Development of an Object-Relational IFC Server

Hoon-sig Kang[1], **Ghang Lee**[2]

[1]Graduate Research Assistant, Building Informatics Group, Department of architectural engineering, Yonsei University, Seoul, Korea
[2]Assistant Professor, Building Informatics Group, Department of architectural engineering, Yonsei University, Seoul, Korea
Correspond to Ghang Lee at glee@yonsei.ac.kr

ABSTRACT: In this paper we propose a framework for an Object Relational IFC Server (OR-IFC Server). Enormous amounts of information are generated in each project. Today, many BIM systems are developed by various CAD software vendors. Industry Foundation Classes (IFC) developed by International Alliance for Interoperability (IAI) is an open standard data model for exchanging data between the various BIM tools. The IFC provides a foundation for exchanging and sharing of information directly between software applications and define a shared building project model. The IFC model server is a database management system that can keep track of transactions, modifications, and deletions. It plays a role as an information hub for storing and sharing information between various parties involved in construction projects. Users can communicate with each other via the internet and utilize functions implemented in the model server such as partial data import/export, file merge, version control, etc. IFC model servers using relational database systems have been developed. However, they suffered from slow performance and long transaction time due to a complex mapping process between the IFC structure and a relational-database structure because the IFC model schema is defined in the EXPRESS language which is object-favored language. In order to simplify the mapping process, we developed a set of rules to map the IFC model to an object-relational database (ORDB). Once the database has been configured, only those pieces of information that are required for a specific information-exchange scenario are extracted using the pre-defined information delivery manual (IDM). Therefore, file sizes will be reduced when exchanging data, meaning that files can now be effectively exchanged and shared. In this study, the framework of the IFC server using ORDB and IDM and the method to develop it will be examined.

Keywords: BIM, IFC, IFC Server, ORDB,IDM

1. INTRODUCTIO

As a result of recent progress in computer technologies, BIM (Building Information Modeling, BIM) Software is presently being developed that makes each object express the forms and attributes of buildings. This is quite unlike the past 2D CAD and 3D CAD software that had only geometric models. Using BIM tools, diverse participators (architects, structure engineers, facility engineers, cost estimator, scheduler) can create and share numerous pieces of information throughout the lifecycle of a construction project. However, because each of the BIM Tools has its own file format, it becomes difficult to exchange this information. According to Gallaher et al.[1], approximately 4.25% of the entire work costs is wasted in the U.S Capital Facilities Industry due to the problem of incompatibility. As such, the exchange and sharing of construction information are becoming salient issues in BIM Processes. To solve these problems, IAI (International Alliance for Interoperability) has presented IFC (International Foundation Classes), which expresses construction information with the EXPRESS data modeling language as a standard construction information model. IFC is one of the standard models and is the most frequently used in the area of construction at this moment. It observes ISO 10303 standards and its 2X3 version is currently posted and used. Most of the BIM Tools can export or import IFC file formats, allowing users the ability to transform each BIM model into an IFC file format to exchange and share with each other. However, according to Fischer and Kam.[2], when IFC models were used in actual projects, three problems emerged: 1) increased time to exchange files due to increased sizes of IFC files resulting from increases in the sizes of models; 2) loss of information in IFC files due to the use of different IFC translators in diverse kinds of software; and 3) version control issues and the authority to use files. Due to these problems, information models became less consistent and were unable to provide necessary information throughout the entire processes of the projects.

These problems were identified even before the study by Fischer and Kam [2]. And as a way to solve these problems, an IFC Model Server was proposed, which would enable multiple users to import/export IFC models through a single web-based database [3]. The IFC Server solved the problems of version control and user authority management, but sizes of IFC files and loss of information in IFC files still remained as problems. Upload and download of models were typically performed via the internet and, for large models, this caused performance problems [4]. To relieve these problems, this paper suggests the adoption of IDM (information delivery manual) in model exchange and the use of ORDB (Object-Relational Database) as an IFC server and report the development progress.

