OUDG: CROSS MODEL DATUM ACCESS WITH SEMANTIC PRESERVATION FOR LEGACY DATABASES

Joseph Fong[1] and Kenneth Wong[2]

[1]Department of Computer Science, City University of Hong Kong, Hong Kong
[1]csjfong@cityu.edu.hk,[2]wting_yan@hotmail.com

ABSTRACT

Conventional databases are associated with a plurality of database models. Generally database models are distinct and not interoperable. Data stored in a database under a particular database model can be termed as "siloed data". Accordingly, a DBMS associated with a database silo, is generally not interoperable with another database management system associated with another database sil. This can limit the exchange of information stored in a database where those desiring to access the information are not employing a database management system associated with the database model related to the information. The DBMS of various data models have proliferated into many companies, and become their legacy databases. There is a need to access these legacy databases using ODBC. An ODBC is for the users to transform a legacy database into another legacy database. This paper offers an end user's tool of Open Universal Database Gateway(OUDG) to supplement ODBC by transforming a source legacy database data into Flattened XML documents, and further transform Flattened XML document into a target legacy database. The Flattened XML document is a mixture of relational and XML data models, which is user friendly and is a data standard on the Internet. The result of reengineering legacy databases into each other through OUDG is information lossless by the preservation of their data semantics in terms of data dependencies.

KEYWORDS

Open universal database gateway, Legacy databases, Flattened XML Documents, Data semantics, Data dependencies, Open database connectivity

1. INTRODUCTION

The database management system (DBMS) of various data models have proliferated into many companies and, over time, have become legacy databases within the companies. However, there is a need to access these legacy databases, e.g., for mass information transmission associated with e-commerce, etc. Legacy databases, e.g., conventional databases, can be associated with a plurality of database models, e.g., database silos. These database silos can be distinct and fail to interoperate without significant costs or loss of data or data semantic information. Siloed data, e.g., data within a database model acts is typically only readily accessible or interoperable within that database model and not with data stored in another database silo, can limit the exchange of information where those desiring to access the information are not employing a related DBMS.

David C. Wyld et al. (Eds) : ITCS, CST, JSE, SIP, ARIA, DMS - 2015
pp. 165–184, 2015. © CS & IT-CSCP 2015 DOI : 10.5121/csit.2015.50117

Additionally, even where a database environment is relatively modern, it can be incompatible with other relatively modern database silos. The plurality of database silos in itself can be an impediment to sharing data among them. As an example, where a first company employs a first database associated with a first database model, a second company employs a second data model for their data, and a third company employs a third data model for their data, sharing of data across the three data silos can be impractical or impossible. Where the first company purchase the second company, incorporating the second company's data can be problematic, e.g., it can require rewriting the data into the first data model at the risk of losing data or semantics. Alternatively, the first company can operate the two databases separately but are then internally faced with the incongruences of the two databases, bear the costs associated with operating or maintaining two separate databases, etc. Further, the first company, even with access to the first and second databases, still can face serious challenges with sharing data with the third company.

The evolution of database technologies intend to meet different users requirements. For example, the Hierarchical and Network (Codasyl) databases(NDB) are good for business computing on the large mainframe computers. The user friendly relational databases(RDB) are good for end user computing on personal computers. The object-oriented databases(OODB) are good for multi-media computing on mini computers. The XML databases(XML DB) are good for Internet computing on the mobile devices. These are first generation Hierarchical and Network databases, second generation relational databases, and third generation Object-Oriented and XML databases.

Flattened XML documents

Flattened XML documents are generic representation of any legacy database instance in any legacy database data model. Flattened XML document is a valid XML document which contains a collection of elements of various types and each element defines its own set of properties. The structure of the flattened XML document data file is a relational table structured XML document. It has XML document syntax with relational table structure. It replaces primary key with ID, and foreign key with IDREF as follows:

```
<?xml version="1.0">
<root>
 <table1 ID="…" IDREF1="…" IDREF2="…" … IDREFN="…">
  <attribute1>…</attribute1>
    …
  <attributeN>…</attributeN>
 </table1>
 …
 <tableN ID="…" IDREF1="…" IDREF2="…" … IDREFN="…">
  <attribute1>…</attribute1>
  …
  <attributeN>…</attributeN>
 </tableN>
</root>
```

For each table, the name of the table (tableN) determines its type name and the name of property (attributeN) determines its property name. Each table defines an ID type attribute that can uniquely identify itself and there are optional multiple IDREF type attributes that can refer to the ID in other tables in the same flattened XML document instance. Each property XML element encloses a property value in a proper textual representation format. In order to ensure a flattened XML document instance to be valid, there must be either an internal or an external DTD document that defines the XML structures and attribute types, in particular for those ID and IDREF type attributes.

Open Universal Database Gateway

An open universal database gateway(OUDG) is a database middleware which provides more flexibility for the users to access legacy databases in their own chosen data model. Users can apply OUDG to transform legacy databases into flattened XML documents, and then further transform them into user's own familiar legacy database for access. Since XML is the data standard on the Internet, it becomes information highway for user to access data.

The reason we choose flattened XML document is due to its openness for DBMS independence. All other data models are DBMS dependent. For example, an Oracle database can only be accessed by Oracle DBMS, and a MS SQL Server database can only be accessed by MS SQL Server DBMS. Nevertheless, users can access flattened XML documents on the Internet by Internet Explorer without programming. Therefore, an Oracle user can access an MS SQL Server database by using OUDG transforming the MS SQL Server database into flattened XML document, and then further transform flattened XML document to Oracle database.

Similarly, the reason we choose relational table structure for the flattened XML document is that relational table structure has a strong mathematical foundation of relational algebra to implement the constraints of major data semantics such as cardinality, isa and generalization to meet users' data requirements by replacing primary keys and foreign keys into ID(s) and IDREF(s) in XML schema.

The OUDG can transform legacy databases into flattened XML document, and then further transform the flattened XML document into one of four target legacy databases: relational, object-oriented, XML and network. The result is that OUDG allows users reengineer a source legacy database into an equivalent target legacy database of user's choice with data semantics preservation.

This paper offers flattened XML documents as universal database medium for the interoperability of all legacy databases that can be accessed by the users using their own familiar legacy database language via OUDG. We consider hierarchical data model same as XML data model in this paper because they are all in tree structure. All proprietary legacy data models can be united into flattened XML document as universal database as shown in Figure 1.

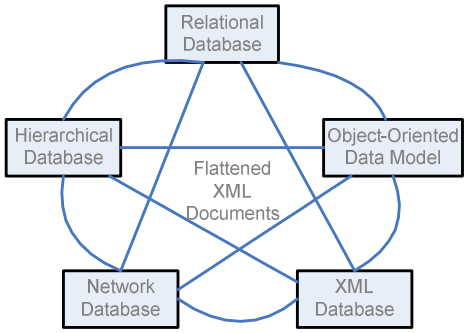


Figure 1. Legacy Databases Interoperability via Flattened XML documents

The major difference between regular tree-based XML document and flattened XML document is that the latter is more user friendly and efficient in database update since its database navigation access path is much shorter than the former. The flattened XML document database navigation access path is limited to 2 levels from root element to sibling elements while the regular XML document database access path is in several levels and much longer in general.

