into the highly common eXtensible Markup Language (XML) language, also known as IFCXML, according to ISO10303-28:2007 standard, and then converts IFCXML into the IFC database in the form of SQLServer. Based on this database, an IFC server named IMSvr was developed [19]. Unfortunately, it seems that the IMSvr server is no longer maintained, and the exact conversion mechanism is not publicly available. About the same time, YOU developed a server called CIS/2 SQL [20]. Although this server was developed for the CIS/2 data format, it could be used as an IFC server in a sense because CIS2 was defined in the same EXPRESS language as IFC. However, the special relationship of ANDOR between the entities in the definition of CIS/2, which is not available in IFC, has led to unnecessary development costs and difficulties. In addition, GUO attempted to build an IFC database by first converting the IFC schema to Java classes and then to MySql using the Hibernate Object Relational Mapping (ORM) framework [25]. What these studies have in common is that they each designed a set of mapping rules to convert the IFC schema into RDB database tables in order to verify whether an IFC database can be built with RDB. Since IFC is defined based on object-oriented concepts, but RDB is not. This leads to an impedance mismatch between the two in the conversion process [26]. In other words, different database table design schemes have different impacts on the data access efficiency and data consistency. At present, since native BIM data is not created based on IFC, it leads to the problem of semantic mismatch and missing information when exporting BIM data from authoring BIM software to IFC format. The impact of such problems could potentially be mitigated by an appropriate database table design scheme, because when the data is ambiguous with the database schema, it cannot be stored in the database due to error reporting [21]. Existing studies have not systematically compared the impact of each solution in the data access efficiency and data consistency trade-off process, making it difficult to determine which database table design solution is most appropriate for storing IFC data. In addition, most of the existing studies are not open source, it is therefore difficult to reproduce the systems in a third-party format and conduct an objective comparison and analysis.

In addition, some studies compare RDB with other types of databases. For example, Lee and Li show that storing IFC data in an object-relational database (OODB) is more efficient than RDB in terms of data access [27]. However, OODB technology is not yet mature, its interoperability among database management system software is low, and APIs for connecting databases and applications are difficult to unify, meaning it is difficult to apply in practice. In this type of research, they did not consider the impact of different database design solutions on the performance of RDB, and there is no strong data support to compare whether the RDB used in their experiments is the most representative one. In order to fill the above research gaps, this paper presents a comprehensive analysis and comparison of all feasible database table design schemes when IFC data is stored in RDB.

3. IFC to RDB mapping rules defined for experiment

The definition of the IFC schema contains data types such as entity, simple type, enum type, user-defined type, collection type, select type [28]. Additionally, there is also an inheritance relationship between entities. It is like the concept of base/derivative classes in object-oriented languages, where lower-level entities can obtain the properties defined by higher-level entities through inheritance without having to repeat the definition of these properties. The essence of converting the IFC schema to RDB is to consider how the above data types and inheritance relationship are represented in the form of database tables. This process is usually called mapping, and the corresponding conversion method is also called mapping rule, since the IFC schema is defined in the EXPRESS language, which is a data model language, not a programming language [28]. Therefore, for the computer to be able to execute the corresponding mapping rule, the IFC model needs to be reproduced in the form of a programming language first. This is one of the reasons why it is difficult to freely convert IFC patterns to RDB according to the pre-designed mapping rules. In this study, we design all the feasible mapping rules for converting IFC to RDB (as illustrate in Figure 1), and then develop the EXPRESS language parser based on c# programming language using ANTLR, and because of this, we are able to develop a program that can automatically convert IFC patterns to RDB according to various mapping rules. Finally, the performance difference of each RDB is compared systematically by Benchmark Test to clarify which mapping rule (database table design pattern) is the most suitable for storing IFC data. The following sub-sections will discuss the specific schemes of various mapping rules and their advantages and disadvantages.

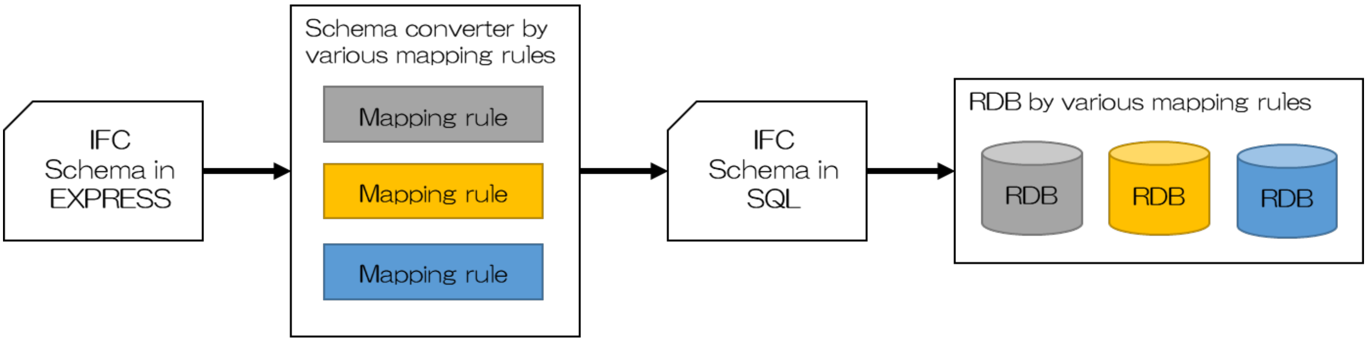


Figure 1. The process of converting IFC to RDB via various mapping rules

3.1. Entity

The methods of mapping IFC schema to RDB can be categorized into two: entity-to-table and attribute-to-table. As shown in the Figure 2, P1 is a way to unify the data of all entities with inheritance relationships in one table. In order to identify the specific subclass entities, an identification column is added to the table. Since it is possible to query the attributes of subclass entities without using JOIN SQL, it has the advantage of fast querying. However, this method must allow NULL values in columns other than the primary key column and the identity column. For example, in the IFC schema definition, there is an entity called IfcTask, and it is a subclass entity of IfcProcess with a required attribute called IsMilestone, while IfcProcess also has another subclass entity called IfcProcedure. And the IsMilestone attribute is not defined in the IfcProcedure entity. As a result, in order to store the data of IsMilestone attribute in a table, IsMilestone column must allow NULL value to exist. This makes the design of the database table ambiguous with the definition of the IFC schema, so the database cannot guarantee data integrity. In addition, there is a multi-level inheritance relationship in the definition of IFC, for example, from the IfcRoot entity derived from as many as 419 child entities, and a total of 602 attributes. This leads to an excessive number of columns in a single table, which can greatly reduce the performance of the database. Based on the above reasons, this study concluded that the P1 method is not suitable for storing IFC data, so it is excluded from this comparison.

The P2 method is a method that creates a table regardless of whether it is an abstract entity or a non-abstract entity. The subclass entity table has a column for storing its own attributes, and a column for storing inheritance relationships. This column is set to FOREIGN KEY CONSTRAINT, which is used to store the PRIMARY KEY of the superclass entity. For example, when an instance of IfcTask is stored in the database, the table T_IfcTask stores its own defined attributes such as Status, WorkMethod, IsMilestone, Priority, TaskTime, PredefinedType, while other attribute values are stored in the corresponding superclass entity table. Since these tables share a PRIMARY KEY information, one can use JOIN SQL to query the attributes defined by the super class entities. However, this has both an advantage and a disadvantage, as each entity creates a database table for each entity, when retrieving data across a range of entity hierarchies, many tables need to be joined, which can lead to database performance degradation. However, unlike the P1 method, one can add NOT NULL constraints to the corresponding columns according to the definition of the IFC schema, which will more accurately reproduce the content defined by the IFC schema and therefore better guarantee the integrity of the data.

The P3 method maps only the non-abstract entities to a database table, and the attribute data of the superclass entities are also stored in this table. This not only reduces the number of tables but also allows you to query specific properties without using JOIN SQL. However, when multiple entities inherit from the same superclass entity, there are many duplicate columns between database tables, which causes unnecessary waste of database space. In addition, all inheritance relationships are not reflected in the database because the database tables are not created for superclass entities. This has a significant impact when querying geometry information. For example, in the definition of IFC schema, the IfcSweptDiskSolid entity forms a one-to-one relationship with the abstract entity IfcCurve through the Directrix property. And IfcCurve has derived from IfcLine, IfcPcurve, IfcSurfaceCurve and other non-abstract entities. For this case, P3 method does not map IfcCurve to database table, so a new column must be added in T_IfcSweptDiskSolid table to indicate which table this table is related to. Although data can be stored when using this method, data integrity cannot be guaranteed in the database because FOREIGN KEY CONSTRAINT cannot be set between tables and the data status of the associated tables cannot be guaranteed. Furthermore, because the IFC schema is also constantly updated, once the definition of the abstract entity is updated, the database built by the P3 method will face a substantial reconstruction. For the database built by the P2 method, when an abstract entity is updated, only the table of the corresponding entity needs to be modified.

The P4 method differs from the above three methods in that it separates entity instances from attributes and stores the attributes in tables of their respective base data types. This minimizes the number of tables in the database and makes data management easier. However, for an entity instance, its attribute information is distributed and stored in multiple tables, and when retrieving information about an attribute of that entity, the associated attribute tables and instance tables must be JOIN SQL, which will probably lead to a reduction in data query efficiency.

