SQL Query Based Data Synchronization in Heterogeneous Database Environment

Md. Iqbal Hossain and Muhammad Masroor Ali
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
Dhaka-1000, Bangladesh
Email: {mdiqbalhossain, mmasroorali}@cse.buet.ac.bd

***Abstract*—Database synchronization is a complex process in heterogeneous database environment. Maintaining the uniformity of data and its structure is a fundamental problem of data synchronization. In this paper we present a methodology to synchronize data and structure in heterogeneous database envrionment. We capture all DDL and DML SQL query from source database and pass this SQL query to target database. A process runs continuously in the back end to read SQL query and passes the SQL query to target system through HTTP. Another process processes the SQL query in the target system. Some errors may happen in target database for data type mismatch or SQL query function mismatch or unavailability. To avoid these errors, we have used two lookup tables and adapt the SQL query. Then we execute the changed query in the target database. Our experimental results evidence the effectiveness of the proposed method.**

***Index Terms*—DDL and DML, heterogeneous database, SQL query capture, database synchronization**

I. Introduction

Data backup is getting a lot of attention as the importance of data is increasing day by day in the business arena as well as in different educational and government organizations. If any database failure happens, it needs very quick recovery of data in order to maintain productivity as well as data integrity. Different databases are fit for different applications. Data needs to be synchronized in heterogeneous environment to get data and structure from different databases by the different applications so that end users at all sites can experience improved availability of uniform data and structure [1]. If a local copy of data is unavailable, users can still access the remote copy of data from target database. For data mining, data analysis, application support [2], we need up-to-date data and structure of live source database. So, we can achieve all these requirements by real time synchronization [3]. To achieve synchronization, a core issue that should be solved is maintaining uniformity of data and its structure. Many commercial applications [4], [5], [6], [7] and methodologies are being used to accomplish this task.

In this paper, we explore the problem of preserving consistency of data in database and its structure in heterogeneous database environment. To solve this problem, we are passing SQL (Standard Query Language) [8] query to the target database through HTTP protocol. We take advantage of the fact that RDBMS compliant data supports SQL query for all database platforms having slight differences in syntax. Another reason behind our choice is that full object replacement requires huge time and bandwidth to process the data compared to SQL query transfer method. Unidirectional synchronization can be done by detecting the SQL query in source database and passing it to the remote target database through HTTP protocol. The main advantage of this method is that it does not process all data of source database. So this method does not depend on size of database.

II. Present state of the problem

In XML based synchronization [9], the authors have described about data synchronization in heterogeneous environment over HTTP and JMS but they do not mention how the method can be used for maintaining structure of database.

WisdomForce DatabaseSync [6] is very costly and it can only do data synchronization. It can not detect any change in schema of source database. For example, if any new table is created in source database, WisdomForce DatabaseSync can not automatically create this table in corresponding schema of the target database. Oracle GoldenGate [5] is more expensive than WisdomForce DatabaseSync. Also for customization, it needs permission from Oracle Corporation. This makes it more expensive. Oracle Streaming [10] is slower. It increases the load on source database as it works on the basis of database triggering.

Oracle Active Data Guard [11] can not be implemented in heterogeneous system because it only supports Oracle to Oracle database. Another problem is that it is used only for database replication but not real time data synchronization. Object-relational mapping solution can be helpful if the modified data is in our hand. So, we need to capture modified data from source database to use this solution. As we are working with heterogeneous database, most of the databases do not provide modified data capture facility. We can have two options here. Either we can extract the modified records and process them using object-relational mapping. Problem of this approach has been pointed out above. Or we can transfer the whole database in case of any changes. This is absolutely impractical.

III. Methodology

A. Sync Architecture

Fig. 1 depicts main components of our synchronization architecture. SQL query capture and transfer module works in

978-1-4577-1583-9/ 12/ $26.00 © 2012 IEEE

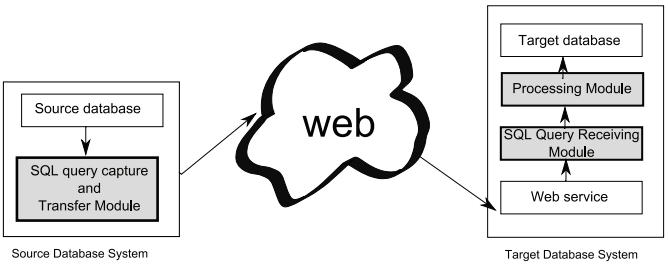


Fig. 1. Sync architecture.

source database system. It captures SQL query from source database and sends it to the remote database system. The module uses HTTP protocol to transfer captured SQL query as well as timestamp of the query. In the target database system there are two main components. One is SQL query receiving module which works with transfer module of source database system to receive SQL query and other information. Then it saves SQL query as text file with timestamp information. Another one is Processing Module which processes and executes SQL query in target database. We will describe the details of these modules in the next sections.

