Secure Migration of Tables across Different Database Management Systems over a Cross-Platform Environment

Pvnss Krishna Murthy, Surabhi Agarwal, Tulsi Garbyal, and Pavan Chakrobarty

***Abstract*—Cross-platform environment resembles a heterogeneous database systems [1],[2] where different sites (nodes that are equipped with a computational and storage resource) may use different ways of storing and representing the data, file formats, access protocols, query languages, etc. The sites in this cross-platform environment communicate through an insecure channel. This paper provides a solution in which relational database can be migrated from server to client based on replication or fragmentation (vertical) without loss of information. This migration is independent of platform and secured by strong cryptographic algorithm. Here, every site may have different Relational Database Management Systems (RDBMSs). By RDMSs, we mean Oracle, MySql, DB2, MS Access etc.**

***Index Terms*—AES, cross-platform environment, RDBMS, SAX, XML.**

I. Introduction

In modern times, computers are main source of information transfer and communication. Computer network may be cross-platform in terms of operating system and language used whereas insecure in terms of transmission errors, data loss, adversary attacks, etc. When information transferred is in form of data which is structured into tables with well defined format and parameters, then we call it relational database and systems that manage them are called relational database management systems (RDBMS). The problem comes when we have to transfer relational database from one site to another i.e., from server-side to client-side cross-platform environment which is insecure. These in systems may use different RDBMSs. In this paper we propose a solution to this problem.

The relational database should be converted into a format prior to transfer that is understandable to each host in a cross-platform environment. The database format that is used is XML, as it is platform independent. The hosts that participate in communication must agree upon some protocol. In order to fulfill reliability constraints, TCP is used. TCP takes care of transmission errors and data losses, but it is insecure in terms of adversary attacks which can be avoided by using cryptography. While transferring relational database, it must be necessary to preserve its schema. The preservation of schema facilitates us to integrate it efficiently at the client RDBMS.

This paper gives an overview of an application which provides an interface where user can select any one of the RDBMSs present at the server-side. User then selects any one of the tables present in the RDBMS for migration. He/she is given a choice to either replicate whole table or a fraction of it (vertical fragmentation) based on which data is extracted and then converted into xml file. Before transfer, xml file is encrypted. At client-side, received file is decrypted and parsed to extract the original data. This data is integrated into one of the client-side RDBMSs specified by the user.

The remaining paper is organized as follows. In Section II, we analyze the previous research work. Section III describes the system architecture. Section IV focused on security aspects. Section V presents the implementation details of our proposed work. Finally, conclusions and future work are presented in section VI, followed by references in section VII.

II. Analysis of Previous Research Work

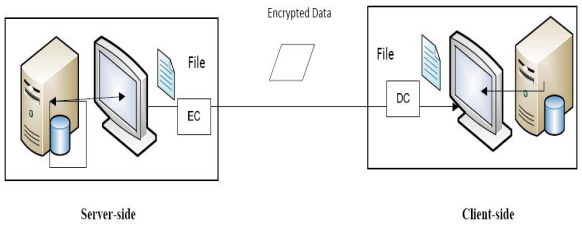

Fig 1. System architecture.

Large amount of work has already been done on distributed database systems [3]. Many algorithms have been implemented to achieve dynamic fragmentation and object allocation in distributed databases [4]. Complexities arouse while considering heterogeneous database systems [1], [2] in which sites are unaware of each other, database software and schemas used by the different systems may be different. Data migration and task Scheduling problem called Data consolidation [5] has been evaluated which applies to data-intensive applications that need more than one pieces of data for their execution. A software framework has been designed to alleviate the problem associated with task migration [6], [7]. Some of the

Manuscript received February 2, 2012; revised April 23. 2012. This work was supported in part by Indian Institute of Information Technology, Allahabad, India.

P.V.N.S.S. Krishna Murthy was a student at Indian Institute of Information Technology, Allahabad, India. He is now working as a Software Developer at Cisco Systems, Inc, Bangalore, India (email: kpvnss@cisco.com).

S. Agarwal was a student at Indian Institute of Information Technology, Allahabad, India. She is now working as a Business Technology Associate at ZS Associates India Pvt Ltd, Gurgaon, India (email: surabhi.agarwal@zsassociates.com).

T. Garbyal was a student at Indian Institute of Information Technology, Allahabad, India. She is now working as an Associate Software Engineer at Accenture, Bangalore, India (email: tulsi.garbyal@accenture.com).

