A Framework for Data Migration between Various Types of Relational Database Management Systems

Ahlam Mohammad Al Balushi
Sultanate of Oman,
International Maritime College
Oman

ABSTRACT
Data Migration is an important event for an organization to be competitive in today's business world. However, it is a critical process that directly influence the quality of data. Data is considered as a critical asset to continue organization's business operations. Therefore, this paper proposes a framework to improve quality of data during data migration between different types of Relational Database Management System (RDBMS).

Keywords
RDBMS, Data Migration, ETL, and Multi-pass Sorted Neighborhood Algorithm.

1. INTRODUCTION
Due to rapid changing in information technologies and expanding scale of organization information system, the resulting data quantity becomes larger and larger [1] and this will effect on the performance of database. Therefore, organizations respond quickly to cope with the changes and improve the database performance by modernizing or replacing their existing database to a new one. Inevitably, a new database designed to be more efficient and inclusive than older database. Furthermore, the new database developed to replace old one that has become outdated and which resists further development in the speed and capacity to be able to handle an ever-increasing amount of data. Further, often new statutory and regulatory requirements require [2] adopt new business applications that thus requires replacing or upgrading the existing database. Also, economic problems in the organization lead to uninstalling commercial and expensive database and installing open-source and cheaper database instead [3]. Consequently, the question arises, how the organization replaced or upgraded its database while retaining the data and customizations. The answer is data migration which is moving data from a source database into a target database. This includes migrate the source schema and source data into their equivalents in a target database. However, the conversion of programs and their queries are outside the scope of this paper because it is a software engineering task.

ETL (Extract, Transform, and load) is a general process of data migration between different types of Relational Database Management System (RDBMS). It is used in some migration tools and manual migration process.

- Extract – it is responsible to extract data from the source database.
- Transform - it is used to apply a series of rules or functions on incoming data (extracted data) such as join, aggregate or filter functions, to make the extracted data in a usable form for migration to the target database.
- Load – process of loading the data into the end target database.

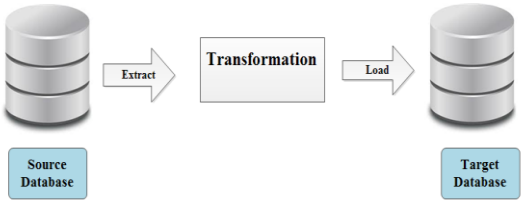

Figure 1 - ETL Process

2. PROBLEM STATEMENT
Data migration is a critical process; because dirty data in the data source and crude movement of data in the shape of unprofessional migration or bad strategy for data migration leads for different risks such as inconsistent data, inaccurate data, incomplete data and duplicate data. Thus, it affects the efficiency of organization's business. However, most Relational Database Management Systems (RDBMS) such as Oracle, SQL Server and MySQL, have their own database transferring utilities for some database migration tasks [4]. Built-in data transfer utility is called also Import/Export. However, the drawback of this feature that it based on two crucial assumptions. First, user must assume the content of the source data is correct, especially with entity integrity, domain integrity and referential integrity [4]. Otherwise, if there is a problem in the source data, built-in utility (Import/Export) will fail to transfer data. Second, transferring utility in relational database management system (RDBMS) assume that target database has an identical structure to that of the source database. Data migration process becomes a difficult challenge when source and target databases are different in their structures [5], because each relational database management system has different property and characteristic. Inevitably, they are differing in naming, support of data types and SQL commands for creating and editing database tables [6]. Therefore, if database administrators will modify the structure in the target database to accommodate changing requirements, they have to change the structure of the source database manually or by writing a customized program. A customized program will increase the cost of data migration because it may include errors and vulnerability.

Due to limitation of built-in data transfer utilities, quite a few third-party products have been developed to migrate data between different database management systems, and are commercially available such as such SwisSQL migration tool, SQL Server 2008 and others. However, these products have problems and disadvantages in data integrity, data correctness and others. For example:

1. Some tools like Oracle (SQL Developer 3.1 Data Dump), SQL Server 2008 and DBConvert do not providing data integrity, which is a vital role in preserving the consistency and accuracy of data in a database.
2. Tools cannot be modified or extended by user [3].
3. There is no possibility to change the features of tables and attributes in the target database during the process of data migration.
4. Tools can not verify the data correctness. They only provides a simple report stating the number of records that have been migrated [4]. Thus, database specialists must perform validation either by writing special process or inspecting the database visually.