Problems:
(1) Most legacy database systems are proprietary. Database vendors do not facilitate tools to export their databases to other legacy databases. Thus, companies need to use ODBC to access heterogeneous databases, which requires programming and much time effort.
(2) Most users cannot access all legacy databases because they do not know all legacy database languages. They rely on ODBC, which is not easy to learn.
(3) It is difficult to convert legacy databases in different data models because the data conversion of legacy database involves data models transformation.

Solution:
Through OUDG, users can use one database language access another legacy databases of relational, object-oriented, network and XML. The operation is more reliable and speedy because same data can be concurrently processed by legacy database and their equivalent flattened XML document.

Academic merit:
The novelty is that it is feasible to replace ODBC by OUDG transforming legacy database into flattened XML document for access. ODBC needs programming, but OUDG is an end user software utility.

Industrial merit:
The application of flattened XML document is for information highway on the Internet for data warehouse, decision support systems (Fong, Li & Huang, 2003). The benefits are information sharing among users for database interoperability.

OUDG as supplement for ODBC
OUDG can supplement ODBC to access any legacy database by transforming(reengineering) them into a flattened XML document for access as universal database which is defined as a database interchangeable to all legacy databases.

At present, most database systems are proprietary. Each DBMS vendor has software tools which convert other legacy databases into databases using their own DBMS(s), but not vice versa for converting their own databases into a target legacy database. The result makes legacy databases not open to each other. However, using OUDG, any legacy database can be transformed into any other legacy database via flattened XML documents. The benefit is that data sharing and data conversion among legacy databases becomes possible. The openness of legacy database is necessary for such application such as data warehousing, data mining and big data.

Figure 2 shows the architecture of an open universal database gateway which transforms legacy databases into each other with different data models via flattened XML document as a replacement for open database connectivity.

Data Semantics preservation in legacy databases
Data semantics describe data definitions and data application for users' data requirements, which can be captured in the database conceptual schemas. The following are the data semantics which can be preserved among the legacy conceptual schemas and their equivalent flattened XML schema:

(a) Cardinality: 1:1, 1:n and m:n relationships set between two classes
A one-to-one relationship between set A and set B is defined as: For all a in A, there exists at most one b in B such that a and b are related, and vice versa. The implementation of one-to-one relationship is similar to one-to-many relationship.

A one-to-many relationship from set A to set B is defined as: for all a in A, there exists one or more b in B such that a and b are related. For all b in B, there exists at most one a in A such that a and b are related.
A many-to-many relationship between set A and set B is defined as: For all a in A, there exists one or more b in B such that a and b are related. Similarly, for all b in B, there exists one or more a in A such that a and b are related.
In relational schema, 1:n is constructed by foreign key on "many" side referring to primary key on "one" side. It can also be implemented by association attribute of a class object on "one" side points to another class objects on "many" side in object-oriented schema. It can also be implemented by owner record occurrence on "one" side and member record occurrences on "many" side in network schema. It can also be implemented by element occurrence with IDREF on "many" side links with element occurrence with ID on "one" side in XML schema. As to m:n cardinality, it can be implemented by two 1:n cardinalities with 2 "one" side classes link with 1 "many" side class.

(b) Isa relationship between a superclass and a subclass
The relationship A isa B is defined as: A is a special kind of B.
In relational schema, a subclass relation has same primary key as its superclass relation, and refers it as a foreign key in isa relationship. In object-oriented schema, isa can be implemented by a subclass inheriting its superclass's OID and attributes. In Network schema, isa can be implemented by an owner record that has same key as its member record in network schema via SET linkage. In XML schema, isa can be implemented by an element links one-to-one occurrence with its sub-element.

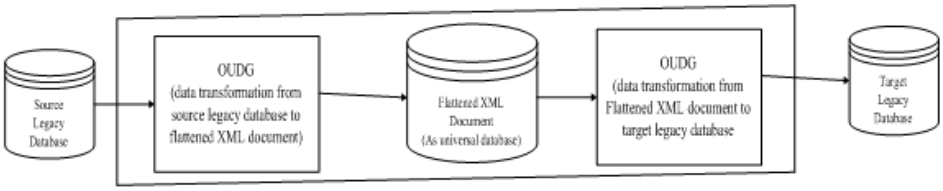


Figure 2. Architecture of OUDG with schema translation and data transformation

(c) Generalization is the relationship between a superclass and its subclasses.
Multiple isa relationships construct generalization with common superclass.
In relational schema, A is a special kind of B, and C is also a special kind of B, then A and C subclasses can be generalized as B superclass. In relational schema, multiple subclass relations and their superclass relation contain the same key, with subclass relations' keys referring to superclass key as foreign key in generalization. In object-oriented schema, multiple subclasses objects contain the same OID as their superclass object in generalization. In network schema, one owner record links with multiple member records through a SET in generalization. In XML schema, multiple subclass elements and their superclass element are in 1:1 linkage with same key attribute in generalization. Generalization can be implemented by multiple isa relationships with multiple subclasses generalized into one superclass.

Initially, OUDG maps major data semantics of cardinality, isa, and generalization into each legacy data model as shown in Table 1 which shows data semantics preservation in legacy data models and Flattened XML document

The preservation of data semantics among legacy databases can be verified by the preservation of their data dependencies as follows:

Definition of FD (functional dependency)

Given a relation R, attribute Y of R is functionally dependent on attribute X of R, i.e., FD: R.X → R.Y, iff each X-value in R has associated with it precisely one Y value in R. Attribute X and Y may be composite.

Definition of ID (inclusion dependency)
ID: Y ⊆ Z states that the set of values appearing in attribute Y must be a subset of the set of values appearing in attribute Z.

Definition of MVD (multi-valued dependency)
Let R be a relation variable, and let A, B and C be the attributes of R. Then B is multi-dependent on A if and only if in every legal value of R, the set of B values matching a given AC pair value depends on the A value, and is independent of the C value.

In general, the mapping and the preservation of the data semantics of cardinality, isa, and generalization among legacy databases schemas can be shown in Figure 3 as follows:

In one-to-many cardinality, for example, each child relation B tuple determines its parent relation A tuple in relational schema; each member record B determines its owner record A in network schema; each "many" side object B determines its associated "one" side object A in object-oriented schema, and each sub-element B occurrence determines its element A occurrence in XML schema.

In many-to-many cardinality, two one-to-many cardinality MVD(s) can construct a many-to-many cardinality. For example, many tuples in relation B determine many tuples in relation A and vice versa (many relation A tuples determine many relation B tuples); many records B determine many records A. Therefore many elements B occurrence determine many elements A occurrences, and vice versa.

Table 1 showing information related to data semantic preservation.