2. Previous Efforts

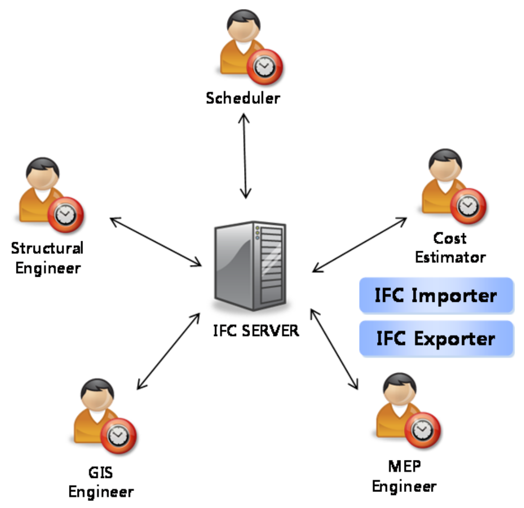


Fig 1. IFC file exchange using an IFC Server

The initial IFC Model Server was first developed in 2002 through the IMSvr project of SECOM in Japan and VTT in Finland [3]. The IFC model server enabled IFC file exchanges through the internet using SOAP (Simple Object Access Protocol) and XML (eXtensible Markup Language). It converted IFC files in EXPRESS format into XML models and then further into SQL code using XSLT (eXensible Stylesheet Language Trans formation, hereinafter XSLT) and an XML Parser. The converted SQL file was read into a relational database (RDB) management system for further use [3].

In the case of EMS (Eurostep Model Server For IFC) developed by Eurostep, SOAP and XML were used. XML based PMQL (Product Model Query Language) was developed as a substitute for the XML Parser for converting a IFC file into SQL code (Karstila, 2002). In the case of these two early IFC model servers, several different servers existed for different sets of information which were required for keeping track of exchanged data. Consequently, the problem of the necessity to access to multiple servers emerged [5].

Later, the 2003 SABLE (Simple Access to the Building Lifecycle Exchange, hereinafter, SABLE) project suggested the use of SABLE servers in order to more easily connect clients with multiple servers. A SABLE server is a sort of middleware server that basically uses XML data exchange methods. It suggests a standard interface for individual interfaces necessary to access to individual servers from existing IFC servers through SOAP, so that all the servers can be accessed. In the standard interface, IFC file Import/Export, file version management, session processing, and authority processing that were done in different IFC servers are now processed altogether. Therefore, clients now are able to use IFC servers more efficiently [6].

However, existing IFC Servers are not effective in IFC exchanges, because IFC models express the entire process of construction projects. This creates the problem of increased sizes of the files to be exchanged and results in reduced transfer speeds. In addition, since the concept of the relation-type database used to store IFC files was different from that of object-based IFC models, there was a mapping problem. To solve these problems, the current study seeks to reduce the sizes of IFC files and to increase the speed of file exchanges using IDM, by manually defining only those pieces of information necessary for the processes of information exchanges. The findings suggest an ORDB that is suitable to the attributes of the EXPRESS language, thereby presenting a method for increasing speeds in IFC file exchanges.

3. IFC and ORDB

3-1. IFC

The elements that constitute IFC object models include the key elements of Entities and Attributes as well as other elements such as Process Diagram, Relationships, and Interfaces. IFC2X3 consists of 653 Entities. Each of these Entities consists of the relations of inheritance and Attributes. IfcWallStandardCase is defined as per Table 1.

Table 1 1 IfcWallStandardCase

```
ENTITY IfcWallStandardCase

  SUBTYPE OF (IfcWall);

END_ENTITY;
```

IfcWallStandardCase appears to have no attributes, but it actually will inherit attributes from the Super type. IfcWallStandardCase will have inheritance structures and thereby 7 Supertypes: IfcRoot à IfcObjectDefinition à IfcObject à IfcProduct à IfcElement à IfcBuildingElement à IfcWall (where • à € depicts the relationship from a supertype to a subtype: i.e., supertype à subtype)

It is not necessary to define each column in IfcWallStandardCase, as the Columns will be inherited from higher classes. For instance, it is not necessary to define GloblaId in IfcWallStandardCase, as Data Types will be inherited from IfcRoot. IfcWallStandardCase will actually inherit 8 Attribute Types from IfcRoot, IfcObject, IfcProduct, and IfcElement, as shown in Table2.