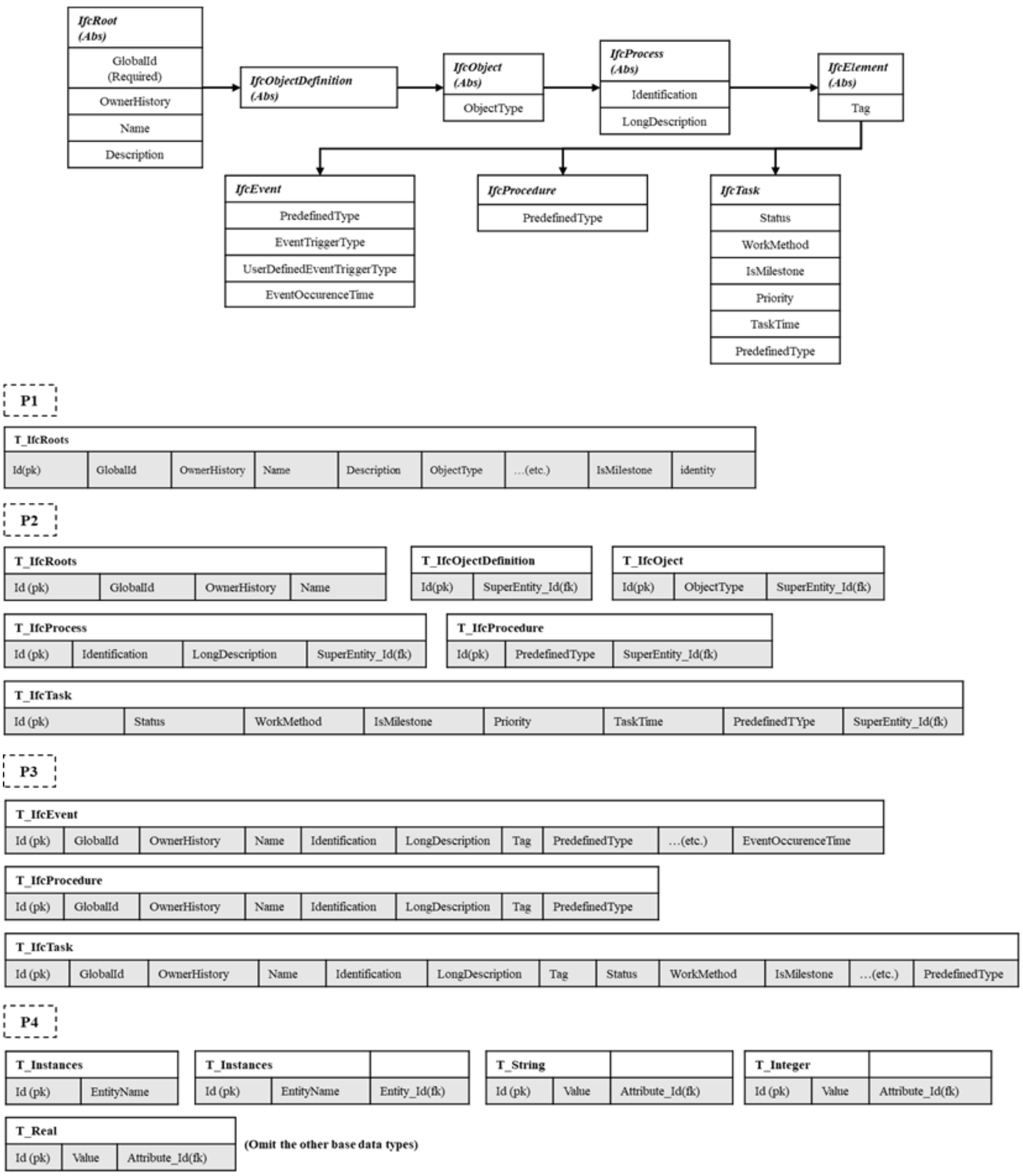


Figure 2. Example of table structure for each design pattern.

3.2. SimpleType, EnumType, UserDefinedType

SimpleType is the most basic data type in the EXPRESS language. They include REAL, INTEGER, NUMBER, LOGICAL, BOOLEAN, STRING, BINARY and so on. As shown in Table 1, REAL, INTEGER, NUMBER, STRING, and BINARY can be converted directly because there is a one-to-one data type in the database. However, for LOGICAL and BOOLEAN types, there is no corresponding data type in the database, so this study uses bit type instead. The bit type is an integer type, and its value can be 1, 0 or NULL. Therefore, if the data type of an attribute of an entity is LOGICAL or BOOLEAN, its attribute value is stored as 1 if TRUE, 0 if FALSE, and NULL if UNKNOWN. Since all EnumType in the IFC schema can be expressed as strings, they can be converted to narchar types when stored in the database. For UserDefinedType conversion rules depend on its underlying data type.

Table 1. Mapping rule about simpletype.

Express SimpleType	SqlServer data type
REAL	float
INTEGER	int
NUMBER	numeric
LOGICAL (TRUE, FALSE, UNKNOWN)	bit
BOOLEAN (TRUE, FALSE)	bit
STRING	nvarchar
BINANY	varbinany

3.3. AggregationType

Since RDB does not support data types in the form of arrays, data of AggregationType in the IFC schema cannot be stored directly. In order to store the data type, there are two possible workarounds. One is to separate the elements of the array by commas and store them as string types. The other is to create a separate table for each attribute of type AggregationType to store each element. For the former not only violates the database's First normal form (1NF), but because all data is converted to string types, if there is some data within the IFC file that violates the IFC schema definition, when these data are stored in the database, the database side is not able to detect it. This results in the application not only having to check the accuracy of the data when importing the IFC file into the database, but also having to convert the data to the original data type after acquiring it, which greatly increases the overall system development effort. On the other hand, the latter separates the attributes of the array type from the entity table and creates a separate table for them. So the data consistency can be ensured by associating the array attribute table with the entity table through FOREIGN KEY CONSTRAINT. The Figureure shows two instances of IfcCartesianPoint with coordinate values (X=0, Y=0, Z=0) and (X=0, Y=0, Z=100) deposited into the database. From the Figure 3, we can know that T_IfcCartesianPoint table and T_IfcCartesianPoint_Has_Coordinate two tables form a one-to-many relationship through IfcCartesianPoint_Id field, and because this field adds FOREIGN KEY CONSTRAINT, if the data in T_IfcCartesianPoint table is deleted or updated, the data in T_IfcCartesianPoint_Has_Coordinate table will also be deleted or updated automatically. Based on the above characteristics, this study adopts the latter design method to ensure data integrity to the maximum extent.

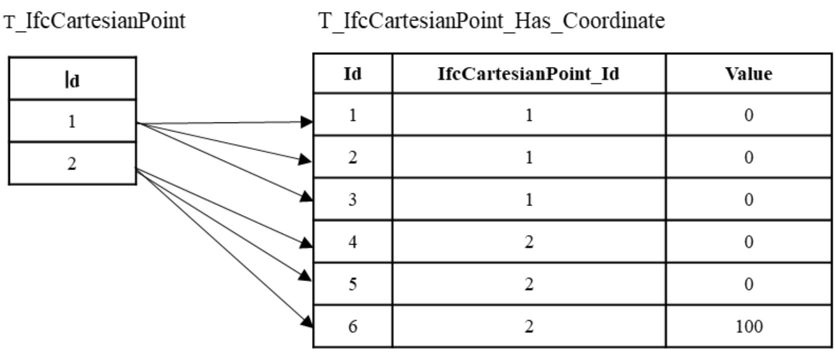


Figure 3. Example of IfcCartesianPoint table.

3.4. SelectType

Compared to other data types, the selection type is special. Its value can be any of the multiple data types defined by the IFC schema. Since the specific data type of the SelectType attribute in IFC data cannot be determined at the database table design stage, it is impossible to set the data type for the fields of this attribute. This makes it difficult to reflect the attributes of the SelectType in the RDB. In order to cope with this issue, this study considers mapping two fields for the attribute of the selection type. One field stores the actual selected entity name, and the other field stores the corresponding value, and both fields are set to string type. In addition, if the selectable data type is an entity type, the PRIMARY KEY of the actual selected entity is stored as the value. Although this design method is relatively simple to implement, the data state of the interrelated entity instances cannot be guaranteed on the database side due to the inability to set up FOREIGN KEY CONSTRAINT for the related tables. The impact of data consistency brought about by SelectType on the database has to be left to the application to deal with it.

