Data Migration Model and Algorithm between Heterogeneous Databases based on Web Service

Guowei Wang and Zongpu Jia
School of Computer Science and Technology, Henan Polytechnic University, Jiaozuo, China
Email: wangguowei@hpu.edu.cn, jiazp@hpu.edu.cn

Manjun Xue
School of Architectural and Artistic Design, Henan Polytechnic University, Jiaozuo, China
Email: xuemanjun@hpu.edu.cn

Abstract—To solve the problems of data conversion between heterogeneous databases that arising from the progress of data migration, the paper presents an improved model of data migration and a data conversion algorithm based on the research on data synchronization via web service. The data migration model is divided into legacy data tier, middle tier and target data tier, which respectively provides data extracting, data converting and data loading functions. By establishing the corresponding relationships of different data fields between heterogeneous legacy database and target database, the data conversion algorithm converts data formats of legacy database into required data formats of target database according to the relationships, then writes the converted XML document to target database to achieve data migration. Experiments show that the model and algorithm can effectively realize data migration.

Index Terms—Data Migration; Heterogeneous Databases; Data Conversion; XML; Web Service; Data Synchronization

I. INTRODUCTION

With the constantly growing of business scale and rapidly developing of computer technology, information application systems in enterprise need to be modified frequently to more easily adapt to new business requirements. Due to the difference of operating system, developing platform and programming language and other factors between legacy system and target system, there emerging a key problem that data from legacy system is mapped to the target system in heterogeneous environment where legacy and new systems have different data structures [1]. Generally, we call this process of data moving the data migration.

The key point of data migration is data conversion. In a computer environment, data is encoded in a variety of ways. For example, the operating system is predicated on certain standards for data and file handling, each computer program handles data in a different manner. Whenever any one of these variable is changed, data must be converted in some way before it can be used by a different operating system, database management system (DBMS) or program. Even different versions of these elements usually involve different data structures. For example, the changing of bits from one format to another, usually for the purpose of application interoperability or of capability of using new features.

Data migration is similar to data integration and data synchronization. Data integration involves combining data residing in different sources and providing users with a unified view of these data. Data synchronization is the process of establishing consistency among data from a source to a target data storage and vice versa and the continuous harmonization of the data over time. The common features of them is that they all need to transport data from one database to another. Despite their similarities, data migration, integration and synchronization differ in two areas:

First, number of data sources and targets. The flow of data from its sources to its targets can be described as many-to-one (integration), one-to-one (migration), and many-to-many (synchronization). This enables us to differentiate these similar data-movement techniques.

Second, diversity of legacy and target data models. Much of the time spent in data-movement development concerns mappings and transforms between source and target data models. The more divergent they are (or the greater the number of data models involved), the more time development will take. So it requires a thorough understanding of data sources from various aspects, such as data type, data format, identifier naming, database objects and application when we achieve an effective data migration procedure [2].

Data of the legacy system is mapped to the target system providing a design for data extraction and data loading. The design relates legacy data formats to the target system’s formats and requirements. Data migration is usually performed programmatically to achieve an automated migration. Programmatic data migration may involve many phases but it minimally includes data...
extraction where data is read from the legacy system and data loading where data is written to the target system.

Data migration is a messy, time-consuming and difficult undertaking. The structuring of data elements translation makes the data migration process very difficult to automate because the data migration project address the areas of reverse engineering, business reengineering, schema mapping and translation, data transformation, application development, human computer-interaction and other complicated works. A survey of services firms found that 72% of organizations deferred moving applications because data migration is too risky.

Thus there is an urgent need to provide data migration model and migration algorithm not only for accessing the data which is stored in legacy systems, but also to provide strategy which formats the migrated data to adapt to the platforms and architectures of target system. Although there are many commercial tools claim to realize data migration automatically, most are specific to particular DBMS and require a particular data structure.

A. Our Contributions

In this paper we investigate data migration based on the research on data integration and data synchronization between heterogeneous databases via service oriented architecture (SOA) and XML. In particular, we developed a general data migration model that can be used to achieve an effective data migration procedure. Secondly, we presented a corresponding relationship library that can be applied in popular DBMS. Finally, we designed a data conversion algorithm based on the migration model and relationships library to achieve data migration between heterogeneous databases through the methods of XML document processing and web service calling.