Data model\ Data Semantic	Relational	Object-Oriented	Network	XML	Flattened XML
1:n cardinality	Many child relations' foreign keys referring to same parent relation's primary key.	A class's association attribute refers to another class's many objects' OID(s) as a Stored OID.	An owner record data points to many member records data via SET linkage.	An element has many sub-elements.	The IDREF(s) of a "many" side sibling element's data refer to an ID of "one" side sibling element data under root element.

m:n cardinality	A relationship relation's composite key are foreign keys referring to 2 other relations' primary keys.	A class's association attribute refers to another class's many objects' OID(s) as an Stored OID, and vice versa.	Two owner records data point to the same member record data via 2 SETs linkages .	A sub-element of an element links another element IDREF referring to the latter's ID. The 2 elements are in m:n cardinality.	An sibling element data has 2 IDREF(s) referring to 2 other sibling elements ID(s) under root element.
Is-a	Subclass relation's primary key is a foreign key referring to its superclass relation's primary key.	A subclass inherit OID(s), attributes and methods of its superclass as its OID plus its own attributes and methods.	An owner record data links to a member record data in 1:1 with same key.	An element occurrence links with a sub-element occurrence in 1:1 linkage.	The IDREF of a subclass sibling element data refers to the ID of a superclass sibling element. Both elements has same key value under root element.
Generalization	2 subclass relations' primary keys are foreign keys referring to same superclass relation's primary keys.	Two subclasses inherit OID and attributes of their identical superclass as their OID plus their own attributes.	An owner record data occurrence points to two member records data occurrence with same key.	An element data occurrence links with two sub-elements data occurrence in 1:1 linkages.	The IDREF(s) of 2 subclass sibling elements data occurrence refer to an ID of a superclass sibling element data occurrence with same key value under root element.

In isa relationship, for example, each B tuple is a subset of A tuple; each record B is a subset of A record; each object B is a subset of object A; and each sub-lement B occurrences is a subset of element A occurrence. In generalization, the data dependencies are similar to isa relationship, except the pair of subclass B and C is a subset of superclass A.

The above data semantics can be preserved in flattened XML documents with sibling elements only, linking with each other via IDREF and ID as shown in Figure 4.

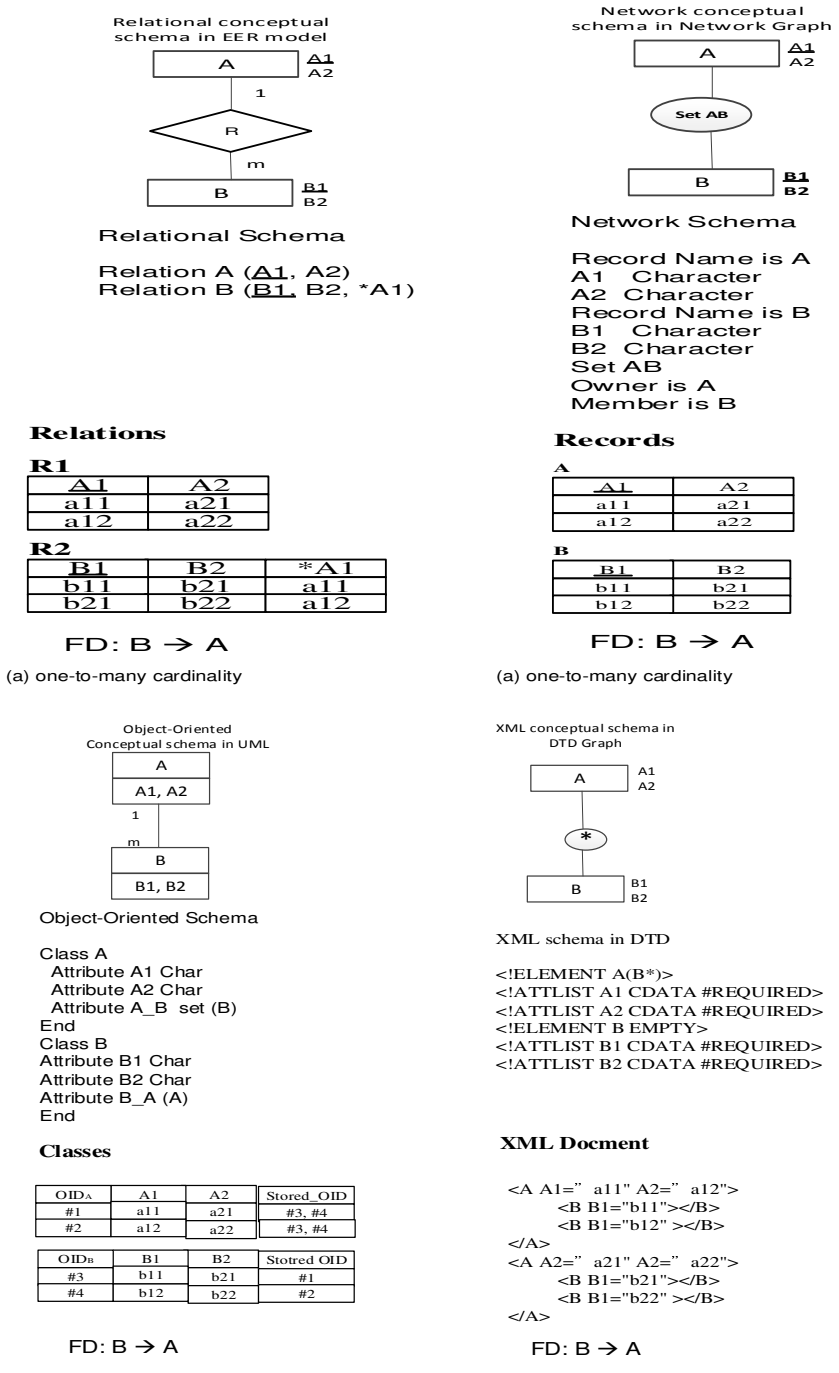
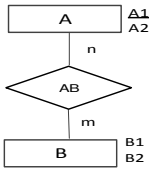


Figure 3a Data semantics preservation in legacy databases (1:n Cardinality)

(b) many-to-many cardinality

Relational conceptual schema in EER model



Relational Schema

Relation A (A1, A2)
Relation B (B1, B2)
Relation AB (*A1, *B1)

Relations

A

A1	A2
a11	a21
a12	a22

B1	B2
b11	b21
b21	b22

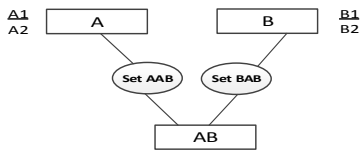
AB

*A1	*B1
a11	b11
a12	b21

MVD: B →→ A
MVD: A →→ B

(b) many-to-many cardinality

Network conceptual schema in Network Graph



Network Schema

Record Name is A
A1 Character
A2 Character
Record Name is B
B1 Character
B2 Character
Record Name is AB
Set AAB
Owner is A
Member is AB
Set BAB
Owner is B
Member is AB

Records

A

A1	A2
a11	a21
a12	a22

B

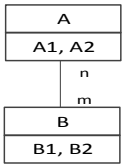
B1	B2
b11	b21
b12	b22

AB

A1	B1
a11	b11
a12	b21

MVD: A →→ B
MVD: B →→ A

Object-Oriented Conceptual schema in UML



Object-Oriented Schema

Class A
Attribute A1 Char
Attribute A2 Char
Attribute A_B set (B)
End
Class B
Attribute B1 Char
Attribute B2 Char
Attribute B_A set (A)
Member is B

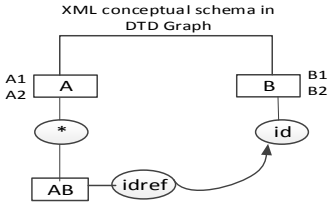
Classes

OID_A	A1	A2	Stored_OID
#1	a11	a21	#3, #4
#2	a12	a22	#3, #4

OID_B	B1	B2	Stotred OID
#3	b11	b21	#1, #2
#4	b12	b22	#1, #2

MVD: A →→ B
MVD: B →→ A

XML conceptual schema in DTD Graph



XML schema in DTD

<!ELEMENT A(AB*)>
<!ATTLIST A1 CDATA #REQUIRED>
<!ATTLIST A2 CDATA #REQUIRED>
<!ELEMENT AB EMPTY>
<!ATTLIST AB_iderf IDREF #REQUIRED>
<!ELEMENT B EMPTY>
<!ATTLIST B id ID CDATA #REQUIRED>
<!ATTLIST B1 CDATA #REQUIRED>
<!ATTLIST B2 CDATA #REQUIRED>