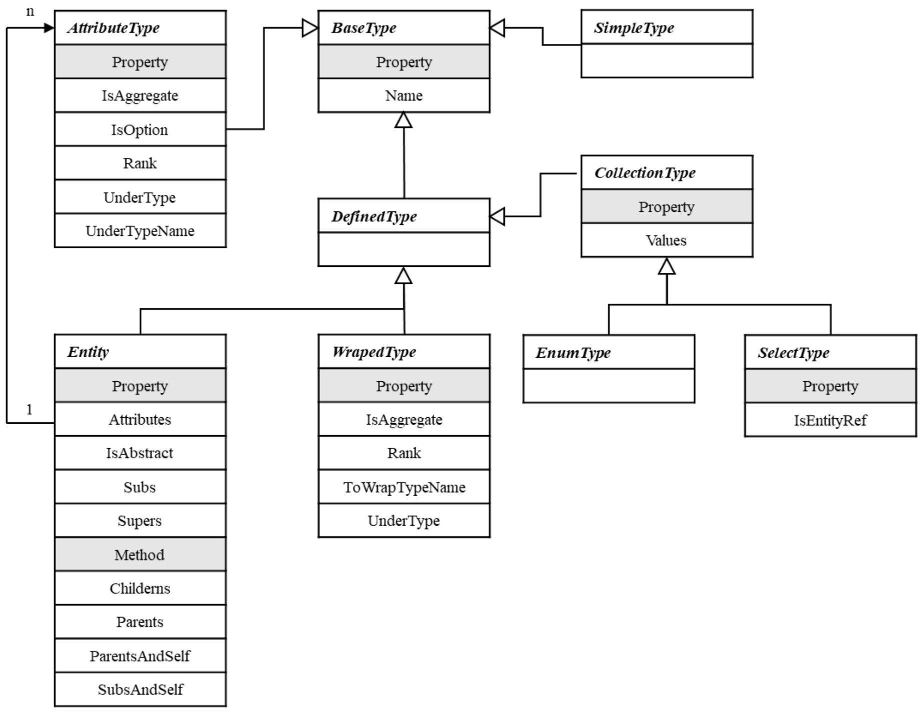


Figure 4. The UML of Ifc schema parser.

4. Research methodology

4.1. Methods for auto-converting IFC to RDB and auto-importing IFC data

In order to know the most suitable database table design scheme for converting IFC to RDB, this study developed a tool to automatically convert IFC schema to RDB according to different mapping rules for IFC4_ADD2_TC1. The tool has three functions, including automatic RDB creation based on IFC schema files, IFC data import and QUERY performance testing.

A specific problem when converting IFC schema to RDB is how to convert the content defined in EXPRESS language into SQL statements for RDB creation. This requires first parsing the IFC schema file and classifying the various data types and the relationships between them, and then converting each data type into a corresponding SQL statement according to the pre-designed mapping rules. Therefore, in this study, we first design a data model specifically for parsing IFC schemas, which becomes the parsing model below. As shown in the Figure 4, a data type named BaseType is defined as the superclass of all data types. From this superclass, SimpleType, Entity, WrappedType, EnumType, SelectType, AttributeType classes are derived. The Enity class defines four attributes, Attributes, IsAbstact, Subs, and Supers; Attributes stores the attributes of each Entity as a data type; IsAbstact distinguishes whether an entity is an abstract entity. The WrappedType class refers to a new IFC data type defined by the user based on SimpleType. The WrappedType class has a property called IsAggregate that is designed to determine if the WrappedType class is an array type, which allows the program to easily identify if the property is an array type, and thus determine if a new table needs to be created independently to store the array elements. The AttributeType class has an attribute called IsEntityRef, which can be used to determine whether the data type that can be selected is an entity type or a value type. It can be used to determine whether the relevant IFC attribute is a required attribute, so that a not null constraint can be created for the field of this attribute when creating the database.

Based on the above parsing model, this study developed a tool to automatically convert IFC schema files (.exp) into SQL statements according to different mapping rules using the open-source syntax parser ANTLR, based on c# programming. The Figure 5 shows the User Interface (UI) for the tool and the converted SQL statements are shown in the Appendix. The database can be created by executing the converted SQL statements. In this study, we also developed a tool to convert IFC data into SQL statements required for importing databases created by different mapping rules. The conversion results are shown in the table. Finally, QUERY performance tests were conducted using these imported IFC data.

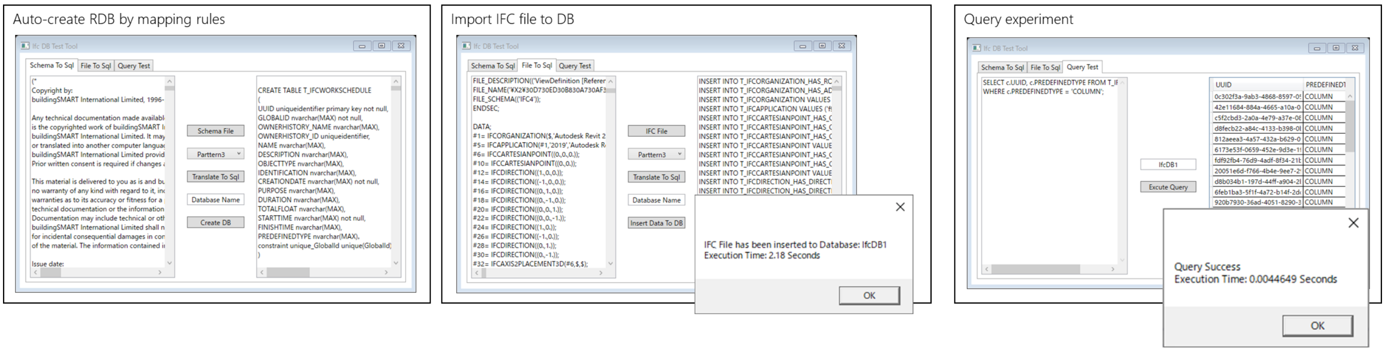


Figure 5. User Interface about IfcDB test tool.

4.2. Query test

This study uses the BUCKY benchmark method to evaluate the performance of the RDB generated based on mapping rules. Originally, BUCKY is a commonly used query-based benchmark specifically designed for testing the performance of ORDBs and RDBs. It is composed of 16 queries for testing five areas, namely inheritance, object references, collection data types methods, and abstract data types [29]. Due to the existence of an RDB version, BUCKY can also be used to compare the performance of RDBs composed of different database table design schemes. This study follows the existing literature on testing IFC databases [27], [30] and selects 10 QUERYs from the original BUCKY that are suitable for testing IFC data. The specific QUERY contents and the purpose of QUERY testing are shown in the Table 2.

After determining the Query content of the BUCKY Test, the next step is to choose which IFC data to be used for the BUCKY Test. the standard approach used in the existing literature is to use IFC files with different number of instances as the experimental objects. These files are selected to allow us to know not only the horizontal difference (e.g., the performance difference between RDBs constructed by different mapping rules for the same number of instances of IFC files), but also the vertical comparison (e.g., the impact on the performance of RDBs constructed by the same mapping rule when storing different numbers of instances of IFC files). The above objectives cannot be achieved by selecting a single test data.

Table 2. Query contents.

No	SqlServer data type	Test goal	Query in this study
1	SINGLE-EXACT	A simple selection operation, independent of other test items	Get columns with PredefinedType = "column
2	HIER-EXACT	Querying data in the inheritance relation	Get columns with PredefinedType = "oo"
3	SINGLE-JOIN	Querying data using the self-join operation	Get multiple columns with the same "PredefinedType" value
4	HIER-JOIN	Querying performance of inherited classed	Get multiple columns with the same "Tag" value
5	SET-ELEMENT	Querying collection-type data that satisfy one condition	Get columns with X="oo" in the relative coordinate system of columns

6	SET-AND	Collection-type data that satisfy two conditions connected using the AND relation	Get columns with X="oo" and Y= "oo" in the relative coordinate system of columns
7	1HOP-NONE	Querying all pairs of data from two tables/classes, which are not in the inheritance relation	Get OwnerHistory information from all columns
8	1HOP-ONE	Like Query7, but includes a where clause for querying an attribute from the one side in the one-to-many relation	Get the OwnerHistory information of the column for which the GlobalId is specified.
9	1HOP-MANY	Like Query7, but includes a where clause for querying an attribute from the one side in the one-to-many relation	Get all columns with CHANGEACTION="NOCHANGE" in the column's OwnerHistory.
10	2HOP-ONE	Two joins and one where clause condition	Get the OwningApplication information in the OwnerHistory of all columns.

4.3. Sampling procedure

This study will test three IFC data sets varying in size and complexity. The first is a simple frame model comprising some columns, beams, floors and walls. The second is an IFC file that integrates the design, equipment and structural sample models included in Revit 2021 and output using the IFC4 Reference View MVD. The third is a concrete frame BIM model of a 20-story high-rise apartment building, including a rebar model. Details of each file are shown in the Figure 6.

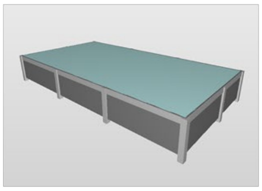
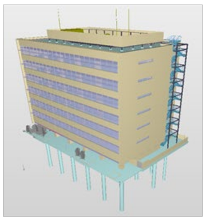
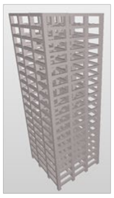
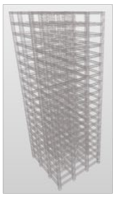
No	capture	Number of instances	Data size
1		1390	89.5 KB
2		263276	24.4 MB
3	 	2240026	153 MB

Figure 6. About Ifc sample data.

5. Result

The results of various experiments are shown in Figure 7. The processing time for data import increases as the data increases for all three methods. When the amount of data is small, P3 has the fastest processing speed, but we cannot confirm a significant difference with P2 and P4. The more the number of instances increases, the more pronounced the difference becomes. In Sample 3, there is a performance difference of up to 30 minutes between P4 and P2 or P3.