Table 2 IfcWallStandardCase Attribute

Attribute	Type	DefinedBy
GlobalId	IfcGloballyUniqueId (STRING)	IfcRoot
OwnerHistory	IfcOwnerHistory (ENTITY)	IfcRoot
Name	IfcLabel (STRING)	IfcRoot
Description	IfcText (STRING)	IfcRoot
ObjectType	IfcLabel (STRING)	IfcObject
ObjectPlacement	IfcObjectPlacement (ENTITY)	IfcProduct
Representation	IfcProductRepresentation (ENTITY)	IfcProduct
Tag	IfcIdentifier (STRING)	IfcElement

Table 3 IfcWallStandardCase Part21

```
#73=IFCWALLSTANDARDCASE ('2UEEWTCGzAYh5bfayGWtG0', #31, 'Basic Wall: Generic - 200mm:102163', $, 'Basic Wall:Generic - 200mm:398', #43, #72, '102163');
```

It will become possible to store the IFC files, defined as Part21, in the defined IfcWallStandardCase. IfcWall StandardCase will then have respective instances, as per Table 4.

Table 4 IfcWallStandardCase Data

Attribute	Part21
GlobalId	'2UEEWTCGzAYh5bfayGWtG0'
OwnerHistory	#31
Name	'Basic Wall: Generic - 200mm:102163'
Description	$
ObjectType	'Basic Wall: Generic - 200mm:398'
ObjectPlacement	#43
Representation	#72
Tag	'102163'

However, in this case, OwnerHistory will not have a single value, but will refer to the value of #31; IFCOWNERHISTORY then the value of #31; IFCO WNERHISTORY will refer to the value of #30; IFCPERSONANDORGANIZATION and the value of #2; IFCAPPLICATION and the value of #30; IFCPERSONANDORGANIZATION will again refer to the value of #28; IFCPERSOMN and #29; IFCORGANIZA TION and then #2; IFCAPPLICATION will refer to the value of #1; IFCORGANIZATION. Through these complicated reference relations, the value defined by the #31 of IFCWALLSTANDARDCASE will come to be known. It is difficult to implement these complicated inheritance relations and designated Attribute references by RDB. Even if they are implemented, each Entity should have all attribute values and referring attribute values, thus the mapping time to put Part21 files into RDB will be increased. To solve this problem, the possibility of mapping into ORDB, invoking the concept of OODB, will be examined.

3-2. ORDB

An existing RDB (Relational Database) stores all data in the form of tables. The tables consist of columns and rows. In this case, each of the columns is specified to have a basic data type, such as a letter type, a figure type, or a date type, etc. To manipulate or extract the data stored in the tables, the standard questioning language, called SQL (Structured Query Language), is provided. However, as database management has become more complicated recently, as in the case of CAD, GIS, and multimedia, problems, as follows, began to appear: 1) Data types cannot be expended in RDBMS. 2) It is difficult to express complex objects using tables. To solve these problems, the OODBMS (Object -Oriented Database Management System) provides object-oriented functions, such as inheritance or aggregation, to an existing RDB. However, the existing RDB could not be developed because its functions for data management, such as high grade questioning languages, etc, were insufficient. ORDBMS (Object-Relational Database Management System) accommodates all of the functions of RDBMS, while adding the concept of objects. By uniting the strengths of RDBMS and OODBMS, excellent data management functions are supported. ORDB enables the use of types made by users, in addition to the data types defined by the system, such as User defined data types in RDB. It also provides unique types where many values can be put into one field, such as Collection data type. In addition, it provides inheritance functions, so that the function to inherit from proved classes to make new classes is enabled.

3-3. IFC and ORDB

IFC and ORDB have similar naming regulations, as shown in Table5. To store IFC files that have the concept of objects in Databases, a process of mapping with ORDB language should be carried out.