B. Outline

The paper is divided into 6 sections. The next section discusses the background and related work involved in data migration and data synchronization projects. Section III details the architecture and function of the data migration model. Section IV outlines the workflow of the migration model. Section V illustrates the proposed data converting algorithm. The experimental results and analysis are presented in Section VI. Finally, the concluding section presents a summary of the paper.

II. BACKGROUND AND RELATED WORK

Existing proposals for data migration between heterogeneous databases can be classified into three different classes: manual data input, using Extract Transform Load tools (ETL), and programming methods based on database technology. We briefly outline the programming method based on database technology, which is the most relevant efforts in the three classes.

Most programming methods based on database technology using middleware to achieve data migration.

David I. Spivak in reference [3] provides new data migration functors based on a straightforward category-theoretic model of database to translate instances from one schema to the other in canonical ways. These functors parameterize projections, unions, and joins over all tables simultaneously and can be used in place of conjunctive and disjunctive queries.

Li Xiaohang et al. in reference [4] propose a method of SQL language reproduction to achieve database duplication. By building a tracks table in source database, the method translates the operation of data table into corresponding SQL language sequences, then executes these SQL sequence in target database according to time order to generate required data. The method can copies data from legacy database to target database but their approach is constrained to particular databases, if one of the fields in a data table changes in data type, length or naming, the business may grind to a halt.

By using ETL tool, reference [5] proposes a generalized data migration tool based on metadata-driven technology, but ETL is complex design structure that runs under strict performance requirements and its optimization is crucial for satisfying business objectives because of the operations and the typical structural complexity [6], meanwhile, due to the complexity of components extensibility, ETL components are tightly-coupled to each others in the current ETL framework [7]. ETL is mostly adapt to those information systems whose version is upgraded from an old one, so the proposal does not working well when differences occurred in business processes or database.

XML-based web service and SOA technologies have been widely applied in digital campus of institutes and informatization of enterprises because of the platform-independent characteristics [8]. Reference [9-12] respectively researched on web service and SOA for data exchanging, integrating, synchronizing and sharing to solve the problems about information isolated island that arising from the procedure of information transmitting between distributed heterogeneous systems. In order to realize data synchronization and avoid data inconsistency caused by one application system, the key points of these works are focused on the construction of data interconnection channel and platform, declaration of rules and other aspects, in the procedure of data synchronization, only relevant data in business legacy systems is updated according to the changes of the basic data that transferred in the channel.

In contrast to numerous industry and information interests about SOA and web service, a few research attention has been given to the theoretical foundations of data migration. Our works mainly aim to the data formatting and whole data moving from one database to another by improving the methods of XML-based web service and SOA technologies and data synchronization that studied in heterogeneous information systems.

III. ARCHITECTURE OF THE DATA MIGRATION MODEL

As shown in figure 1, the data migration model consists of legacy data tier, middle tier and target data tier, which respectively provides data extracting, data converting and data loading functions.
### A. Legacy Data Tier
 Legacy data tier includes legacy application system and its database that need to transform its data to target heterogeneous databases. Legacy data tier is the data source of the migration model.

### B. Middle Tier
 Middle tier contains data extracting module, data converting module, exception processing module and data loading module.
Data extracting module mainly realizes original data acquisition. By designing a query designer, data extracting module establishes database connections of legacy database and target database by calling the database connection web service that registered in universal description discovery and integration (UDDI) center, then gets the data table structures from legacy database and target database, finally establishes the relationships between specific fields that selected from data tables. After executing the query, the retrieved data will be written to a XML document to facilitate data conversion based on the converting algorithm.

Data converting module includes the corresponding relationship library of heterogeneous data type and the implementation of data converting algorithm, which were encapsulated into web service for service calling.

Exception processing module responses the errors or inconvertible data or other information to database administrator (DBA) for another process according to migration strategies.

Data loading module processes the response XML of data converting web service, then writes the results to target database via data writer.

C. Target Data Tier

Target data tier is composed of the target database and its application system programs, migrated data is eventually written to target database.

D. Web Service

In our works, we establish database connection web service interface of popular DBMS to enhance the flexibility and expansibility of the model. Administrators select or enter the type, name, account, password of database, as well as parameters such as IP address of database server to generate database connection string automatically.