Q1 and Q3 are queries designed using attributes defined by the entities themselves. When the number of instances is small, P4 has the fastest processing time, but as the number of instances increases, the processing time of P4 increases significantly. However, beyond a certain amount of data, P4 has the slowest processing time; P2 and P3 can retrieve data in almost constant time regardless of the amount of data. Also, the processing speeds of P2 and P3 are not so different.

Q2 is a query that searches for inherited attributes. When the number of instances is small, P4 has the fastest processing speed. Also, the processing speed of P2 and P3 is almost the same. When the number of instances increases, P2 and P4 become slower, but the speed is almost the same for P3. As a result, P3 has the best performance when the amount of data is large.

Q4 is a query to retrieve multiple instances with the same inherited attributes; in P2 and P4, the processing speed decreases as the data volume increases; in P3, the processing speed remains the same when the data volume is within a certain range but decreases when it exceeds that range. In the case of handling large amounts of data, the performance of P4 is significantly lower than that of P2 and P3.

Q5 and Q6 are queries to obtain aggregate-type data. As the amount of data increases, the processing speed of all three methods slows down, but P4 has a significantly greater reduction than P2/P3. The results of Sample 3 show that P4 takes more than 1000 ms for a single search, while P3 is slightly faster than P2, but not by much.

Q7-Q10 and the rebar query test are queries that retrieve data using reference relationships between entities; P3 has the fastest processing speed and can retrieve data in a fixed amount of time regardless of the increase in data volume; P2 and P4 slow down as the data volume increases; P2 is slower than P3 but P2 is slower than P3, but the maximum difference in speed is only 70 ms. The performance difference between P4 and P2/P3 increases as the data volume increases. Based on the results of Sample 3, P4 requires about 8s of processing time in Q10.

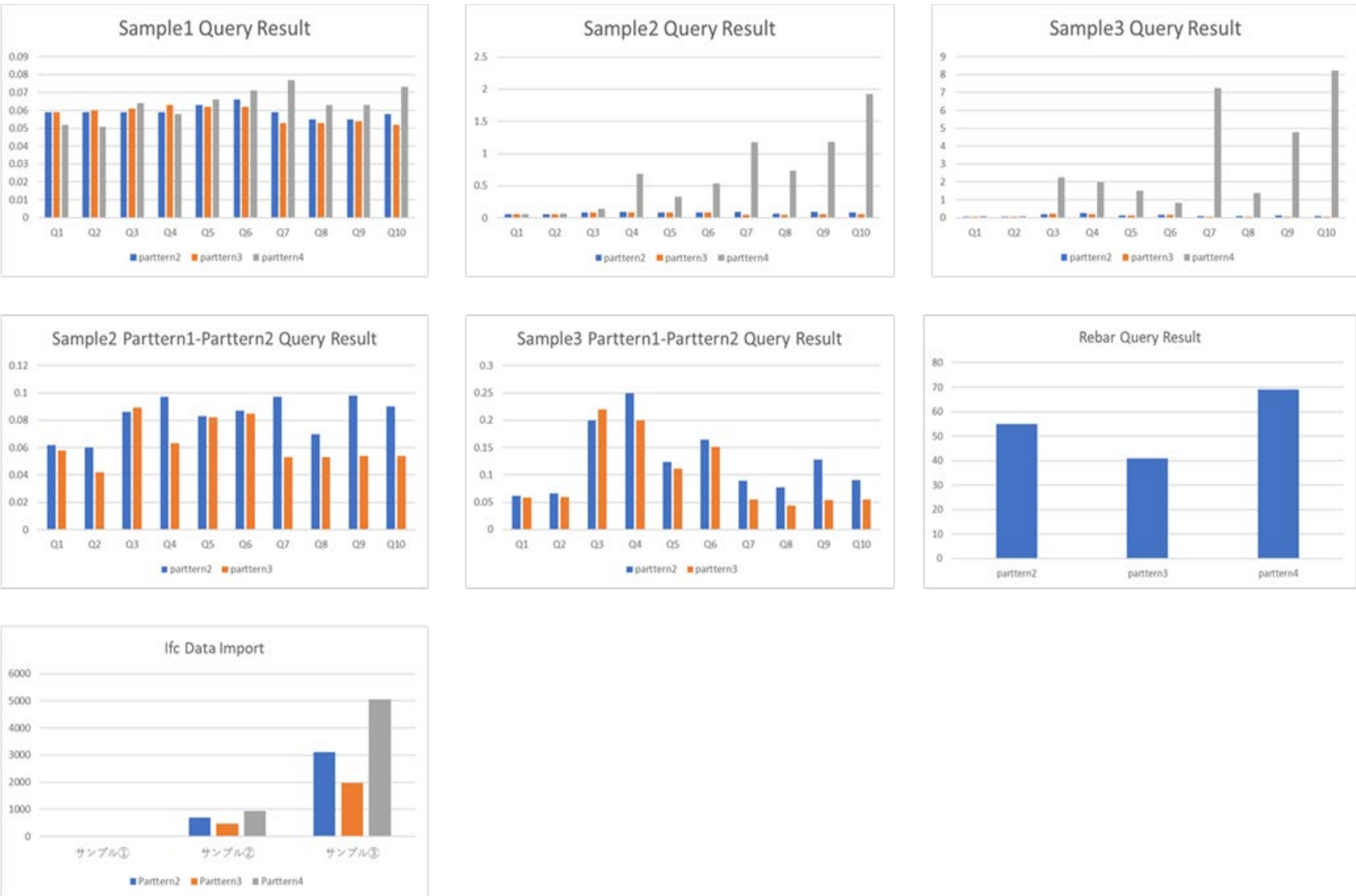


Figure 7. The result of experiment.

6. Discussion and concluding remarks

6.1. Discussion

Based on the data import results, the P3 method is expected to have the fastest processing speed because it covers all attributes in a single table and can store data with only one SQL execution with one IFC instance as one record; the P2 method is slower than P3 because it requires the data to be stored in multiple tables by separating the attributes of one IFC instance. P4 requires importing as many data as the number of attributes, so performance deteriorates as the amount of data and the types of data types increase.

Based on the results of Q1 and Q3, P4 needs to retrieve a specific instance from the instance table. When dealing with large data volumes, the number of records in the instance table becomes very large, and it takes time to retrieve the target instance. The reason for this is thought to be that the performance of P4 degrades with the volume of data. On the other hand, in P2 and P3, the amount of data does not affect the processing speed because a table is created for each instance and the process of identifying a specific instance is not necessary.

The result of Q2 shows that the number of instances affects the processing speed for P4, while for P3, the same processing speed can be obtained regardless of whether the attribute to be searched is inherited or not, since the inherited attribute and the attribute defined by the user are expressed in a single table in full text. In P2, in order to search for inherited attributes, it is necessary to use JOIN processing to retrieve information from the inherited table to the inherited source table, which may be the reason for the slightly lower performance than in P3.

Based on the results of Q4, the number of data types in the stored IFC data affects performance because P4 stores data of the same data type in a single table. About P2 and P3, data is stored in each entity in a distributed manner. Furthermore, in P2, the number of fields for a single table is smaller than in P3 because there is also a table for abstract type entities. Therefore, in Sample 1, the retrieval speed of P2 is faster than that of P3. However, as the amount of data increases, the effect of JOIN becomes more pronounced, which may be the reason why P2 is slightly slower than P3 in Samples 2 and 3.

Since all three patterns use the same method to store aggregate type data, it is not possible to reflect how the DB design method affects performance. However, based on the results of Q5 and Q6, there is no significant difference in performance when comparing P2 and P3, so P2 is considered better in terms of data integrity.

Based on the results of Q7-Q10 and the rebar query test, when retrieving data using the reference relationship between entities, P4 requires a round-trip search to three tables: the table that stores instances in an integrated manner, the table that stores attribute information, and the table that stores actual values. The reason for this is that the longer the search route, the more significant the performance degradation. P3 stores the table name and Id of the related destination in the related source table, which is considered to have the best search efficiency because only the related destination and two related source tables can be processed by JOIN in a long search route. However, since foreign keys cannot be set, it is impossible to update data in the source table as data changes in the destination table. In P2, the search route is longer because the related source table stores the Id of the super entity of the target entity table, which results in many joined tables. Therefore, the search efficiency is slower than in P3. In addition, the performance difference tends to increase as the hierarchy of related entities increases.

There are many other factors that affect RDB performance. For example, attributes of entities to be queried, SQL algorithms, PC specifications, etc. Based on the results of this study, the entity-to-table method is considered more appropriate than the attribute-to-table method for storing IFC data. Among the entity-to-table methods, P3 can perform better than P2 in importing IFC data and querying using relationships between entities. However, there is not a significant difference between the two in terms of retrieving simple entity attributes. P2 is also able to more reliably maintain data consistency and integrity. Since the IFC server services the building lifecycle, we believe that data consistency is more important than performance, and therefore, using P2 as the basic method and combining it with methods that improve query performance is the most appropriate method for building an IFC server at present.