B. SQL query capture and transfer module

A process runs in back end to get SQL query from source database and transfers it to the target database system. Fig. 2 shows the flow chart of this module. At first, the process connects with source database. Then it reads a text file to acquire the last synchronized timestamp. The process starts searching the new SQL statements that are created after this timestamp. The way of capturing new SQL query depends on source database. Most of the RDBMS's have features for capturing SQL query. If source database is Oracle, we can use Oracle audit [12] to capture SQL query.

After getting new queries, it transfers all SQL queries one by one and updates timestamp in timestamp log file. Then the engine waits a certain predefined interval to start again. This waiting time is fixed by administrator and it can be changed according to one's requirement.

For a single query, the process sends request to query receiving module of target database system. For this, we use HTTP request to web server. If the process fails to send single query (for example because of network error) it will stop processing to avoid inconsistency. Database system administrator can observe the error log file and start the engine again. We consider that a miss operation is more harmful than the system sitting idle waiting for administrator's action.

C. SQL query receiving module

This module gets request from the engine of source database system. It takes two web request parameters. One is the SQL query itself and the other is the timestamp. After getting these data from source database engine, it saves every SQL query in text file. The file will always be saved in a specific "Received" directory of the same system. The content of text file

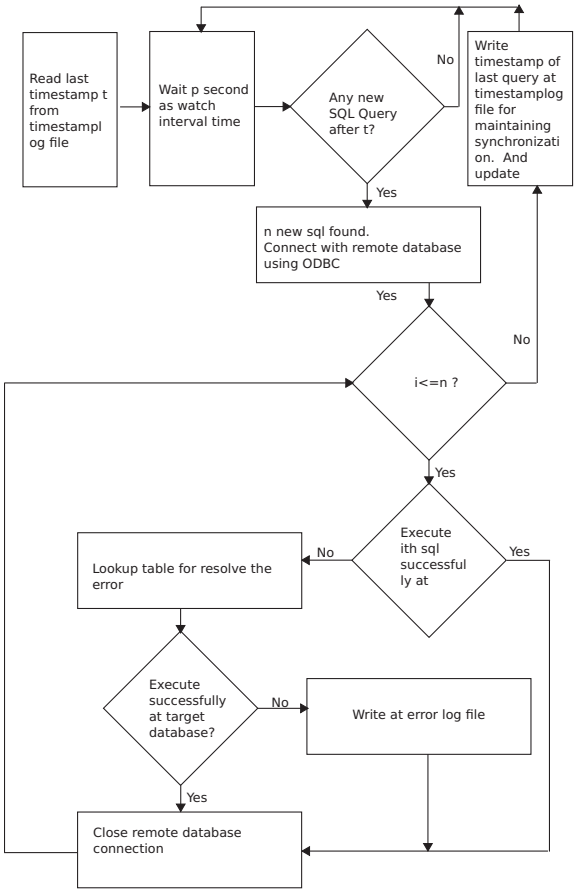


Fig. 2. Flow chart of the capturing SQL query and transfer module

is SQL query and the name of file is timestamp. We are keeping timestamp information because it is very important to process query sequentially. If the execution of SQL query is in different order compared to source database then unexpected situation may arise.

IV. Processing Module

Another engine runs in back end to process every query and executes it in the target database. Fig. 3 shows the flow chart of this module. After a certain interval time the process checks whether any new query arrives or not in "Received" directory. If any new query is found, it sorts the queries in ascending order according to timestamp to avoid unwanted result. Then it starts processing every query.

It is very natural that for different RDMS's there will be some mismatches in the syntax of SQL query for data types and functions in target database. So in the query adapt section, at first it checks whether the query type is DDL or DML. If the query is DDL, it will replace source data type to target data type by using the data type map table which is predefined. On the other hand, if the query type is DML it will replace pattern of function by using the function map table. Function map table is such a table where source function pattern is mapped with target function pattern. This is the most important part of