Table 5 Comparison of RDB, ORDB, and IFC

RDB	ORDB	IFC
Table	Class	Entity
Column	Attribute	Type
Row	Instance	Instance

-	Subclass	Subtypes
-	Superclass	Supertypes

IFC files have essentially 3 kinds of object concepts, as follows: 1) The concept of inheritance, such as Subtypes and Supertypes. 2) User defined attribute values may be brought in as the values of Attributes. 3) Multiple instances such as SET, BAG, LIST, and ARRAY can be inserted into one column. The method to map IFC schemas having these concepts of objects into Cubrid, a commercial ORDB, is as follows: 1) The Subtypes and Supertypes in IFC may be defined as UNDER and ADD SUPERCLASS in ORDB.

Table 6 A Mapping Method for the Subtype/Supertype Relation

IFC 2X3 Schema: IfcWall Entity
ENTITY IfcWall SUPERTYPE OF (ONEOF (IfcWallStandardCase)) SUBTYPE OF (IfcBuildingElement); END_ENTITY;
ORDB QUERY
Subtype create Query CREATE IfcWallStandardCase CREATE IfcWall UNDER IfcWallStandardCase Supertype create Query ALTER CLASS IfcWall ADD SUPERCLASS IfcBuildingElement

2) In object relation type databases, user defined values of Attributes can be directly used as data types. Therefore, in order to enable the use of values of a column of a table as the values of a row of another table, the use of nested tables is enabled.

Table 7 A Mapping Method for a Nested Table

IFC 2X3 Schema : IfcWall Entity

ENTITY IfcProduct
ObjectPlacement : OPTIONAL IfcObjectPlacement;
Representation:OPTIONAL IfcProductRepresentation;
END_ENTITY;

ORDB QUERY

Create Class IfcProduct (
ObjectPlacement IfcObjectPlacement;
Representation IfcProductRepresentation;
)

3) In relation type models, a column can have one value only; thus, when multiple values are necessary, an additional table should be created and then the two tables should be combined. In contrast, object relation type databases can have multiple values using Collection Types. The SET, BAG, LIST, ARRAY types in EXPRESS can be mapped as SET, MULTISET, or LIST, respectively in ORDB, but ARRAY is not supported.

Table 8 A Mapping Method for SET Type

IFC 2X3 Schema : IfcPresentationLayerAssignment Entity

ENTITY IfcPresentationLayerAssignment
AssignedItems : SET [1:?] OF IfcLayeredItem;
END_ENTITY;

ORDB QUERY

CREATE CLASS IfcPresentationLayerAssignment (
AssignedItems set (IfcLayeredItem)
);

Table9 A Mapping Method for BAG Type

IFC 2X3 Schema: IfcNoOfLayers Function

FUNCTION IfcNoOfLayers
Association : BAG OF IfcRelAssociates := [];
END_FUNCTION;

ORDB QUERY

CREATE CLASS IfcNoOfLayers (
Association multiset (IfcLayeredItem)
);

Table 10 A Mapping Method for LIST Type

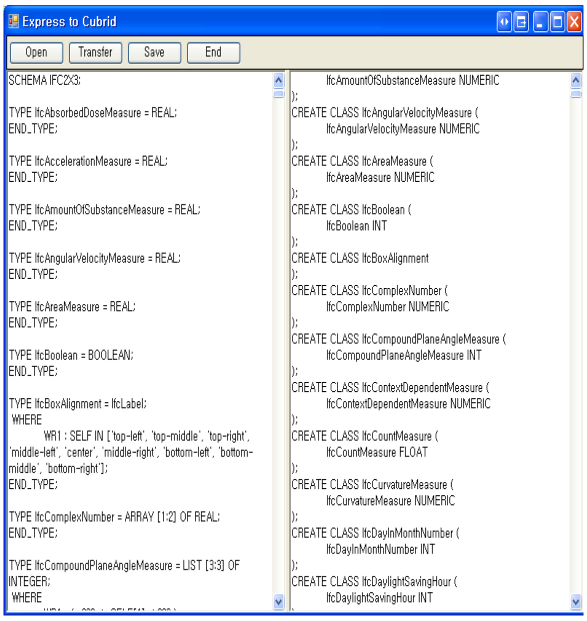
IFC 2X3 Schema IfcDirection Entity

ENTITY IfcDirection
 	
DirectionRatios : LIST [2:3] OF REAL;
 	