6.2. Concluding remarks

This study systematically analyzes all feasible database design schemes for converting IFC schema to RDB and develops tools for automatically converting IFC schema files to RDB according to different data table design schemes. Accordingly, the performance of various database design schemes is compared systematically by various query tests. Through the experimental results, we find that when storing IFC data in the form of RDB, the attribute to table method is much lower than the entity to table method in terms of data import and data fetching performance, although the overall number of database tables is small to facilitate data management. In the entity to table method, the method of creating database tables for both abstract entities and non-abstract entities (the P2 method mentioned above), when compared with other methods, can maintain data consistency to the greatest extent while also performing at a relatively high level. One of the main difficulties when developing BIM server is to strike a balance between the access efficiency and data consistency due to different database table design schemes. However, we know little about the optimal solution that maximizes both performance and data consistency. The method proposed in our study thus fill this void. Knowing this would reduce the difficulty of developing BIM server, in addition, in recent years, commercial IFC model server services have emerged and are used in the development of cloud-based BIM applications for design, construction and maintenance. bSI has been working on standardization of a common API specification for IFC model servers, named openCDE API, and the results of this study could help bSI to carry out such work. Nonetheless, although this study compares the optimality of IFC RDBs built from various mapping rules using data access and ensuring data consistency as metrics, there may be other factors affecting the optimality, such as the completeness of IFC schema representation of the mapped tables, which could be further explored and studied in the future. It should be noted that the P2 method proposed in our study can only ensure maximum data consistency for data types other than the SeletcType in the IFC schema, which is conceptually a multiple inheritance. This is due to the fact that the existing RDBS is not suitable for storing multiple inheritance, and therefore, future studies need to conduct a more in-depth analysis of the storage method for SelectType.

Declaration of competing interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] ISO, ISO 19650-1:2018(en), Organization and digitization of information about buildings and civil engineering works, including building information modelling (BIM) — Information management using building information modelling — Part 1: Concepts and principles, 2018.

[2] S. Herle, R. Becker, et al, "How to Obtain Interoperability Between Geospatial and Building Information Modelling?" in Journal of Photogrammerty Remote Sensing and Geoinformation Science, 88 (8), 2020-02. [Online]. Available: https://www.researchgate.net/publication/339040025_GIM_and_BIM_How_to_Obtain_Interoperability_Between_Geospatial_and_Building_Information_Modelling

[3] El-Mekawy, Mohanmed, et al, "An evaluation of IFC-CityGML unidirectional conversion." International Journal of Advanced Computer Science and Applications 3.5, 2012. [Online]. Available: https://www.proquest.com/docview/2656741169?pq-origsite=gscholar&fromopenview=true

[4] Zhu, J., & Wu, P. (2021). Towards effective BIM/GIS data integration for smart city by integrating computer graphics technique. Remote Sensing, 13(10), 1889. [Online]. Available: https://www.mdpi.com/2072-4292/13/10/1889

[5] Donkers, S., Ledoux, H., Zhao, J., & Stoter, J. (2016). Automatic conversion of IFC datasets to geometrically and semantically correct CityGML LOD3 buildings. Transactions in GIS, 20(4), 547-569. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/tgis.12162

[6] MLIT Japan, "Manual for the Integration of BIM Models in 3D City Models with CityGML," in Handbook of 3D City Models, 2021. [Online]. Available: https://www.mlit.go.jp/plateau/file/libraries/doc/plateau_doc_0003_ver01.pdf.

[7] El-Mekawy, M., Östman, A., & Hijazi, I. (2012). A unified building model for 3D urban GIS. ISPRS International Journal of Geo-Information, 1(2), 120-145. [Online]. Available: https://www.mdpi.com/2220-9964/1/2/120

[8] Redmond, A., & Smith, B. (2011, October). Exchanging partial BIM information through a cloud-based service: Testing the Efficacy of a Major Innovation. In Proc. of the IBEA Conference 2011. [Online]. Available: https://www.researchgate.net/publication/315831814_Exchanging_Partial_BIM_Inform

ation_through_a_Cloud-Based_Service_testing_the_efficacy_of_a_major_innovation

[9] Wyszomirski, M. (2022). Analysis of the possibility of using key-value store NoSQL databases for IFC data processing in the BIM-GIS integration process. Polish Cartographical Review, 54(1), 11-22. [Online]. Available: https://sciendo.com/it/article/10.2478/pcr-2022-0002

[10] H. David, “BIMserver and the potential of server-side BIM”, 2009.02.26. [Online]. Available: https://www.stress-free.co.nz/bimserver_and_the_potential_of_serverside_bim

[11] Beetz, J., van Berlo, L., de Laat, R., & van den Helm, P. (2010, November). BIMserver. org–An open source IFC model server. In Proceedings of the CIP W78 conference (p. 8). [Online]. Available: https://www.researchgate.net/publication/254899282_Bimserverorg_-_an_Open_Source_IFC_model_server

[12] T. Watanuki and Y. Nobuyoshi, “MANAGEMENT OF 3D PRODUCT MODELS OF THE BRIDGE USING SEVERAL KINDS OF DBMS,” in Journal of Japan Society of Civil Engineers, 71(2), I_87-I_98, 2015. [Online]. Available: https://ci.nii.ac.jp/naid/130005142641.

[13] Guyo, E., Hartmann, T., & Ungureanu, L. (2021). Interoperability between BIM and GIS through open data standards: An overview of current literature. sign, 3, 5-9. [Online]. Available: http://ceur-ws.org/Vol-3081/10paper.pdf

[14] buildingSMART, “bsi-Standards”, [Online]. Available: https://www.buildingsmart.org/standards/bsi-standards/

[15] buildingSMART Japan, “About MVD” (in Japanese), [Online]. Available: https://www.building-smart.or.jp/ifc/mvd/

[16] Tang, S., Shelden, D. R., Eastman, C. M., Pishdad-Bozorgi, P., & Gao, X. (2020). BIM assisted Building Automation System information exchange using BACnet and IFC. Automation in Construction, 110, 103049. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0926580519301645

[17] Japan Housing and Wood Technology Center, “Project to Create a Digital Construction Casebook Using BIM Report” (in Japanese), 2020.3. [Online]. Available: https://www.howtec.or.jp/files/libs/3299/202005081042514547.pdf

[18] Wikipedia, “Relational database”, [Online]. Available: https://en.wikipedia.org/wiki/Relational_database

[19] Han, J., Haihong, E., Le, G., & Du, J. (2011, October). Survey on NoSQL database. In 2011 6th international conference on pervasive computing and applications (pp. 363-366). IEEE. [Online]. Available: https://ieeexplore.ieee.org/document/6106531

[20] Y. Adachi, Overview of IFC Model Server Framework, European Conference of Product and Process Modeling ECPPM 2002.

[21] SJ. You, D. Yang, CM. Eastman “RELATIONAL DB IMPLEMENTATION OF STEP BASED PRODUCT MODEL”, CIB World Building Congress 2004, 2004. [Online], Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.86.1723&rep=rep1&type=pdf

[22] Louis Davidson, “Pro SQL Server Relational Database Design and Implementation: Best Practices for Scalability and Performance”, 2020.12.15.

[23] T. Watanuki and Y. Nobuyoshi, “A NOVEL METHOD FOR STORING BIM DATA IN A GRAPH DB AND EXTRACTING INFORMATION:Devising a data model which is not affected by versions of IFC schema,” in The Architectural Institute of Japan’s Journal of Architecture and Planning(772), 1377-1385, 2020-06. [Online]. Available: https://ci.nii.ac.jp/naid/130007866585.

[24] J. Beetz, L. Berlo, R. Laat, et al, “Bimserver.org – an Open Source IFC model server,” in Proceedings of the CIB W78, 2010: 27th International Conference –Cairo, Egypt, 16-18 November. [Online]. Available: https://www.researchgate.net/publication/254899282_Bimserverorg_-_an_Open_Source_IFC_model_server.

[25] HL. Guo, Y. Z, XT. Ye, et al, "Automated mapping from IFC data model to relational database model," in Journal of Tsinghua University (Sci & Technol), vol.61, No2, 2021. [Online], Available: https://www.researchgate.net/publication/344329568_Automated_mapping_from_IFC_data_model_to_relational_database_model

[26] SB. Bernhard, E. Friedrich, "ifcSQL-Database", 29 October 2021. [Source code]. https://github.com/IfcSharp/IfcSQL.

[27] G. Lee, J. Jeong, J. Won, et al, "Query performance of the IFC model server using an object-relational database approach and a traditional relational database approach," in Journal of Computing, vol.28, Issue2, 2014 March. [Online]. Available: https://ascelibrary.org/doi/abs/10.1061/(ASCE)CP.1943-5487.0000256.

[28] XL. Li, H. Yi, XY. Wu, "Implementation of EXPRESS data model on relational database," in Computer Integrated Manufacturing System, 1999(1): 64-68, (in Chinsese). [Online]. Available: http://qikan.cqvip.com/Qikan/Article/Detail?id=3477001.