XML Docment

<A A1=" a11", A2=" a21">
<AB idref=" 1"></AB>
</A>
<A A1=" a12", A2=" a22">
<AB idref=" 1"></AB>
</A>
<B B1="b11" B2=" b12" id=1"></B>

MVD: A →→ B
MVD: B →→ A

Figure 3b Data semantics preservation in legacy databases (m:n Cardinality)

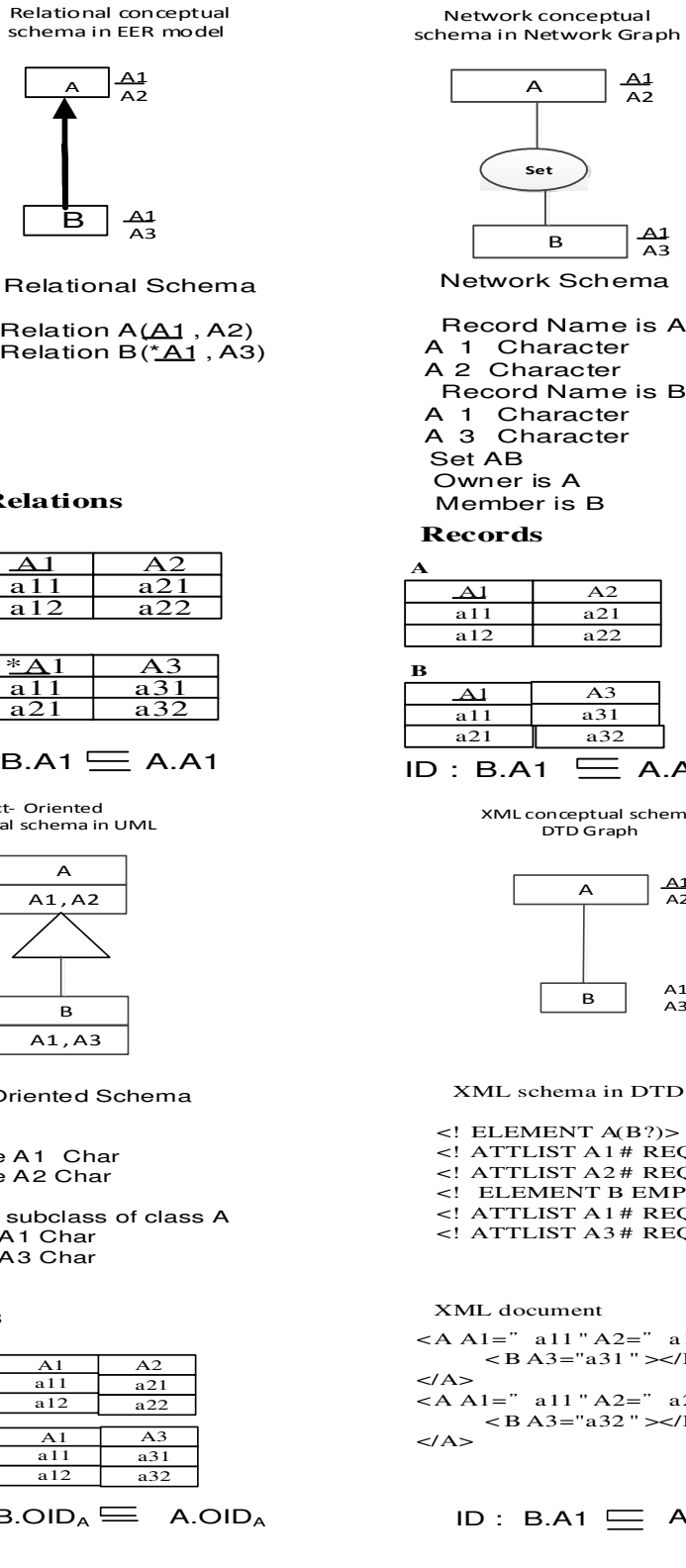
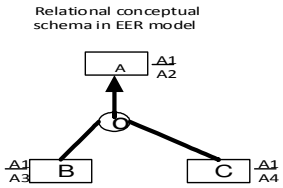


Figure 3c Data semantics preservation in legacy databases (ISA relationship)



Relational Schema

Relation A(A1 , A2)
Relation B(*A1 , A3)
Relation C(*A1 , A4)

Relations

A

A1	A2
a11	a21
a12	a22

B

*A1	A3
a11	a31
a21	a32

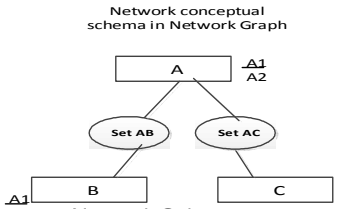
C

*A1	A4
a11	a41
a21	a42

ID : B.A1 ⊆ A.A1

ID : C.A1 ⊆ A.A1

(a) one-to- many cardinality



Network Schema
Record Name is A
A 1 Character
A 2 Character
Record Name is B
A 1 Character
A 3 Character
Record Name is C
A 1 Character
A 4 Character
Set AB
Owner is A
Member is B
Set AC
Owner is A
Member is C

Records

A

A1	A2
a11	a21
a12	a22

B

*A1	A3
a11	a31
a21	a32

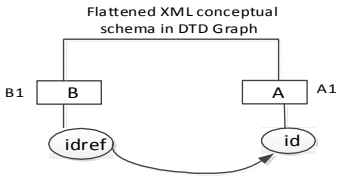
C

A1	A4
a11	a41
a12	a42

ID : B.A1 ⊆ A.A1

ID : C.A1 ⊆ A.A1

(b) many-to- many cardinality



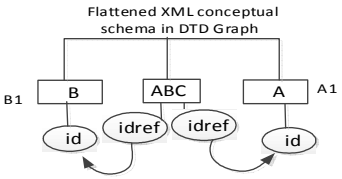
Flattened XML Document schema in DTD

```
<! ELEMENT ROOT (A , B)>
<! ELEMENT A EMPTY >
<! ATTLIST A id ID # REQUIRED >
<! ATTLIST A A 1 CDATA # REQUIRED >
<! ELEMENT B EMPTY >
<! ATTLIST B idref IDREF # REQUIRED >
<! ATTLIST B B 1 CDATA # REQUIRED >
```

Flattened XML Document Data

```
<ROOT>
    <A A1="a11 " id="1"></A>
    <B B1="b11 " idref=1"></B>
    <B B1="b12 " idref=1"></B>
</ROOT>
```

FD : B.iderf → A.id



Flattened XML Document schema in DTD

```
<! ELEMENT ROOT (A , AB , B)>
<! ELEMENT A EMPTY >
<! ATTLIST A id ID # REQUIRED >
<! ATTLIST A A 1 CDATA # REQUIRED >
<! ELEMENT B EMPTY >
<! ATTLIST B id ID # REQUIRED >
<! ATTLIST B B 1 CDATA # REQUIRED >
<! ELEMENT AB EMPTY >
<! ATTLIST AB idref 1 IDREF # REQUIRED >
<! ATTLIST AB idref 2 IDREF # REQUIRED >
<! ATTLIST AB C CDATA # REQUIRED >
```