3. RELATED WORK

Progress and growth of information technology led some researchers to propose expert system to help them make a convenient decision by using knowledge base. Therefore, Walek and Klimes (2012) proposed fuzzy expert system for data migration between different relational database management systems (RDBMS). The main reasons to propose this system was the difference between Relational Database Management System in the name and definition of data types. For instance, Oracle contains Long, Blob, Clob; while SQL Server contains Text, Ntext, and LongVarchar. Moreover, some tools that use for migrating data between different RDBMS have problems with foreign key, data integrity, etc. Therefore, this expert system contains a knowledge base which composed of IF-THEN rules and based on the input data suggest the appropriate data types of columns of database tables [3]. Also, it is able to optimize the data type in the target Relational Database Management System (RDBMS) tables based on processed data of the source RDBMS tables [6].

Vitthal, Baban, Warade and Chauldhari (2013) presented a system to be as a tool to manage and store the movement of data to a target database in cost-efficient way, especially if the source and target databases are heterogeneous They designed the system to migrate data from existing database to a new database without any manual intervention. Therefore, it was easy and efficient tool, so any person who has a basic technical knowledge can use it. Further, it was designed to work on any windows platform. Further, migration tool provided source databases as MS Access, and destination databases as SQL Server and Oracle [7]. The migration tool allowed user to migrate tables along with constraints such as primary key, foreign key, index key, unique key, and check constraints. The other part, it provided facility for selecting the fields of a table or an entire table itself for migration [7]. Views were supported for migration to hide a part of database from certain users. In addition, the tool produced a status report after migration occurred to show if the table schema is migrated, constraints have been migrated and records are inserted successfully or pending or failed. Finally, Vitthal, Baban, Warade and Chauldhari checked quality of migrated data based on criteria, such as consistency, completeness, records count, total balanced, and system integrity.

4. A FRAMEWORK FOR DATA MIGRATION BETWEEN DIFFERENT RELATIONAL DATABASE MANAGEMENT SYSTEMS

The main aim of this paper is to propose a framework for migrating database tables and their data between various types of Relational Database Management Systems. The proposed framework is designed based on one of the existing approach which is expert system for data migration between different Database Management Systems. That is mentioned in the related work section, but with some enhancements to make data migration between different Relational Databases more efficient and effective. In the other word, proposed framework solves gaps that exist in the expert system approach.

The proposed framework consists of three main steps which are divided into sub steps as shown in the below figure.

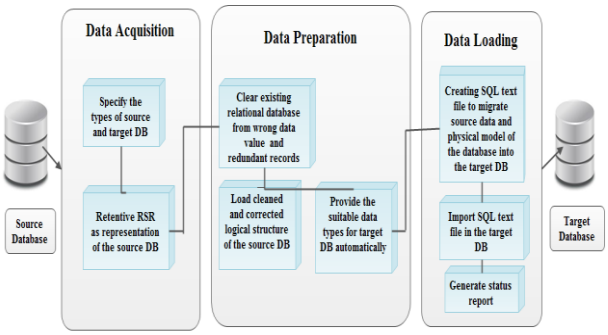

Figure 2 - Framework for data migration between different RDBMS

Data Acquisition

1. Specify and identify the source and target relational database systems [6] such as Oracle, SQL Server, and MySQL. This requires defining the version of the source and targeting database. Due to, various versions of the same type of database can be different [6]. For example:
   - Oracle - versions: 9i, 10g, 11g
   - SQL Server - versions: 2008, 2008R2, 2012
   - MySQL - versions: 4.0, 4.1, 5.0
2. Extract or retrieve Relational Schema Representation (RSR) of the source Relational Database, to be used as source information for preparation of data migration process. RSR gives the users holistic view of metadata obtained from existing Relational Database System. For instance, database table names and attributes' properties such as attribute names, data types, max length, and constraints (primary key and foreign key).

Data Preparation

1. Data cleaning is a process of detecting and removing missing data, inconsistent and redundant data, and so on, from source data in order to improve data quality. Data quality problems are classified into single-source and multi-source problems which are further categorized into schema and instance related problems respectively [8]. This paper focuses on solving part of single-source problems that are lack of integrity constraints (schema

level) and missing values, redundancy and wrong data (instance level).