[29] Carey, M. J., DeWitt, D. J., Naughton, J. F., Asgarian, M., Brown, P., Gehrke, J. E., & Shah, D. N. (1997, June). The BUCKY object-relational benchmark. In Proceedings of the 1997 ACM SIGMOD international conference on Management of data (pp. 135-146). [Online]. Available: https://dl.acm.org/doi/abs/10.1145/253260.253283

[30] Maatuk, A. M., Ali, M. A., Moftah, R. A., & Elkobaisi, M. R. (2016, September). Performance evaluation of an RDB and an ORDB: A comparative study using the BUCKY benchmark. In 2016 International Conference on Engineering & MIS (ICEMIS) (pp. 1-7). IEEE. [Online]. Available: https://ieeexplore.ieee.org/document/7745312

Appendix

<table>
<tr><th>IFC</th><th>SQL</th></tr>
<tr><td rowspan="6">ENTITY IfcRoot
GlobalId : IfcGloballyUniqueId;
OwnerHistory : OPTIONAL IfcOwnerHistory;
Name : OPTIONAL IfcLabel;
Description : OPTIONAL IfcText;
UNIQUE
UR1 : GlobalId;
END_ENTITY;

ENTITY IfcObjectDefinition
SUBTYPE OF (IfcRoot);
END_ENTITY;

ENTITY IfcObject
SUBTYPE OF (IfcObjectDefinition);
END_ENTITY;

ENTITY IfcProduct
SUBTYPE OF (IfcObject);
ObjectPlacement : OPTIONAL IfcObjectPlacement;
Representation : OPTIONAL IfcProductRepresentation;
END_ENTITY;

ENTITY IfcElement
SUBTYPE OF (IfcProduct);
Tag : OPTIONAL IfcIdentifier;
END_ENTITY;

ENTITY IfcBuildingElement
SUBTYPE OF (IfcElement);
END_ENTITY;

ENTITY IfcColumn
SUBTYPE OF (IfcBuildingElement);
PredefinedType : OPTIONAL IfcColumnTypeEnum;
END_ENTITY;</td>
<th>Parttern2</th></tr>
<tr><td>CREATE TABLE T_IfcRoot
(
UuId uniqueidentifier primary key not null,
GlobalId nvarchar(max) not null,
Name nvarchar(max),
Description nvarchar(max),
constraint unique_GlobalId unique(GlobalId)
)

CREATE TABLE T_IfcObjectDefinition
(
UuId uniqueidentifier primary key not null,
superEntity_Id uniqueidentifier not null,
foreign key(superEntity_Id) references T_IfcRoot(UuId)
)

CREATE TABLE T_IfcObject
(
UuId uniqueidentifier primary key not null,
superEntity_Id uniqueidentifier not null,
foreign key(superEntity_Id) references T_IfcObjectDefinition(UuId)
)

CREATE TABLE T_IfcProduct
(
UuId uniqueidentifier primary key not null,
superEntity_Id uniqueidentifier not null,
ObjectPlacement_Id uniqueidentifier,
Representation_Id uniqueidentifier,
foreign key(superEntity_Id) references T_IfcObjectDefinition(UuId),
foreign key(ObjectPlacement_Id) references T_IfcObjectPlacement(UuId),
foreign key(Representation_Id) references T_IfcProductRepresentation(UuId)
)

…(IfcElement, IfcBuildingElementを省略する)

CREATE TABLE T_IfcColumn
(
UuId uniqueidentifier primary key not null,
superEntity_Id int not null,
PredefinedType nvarchar(max),
foreign key(superEntity_Id) references T_IfcBuiltElement(UuId)
)</td></tr>
<tr><th>Parttern3</th></tr>
<tr><td>CREATE TABLE T_IfcColumn
(
UuId uniqueidentifier primary key not null,
GlobalId uniqueidentifier not null,
Name nvarchar(max),
Description nvarchar(max),
ObjectPlacement_Name nvarchar(max),
ObjectPlacement_Id uniqueidentifier,
Representation_Name nvarchar(max),
Representation_Id uniqueidentifier,
Tag nvarchar(max),
constraint unique_GlobalId unique(GlobalId)
)</td></tr>
<tr><th>Parttern4</th></tr>
<tr><td>CREATE TABLE T_Instances
(
UuId uniqueidentifier primary key not null
)

CREATE TABLE T_Attributes
(
UuId uniqueidentifier primary key not null,
Instance_Id uniqueidentifier,
foreign key(Instance_Id) references T_Instances(UuId)
)

CREATE TABLE T_String
(
UuId uniqueidentifier primary key not null,
Attribute_Id uniqueidentifier,
Value nvarchar(max),
foreign key(Attribute_Id) references T_Attributes(UuId)
)

…(その他データ型のテーブルを省略する)</td></tr>
</table>

Appendix A. Result of converting Ifc schema to Sql.

<table>
<tr><th>IFC</th><th>SQL</th></tr>
<tr><td rowspan="6">#42=IFCOWNERHISTORY(#39,#5,$,.NOCHANGE.,$,$,$,0);
#224=IFCLOCALPLACEMENT(#134,#223);
#216=IFCPRODUCTDEFINITIONSHAPE($,$,(#214));
#226=IFCCOLUMN('3g3QWhC7bC8xkTpZj1QVmC',#42,'RC¥X2¥67F1¥X0¥-¥X2¥89D2¥X0¥:500x500:271165',$,'500x500',#224,#216,'271165',.COLUMN.);</td><td>Parttern2</td></tr>
<tr><td>INSERT INTO T_IfcRoot VALUES(‘4a2b29e7-7de7-4bd7-974f-3f80048c7dba‘, ‘3g3QWhC7bC8xkTpZj1QVmC’, ‘744E0D25-BB85-4F6A-9481-E812D8B57B9F’,'RC¥X2¥67F1¥X0¥-¥X2¥89D2¥X0¥:500x500:271165’,)

INSERT INTO T_IfcObjectDefinition VALUES(‘4a2b29e7-7de7-4bd7-974f-3f80048c7dba‘, 4a2b29e7-7de7-4bd7-974f-3f80048c7dba’)

INSERT INTO T_IfcObject VALUES(‘cb4546fa-869f-4d63-94e5-dca3602836b0‘, 4a2b29e7-7de7-4bd7-974f-3f80048c7dba’, ‘500x500’)

INSERT INTO T_IfcProduct VALUES(‘73f16b64-df4c-4889-aca2-58ad7e7e1dd5‘, ‘cb4546fa-869f-4d63-94e5-dca3602836b0’, ‘324ab512-ebcb-48cf-8dbe-c3194fbb86c0’, ‘67fadcee-87e5-40cc-b87b-54371a979b96’)

INSERT INTO T_IfcElement VALUES(‘5c868575-5646-4b37-bc44-1186b89b8ef4‘, ‘73f16b64-df4c-4889-aca2-58ad7e7e1dd5’, ‘271165’)

INSERT INTO T_IfcBuildingElement VALUES(‘814d066a-4115-4e3d-81a8-0020c05ca8d9‘, ‘5c868575-5646-4b37-bc44-1186b89b8ef4’)

INSERT INTO T_IfcColumn VALUES(‘7C7F5243-3527-41C9-B4A9-4C44F638B7E6‘, ‘814d066a-4115-4e3d-81a8-0020c05ca8d9’, ‘COLUMN’)</td></tr>
<tr><td>Parttern3</td></tr>
<tr><td>INSERT INTO T_IfcColumn VALUES(‘7C7F5243-3527-41C9-B4A9-4C44F638B7E6‘, ‘3g3QWhC7bC8xkTpZj1QVmC’, ‘T_IfcObjectPlacement’, ‘744E0D25-BB85-4F6A-9481-E812D8B57B9F’,'RC¥X2¥67F1¥X0¥-¥X2¥89D2¥X0¥:500x500:271165’, , ‘500x500’, ‘T_IfcProductdefinitionshape’, ‘324ab512-ebcb-48cf-8dbe-c3194fbb86c0’, ‘T_IfcProductDefinitionShape’, ‘67fadcee-87e5-40cc-b87b-54371a979b96’, ‘271165’, ‘COLUMN’)</td></tr>
<tr><td>Parttern4</td></tr>
<tr><td>INSERT INTO T_Instances VALUES(‘7C7F5243-3527-41C9-B4A9-4C44F638B7E6‘)

INSERT INTO T_Attributes VALUES(‘AAFEF574-18E3-475C-97A1-2BE601E3DC16’, ‘7C7F5243-3527-41C9-B4A9-4C44F638B7E6‘, ‘Name‘)
INSERT INTO T_Attributes VALUES(‘A9490C0D-6EF0-4FA9-97FA-85B5C8D9A95F’, ‘7C7F5243-3527-41C9-B4A9-4C44F638B7E6‘, ‘Description‘)
INSERT INTO T_Attributes VALUES((‘6A26F365-CAA6-493E-B72C-8D07F48B177B, ‘7C7F5243-3527-41C9-B4A9-4C44F638B7E6‘, ‘ObjectType‘)
INSERT INTO T_Attributes VALUES(‘6A26F365-CAA6-493E-B72C-8D07F48B177B‘, ‘7C7F5243-3527-41C9-B4A9-4C44F638B7E6‘, ‘Tag‘)
INSERT INTO T_Attributes VALUES(‘45BA521E-7279-493F-89C3-7C5657A355CF‘, ‘7C7F5243-3527-41C9-B4A9-4C44F638B7E6‘, ‘PredefinedType ‘)

INSERT INTO T_String VALUES(‘AAFEF574-18E3-475C-97A1-2BE601E3DC16’, ‘'RC¥X2¥67F1¥X0¥-¥X2¥89D2¥X0¥:500x500:271165’)
INSERT INTO T_String VALUES(‘A9490C0D-6EF0-4FA9-97FA-85B5C8D9A95F’,)
INSERT INTO T_String VALUES((‘6A26F365-CAA6-493E-B72C-8D07F48B177B, ‘500x500’‘)
INSERT INTO T_String VALUES(‘6A26F365-CAA6-493E-B72C-8D07F48B177B‘, ‘271165‘)
INSERT INTO T_String VALUES(‘45BA521E-7279-493F-89C3-7C5657A355CF‘, ‘COLUMN ‘)

…(その他データ型のテーブルを省略する)</td></tr>
</table>

Appendix B. Result of converting Ifc data to Sql.