P. Chakraborty is Assistant Professor at Indian Institute of Information Technology, Allahabad, India and is currently working with Robotics Department (email: pavan@iiita.ac.in).

solutions are proposed which can transfer data from hierarchical database to relational database [8]. Our framework was used to transfer task, data and state information across the platforms.

III. System Architecture

Fig 1 describes the basic system architecture which consists of server-side and client-side. Server-side is the source of information from where data is migrated. Client-side is the receiver of information which requests connection with the server for migration of data. At the server-side, files are transferred in encrypted form, represented by EC. The transferred file is decrypted at the client-side represented by DC.

A. SERVER-SIDE

Following steps are involved at the server side during the transfer of data: 1) Extraction of a table for migration, 2) Converting the table into a format that is useful in efficient transfer and specifying the client side RDBMS details, 3) Establishing the connection with the server which is in waiting state (Pull Mechanism), 4) Transferring the table and record information along with client side RDBMS details.

i) Extraction of a table

Extraction of tables includes series of steps. Initially user is provided with the list of RDBMSs present at the server side from which any one can be selected. User gets access to the database only after submitting correct authentication details like username, password and database name (if needed). These authentication details are required for the database connectivity. Once connection is established, list of tables are extracted and made available to the user from which he/she can select any one of them. According to the user's choice, table information (column-names, data-types, default values etc) is extracted and primary key attributes are identified. The list of column names is provided to the user from which he/she can select either all the columns of the table (replication) or just few of them (fragmentation). For successful fragmentation, it is necessary to select all primary key attributes and hence if any of the primary key attributes is left unselected by the user, an error is invoked and processing starts all over again. Based on the selected column names, data is extracted from the database. Implementing above mentioned steps sequentially, results in successful extraction of tables.

ii) Conversion of the table and Specification of client-side RDMS details

After the extraction of table, records of the table are stored in an XML format in order to transfer through the network. The main advantage of xml is that it is not necessary to use same language to convert a table into XML format and vice versa i.e., we can use any language like C++ to convert into xml format and later the xml file can be parsed using any language like java. Because of these reasons xml can be used in heterogeneous environment. User must also specify client-side RDBMS details like RDBMS name, user-name and password. Besides transferring only xml file which contains records (data), the table description (data structure) along with client-side RDBMS authentication details should also be transferred in a form of the text file. Following details are stored in a text file: 1) server-side database platform name, 2) client-side database platform name, 3) client-side RDBMS authentication on details (username, password), 4) database name (if required, like for MYSQL), 5) table name, 6) number of records that are stored in xml file, 7) number of primary key attributes and finally 8) the list of attributes. Each attribute can be described by following parameters: 1) attribute name, 2) attribute type, 3) key (null or not-null), 4) default value. Table I shows how attributes are stored in the text file. This text file is transferred along with the XML file from server-side to client-side.

TABLE I: Attribute List

Attribute Name	Attribute Type	Key	Default Value

Server-side and client-side platform names are needed in the text file because the client-side must know about the previous database platform and the current database platform for the type conversion of attributes. We need number of records to check whether whole xml file is transferred or not which can be done by matching this number against the number that is obtained after parsing of the xml file. Successful xml file transfer results in correct match. We need number of primary key attributes (n) because the client-side confirms that the top n elements in the list of attributes form a list of primary key attributes.

iii) Establishment of the connection

In order to transfer files first of all a connection must be established between server-side and client-side. Socket programming is used for connection establishment i.e., sockets are created on both sides. Server side acts as a central server so this has to be started first i.e., it goes into waiting state as it is a Pull mechanism. Here, server has to be assigned a port number to which it listens in order to process the client's request. Both server-side and client-side must agree on the same protocol and client-side must know the Internet Protocol (IP) address and Port number of the server for connection establishment.

iv) Transfer of files from server-side to client- side

Once connection is established, the records that is stored in the xml file, the table information (server-side RDBMS name, table name, number of records in the xml file, number of primary key attributes, attributes information) and client-side authentication details (stored in text file) are transferred to the client. Exceptions like connection link failure and server break down must handled at both ends. Exception handling must be implemented to increase the robustness.