END_ENTITY;

ORDB QUERY

CREATE CLASS IfcDirection (
 	
DirectionRatios list (REAL)
 	
);

In addition to the 3 types of object concepts, it is difficult for users to firsthand transform data types and Inverse Rule, Derive Rule, Domain Rule in EXPRESS into ORDB Queries. To solve this problem, a program to automatically transform EXPRESS formats into Cubrid Queries, using mapping rules, was developed, as shown in Fig 2.



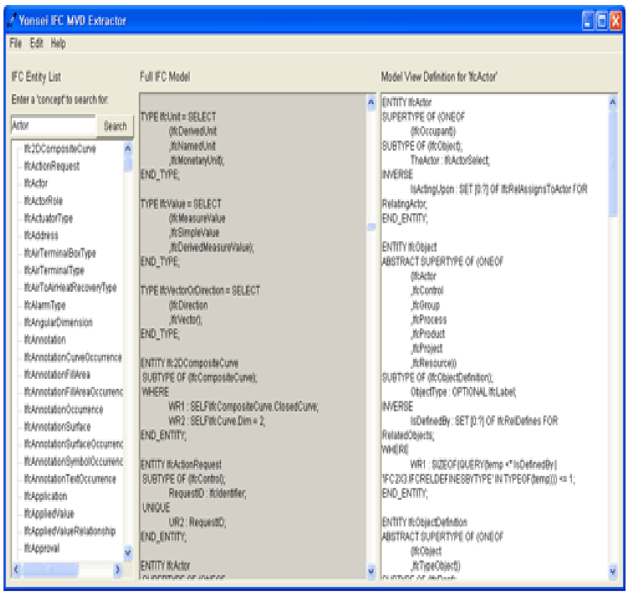
```
SCHEMA IFC2X3;

TYPE IfcAbsorbedDoseMeasure = REAL;
END_TYPE;

TYPE IfcAccelerationMeasure = REAL;
END_TYPE;

TYPE IfcAmountOfSubstanceMeasure = REAL;
END_TYPE;

TYPE IfcAngularVelocityMeasure = REAL;
END_TYPE;

TYPE IfcAreaMeasure = REAL;
END_TYPE;

TYPE IfcBoolean = BOOLEAN;
END_TYPE;

TYPE IfcBoxAlignment = IfcLabel;
 WHERE
        WR1 : SELF IN ['top-left', 'top-middle', 'top-right',
'middle-left', 'center', 'middle-right', 'bottom-left', 'bottom-
middle', 'bottom-right'];
END_TYPE;

TYPE IfcComplexNumber = ARRAY [1:2] OF REAL;
END_TYPE;

TYPE IfcCompoundPlaneAngleMeasure = LIST [3:3] OF
INTEGER;
 WHERE
```

```
        IfcAmountOfSubstanceMeasure NUMERIC
);
CREATE CLASS IfcAngularVelocityMeasure (
        IfcAngularVelocityMeasure NUMERIC
);
CREATE CLASS IfcAreaMeasure (
        IfcAreaMeasure NUMERIC
);
CREATE CLASS IfcBoolean (
        IfcBoolean INT
);
CREATE CLASS IfcBoxAlignment
);
CREATE CLASS IfcComplexNumber (
        IfcComplexNumber NUMERIC
);
CREATE CLASS IfcCompoundPlaneAngleMeasure (
        IfcCompoundPlaneAngleMeasure INT
);
CREATE CLASS IfcContextDependentMeasure (
        IfcContextDependentMeasure NUMERIC
);
CREATE CLASS IfcCountMeasure (
        IfcCountMeasure FLOAT
);
CREATE CLASS IfcCurvatureMeasure (
        IfcCurvatureMeasure NUMERIC
);
CREATE CLASS IfcDayInMonthNumber (
        IfcDayInMonthNumber INT
);
CREATE CLASS IfcDaylightSavingHour (
        IfcDaylightSavingHour INT
```