Besides, we encapsulate data converting algorithm into web services and register them in UDDI centre. Service requesters call the web service and perform the corresponding operation, the results were returned by way of simple object Access Protocol (SOAP) after the completion [13].

IV. Workflow of the Migration Model

As shown in figure 2, the workflow of the migration model consists of data extracting procedure, data converting procedure and data loading procedure.

A. Procedure of Data Extracting

- DBA visits the query designer of data extracting module then selects legacy database and target database that need to migrate data.
- Query designer visits UDDI center then retrievals and calls database connection web service of legacy database and target database to generate database connection string for database connection.
- After connected to database, data tables of legacy database and target database will be displayed in the query designer. DBA chooses the needed data tables and data fields.
- Query designer retrieves the desired data from legacy database and target database. Retrieved data including structured information of data table such as name, attribute information of fields such as name, type, length and constraint information such as names and values of primary key or foreign key, which were used for data conversion process when middle tier creates fields mapping rules and relationships.
- Data extracting module preprocesses the data that retrieved from database, then translates the data into XML document for convenient data processing of web services in data converting module.

B. Procedure of Data Converting

- Data converting module calls the data conversion web service that registered in UDDI center, then translates the response XML document into compatible data for target database formats according to the corresponding relationship library and data converting algorithm.
- During the process of data conversion, whether the operation brings a successful result or not, data converting module will sends a response message to notify DBA, if the data conversion is not well done, it will records the error data in database.
- DBA issues instructions for data migration based on the response message. If the received response is message about error conversion, DBA will issues exception processing instructions according to data migration strategies, otherwise issues normal instructions.

C. Procedure of Data Loading

- After received data migration instructions, data loading module processes the response XML document that returned by data converting web service.
- Data loading module writes the processed data to target database via data writer calling.

V. Data Converting Algorithm

There are several factors that may considerably complicate the process of data migration in heterogeneous environment, while problems of data retrieving from legacy database, data writing to target database, web service encapsulating and calling, exchanging between XML and relational database have been widely studied by many researchers, in this section, we mainly realize the definition and implementation of corresponding relationship library and data converting algorithm.

A. Definition of Corresponding Relationship Library

Heterogeneous databases use different data types to store same data, besides, representations and semantics of the same data type are all different from each other,
which mainly reflected in the name of data type and length of stored data. For example, integer data in Oracle database is represented by NUMBER, but in Informix database is INTEGER, moreover, the length of the CHAR data type in Oracle database is 2000, while in Informix database is 255.

Essentially, corresponding relationship library can be considered as the collections of correspondence between heterogeneous database data type. Table 1 shows the commonly used data types in Oracle database and SQLServer2000 database.

Table 1. DATA TYPES IN ORACLE AND SQLSERVER2000

<table>
<thead>
<tr>
<th>SQL Server 2000</th>
<th>Oracle</th>
</tr>
</thead>
<tbody>
<tr>
<td>bigint, bit, decimal, int, money, numeric, smallint, smallmoney, tinyint</td>
<td>NUMBER</td>
</tr>
<tr>
<td>char, nchar, nvarchar, varchar</td>
<td>VARCHAR2</td>
</tr>
<tr>
<td>date, datetime, smalldatetime</td>
<td>DATE</td>
</tr>
<tr>
<td>float, real</td>
<td>FLOAT</td>
</tr>
<tr>
<td>image, sql_variant, text, ntext</td>
<td>LONG RAW</td>
</tr>
<tr>
<td>sysname</td>
<td>CHAR</td>
</tr>
</tbody>
</table>

Definition 1. Corresponding relationship library
Suppose that function \( f = (c, v) \) converts the data type of fields \( c \) from legacy database to desired data type \( v \) of target database.

Suppose the corresponding relationship library \( F \) is the collection of \( f \).

Suppose that legacy database \( D \) is the collection of data table \( T \) and constrains, \( C \) is a set of fields of data table.

Suppose data contents of data table is a set of \( n \) dimensional vector \( \sigma \), \( \sigma = (a_1, a_2, \ldots, a_n) \), each component of \( \sigma \) is \( a_i = \sigma[c_i] \), \( a_i \in D(c_i) \).