B. CLIENT-SIDE

Following steps are involved at the client-side while receiving and integrating the data: 1) receiving the files sent by the server in the same format, 2) Parsing the XML file, 3) manipulation of the attribute information and Integration into the client-side RDBMS. Fig 2 shows steps followed at the client-side.

i) Receiving the Files from the Server-side

At the client site, first task is to receive two files, one is an XML file and the other is a text file. For generalization, user is given an option to choose IP address and port number for communication with the server side. If the data is available at specified location (given by user), it is received otherwise an error message is displayed. The connection is closed after successful transfer. The details like server-side RDBMS name, client-side RDBMS name, table name, number of records in the xml file, number of primary key attributes, attributes information and client-side authentication information are extracted from the description file(text file).

ii)Parsing the XML file

Next task is to parse the transferred xml file and to extract useful RDBMS data stored within it. The information like number of records in an xml file and number of primary key attributes is given as an input to xml parser. The parsed data is then stored in some table like data structure so that the value of a corresponding attribute is accessed in constant time with respect to each tuple.

iii) Manipulation of the attribute information and Integration into client-side RDBMS

If the server-side RDBMS and client-side RDBMS names are same then no manipulation of the attributes information described in the Table 1 is required, else necessary changes has to done in order to integrate it into the client-side RDBMS. Here, manipulation means changing the attribute types, their corresponding default values and key values in order to make it compatible with client-side RDBMS. Based on the client-side authentication details, parsed data and the manipulated attribute information are integrated into the client-side RDBMS.

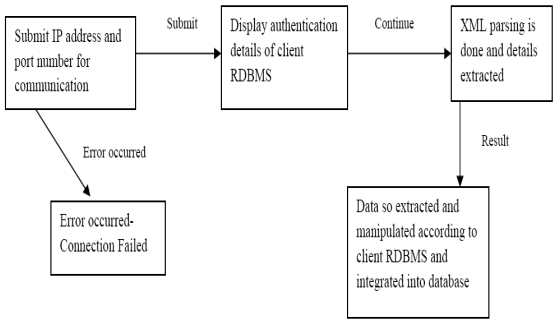

Fig. 2. Steps followed at client-side.

IV. Security Aspects

Our main task is to migrate the data through a network in secure manner. XML file and description file that are to be transferred contains sensitive information (table schema, tuples etc) which when leaked or tampered may result in severe consequences like destroying the integrity of the data. So, in order to ensure the security of transmission, cryptographic techniques have to be used. Many cryptographic techniques evolved and tested against four major cryptographic attacks namely plain-text only attack, cipher-text attack, chosen plain-text attack and chosen cipher-text attack. Few of the techniques widely used are RSA, DES (Data Encryption Standard), AES (Advanced Encryption Scheme), RC4, SHA (Secure Hash Algorithm), MAC (Message Authentication Code) etc. Among these AES is the most secure Symmetric-Key Cryptographic technique which ensures authenticity and privacy of data and is used for transferring highly confidential information. Thus AES is used in encryption and decryption (as shown in Fig-1) of Xml file and description file. AES is a block–cipher algorithm where block and key can be of 128, 192 or 256 bits length. Based on the level of security key size can be specified. It is resistant against all above mentioned attacks, efficient and is known for its design simplicity [9], [10]. In Symmetric-Key algorithms as in AES, a secret key has to be shared between communicating parties. As its security relies on the shared key this has to be distributed securely. For this purpose a Public-key Cryptographic algorithm which ensures confidentiality like RSA can be used. Even though RSA had been broken in the past, a key length of 1024 bits is highly resistant [9]. Fig 3 shows AES key transfer from host A to host B. Both of the hosts generate a pair of public and private keys i.e., Pu_a, Pr_a are generated by A and Pu_b, Pr_b are generated by B. Pu_a and Pu_b are known to both the parties but Pr_a is known only to A and Pr_b is known only to B. So, A encrypts the AES key using Pub. The encrypted file (EF) is transferred to host B where it is decrypted using its own Pr_b. This encryption and decryption follows RSA algorithm.