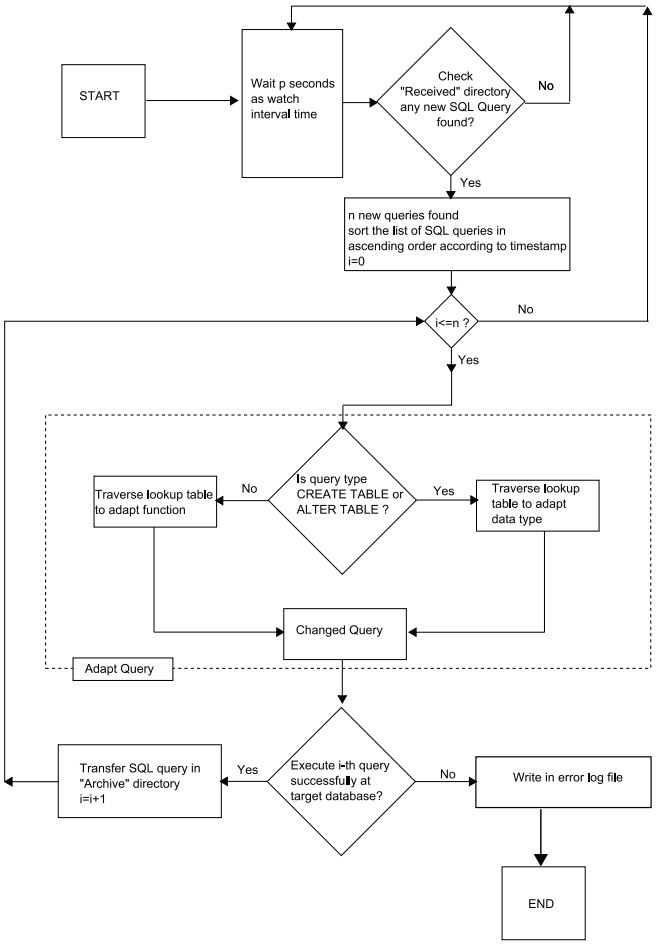


Fig. 3. Flow chart of the processing module.

our methodology. In the next section we highlight this error handling technique in detail.

After adapting SQL query, it executes changed query in the target database. If execution succeeds, it will move the SQL query (text file) from "Received" directory to "Archive" directory and start to process the next query. On the other hand, if it fails, it will save the error with timestamp in error log file and stop processing.

A. Error Handling

As we have already mentioned that RDBMS compliant database in heterogeneous environment supports generic SQL query which is the same but there are some function and date type mismatches. So, error may happen for both DDL and DML query.

First of all, for DDL, some data type may not be supported in target database. That is why we need data type mapping between source database and target database. For example in Oracle, the following query will execute properly. `create table student (id number, name varchar2(100));` But in MSSQL, that query will fail to execute because MSSQL does not support `varchar2`. As an alternative we can use `varchar`.

TABLE I
LIST OF CASE FOR DATA TYPE MAP BETWEEN ORACLE AND MSSQL.

Oracle	*MSSQL Server*	*Applicable Case*
FLOAT	FLOAT	Case 1
CHAR(n)	CHAR(n)	Case 1
VARCHAR2(n)	VARCHAR(n)	Case 2
CLOB	TEXT	Case 2
BLOB	IMAGE	Case 2
DATE	DATETIME	Case 2
BFILE	–	Case 3

TABLE II
A SAMPLE OF LOOK UP TABLE FOR DATA TYPE MAPPING.

Oracle	*MSSQL Server*
number	float
real	float
clob	varchar(max)
date	datetime
interval	datetime
nvarchar2	nvarchar
varchar2	varchar
long raw	image

Similarly, for DML, some functions in source database can differ from target database. For example, if someone executes substring function like, $substr(char\_exp, start, length)$ to get the part of the string as return of the function, it will execute in source database as it is Oracle. But if target database is MSSQL, `substr(char_exp,start,length)` will fail to execute because MSSQL does not support it. So, we should replace the query `substr(char_exp,start,length)` with `substring(char_exp,start,length)` to adapt SQL query for MSSQL database. We call error handler to solve this type of problem. In the following sections, we discuss how we have solved error in DDL and DML query.

B. Error handling for DDL query

Error in DDL occurs due to data type mismatch in heterogeneous database. Let us examine Table I for data type mapping. Three cases may arise:

Case 1: Same data type for source and target database.
Case 2: Data type is not supported in target database but has alternative.
Case 3: Target database does not support data type and has no alternative.

In Case 1, this type of SQL query will execute without any error in target database because the data type is exactly the same as in source database. Secondly, in Case 2, we will use a lookup table for data type mapping. Table II shows the lookup table for Case 2.

If query type is `CREATE TABLE` or `ALTER TABLE`, we traverse data type lookup table and replace data type with alternative data type (in 2nd column of Table II). So the query will be executed to the target database without any error.