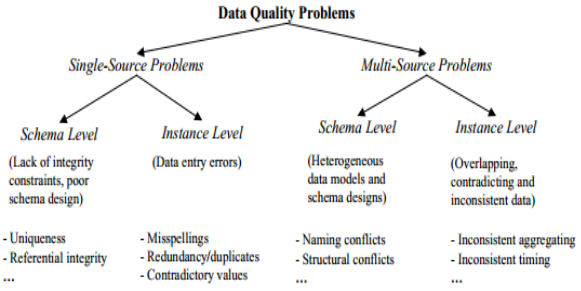

Figure 3 - Classification of data quality problems in data sources [8]

a. Data cleaning involves the following steps in the 'Instance Level':

- Remove data redundancy from data source through employing Multi-pass Sorted Neighborhood Algorithm. Its basic idea is to carry out Sorted Neighborhood Algorithm several times independently [9]. The idea of basic Sorted Neighborhood Algorithm can be summarized in three steps.
    - Create sort key - Compute a key for each record in the database by extracting relevant fields or portions of fields [9]. Typically, the key may be made of three letters from each field, but one of the fields is taken as primary field [10].
    - Sort data - Merge sort or quick sort or any good sorting methods can be used to sort dataset based on the key created in the first step.
    - Merge - A fixed size window is moved to compare the records [11]. Every new record entering in the window is compared with the previous records in the window [11]. If size of the window is w, then every new record entering the window is compared with previous w-1 records to find "matching" records [9].

Multi-pass Sorted Neighborhood Algorithm creates different sort key in each time and uses a relatively small sliding window. In merge step, the algorithm assumes that the similarity of the record is transitive and calculates the transitive closure [12]. Transitive closure means that if record R1 is a duplicate of record R2 and record R2 is a duplicate of record R3, record R1 should be duplicate of record R3 [9]. This algorithm solves problem of redundant data efficiently.

- Remove missing values or empty records by using specific query to display all the empty records in the specific table. Then, allow the user to remove these records in order to eliminate missing information. For example, SELECT * ROM table_name WHERE column_name IS NULL;

- Remove wrong data value through business rules which is the simplest and efficient way to eliminate wrong data from data source. The proposed framework checks whether the value in each field is in accordance with the business logic [13]. Then rectify wrong data with the correct data. For example:

Rules:

- *One section can have a maximum of 30 students*
- *Teacher teaches from 0 to 3 courses*

Section ID	Course ID	Teacher ID	Capacity
Section1	COMP1109	AC1198	40
Section2	COMP1114	AC1198	30
Section1	COMP3106	AC1198	35
Section3	COMP2108	AC1198	30

This table does not satisfy the rules because the teacher should not teach more than 3 courses. Also, one section can have maximum 30 students

Figure 4 - Example of wrong data value

The syntax of business rule as follows:

R#: IF (AND (OR {<CONDITION>})) THEN ({<ACTION>})

R# means a business rule, "AND" it connects two conditions that all must be fulfilled. "OR" it connects two conditions but at least one of them must be fulfilled. "{}" includes one or more conditions. Once "IF" condition is fulfilled, the <ACTION> will be taken [13].

b. Data cleaning involves the following step in the 'Schema Level':

- Check whether entity integrity (primary key) and referential integrity (foreign key) are exist in the each database tables or not, which are provided from metadata. Missing constraints can cause data errors such as non-uniqueness, nulls and missing relationships. For example, the figure 5 shows that "A faculty contains many employees". This indicates that Faculty table is not managed by any employee, which means a foreign key constraint in the Faculty table is missing. Thus, there is a relationship is missing. The foreign key (Manager_ID) is the missing key in the Faculty table. The right solution shown in the figure 6, "A faculty contains many employees, but managed by one employee".