Flattened XML Document Data

```
<ROOT>
    <A A1="a11 " id="1"></A>
    <B B1="b11 " id="2"></B>
    <A B C="c11 " idref1="1" idref2="2" ></AB>
    <A B C="c12 " idref1="2" idref2="1" ></AB>
</ROOT>
```

MVD : A.id →→ B.id

MVD : B.id →→ A.id

Figure 4a Data semantics preservation in flattened XML documents (1:n & m:n cardinalities)

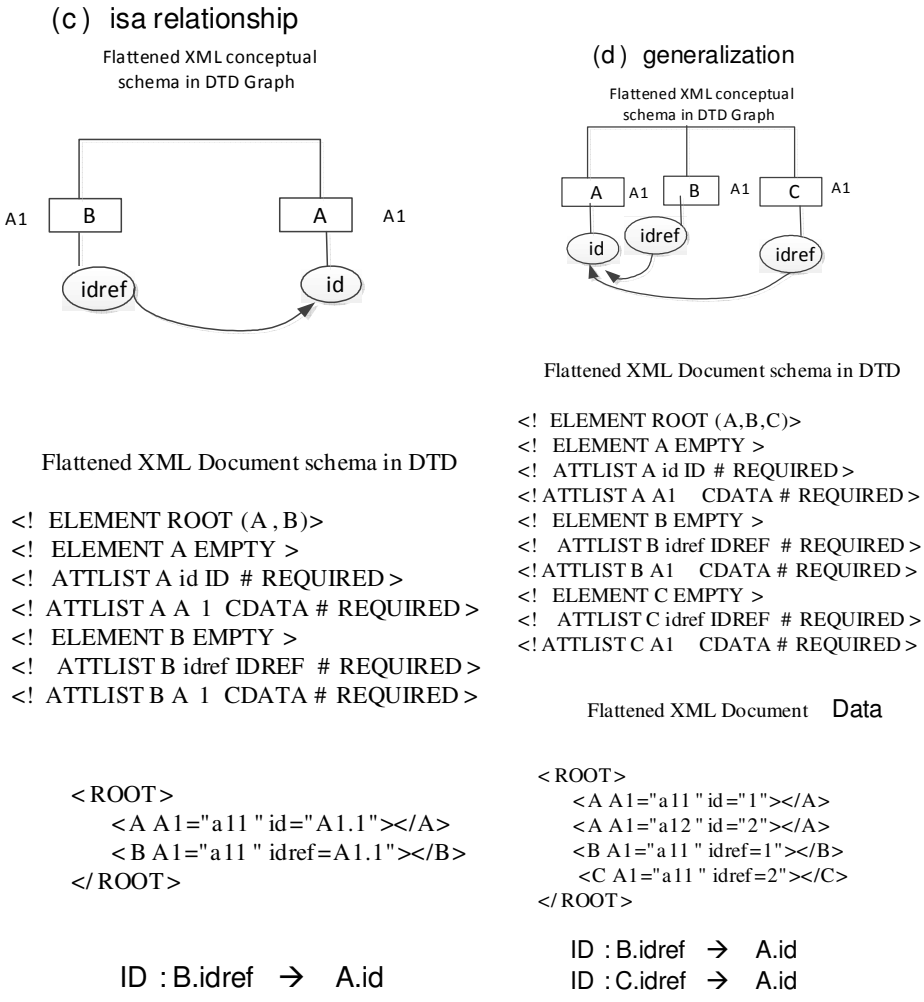


Figure 4b Data semantics preservation in flattened XML documents (ISA, generalization)

2. RELATED WORK

On data transformation

Shoshani, A.[1] defined logical level approach data conversion by using source and target schemas to perform data type conversion instead of physical data type conversion. He provided a general methodology of using logical level approach of downloading source legacy database into sequential file and uploading them into target legacy database for data transformation.

Lum et al [2] showed how to construct data conversion languages SDDL and TDL to extract and restrict data from source legacy database into target legacy database.

The above two paper differ from this paper such that they apply sequential file as medium for legacy databases exchange, but this paper applies flattened XML document as medium for legacy databases exchange.

Fong and Bloor [3] described mapping navigational semantics of the network schema into a relational schema before converting data from network database to relational database.

Fong [4] presented a methodology of transforming object-oriented database objects into Relational database by using SQL Insert statements.
Fong and Shiu [5] designed a Semantic Export Markup Language as a data conversion language to export component of relational database into XML database.
Fong et al.[6] applied logical level approach for data materialization between relational database and object-oriented database using sequential file as medium.

On Heterogeneous database
Given huge investment for a company put into heterogeneous databases, it is difficult for company convert them into homogeneous databases for new applications. Therefore, researchers have come up with a solution of universal databases that can be accessed as homogeneous databases by the user [7] . For instance, we can provide an relational interface to non-relational database such as Hierarchical, Network, Object-Oriented and XML [8] .
Hsiao & Kamel [9] offered a solution of multiple-models-and-languages-to-multiple-models-and–languages mapping to access heterogeneous databases.
Their papers propose a universal database for Hierarchical, Network and Relational databases while this paper proposes a universal database for Network, Relational, Object-Oriented and XML databases.

On Universal database
Fong et al. [10] applied universal database system to access universal data warehousing for the integration of both relational databases and object-oriented databases with star schema and OLAP functions.
Silverston & Graziano [11] used a universal data model in a diagram to design the conceptual schema of different legacy data models of any legacy database.
The above papers differ from this paper such that the above paper proposes a universal data model diagram for universal database conceptual schema while this paper proposes using DTD Graph in Figure 3 and 4 as conceptual schema for universal database.

On Homogeneous database
Sellis, Lin & Raschid [12] presented a solution to decompose and store the condition elements in the antecedents of rules such as those used in production rule-based systems in homogeneous databases environment using relational data model.
This paper differs from the above paper such that it uses flattened XML document environment for universal database.

On schema translation
Funderburk et al. [13] proposed DTD Graph as XML conceptual schema which is identical to DTD, but in a graph format.

On Cloud Database
Derrick Harris [14] defines cloud database as databases in virtual machines.
This paper plans to include cloud computing for further research in future.

On Flattened XML document
Fong et al. [15] converted an XML document into Relational database by transforming XML document into flattened XML document with relational table structure by Extensible Stylesheet Language Transformation.
Compared with the above references, this paper has 3 uniqueness:
(1) Cover more data model
All other database research paper only involve 2 or 3 data models in the universal database. This paper involves 4 data models such as Network, relational, object-oriented and XML.
(2) Use cloud platform

The reference papers do not use cloud platform to implement universal database. This paper performs prototype in cloud platform.
(3) Flattened XML document as middleware
This paper applies flattened XML document as medium to transform legacy databases among each other which is not done by other research papers.

Above all, this paper extends the work of universal database into an "open" universal database gateway such that the universal database is not limited to a particular DBMS, but can be any legacy database of user's choice. Similarly, OUDG is more user friendly than ODBC because it requires less programming effort.