Suppose data acquisition function is \( DATA_{ACQ} : (t, F) \rightarrow \{ \sigma \} \), \( t \) is the name of data table from \( T \).

B. Implementation of Data Converting Algorithm

1) Basic knowledge
Definition 2. Relational schema
A relational schema is denoted by a 5-tuple \( R = (T, C, F, PK, FK) \) [14], where

- \( T \) is a finite set of table names.
- \( C \) is a function from a table name to a set of column names.
- \( F \) is a function from a column name \( c \) to its column type definition. \( F(c) = \alpha \cdot \alpha = (\sigma, \mu, \lambda, \pi, n) \), where \( \sigma \) is metadata data type, \( \mu \) means that whether the value of \( c \) is unique or not, \( \lambda \) means that whether the value of \( c \) is null or not, \( \pi \) is the default value of \( c \), if \( c \) does not has a default value, \( \pi \) is \( 
\end{equation}

- \( PK \) is a finite set of relational integrity primary key constrains that can be either retrieved from databases directly or provided by human experts.
- \( FK \) is a finite set of relational integrity foreign key constrains that can be either retrieved from databases directly or provided by human experts.

Definition 3. XML schema
An XML schema is denoted by a 5-tuple \( S = (E, A, ST, CT, r) \) [15], the meaning of each entry as follows:

- \( E \) is a finite set of element names.
- \( A \) is a function from an element name to a set of attribute names.
- \( ST \) is a function from an element name to its element type definition.
- \( CT \) is a function from an attribute name to its attribute type definition.
- \( r \) is a finite set of root elements.

2) Data Converting Algorithm
Suppose that the relational schema of source database \( D \) is \( R = (T, C, F, PK, FK) \), XML schema is \( S = (E, A, ST, CT, r) \) and data converting rule is \( M = (R, S, M_a, M_e, M_f, M_p) \). Inputs of data converting algorithm are \( R, S, M \), output of data converting algorithm is XML \( e \).

The data converting algorithm as follows:
1. Initialize XML \( e = m.e.n \).
2. Assign \( t_i = m.e.t \).
3. For each \( a_i \in t_i.A_i \):
   a. If there exist a mapping rule \( m_a.a = a \) in \( M_a \) then initialize \( p \), and respectively assign \( a_n \) and \( \pi[m_a.c] \) to its name and value, finally add \( p \) into \( e \).
   b. If there is not exist a mapping rule \( m_a.a = a \) in \( M_a \) then do nothing but continues next 3.
4. For each \( e' \) in \( t_i.E_i \):
   a. If there exist a mapping rule \( m_e.e = e' \) in \( M_e \) then initialize simple content \( se \) and assign \( se = \pi[m_e.c] \), finally add \( se \) into \( e \).
   b. If there is not exist a mapping rule \( m_e.e = e' \) in \( M_e \), then do nothing but continues next 4.
5. If \( e' \in CT \) then
   a. If there exist a mapping rule \( m_{e'} = e' \) in \( M_e \) then initialize \( F = \phi \). For each source constrain \( sc = (m_e', \{c_i \in D_i\}) \) of \( m' \) in \( M_{e'} \), add \( f_e = (c_i, D_i) \) into \( F \). For each relational constraint \( rc = (m_e, m_{e'}, \{c_i \in D_i\}) \) from \( m_i \) to \( m_{e'} \) in \( M_{e} \), assign \( f_i = (c_i, \{ \pi[c_i] \}) \) and add it into \( F \).
   b. If there is not exist mapping rules \( m_{e'} = e' \) in \( M_{e} \) then do nothing but continues next 4.
5. Calculate set $\Sigma$ from $m', t$ that satisfied given condition of $F$ by calling `DATA_ACQ(m', t, F)`.  
6. For each $\tau'$ in $\Sigma$, recursively call `DATA_CON` and add the calculating result $e'' = \text{DATA_CON}(m', \tau'.C_{\phi}(m', t))$ into XML $e'$.  
7. Return XML $e'$.  