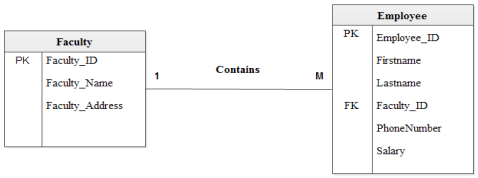

Figure 5 - Example of missing FK

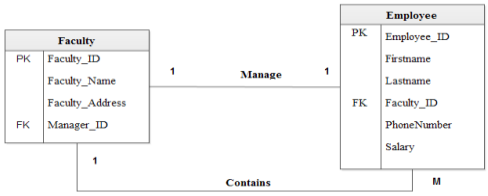

Figure 6 - Example of correct relationship between Faculty and Employee

Therefore, the framework will allow the user to create the foreign key (Manager_ID) for Faculty table. Then allow the user to search for managers and insert them in the Faculty table.

1. Load the cleaned and corrected logical structure of the source Relational Database System, that are constraints (primary key and foreign key) attributes, relationship between tables, and views. This process divides into

three parts:

a) Loading database table names, list of attributes, data types of attributes, and max length.

b) Loading of views are supported.

c) Loading relationships between all tables in the database. This step requires identifying the following:

- PK_Table - name of referenced table.
- PK_Column - column contains the primary key of the referenced tables.
- FK_Table - name of the referencing table.
- FK_Column - column contains the foreign key of the referencing table.

2. Provide the suitable data types for the target database tables and their attributes automatically by considering the following:

a) Type of source database

b) The name of attributes in the source tables

c) Type of the target database

Each Relational Database has specific data types that are differ in the name and definition. For instance, if a source relational database is MySQL database and target database is Oracle, so the suitable data type for source data type VARCHAR is VARCHAR2.

Data Loading

1. Generate SQL text file that contains data and physical model of the target database (such as database tables, constraints (primary and foreign keys), attributes, data types of attributes, and max length) [6], to import it in the target database. The SQL text file consists of the following SQL commands:

- Create commands to create database tables and attributes of the target database.
- Alter commands to add primary and foreign keys in the tables of the target database.
- Insert commands to insert source data in the target database tables.

2. Import SQL text file into the target database.

3. Generate status report to show that data migration process is successful, failure or pending [7].

5. FINDING AND RESULT

The proposed framework will be verified by creating a prototype for data migration between different Relational Database Management Systems, which is implemented in a part of the university database. It includes Student table, Faculty table, Course table, Enrollment table, Section table and Employee table, as shown in the following figure.

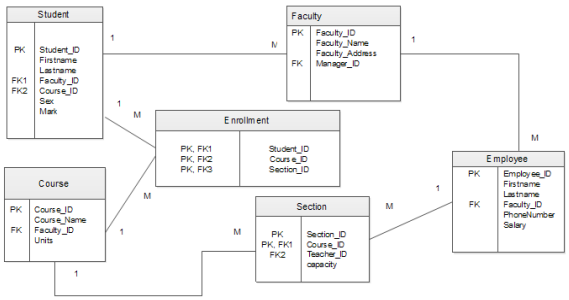

Figure 7 - Part of University Database

The following steps illustrate the process of proposed framework for data migration between different RDBMS.

a) Specify and identify the type of the source and target relational database.

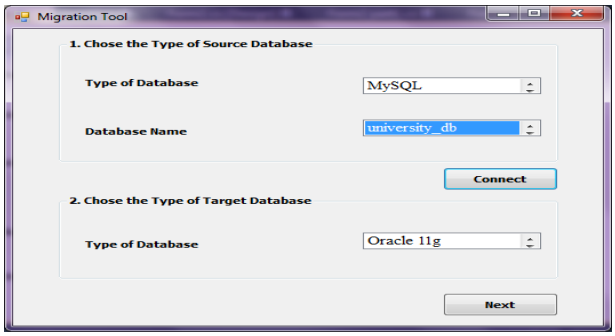

Figure 8 - Specify types of source and target database

As shown in figure 8 a user is asked to select the source relational database, which is MySQL, and then the name of database (university database) to be migrated in the target database. After that, the connection is tested, and user is asked to select the target database that will allow migration take place. The target database is Oracle 11g. Further, once the user click on the next button, all database tables that exist in the university database will be retrieved from MySQL database as illustrated in figure 9.