抄録

Building Information Model（BIM）データは、膨大かつ多種多様なデータ形式が存在しているため、大量なBIMデータを永続的に保存することと、異なる形式のBIMデータへの対応が必須になっている。そのような中で、BIM データを利活用する可能性を広げるために、サーバーの構築が重要になっている。BIM サーバーを利用することで、煩雑な BIM データを統合・共有できるのみならず、建設業界以外への情報連携することも可能になる。本研究では、buildingSMART International（bSI）が策定している BIM データの国際標準仕様である Industry Foundation Classes（IFC）を対象に、IFC を Relation Database（RDB）に変換した後の、RDB におけるデータ整合性とパフォーマンスのトレードオフについて分析を行ったものである。データベースは，従来のファイルベース·よりも、データの関連付け、データ保存量、データの共有・管理にメリットを得ている。中でも、Relation Database（RDB） は安定した性能があり、標準化されている SQL を使う、データの整合性を保つ仕組みがある、という特徴があり、多くの研究者がRDBを利用してBIMサーバーの構築している。しかし、大量かつ多様な BIM データの RDB 化では、RDB におけるデータ整合性とパフォーマンスのトレードオフを行うことが避けられない。既存の研究では、IFC ベースの RDB が構築できるか否かに焦点が当てられており、これらのトレードオフがシステムに与える影響はまだ解明されていない。そこで、本研究は、IFC データを RDB に格納するために、実現可能なすべてのテーブル設計パターンを定義し、更に Query 実験によって性能とデータ整合性の両方から最適な設計パターンを見出した。その結果、エンティティごとにテーブルを作成し、外部キーを持たせることによって継承関係を維持するパターンが、データの整合性を最大限に保持しかつ一定のパフォーマンスを得ることができることを明らかにした。

Response letter

Official Review of Paper10 by AnonReviewer 2

● ***Overall: While it is stated in Section 2 that the authors "develop a method for building a BIM server," the contribution could be somewhat identified in the automated process to convert IFC schema to SQL statement; which is only a part of configuring the BIM server for a specific purpose. The paper fails to introduce the research to identify why this technique is worth developing in the broader context. Since the article is titled "The study of BIM server focusing on IFC to RDB conversion," the following questions should be answered: why does the conversion matter? What is the purpose of such conversion and what is the scope (and limitation)? What are the expected use cases and why are they important***

We apologize for the purpose of the study was not stated clearly enough in the previous version, and that the background of the study was too long. This made it difficult to understand the research objective of our study. We have greatly improved the clarity of the paper and thoroughly revised the background, research objective, research methods, and the data description in the revised version.

● ***Background: In Section 1, the authors state "this study proposes that BIM data should not be contained in a 3D urban model as a mere information source, but should aim to become a data services provider," which is literally inconsistent with the original purpose of using BIM. Despite some examples being drawn, the authors should refer to more existing publications to reinforce the argument.***

We greatly appreciate the reviewer's suggestion. We have revised the background section to summarize the existing research regarding the approaches to integrate BIM with GIS, which includes two commonly used approaches, namely data format conversion and standard extension. However, we believe that both methods have limitations for the use of BIM data, for the reason that they are all about extracting and transferring the required BIM data into a specific data format to be used at a certain time for specific needs. With the continuous development of the city, the corresponding needs are constantly changing, and therefore, the two methods above may be difficult to cater the changing needs of the future city construction.
Moreover, since previous studies suggest that BIM server could store all BIM data in a unified form internally and exposes the required BIM information to client applications in the form of APIs according to users' requirements [11], [12]. Because the BIM server is in a separate state from the client application, it would made it easier to develop new applications using BIM data. Therefore, we believe that the use of BIM server will be more able to cope with the changing needs of future urban construction than data format conversion or standard extension.

The full revised discussion is on Section 1.1 page 1 to 2.

● ***Research Objectives: In addition to the above, what are the expected contributions? What is more than "just good to have?"***

We apologize for the lack of clarity in stating the research objective of our study. In the revised version, we have indicated that the purpose of this study is to define and establish all feasible database table design solutions for storing IFC data into RDB, and evaluates the relative performance of each solution through query experiments to find a design solution that maximizes both performance and data consistency. We expect this study to make at least following contribution. First, when building IFC servers, the different database table design schemes mean that it is inevitable to make trade-offs between the access efficiency and data consistency. However, the existing literature remains silent on the impact of these trade-offs on the system, as it mostly focuses on whether RDB can be built or not. Such an impact can be assessed through our analysis. Secondly, some studies have been conducted to compare RDB with other types of databases [19], [20]. This paper provides an understanding of whether the database design solutions used in these studies are the most representative for RDBs. Finally, since this paper lists all the possible database design options and performs performance tests on each of them, it provides options for developers who are trying to use RDB to build IFC servers in terms of database design options.

The full revised discussion is on Section 1.2 page 2.

● ***Literature Review: The mentioned literature is biased and unexhaustive. There exist tens of academic publications regarding IFC conversion, for example, Automatic conversion of IFC datasets to geometrically and semantically correct CityGML LOD3 buildings (Donkers et al, 2016) or An evaluation of IFC-CityGML unidirectional conversion (El-Mekawy et al, 2012), as long as I searched. The authors must introduce the relevant existing research, allocate their argument to them, and prove that the parallel efforts are made compared to the reference.***

We would like to thank the reviewer for pointing us to the related literature and helping us to distinguish our contribution from existing studies. We again apologize for the unclear research objective stated in the previous version, which might create misconception that this study is related to the issue of IFC and cityGML conversion. Since this study is about the optimal design of database tables when storing IFC data in a RDB, we only refer to prior studies on IFC databases in our literature review because they are closely related to our study. Nonetheless, we have incorporated those studies suggested by the reviewer into the background section as they are relevant to our study in a broader context.

The full revised discussion is on Section 2 page 2 to 3.

● ***Research Methodology: The current statements have little to do with the actual methodology. If the significance lies in the converter and the evaluation process, it must be detailed in the section. It is slightly mentioned in Section 7, but this should be fully described in the earlier part. It also should be considered to implement the existing research framework to reorganize the paper.***

We greatly appreciate your comments. In the revised version of the paper, we have added a section 'IFC to RDB mapping rules defined for experiment' to specifically describe all possible database design solutions. The 'Research methodology' section now describes how the designed solution is implemented, how the RDBs composed of different solutions are created, and how the performance of each RDB is evaluated by Query Test. This would allow us to derive the most suitable solution.

● ***Research Core: Sections 5 and 6 are unnecessary. The authors should focus on the necessary concept by using the reference and elaborate more on what actually they developed/analyzed. The meaning of provided figures is very vague; the authors should thoroughly revise the figures to address the original contribution of the paper.***

Following your suggestion, we have re-organized the structure of our paper. We have shortened and revised the original Sections 5, 6 changed to Section 3, 4 in new version. Section 3 will first briefly describe the details of the IFC schema. Then it discusses the specific schemes of various mapping rules and their advantages and disadvantages. Section 4 introduces the automatic IFC to RDB conversion tool developed in this study. We have also revised the discussion on the figures provided in the original version. The Figure 2 lists the diagram of the various Mapping rules when converting the IFC schema to RDB, and The Figure3 provides an example showing two instances of IfcCartesianPoint with coordinate values (X=0, Y=0, Z=0) and (X=0, Y=0, Z=100) deposited into the database.

● ***Section 7 seems to be the core of the paper, however, the authors do not clearly identify the problem and elaborate on the efforts they made. Some technical challenges are highlighted, but the fundamental question then comes back to: "what is better than the existing ones, and why do we need it." The paper should be streamlined to explain why that is the missing piece to solve the problem in line with the broader context.***

We are grateful for the constructive suggestions and have revised the discussion to provide the rationale behind the approach used in our study. Since the IFC schema is defined in the EXPRESS language, which is a data model language, not a programming language. Therefore, for the computer to be able to execute the corresponding mapping rule, the IFC model needs to be reproduced in the form of a programming language first. This is one of the reasons why it is difficult to freely convert IFC patterns to RDB according to the pre-designed mapping rules. In this study, we design all the feasible mapping rules for converting IFC to RDB, and then develop the EXPRESS language parser based on c# programming language using ANTLR, and because of this, we can develop a program that can automatically convert IFC patterns to RDB according to various mapping rules.

The full revised discussion is on Section 1.2 page 2.