3. METHODOLOGY OF OUDG AS ODBC SUPPLEMENT

This paper proposes OUDG as a database middleware to access legacy databases via flattened XML documents as an universal database as follows:
First Legacy databases → Phase 1: Second Flattened XML documents (universal database)
→ Phase 2:Third Legacy databases
The major functions of OUDG are:

Phase I: Transform first legacy databases into flattened XML documents
Any one of the four first legacy database can be transformed into the flattened XML document as follows:

Case 1: Transform first legacy relational databases into second flattened XML documents
Firstly, we perform the preprocess of mapping relational schema into flattened XML schema. Secondly, we perform their correspondent data conversion. The input is a relational database and the output is an flattened XML document. The system will read relational table according to the legacy relational schema. In one-to-many data semantic, it will post parent and child relations into 2 table structured sibling XML elements linked with id and idref. In many-to-many data semantic, it will post 2 relations and their relationship relation into 3 table structured XML sibling elements linked with idref(s) and id(s). In isa data semantic, it will post superclass and subclass relations into 2 table structured XML sibling elements linked with id and idref with the same key as shown in Figure 3 and Figure 4.

Case 2: Transform first XML databases into second flattened XML documents
Firstly, we perform the preprocess of mapping XML schema into flattened XML schema. Secondly, we perform their correspondent data conversion. The input is an XML database and the output is a flattened XML document with relational table structure. The system will read XML document according to the XML schema. In one-to-many data semantic, it will post element and sub-element into 2 XML sibling elements linked with id and idref. In many-to-many data semantic, it will post 3 elements linked with id(s) and idref(s) into 3 XML sibling elements linked with id(s) and idref(s). In isa data semantic, it will post superclass and subclass elements into 2 XML sibling elements linked with id and idref with the same key as shown in Figure 3 and Figure 4.

Case 3: Transform first legacy Object Oriented database into second flattened XML document
Firstly, we perform the preprocess of mapping object-oriented schema into flattened XML schema. Secondly, we perform their correspondent data conversion. The input is an OODB and the output is a flattened XML document. The system will read OODB according to OODB schema. In one-to-many data semantic, it will post object and set of associated objects into 2 XML sibling elements linked with id and idref. In man-to-many data semantic, it will post 2 sets of associated objects with a common object into 3 XML sibling elements such that a sibling

element with 2 IDREF(s) referring 2 sibling elements with 2 ID(s)). In isa data semantic, it will post superclass and subclass objects with same OID into 2 XML sibling elements linked with id and idref with the same key as shown in Figure 3 and Figure 4.

Case 4: Transform first legacy Network databases into second flattened XML documents
Firstly, we perform the preprocess of mapping network schema into flattened XML schema. Secondly, we perform their correspondent data conversion. The input is a Network database(NDB) and the output is a table structured flattened XML document. The system will read NDB according to NDB schema. In one-to-many data semantic, it will post owner and member records into 2 XML sibling elements linked with id and idref. In many-to-many data semantic, it will post 2 owners and 1 common member records into 3 XML sibling elements linked with id(s) and idref(s). In isa data semantic, it will post an owner and a member records into 2 XML sibling elements linked with id and idref with the same key as shown in Figure 3 and Figure 4.

Phase II: Transform second flattened XML documents into third legacy databases
In step 2, we map the flattened XML schema into another legacy database schema, followed by the data transformation of the flattened XML documents into a legacy database according to the mapped legacy database schema. In this way, each source database data type can be read by the legacy database schema. Therefore, there is no need for physical data type conversion in this approach. Therefore, we can post the flattened XML document into a legacy database of relational, object-oriented, network or XML.

Case 5: Transform second flattened XML documents into third relational databases
Firstly, we perform the preprocess of mapping flattened XML schema into relational schema. Secondly, we perform their correspondent data conversion. The input is a flattened XML document and the output is a relational database. The system will read flattened XML document according to flattened XML document schema. In one-to-many data semantic, it will post 2 XML sibling elements into parent and child relations. In many-to-many data semantic, it will post 3 XML sibling elements linked with id(s) and idref(s) into 2 parents and 1 child relations. In isa data semantic, it will post 2 XML sibling elements into superclass relation and sub-class relation as shown in Figure 3 and Figure 4.

Case 6: Transform second flattened XML documents into third object-oriented databases
Firstly, we perform the preprocess of mapping flattened XML schema into object-oriented schema. Secondly, we perform their correspondent data conversion. The input is a flattened XML document and the output is an object-oriented database. The system will read flattened XML document according to flattened XML document schema. In one-to-many data semantic, it will post 2 XML sibling elements into 2 associated objects with OID and Stored OID. In many-to-many data semantic, it will post 3 XML sibling elements linked with id(s) and idref(s) into 3 associated objects. In isa data semantic, it will post 2 XML sibling elements into 2 superclass and sub-class objects as shown in Figure 3 and Figure 4.

Case 7 Transform second flattened XML documents into third network databases:
Firstly, we perform the preprocess of mapping flattened XML schema into network schema. Secondly, we perform their correspondent data conversion.

The input is a flattened XML document and the output is a network database. The system will read flattened XML document according to flattened XML document schema. In one-to-many data semantic, it will post 2 XML sibling elements into 2 owner and member records. In many-to-many data semantic, it will post 3 XML sibling elements linked with id(s) and idref(s) into 2 owners linked with 1 member record with the same key. In isa data semantic, it will post 2 XML

sibling elements into 2 owner and member record with the same key as shown in Figure 3 and Figure 4.

Case 8: Transform second flattened XML documents into third legacy XML databases
Firstly, we perform the preprocess of mapping flattened XML schema into XML schema. Secondly, we perform their correspondent data conversion.

The input is a flattened XML document and the output is an XML document. The system will read flattened XML documents according to flattened XML documents schema. In one-to-many data semantic, it will post 2 XML sibling elements into 2 XML element and sub-elements. In many-to-many data semantic, it will post 3 XML sibling elements linked with id(s) and idref(s) into 2 pairs of XML elements linked with same sub-element. In isa data semantic, it will post 2 XML sibling elements with the same key into XML element and sub-elements with the same key as shown in Figure 3 and Figure 4.

4. CASE STUDY

A logistic system records the customer shipment information including which orders are being packed and what the packing information is. Based on the XML schema below, there are three intermediate independent entities: PL_INFORMAION recording the general information of the shipment, PL_LINE_INFORMATION storing the packing information — particularly information about the BOXES — and ORDER_INFORMATION storing the information of orders such as the product information. A many-to-many relationship between ORDER_INFORMATION and PL_LINE_DETAIL must be resolved early in the modeling process to eliminate repeating information when representing PL_INFORMATION or ORDER_INFORMATION (MySQL 2013). The strategy for resolving many-to-many relationship[s] is to replace the relationship with two one-to-many cardinality with an association entity and then relate the two original entities to the association entity. As a result, these two one-to-many relationships are between PL_LINE_INFORMATION and PL_LINE_DETAIL, and between ORDER_INFORMATION and PL_LINE_DETAIL. Similarly, the ORDER_INFOR MATION can be divided into BulkOrder and CustomerOrder in generalization as shown in Figure 6.