VI. EXPERIMENTAL RESULTS AND ANALYSIS  

A. Experiment  
Main fields of table that named `tuition` in legacy database as follow:  
```
tuition(stu_no,stu_name,stu_school,stu_dept,term,tui_sd,...)  
```

Our work is to migrate the data of tuition table from legacy database to two tables of target database that respectively named `stu` and `stu_tuition`:  
```
stu(stu_id, stu_name, stu_school, stu_class,...)  
stu_tuition(stu_id, term_name, tuition_amount,...).  
```

The relational schema source table is $R = (T, C, F, PK, FK)$, the meaning of each entry as follows:  
- $T = \{\text{tuition}\}$  
- $C(\text{tuition}) = \{\text{stu_no, stu_name, dept_name, term, tui_sd}\}$  
- $F(\text{stu_no}) = \{\text{string, unique, not_nullable, } e, e\}$  
- $F(\text{tui_sd}) = \{\text{number, not_unique, not_nullable, } 0\}$  
- $PK(\text{tuition} \rightarrow \{\text{stu_no}\})$  
- $FK = \emptyset$

The XML schema as follows:  
```
E = { 
  stu_school("stu_school",TSCHOOL,e),
  stu_dept("stu_dept",TDEPT,e),
  stu(stu",TSTU,e)
}
A = \{TSCHOOL,"stu_school",string,"required"),
    TDEPT.stu_class("stu_class",string,"required"),
    TDEPT.stu_name("stu_name",string,"required"),
    TDEPT.stu_id("stu_id",string,"required"),
    TSTU.stu_id("stu_id",string,"required"),
    TSTU.term_name("term_name",string,"required"),
    TSTU.tuition_amount("tuition_amount",decimal,"required")\}  
```

According to the data converting algorithm, returned XML contains department information and tuition information of students, which can be easily write to target database.  

B. Results and Analysis  
In this section we analyze the results and show some graphs that summarize the behavior of the algorithm by comparing our works with method of data synchronization and Data Transformation Services (DTS) of SQL Server2000 and a similar method that presented in reference [16]. We evaluate our method in two aspects: the execution time and the memory utilization.  

1) Comparison with Data Synchronization  
Principle of most data synchronization methods between heterogeneous database uses triggers [17] or agent [18] or other way to monitor the data changes of local database, then adds, updates or deletes corresponding data in target database via web service or XML, the data operation is occurred at random and only a few data is modified. While the method of our works formats the fields of data table in legacy systems in a certain periods of time according to the data format of target database through corresponding relationship library and data converting algorithm, then moves the data to target database, it needs to migrate a large quantities of data. So it is an improved method based on data synchronization.  

2) Comparison with Similar Data Migration Methods  
Method in reference [16] translates the data type of fields in table of legacy database into target database and generates SQL string to create source database based on corresponding relationship, which is similar to our works. Besides, both methods are exactly same in programming language, DBMS and database servers.

We perform experiments on machines equipped with 2.13 GHz Intel Core 2 processor and 1.99 GB RAM and windows server 2003 enterprise edition OS, DBMS in server of legacy system is SQL server2000 while in target server is oracle9i.

Since reference [16] only gives the performances on migrating 10000 and 1000000 rows of data, we compare our works with the method of reference [16] on execution time and memory utilization only in these two cases. As shown in table II, reference [16] takes a little more time consuming and a little less memory utilizing than our works to process 10000 rows of data. However, in the procedure of 1000000 rows of data migration, reference [16] takes much more time consuming and memory utilizing than our works. This is due to the following two facts. First, reference [16] needs to connect both database server of legacy system and target system for different fields relating when the data migration occurs, if the quantity of data is too large, it needs much more time to retrieve required data type. Second, reference [16] only uses the calculating ability of database servers to process migrated data. While in our works, we construct corresponding relationship library and deploy it to a server in middle tier before the procedure of data migration, and also, we use this machine to run the data converting algorithm. It needs to connect the server just one time, then reads the whole corresponding relationships into memory, so the memory utilization of database server and execution time are only a bit variations when migrates a few rows of data, but with the increasing of data quantity, the percentage of memory utilization for corresponding relationship library has declined to a low level, comparatively, the execution time superiority of our works appeared.

DTS is a set of objects and utilities used to automatically perform extract, transform and load operations to or from a database. The objects are DTS packages and their components, and the utilities are
called DTS tools. DTS is widely used with Microsoft SQL Server databases, meanwhile it could be used independently with other databases. DTS allows data to be transformed and loaded from heterogeneous sources using OLE DB or ODBC into any supported database.