● ***Section 8 only has a poor description of what was done. What are the details of the data? Why are these different types deployed? Where is the reference for BUCKY Benchmark, what is it, and why is it adopted? What was the purpose to test these three data sets? Overall, what can be scientifically concluded from the benchmark test and what is the implication (to be addressed in the Discussion section)?***

We apologize for the mixed signals provided in the original version and have revised the discussion. In this study, the data import, BUCKY Benchmark, and rebar query are used to test the relative performance between IFC-RDB conversions constructed by various mapping rules or database table design schemes. BUCKY is a commonly used query-based benchmark specifically designed for testing the performance of ORDBs and RDBs. It is composed of 16 queries for testing five areas, namely inheritance, object references, collection data types methods, and abstract data types [29].Due to the existence of an RDB version, BUCKY can also be used to compare the performance of RDBs composed of different database table design schemes. This study follows the existing literature on testing IFC databases [27],[30] and selects 10 QUERYs from the original BUCKY that are suitable for testing IFC data. Depending on the number of instances, we have selected three sets of files in our experiments. These files are selected to allow us to know not only the horizontal difference (e.g., the performance difference between RDBs constructed by different mapping rules for the same number of instances of IFC files), but also the vertical comparison (e.g., the impact on the performance of RDBs constructed by the same mapping rule when storing different numbers of instances of IFC files).

The full revised discussion is on Section 4.2 page 8.

● ***Discussion and Conclusion: The conclusion is not clear and conclusive. The current sections only say that the authors created an automatic IFC conversion tool that sounds potentially useful in building RDB. What are the significances compared to the existing approach? What was technically overcome, that has not been resolved in the existing efforts? What are the limitations? And most importantly, how do you answer the question you formed in the introduction?***

We are grateful for the valuable and constructive comments, and have revised the section to improve the relevance of this study. This study systematically analyzes all feasible database design schemes for converting IFC schema to RDB and develops tools for automatically converting IFC schema files to RDB according to different data table design schemes. Accordingly, the performance of various database design schemes is compared systematically by various query tests. Through the experimental results, we find that when storing IFC data in the form of RDB, the attribute to table method is much lower than the entity to table method in terms of data import and data fetching performance, although the overall number of database tables is small to facilitate data management. In the entity to table method, the method of creating database tables for both abstract entities and non-abstract entities (the P2 method mentioned above), when compared with other methods, can maintain data consistency to the greatest extent while also performing at a relatively high level. Therefore, we believe that this method is the most suitable method for building IFC RDB at this stage. In addition, in recent years, commercial IFC model server services have emerged and are used in the development of cloud-based BIM applications for design, construction and maintenance. bSI has been working on standardization of a common API specification for IFC model servers, named openCDE API, and the results of this study could help bSI to carry out such work. Nonetheless, although this study compares the optimality of IFC RDBs built from various mapping rules using data access and ensuring data consistency as metrics, there may be other factors affecting the optimality, such as the completeness of IFC schema representation of the mapped tables, which could be further explored and studied in the future.

The full revised discussion is on Section 6 page 11 to 12.

Some additional comments:

● ***Please write the full form of the acronym for the first time, like RDB.***

Thank you for the suggestion. We have revised abbreviation thought out the text.

● ***Please revise the Japanese abstract.***

Following your suggestion, We have revised the Japanese abstract.

Official Review of Paper10 by AnonReviewer 3

● ***Section 1 Background covers the necessity of BIM data service by touching on the latest 3D city model use cases. The author also states that the difficulties to realize the BIM data service as wide range of data, and complexities of discipline BIM data handling process and specifications. It can be valid statement, and would have been better if the role of IDM (Information Delivery Manual) and MVD(Model View Definition) are mentioned here to describe the BIM process and BIM data specifications.***

We greatly appreciate your suggestion. We have added the discussions on the role of IDM (Information Delivery Manual) and MVD(Model View Definition)into the Section 1.1.

"……buildingSMART International (bSI) has created a series of international standards such as Industry Foundation Classes (IFC), Information Delivery Manual (IDM) and Model View Definition (MVD). Among them, IFC defines the BIM data structure, IDM defines the business process of data conversion, and MVD can define the data subset of IFC model according to the Exchange Requirements (ER) defined by IDM [14].(page 2)"

● ***Section 2 Research objectives: The term of "original BIM data" is more commonly used as "native BIM data".***

Thank you for the suggestion. We have changed the term "original BIM data" to "native BIM data" in the revised version.
"At present, since native BIM data is not created based on IFC, it leads to the problem of semantic mismatch and missing information when exporting BIM data from authoring BIM software to IFC format. (page 2)"

● ***Section 3 Literature review covers comprehensive previous studies on BIM data model, especially EXPRESS language, mapping to different kinds of database systems.***

We would like to thank the reviewer for the comments.

● ***Section 4 Research Methodology: In this section, the authors direct the use of RDB to store IFC data into IFC server. It is better that further discussion of comparison between RDB approach and other database systems, such as key-value, big table, and so on.***

Following your suggestion, we have revised the literation review to include the discussion on the advantages and disadvantages when using a non-relational database to store BIM data. The detailed discussion is as followed.

“In terms of research on NoSQL databases, Watanuki used graph database to design a data model storage method independent from IFC schema version differences [23].Beetz et al. developed a BIM server using a key-value storage Database Management System (DBMS) [24]. Although NoSQL databases are fast, efficient, simple to extend and have some advantages in high concurrency processing, since most non-relational databases do not support transaction processing, they cannot guarantee data integrity and security during frequent data updates and deletions. Moreover, most NoSQL databases do not support SQL language queries, which makes it not only costly to learn and use, but also difficult to perform more complex queries [22]. (page 2 to 3) ”

Given that the use of RDB approach is the main focus of this study, the comparison between RDB and other database systems might not be closely related to the current project. However, we will compare RDB with other database systems in much more detail in a later study.

● ***Section 5 The structure of the IFC schema: This section seems to be the description of structure of EXPRESS language. It would be better that overview of IFC class hierarchy is mentioned here, so that the mapping rules described in the next section 6 will become much clearer.***

Thank you for the insightful advice。We have added a discussion regarding the overview of IFC class hierarchy to Section 3 in the revised manuscript.
“Additionally, there is also an inheritance relationship between entities. It is like the concept of base/derivative classes in object-oriented languages, where lower-level entities can obtain the properties defined by higher-level entities through inheritance without having to repeat the definition of these properties. (page 3)”

● ***Section 6 Different types of mapping rules: This section is a main body to describe the IFC schema to relational database (RDB) table mapping methodology. The authors state the table design method in the first half of the section, however the relation between the method pattens and mark number P! to P4 in the figure 1 is ambiguous. And the term of “DBing” should be replaced to a more general term.***

We are grateful for the advice。For the ease of the reader, we have re-organized the discussion on the various mapping rules used in our study. To improve clarify, we have replaced the word ″DBing″ with ″storing IFC data″, which is on Section 3 page 4.

● ***Section 7 About the tool to auto-convert IFC to RDB: It is better to specify the version of IFC4 schema, because IFC4 contains multiple releases at the moment. The part of discussion about the figure 3 of UML of IFC schema parser can be included in the Section 6.***

Following your suggestion, we have revised the discussion in Section 3.3 to clarify the version of IFC4 schema used in our study is IFC4_ADD2_TC1.

“In order to know the most suitable database table design scheme for converting IFC to RDB, this study developed a tool to automatically convert IFC schema to RDB according to different mapping rules for IFC4_ADD2_TC1. (page 7)”

We have also moved the figure 4 of UML to Section 3 in the revised version.

● ***Section 8 Experiment: The sentence, “benchmark tests were conducted on various DBs” could be misinterpreted as tests were conducted on different DB systems. In my understanding, multiple mapping rules were tested and compared.***

We apologize for the confusion. We have revised the indicated sentence to be as followed.
“This study uses the BUCKY benchmark method to evaluate the performance of the RDB generated based on mapping rules. (Section 4.2 page 8)”

● ***Section 8 Discussion should be 9 in numerical order, and Concluding remarks. The completeness of IFC schema representation of the mapped tables should be evaluated and discussed in the future. In recent years, commercial IFC model server services have been spreading, and have been used in the development of cloud-based BIM applications in design, construction, and maintenance phases. The buildingSMART International has been working on standardizing the common API specification for IFC model server, named openCDE API, and it is expected that the findings from this paper will contribute to such buildingSMART efforts.***

Thank you very much for your suggestion. We have revised the section numbers throughout the text and incorporated your suggestion into Section 6.2 on page 12.

Some additional comments:

● ***Building Information Modeling, BIM, BIM data, and BIM model: The abbreviation BIM is usually known as Building Information Modeling to imply the process using digital representation in the built environment. On the other hand, the term of data of BIM itself appears BIM model or BIM data, and sometimes BIM as Building Information Model. The quotation of the building information modeling's definition from ISO 19650-1: use of a shared digital representation of a built asset to facilitate design, construction and operation processes to form a reliable basis for decisions.***

We had made clear the meaning of BIM in our study is BIM model. To further clarify, we footnote the definition of BIM used in our study on page 1 reference number 1.

● ***References [7], the following citation can be better reference: Y. Adachi, Overview of IFC Model Server Framework, European Conference of Product and Process Modeling ECPPM 2002***

Thank you very much for your suggestion. We have changed the reference [20] to the Y. Adachi, Overview of IFC Model Server Framework, European Conference of Product and Process Modeling ECPPM 2002