In Figure 6, there are six relations in relational database. Each table has its primary key and foreign key. Their data dependencies are such that each foreign key determines its referred primary key in functional dependency (FD) in one-to-many cardinality with foreign key on the "many" side, and subclass foreign key is a subset of its referred primary key in inclusion dependency (ID). For relations in many-to-many cardinality, their primary keys are in multi-valued dependencies(MVD) to each other as follows:

FD: PL_Line_Information.PL_Information_Seqno → PL_Information.PL_Information_Seqno
ID: Bulk_Order.BulkOrder.Order_Number ⊆ Order_Information.Order_Number
ID: TailorMadeOrder.Order_Number ⊆ Order_Information.Order_Number
MVD: PL_Line_Information.PL_Information_Seqno →→ Order_Information.Order_Number
MVD: Order_Information.Order_Number →→ PL_Line_Information.PL_Information_Seqno

In Figure 7, the relations are transformed into flattened XML document. The input relational conceptual schema is Extended Entity Relationship model(Chen, 1976). We map input relational schema into an flattened XML OUDG schema with relational structure in two levels tree. Notice that the second level elements (under root elements) are linked together using idref referring to id, which is similar to foreign key referring to primary key. There are seven elements. The second level elements has id(s) and/or idref(s). Their data dependencies are such that each idref

determines its referred id in FD for one-to-many cardinality, and each subclass idref is a subset of its superclass id in ID. For elements in many-to-many cardinality, their id(s) are in MVD as follows:
FD: idref1 → id1
ID: idref3 ⊆ id3
MVD: id2 →→ id3
MVD: id3 →→ id2

In Figure 8, the flattened XML document is transformed into XML document. Elements Pl_information and Pl_line_information are in element and sub-element 1:n association. Elements Pl_line_information and Order_information are in m:n association through element Pl_line_detail linked by pairs of idref referring to id. Elements Order_information and Bulk_Order are in isa association. Elements Order_information and TailorMadeOrder are also in isa association. Their data dependencies are such that each sub-element can determine its element in FD. Each sub-class key is a subset of its superclass key in ID. Two one-to-many cardinality with the same element on the "many" side is equivalent to a many-to-many cardinality of the two "one" side elements in MVD as follows:
FD: PL_Line_Information.PL_Information_Seqno → PL_Information.PL_Information_Seqno
ID: BulkOrder.TechnicalOrderNo ⊆ Order_Information.Order_Seqno
ID: TailorMadeOrder.CustomerOrderNo ⊆ Order_Information.Order_Seqno
MVD: PL_Line_Information.PL_Information_Seqno →→ Order_Information.Order_Seqno
MVD: Order_Information.Order_Seqno →→ PL_Line_Information.PL_Information_Seqno

In Figure 9, the flattened XML document is transformed into Object-Oriented database. It shows the mapping of flattened XML schema into UML as object-oriented conceptual schema. There are six classes. Each class has its OID (object identity), which is similar to primary key in relational schema, and Stored OID, which is similar to foreign key in relational schema. Their data dependencies are such that each Stored OID key determines its referred OID in FD, and each subclass OID is a subset of its superclass OID in ID. The class PL_Informationa nd class PL_Line_Information are in 1:n association in FD. Classes PL_line_Informaton and class Order_Information are in m:n association through class PL_Line_Detail. Subclass BulkOrder and subclass TailorMadeOrder are in generalization under same superclass Order_Information in ID as follows:
FD: PL_Line_Information.Stored_OID → PL_Information.OID
ID: Bulk_Order.OID ⊆ Order_Information.OID
ID: TailorMadeOrder.OID ⊆ Order_Information.OID
MVD: PL_Line_Information.OID →→ Order_Information.OID
MVD: Order_Information.OID →→ PL_Line_Information.OID

In Figure 10, flattened XML document is transformed into Network database. Record Pl_informations and record Order_information are under network DBMS as first records for database navigation access path. The path can go from record Pl_information to Pl_line_information in owner and member record in 1:n relationship in FD. Records Pl_line_information (owner), Order_information(owner) and Pl_line_detail (member) are in flex structure such that records Pl_line_information and Order_information they are in m:n relationship in MVD. Records Order_information and BulkOrder are in isa relationship since they have same key value in ID. Similarly, records Order_information and TailorMadeOrder are in isa relationship due to same key value in ID. The set records are pointers only. Their data dependencies are as follows:
FD: PL_Line_Information.PL_Information_Seqno → PL_Information.PL_Information_Seqno
ID: Bulk_Order.TechnicalOrderNo ⊆ Order_Information.OrderSeqno
ID: TailorMadeOrder.CustomerOrderNo ⊆ Order_Information.OrderSeqno
MVD: PL_Line_Information.PL_Information_Seqno →→ Order_Information.OrderSeqno

MVD: Order_Information.OrderSeqno →→ PL_Line_Information.PL_Information_Seqno

Papers in this format must not exceed twenty (20) pages in length. Papers should be submitted to the secretary AIRCC. Papers for initial consideration may be submitted in either .doc or .pdf format. Final, camera-ready versions should take into account referees' suggested amendments.

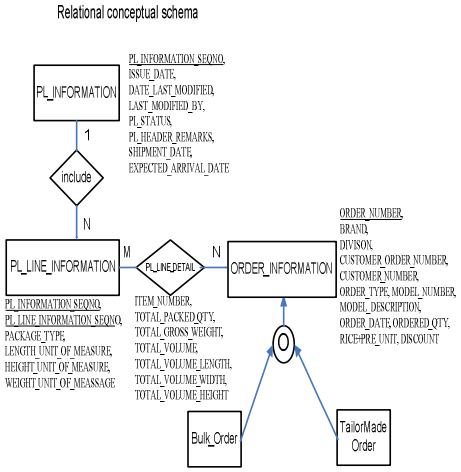


Fig. 6 First source legacy Relational database MySQL with EER model

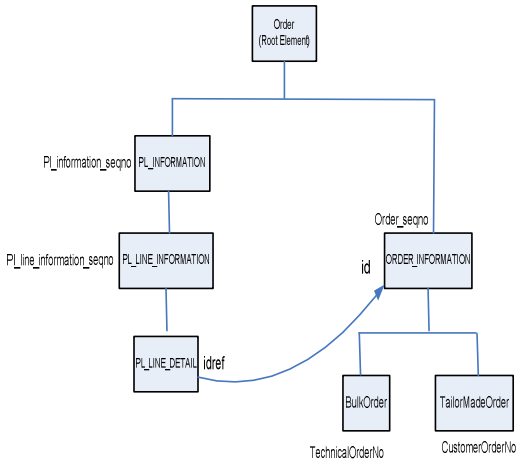


Fig. 8. Transformed second legacy XML document from flattened XML document

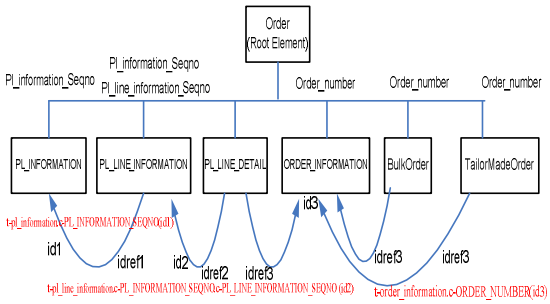


Fig. 7. Transformed flattened XML document and map from first legacy relational database

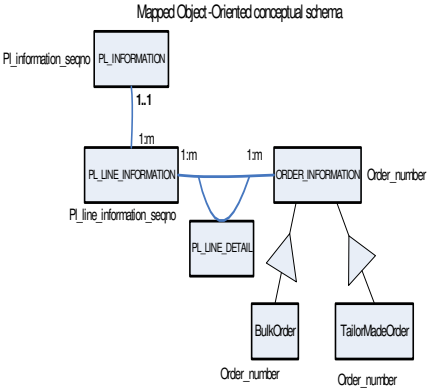


Fig. 9. Transformed second legacy Object Oriented database from flattened XML document