Fig 2. EXPRESS To Cubrid Query

4. IFC Server using IDM

IAI presents a method to prepare the processes of exchanges of IFC models and the information necessary for the exchanges with a manual called IDM, in an attempt to improve the compatibility of IFC files. IDM consists of Reference Processes, Process Map, Business Rules, Verification Tests, Exchange Requirements, Functional Parts, and Concepts, etc. PM expresses the processes of use and creation of model information in process models for easy understanding. ER is made in the form of a manual so that general users (architects, engineers, executors) can easily define when information exchanges are necessary in project processes. FP defines the subset of IFC models, so that software vendors can provide the ER required by users. The method to exchange IFC files using IDM does not use all of the information of IFC models. It defines part of IFC files necessary in the process of information exchanges as View, and then exchanges information through the IFC View, thereby reducing the time required for IFC file exchanges. To define and use this View, the Yonsei IFC MV(Model View) Extractor shown in Figure 3 can be used. The IFC MV Extractor is a tool developed in 2007 [9] that allows users to enter necessary IFC concepts in IFC Schema in order to extract related IFC Entities and types.



```
IfcZDCompositeCurve
IfcActionRequest
IfcActor
IfcActorRole
IfcActuatorType
IfcAddress
IfcAirTerminalBoxType
IfcAirTerminalType
IfcAirToAirHeatRecoveryType
IfcAlarmType
IfcAngularDimension
IfcAnnotation
IfcAnnotationCurveOccurrence
IfcAnnotationFillArea
IfcAnnotationFillAreaOccurrenc
IfcAnnotationOccurrence
IfcAnnotationSurface
IfcAnnotationSurfaceOccurrenc
IfcAnnotationSymbolOccurrenc
IfcAnnotationTextOccurrence
IfcApplication
IfcAppliedValue
IfcAppliedValueRelationship
IfcApproval
```

```
TYPE IfcUnit = SELECT
        (IfcDerivedUnit
        ,IfcNamedUnit
        ,IfcMonetaryUnit);
END_TYPE;

TYPE IfcValue = SELECT
        (IfcMeasureValue
        ,IfcSimpleValue
        ,IfcDerivedMeasureValue);
END_TYPE;

TYPE IfcVectorOrDirection = SELECT
        (IfcDirection
        ,IfcVector);
END_TYPE;

ENTITY Ifc2DCompositeCurve
 SUBTYPE OF (IfcCompositeCurve);
 WHERE
        WR1 : SELF\IfcCompositeCurve.ClosedCurve;
        WR2 : SELF\IfcCurve.Dim = 2;
END_ENTITY;

ENTITY IfcActionRequest
 SUBTYPE OF (IfcControl);
        RequestID : IfcIdentifier;
 UNIQUE
        UR2 : RequestID;
END_ENTITY;

ENTITY IfcActor
```

```
ENTITY IfcActor
SUPERTYPE OF (ONEOF
        (IfcOccupant))
SUBTYPE OF (IfcObject);
        TheActor : IfcActorSelect;
INVERSE
        IsActingUpon : SET [0:?] OF IfcRelAssignsToActor FOR
RelatingActor;
END_ENTITY;

ENTITY IfcObject
ABSTRACT SUPERTYPE OF (ONEOF
        (IfcActor
        ,IfcControl
        ,IfcGroup
        ,IfcProcess
        ,IfcProduct
        ,IfcProject
        ,IfcResource))
SUBTYPE OF (IfcObjectDefinition);
        ObjectType : OPTIONAL IfcLabel;
INVERSE
        IsDefinedBy : SET [0:?] OF IfcRelDefines FOR
RelatedObjects;
WHERE
        WR1 : SIZEOF(QUERY(temp <* IsDefinedBy |
'IFC2X3.IFCRELDEFINESBYTYPE' IN TYPEOF(temp))) <= 1;
END_ENTITY;

ENTITY IfcObjectDefinition
ABSTRACT SUPERTYPE OF (ONEOF
        (IfcObject
        ,IfcTypeObject))
```

Fig 3. Yonsei IFC MV Extractor

5. ORDB IFC Server Framework

The OR-IFC server suggested in this study is indicated by the structure shown in Figure 5. Project participators can upload IFC files on the OR-IFC and select only the part necessary for the stage of their process for Export/Import, through the IFC view created through IDM.