In figure 3 and figure 4, we compare the performance of execution time and memory utilization between DTS and our works under same environment by migrating 10000 to 1000000 rows of data from database server of legacy system to target system.

As shown in the figures, with the increasing of migrated data, execution time of DTS becomes much more than that of our works, it approximately takes more 1200s to process 1000000 data records, meanwhile, memory utilization of our works reveals a general trend of steady rise but fluctuates around 40M in DTS, it roughly uses 160M memory in our works while still keeps 40M in DTS when data quantity is 1000000.

<table>
<thead>
<tr>
<th>Rows of data</th>
<th>10000</th>
<th>1000000</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Execution time of our work</strong></td>
<td>22s</td>
<td>24m</td>
</tr>
<tr>
<td><strong>Execution time of reference [16]</strong></td>
<td>21s</td>
<td>28m</td>
</tr>
<tr>
<td><strong>Memory utilization of our work</strong></td>
<td>38M</td>
<td>154M</td>
</tr>
<tr>
<td><strong>Memory utilization of reference [16]</strong></td>
<td>37M</td>
<td>192M</td>
</tr>
</tbody>
</table>

In figure 3 and figure 4, we compare the performance of execution time and memory utilization between DTS and our works under same environment by migrating 10000 to 1000000 rows of data from database server of legacy system to target system.

As shown in the figures, with the increasing of migrated data, execution time of DTS becomes much more than that of our works, it approximately takes more 1200s to process 1000000 data records, meanwhile, memory utilization of our works reveals a general trend of steady rise but fluctuates around 40M in DTS, it roughly uses 160M memory in our works while still keeps 40M in DTS when data quantity is 1000000.

DTS keeps the database in a connected state and loads the data that to be processed into memory in batches, for example, DTS loads 1000 rows of data into memory in a batch. It needs to retrieve required data repeatedly according to the total quantity of data and parts quantity of each batch, which causes DTS spends more time to process data and keeps a stable memory utilization.

Method of our works fills all the needed data into datasets by programming, then loads the datasets into memory at one time, finally updates database to realize data migration, so it takes a few execution time but needs more memory to store the migrated data along with the growing of data quantity.

But using DTS, the data tables need to be preprocessed, which is a time-consuming and complex work that requires professionals who have thorough understanding of database principle, the method presented in this paper can automatically migrate data through correspondence and programming, although the memory utilization is high, data migration is processed before the running of target system, so it has no impact on the performance of target system.

VII. CONCLUSIONS AND FUTURE WORK

Principle of data migration methods between heterogeneous database can be considered as one way data synchronizing from legacy systems to target systems. We presented data migration model and data converting algorithm based on data synchronization and explained in detail the architecture and realization for data migrating between heterogeneous relational databases. A comprehensive analysis of their performance has shown desired results and allowed us to focus on further developments in specific aspects.

The data migration model and data converting algorithm are able to migrate 1000000 rows of data from SQLServer2000 database to Oracle database by using 160M memory in 1500s. The execution time and memory utilization are mainly affected by the quantity of data to be migrated, the data converting algorithm allows us to cope with data type differences in a fairly efficient way without any impact on the performance of target system. However, in practical applications, rule based data migration method is difficult to solve all the problems. The method of our works is insufficient to deal with the matters when the content representation of fields in data table of legacy system is different from that of target system, in certain circumstances it needs manual operations or SQL statements to realize the whole data migration.

Given these results we identified a few points which will be part of future work. The most important one is the integration of data type conversion and data content conversion, which is the key points to realize automatic data migration. Finally, for optimizing memory utilization, improvements of the data converting algorithm may be required.

REFERENCES


Guowei Wang is a Ph.D candidate in Henan Polytechnic University. The major research fields include system architecture of computer, computer security and database applications.

Zongpu Jia received his Ph.D in system architecture from Jilin University, China, in 2008. He is a professor in school of computer science and technology, Henan Polytechnic University. His research fields include computer network technology, computer control technology, and information systems.

Manjun Xue earned her master degree from Henan Normal University, China, in 2012. Her research fields mainly include computer-aided design and website optimization.