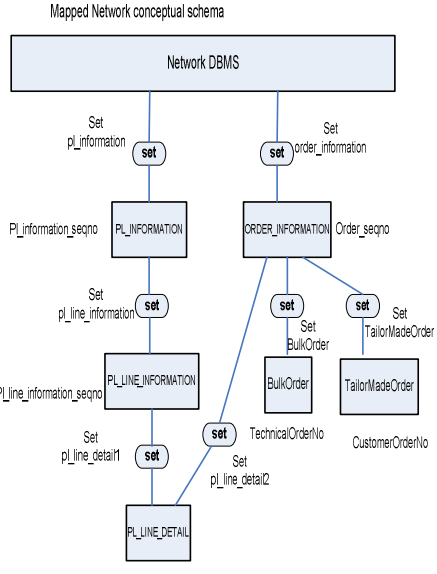


Fig. 10. Translated second Network database from flattened XML document

5. CONCLUSION

Since relational database is the most user friendly legacy database, and XML database is the most portable database for information highway on the Internet. In this project, we propose a Flattened XML database as universal database such that it is most user friendly and portable as a database middleware for all legacy databases.

The uniqueness of this paper are:

(1) Openness of an universal database: The reason we choose flattened XML document is due to its openness, and DBMS independence. All other data models are DBMS dependent.
Nevertheless, users can use OUDG to access any legacy database via flattened XML documents on the Internet by Internet Explorer without programming.

(2) Recovery of legacy database: Since flattened XML document is a replicate legacy database, it can be used to recover any legacy database whenever the production legacy database is down. As a result, replicate XML document can be parallel processing with legacy database in non-stop computing.

(3) Heterogeneous database integration for data warehousing: By transforming all in-house legacy databases into one legacy database as the data cube, companies can use OUDG to integrated their legacy databases into a data warehousing for decision support system.

(4) Portability of Flattened XML document as Universal database: The OUDG solution is not limited to using a particular DBMS, but allows users of any legacy database access other legacy database.

In summary, the proposed OUDG unites all legacy databases data models into flattened XML schema. The portability of the proposed flattened XML document can be transferred into any open platform. The data conversion methodology of this OUDG is to download the raw data of source database into flattened XML document using source database schema, and upload the flattened XML document into target database using translated target database schema, which is a logical level approach, and which can avoid physical data type conversion. Therefore, the methodology can transform any legacy database into any other legacy database. The reason of using flattened XML document as medium is to reduce the number of programs for the data conversion; otherwise, we need 4 * 4 = 16 programs, instead of the current 4 + 4 = 8 programs to do the data conversion for the four legacy database models: relational, network, object-oriented and XML. Above all, all legacy databases can be transformed into each other via flattened XML documents for data access in the same way as computers connect to each other via computer network for information retrieval.

REFERENCES

[1] Shoshani, A., (1975) "A Logical-Level Approach to Data Base Conversion", ACM SGMOD International Conference on Management of Data, pp.112-122.

[2] Lum, V.Y., Shu N.C. & Housel B.C. (1976) "A General Methodology for Data Conversion and Restructuring", IBM Journal of research and development, Volume 20, Issue 5, pp.483-497.

[3] Fong, J.& Bloor C. (1994) "Data Conversion Rules from Network to Relational Databases", Information and Software Technology, Volume. 36 No. 3, pp. 141-154.

[4] Fong, J. (1997) "Converting Relational to Object-Oriented Databases", SIGMOD RECORD, Volume 26, Number 1, pp53-58.

[5] Fong, J. & Shiu, H. (2012) "An interpreter approach for exporting relational data into XML documents with Structured Export Markup Language", Journal of Database Management, volume 23, issue 1.

[6] Fong, J., Pang, R., Fong, A., Pang, F. & Poon, K. (2003) "Concurrent data materialization for object-relational database with semantic metadata", International Journal of Software Engineering and Knowledge Engineering, Volume 13, Number 3, pp.257-291.

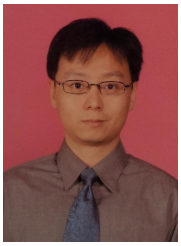
[7] Fong, J. & Huang, S. (1999) "Architecture of a Universal Database: A Frame Model Approach", International Journal of Cooperative Information Systems, Volume 8, Number. 1, pp. 47-82.

[8] Fong, J. (1996) "Adding Relational Interface to Non-relational Database",IEEE Software, pp. 89-97.

[9] Hsiao, D. & Kamel, M. (1989) "Heterogeneous Databases: Proliferations, Issues, and Solutions", IEEE Transactions on Knowledge and Data Engineering, Voumn 1, No. Pp.45-62.
[10] Fong, J., Li, Q. & Huang, S. (2003) "Universal Data Warehousing Based on a Meta-Data Modeling Approach", International Journal of Cooperative Information Systems, Volume 12, Number 3, pp.325-363.
[11] Silverston, L. & Graziano, K.(2013) www.360doc.com/content/08/0830/01/1032_1590731.shtml
[12] Sellis, T., Lin, C. & Raschid, L. (1993) "Coupling Production Systems and Database Systems: A Homogeneous Approach", IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 5, NO.
[13] Funderburk J. (2002) "XTABLES: Bridging Relational Technology and XML", IBM Systems Journal, Vol 41, No. 4, PP 616 –641.
[14] Harris, D. (2012) "cloud-databases-101-who-builds-em-and-what-they-do", GIGAOM, http://gigaom.com/cloud/cloud-databases-101-who-builds-em-and-what-they-do/
[15] Fong, J., Shiu, H. & Wong, J. (2009) "Methodology for data conversion from XML documents to relations using Extensible Stylesheet Language Transformation", International Journal of Software Engineering and Knowledge Engineering,Volume 19, Number 2, pp. 249-281

AUTHORS

Dr Joseph Fong is an Associate Professor at City University of Hong Kong. He is a fellow of Hong Kong Computer Society, the founder chairman of the Hong Kong Computer Society Database Special Interest Group, and the honorable founder chairman of Hong Kong Web Society and International Hybrid Learning Society. Fong had worked in the industry in US for 11 years, and in Hong Kong as an academician since 1988. His research interests are in database, data warehousing, data mining, XML and eLearning. His above 100 publications include SCI Journals, Conferences, Patent (US), books, and an authored text book on "Information Systems Reengineering, Integration and normalization" 3rd edition by Springer in 2015. He had been program manager of M.Sc. of Staffordshire University for a decade and teaches Data warehousing and data mining, Data Engineering, and Database Systems. Dr. Fong is a former editorial board member of International Journal of Web Information Systems.



Mr Wong Ting Yan, Kenneth. has been graduated from the first degree, Computer Engineering in Electronic Engineering Department in City University of Hong Kong at 2002, and Master of Science in Information Engineering in Chinese University of Hong Kong at 2008, and ***Degree of Master of Philosophy in Computer Science*** in City University of Hong Kong in 2014. He has worked in several educational institutes for almost 10 years, included primary school, secondary schools, and has experienced to work as teaching assistant and research associate in Open university of Hong Kong and City University of Hong Kong respectively.