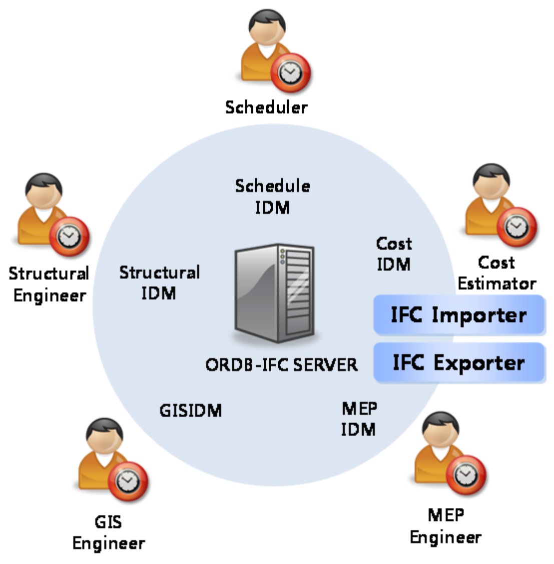


Fig 4. ORDB-IFC Server Framework

6. CONCLUSION

IFC servers have been developed in several projects for efficient data sharing and compatibility, but they have not been commercially successful. The OR-IFC server presented in this paper, unlike existing IFC servers, uses the ORDB added with the concept of objects and the information necessary in processes is defined as IDM, in order to propose an OR-IFC server that will enable efficient data exchanges. The components necessary for OR-IFC servers and OR-IFC frameworks are also suggested. Along with the IDM, which is still conceptual, if the mapping of IFC to ORDB is successfully implemented, it will become possible to more efficiently share and manage data. Future studies will be made on the approaches for inputting actual IFC data into ORDB and outputting them, while other studies will be made in relation to the comparison of performances between RDB and ORDB. Still other investigations will be carried out to determine the optimal way to enter extracted IFC views into ORDB using IDM.

ACKNOWLEDGEMENTS

This work was supported by the Korean Institute of Construction & Transportation Technology Evaluation and Planning (KICTEP) with the program number of "06-Unified and Advanced Construction Technology Program-E01. "

REFERENCES

[1] Gallaher, M.P., O'Connor, A. C., Dettbarn, J. L., Jr., Gilday, L.T. "Cost Analysis of Inadequate Interoperability in the U.S. Capital Facilities Industry.", NIST Advanced Technology Program, Information Technology and Electronics Office, Gaithersburg, Maryland 20899, 2004

[2] M.Fischer, C.Kam, "PM4D Final Report€, CIFE Technical Report(TR143)", CIFE, Stanford University, 2002.

[3] Adachi, Y., "IFC Model Server development Project • official webpage" VTT building and SECOM, Finland, 2002. http://cic.vtt.fi/projects/ifcsvr/index_exc.html

[4] K.A. J, rgensen, J. Skauge, P. Christiansson, K. Svidt, K.B. S, rensen, J. Mitchell, "Use of IFC Model Servers - Modelling Collaboration Possibilities in Practice", Aalborg University & Aarhus School of Architecture, 2008.

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

[6] Houbaux, P. "SABLE of BLIS-Project • official webpage", Eurostep, 2003 http://www.blis-project.org/~sable

[7] Kiviniemi, A. "Review of the development and implementation of IFC compatible BIM", Erabuild, 2007.

[8] Lee, G. "Concept-Based Method for Extracting Valid Subsets from an EXPRESS Schema" *Journal of Computing in Civil Engineering*, Vol. 22, No. 2, pp. 128-135, 2009.

[9] Lee, G. "IFC Model view extractor • official webpage", 2007. http://biis.yonsei.ac.kr:8080/mvd.htm

[10] Wix, J. "The Information Delivery Manual. • official webpage" , 2007. http://idm.buildingsmart.no/confluence/display/IDM/Home