V. Implementation Details

For implementation of above mentioned steps we have used java as our programming language. Java is platform independent and it contains both light-weight and heavy-weight GUI tools which can be easily implemented. Java abstracts many details from the developer that's why it is easy to use. We have used Oracle-10g and MySql as RDBMSs as they are most common. Cross-platform environment has been achieved by using three different operating systems namely Ubuntu, Fedora and Windows. Here, all the three operating systems contain both Mysql and Oracle and any one of them can become a server leaving remaining two as clients. For database connectivity we have used JDBC driver with distinct connection strings for different RDBMSs. Simple API for xml (SAX) was used for creation and parsing of XML document. SAX is used in streaming XML documents as it is event based and inherently sequential [11]. For generating RSA key-pair, OpenSSL was used. Before integration, data manipulation like 'date'/'datetime' data-type in MySql to 'date' in Oracle has been done. Only few of the integrity constraints of server-side RDBMS can be preserved in client-side RBMS like primary/foreign key constraints, null/default value and changing enum (in Mysql) to 'check' constraint (when migration is done from Mysql to Oracle). Thus we can migrate the data securely from server-side to client-side in cross-platform environment with some limitations.

VI. Conclusion

Secure Migration of tables across different database management systems over a cross-platform environment can

be efficiently done. Data is effectively transferred and stored without any loss of information. Location and processing complexities are hidden from the user. There are extreme possibilities for future work like: i) languages like SYBASE and DB2 along with Oracle and MySql can be included thus covering 90% of database application using users, ii) application can be made OS independent for all kind of available OS like Windows XP, Vista, Windows 7, Linux (Mandriva, RedHat, Fedora, Ubuntu etc.) , Mac OS etc. This will help the user to use the application at any OS and platform. This is especially helpful in business environment where communication of database takes place at heterogeneous level [5], iii)when systems are connected to large networks, many failures like site failure, partition failure, network failure etc can occur during transfer. Error-handling can be extended and developed to handle all kinds of failure and prevent data lose, iv) security level can be increased or decreased based on the importance of the data that has to be transferred in order to make it efficient, v) concurrent access of server by the clients can be synchronized by using the concept of multi-threading.

REFERENCES

[1] J. Zhiquan, L. Chengfei, S. Zhongxiu, Z. Xiaofang, C. Peipei, and G. Jianming, "Design and Implementation of a heterogeneous distributed database system," in *Journal of Computer Science and Technology, published by Springer Boston*, vol. 5, no. 4, pp. 363-373.

[2] Z. Gao, S. Luo, Y. Lin, and D. Ding, "A Grid-Based Integration Model of Heterogeneous Database Systems," in *2009 International Conference on Information Technology and Computer Science*, vol. 2, pp. 126-129. 2009.

[3] A. Silberschatz, H. F. Korth, and S. Sudarshan, *Database Systems Concept*, 4th ed, The Mc-Grill Companies, vol. 1, pp. 705-749. 2001.

[4] A. Sleit, W. AlMobaideen, S. Al-Areqi, and A. Yahya, "A Dynamic Object Fragmentation and Replication Algorithm In Distributed Database Systems," in *American Journal of Applied Sciences*, vol. 4, no. 8, pp. 613-618. 2007.

[5] P. Kokkinos, K. Christodoulopoulos, A. Kretsis, and E. Varvarigos, "Data Consolidation: A Task Scheduling and Data Migration Technique for Grid Networks," in *Eighth IEEE International Symposium on Cluster Computing and the Grid*, 2008.

[6] M. Tungare, P. S. Pyla, and M. Sampat, P. Quinones, and M. A, "Syncables: A Framework to Support Seamless Data Migration Across Multiple Platform,s" in *Portable Information Devices, PORTABLE07 IEEE International Conference*, pp. 1-5.2007.

[7] X. Wang, L. Huang, and Y. Zhang, "A Grid Middleware--DISQ for Data Integration," in *2008 International Conference on Computer Science and Software Engineering*, vol. 3, pp. 62-65, 2008.

[8] A. Meier, R. Dippold. J. Mercerat, A. Muriset, J.-C. Untersinger, R. Eckerlin, and F. Ferrara, *Hierarchical to relational database migration*, Vol. 11, no. 3, pp 21-27, 1994.

[9] W. Stallings, *Cryptography and Network Security Principles and Practices*, 4th ed, Prentice Hall, pp. 140-280, 2005.

[10] Z. J. Chowdhury, D. Pishva, and G. G. D. Nishantha, "AES and Confidentiality from the inside out," in *2010 12th International Conference on Advanced Communication Technology*, vol. 2, pp. 1587-1591.

[11] Y. Pan, Y. Zhang, and K. Chiu, "Hybrid Parallelism for XML SAX Parseing," in *2008 ICWS '08. IEEE International Conference on Web Services*, pp 505-512, 2008.