TABLE III
SOME EXAMPLES OF FUNCTION MISMATCH.

Oracle (source database)	*MSSQL (target database)*
`months_between (datetime2, datetime1)`	`datediff (mm, datetime1,datetime2)`
`to_char(date, format)`	`datename (datepart, date)`
`ln(n)`	`log(n)`
`nvl(variable, new_value)`	`isnull(variable, new_value)`
`substr(char_exp, start, length)`	`substring(char_exp, start, length)`
`chr (integer_expression)`	`char (integer_expression)`
`instr (exp, char_string, 1, 1)`	`charindex(exp, char_string)`
`ceil(n)`	`ceiling(n)`
`mod(x, y)`	`x%y`

TABLE IV
AN EXAMPLE OF LOOKUP TABLE FOR PATTERN REPLACEMENT.

Source Pattern	*Target Pattern*
months_between \((.*),(.*)\)	datediff (mm,$1,$2)
ln\((.*)\)	log($1)
nvl\((.*),(.*)\)	isnull($1,$2)
substr \((.*),(.*),(.*)\)	substring ($1,$2,$3)
chr\((.*)\)	char($1)
instr \((.*),(.*),1,1\)	charindex($1,$2)
hline ceil\((.*)\)	ceiling($1)
mod\((.*),(.*)\)	$1%$2

Finally, for Case 3, we are unable to handle this error because the data type is not supported in target database and it has no alternative data type in target database. These errors are reported in error log file. In the discussion section, we will discuss more about this case.

C. Error handling for DML query

We use function in SQL query for our regular data manipulation. It differs from database to database. For handling this, we use another lookup table. User will put list of mismatched functions from his target database to source database. For example, in our case, source database is Oracle and target database is MSSQL. Table III shows some function mismatches between these two databases.

For avoiding function mismatch, we will replace source pattern with target pattern using regular expression. Table IV shows some function patterns.

In our method, to avoid the error in DML query, it matches pattern from lookup table and changes original SQL query by replacing pattern to adapt the SQL query for the target database.

V. EXPERIMENT

We developed various modules and implemented our idea. In our setup, SQL query capture and transfer module runs at source database server. This engine has been developed using C/C++ and PHP programming language. Oracle provides Oracle C++ Call Interface (OCCI) [13] to connect with the Oracle database using C/C++. We have used various methods and classes of OCCI in our C/C++ code in this module. In this module we have also used curl library of php in command line to post HTTP data in webserver's query receiving module. To create object files and executable file, we passed necessary libraries as command line parameters.

On the other hand, we have developed SQL query processing module in target database system. As the webserver apache2 runs in target database system we use PHP here also. PHP supports regular expression replacement and gives command line execution facility. Query processing module gets a text file containing the SQL query from "Received" directory as input, then adapts data types and SQL functions in the SQL query according to lookup tables. After successful execution of the adapted query in target database it transfers the text file in "Archive" directory.

VI. RESULT

We have examined both the DML and DDL operations. In DML, we have observed insert, update and delete queries. And in DDL, we have observed create table, alter table, drop table statements. We have also looked at what happens if error arises in SQL query for function and data types mismatch. Figure VI shows a example how the DML are being adapted and affecting in the target database. We have also observed that data and structure are being properly synchronized in the experimental environment.

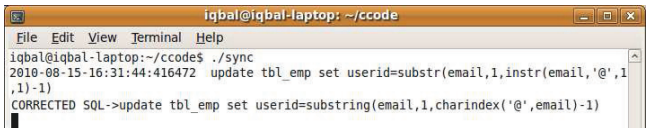
VII. DISCUSSION

As previously mentioned, in XML based synchronization [9], the authors have described about data synchronization in heterogeneous environment over HTTP and JMS but they have not mentioned how the method can be used for maintaining structure of database. This method transfers data using XML file. All data are being transferred in text format without passing any data type information. We think, it is a poor design because it does not consider data structure. In our case, we have emphasized both on data and its structure.

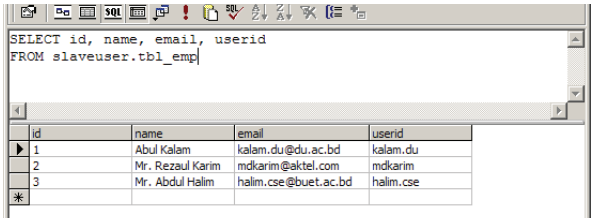
We have also reviewed some products launched by Oracle Corporation. Some of those do not support heterogeneous database; for example, Oracle Active Data Guard [14]. Some provide database replication which provides an extra standby database. That does not serve our purpose. Main disadvantage for all Oracle products is that the source database must have to be Oracle. As these are proprietary software, these are not customizable and are expensive. In this sense, our methodology and its implementation is open source, easily customizable and necessary parameter configurable. It supports heterogeneous database.

WisdomForce DatabaseSync [6] is very costly and can do only data synchronization. But it can not detect any change in schema of source database. For example, if any new table is created in source database, WisdomForce DatabaseSync can

(a) `Update table` query using function not supported in target database



(b) Adapted query in processing screen



(c) Query affected at target database

Fig. 4. Error handling in DML query.

not automatically create that table in corresponding schema of the target database. But our proposed method works successfully for this issue. It detects create or drop object even modification of objects and affects the target database.

Most of the products mentioned above do not support multiple target databases. But our method can be implemented for multiple target databases.

VIII. Scope of Improvement

We believe that our methodology can be improved for unsupported data types and functions which have no alternative in the target database. For data type, instead of failing such a case we can parse the data from SQL query and use some chosen default data type to store that data. On the other hand, for function, we can manipulate resultant data of functions and can update target database using that resultant data.

IX. Conclusion

We have demonstrated our methodology for maintaining uniformity in data and database structure in heterogeneous database environment. We have researched on uniformity maintenance of data and its structure before starting the real time data synchronization among the heterogeneous database systems. So far as we know, this is the first implementation of database synchronization in heterogeneous environment based on SQL query. In this research, we have designed a new methodology, implemented it and discussed it in detail. The main advantages of this method are that it does not depend on database size, uses low network traffic, can handle data and data type and supports heterogeneous databases.

Acknowledgement

This work is part of a Master's thesis work done in the Department of Computer Science and Engineering (CSE) at Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh. The authors gratefully acknowledges the cooperation and support received from BUET.

References

[1] Y. Geng, Z. Zhao, and X. Kong, "Synchronization system among remote heterogeneous database systems," in *Knowledge Acquisition and Modeling Workshop*, Wuhan, 2008, pp. 990 – 993.

[2] X. Yu, "The application study of heterogeneous database synchronization," in *International Conference on Electronic Computer Technology (ICECT)*, Kuala Lumpur, 2010, pp. 79–81.

[3] Y. Hao, D. Xing-chun, and J. Guo-quan, "Research on data synchronization in oracle distributed system," in *International Seminar on Future Information Technology and Management Engineering*, Leicestershire, United Kingdom, 2008, pp. 540–542.

[4] "Dbmoto: Real-time data replication and data integration using snapshot, incremental and synchronization replications," http://www.hitsw.com/products_services/dbmoto/dbmoto_dsheet.html.

[5] "Oracle GoldenGate," http://www.oracle.com/.

[6] "WisdomForce WisdomForce DatabaseSync Real Time Change Data Capture and Replication," http://www.wisdomforce.com.

[7] M. I. Hossain and M. M. Ali, "SQL Query Based Data and Structure Uniformity Maintenance in Heterogeneous Database Environment," Master's thesis, Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka 1000, Bangladesh, 2010.

[8] J. R. Groff and P. N. Weinberg, *SQL: the complete reference*, ser. Osborne ORACLE Press Series. McGraw-Hill/Osborne, 2002.

[9] W. Xiaoli and Y. Yuan, "Xml-based heterogeneous database integration system design and implementation," in *3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), 2010*, Chengdu, 2010, pp. 547 – 550.

[10] L. Wong, N. S. Arora, L. Gao, T. Hoang, and J. Wu, "Oracle streams: a high performance implementation for near real time asynchronous replication," in *IEEE 25th International Conference on Data Engineering, 2009. ICDE '09.* Shanghai: IEEE, Apr 2009, pp. 1363–1374.

[11] L. Carpenter, C. Kim, J. Meeks, and B. Burke, *Oracle Data Guard 11g Handbook*, ser. Osborne ORACLE Press Series. McGraw Hill Professional, 2009.

[12] R. Ben-Natan, *HOWTO Secure and Audit Oracle 10g and 11g*. Auerbach Publications, 2009.

[13] M. Scardina, B. Chang, and J. Wang, *Oracle Database 10g: XML & SQL : design, build & manage XML applications in Java, C, C++ & PL/SQL*, ser. Oracle Press. McGraw-Hill/Osborne, 2004.

[14] "Oracle - goldengate statement of direction," http://www.oracle.com, Oracle Corporation,World Headquarters,500 Oracle Parkway,Redwood Shores, CA 94065,U.S.A., Nov. 2009.