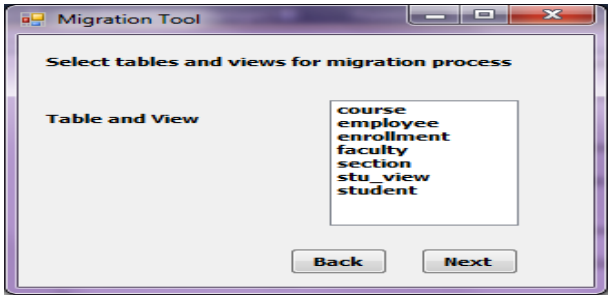

Figure 9 - retrieve all database tables and view

In this step, the user is allowed to select tables and views for data migration process.

b) Retrieves Relational Schema Representation (RSR) of the source relational database. Relational Schema Representation (RSR) displays the details of the university database that including all database tables with their attributes properties: name of attributes, data types, max length, and constraints (primary key and foreign key) as shown in figure 10.

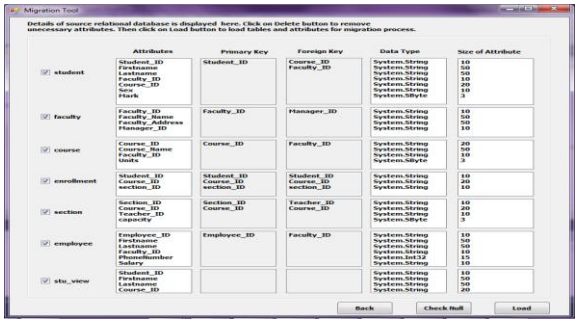

Figure 10 - Retrieve RSR for database tables and views

c) Data cleaning approach is used to detect and remove wrong data, redundancies, inconsistencies and missing values. The prototype focuses on implementing part of data cleaning which is missing values. In this step the user is allowed to check if data source contain null values or not by click on CheckNull button. Then, delete all records that contain null values as shown in figure 11. After that, load the attributes and their properties to be migrated in the Oracle 11g.

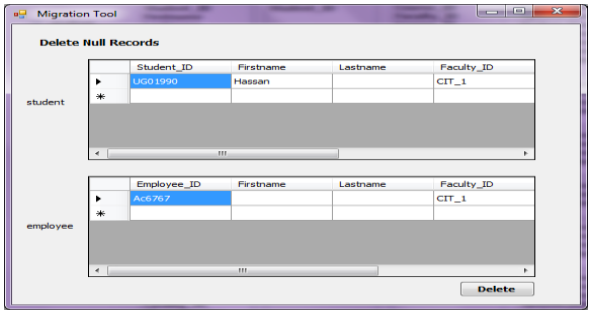

Figure 11 - Delete empty records from source data

d) Load the cleaned and corrected logical structure of the source relational database. Logical structure of the source database (MySQL) is shown in the following figures, in first figure database table's names, their attributes, and the size of attributes are loaded after selected in the previous step for migration. The second figure displays the relationships between database tables.

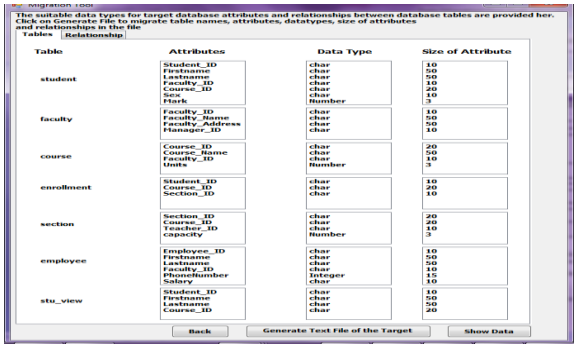

Figure 12- Logical structure of source database

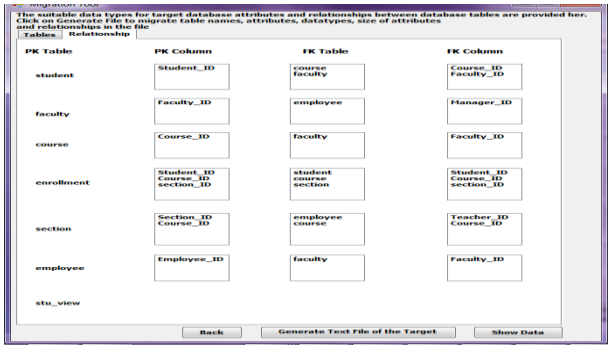

Figure 13 relationship between database tables

e) Provide the suitable data types for the target database tables and their attributes automatically. The following figure shows the suitable data types for the specific attributes in the target database (Oracle 11 g).

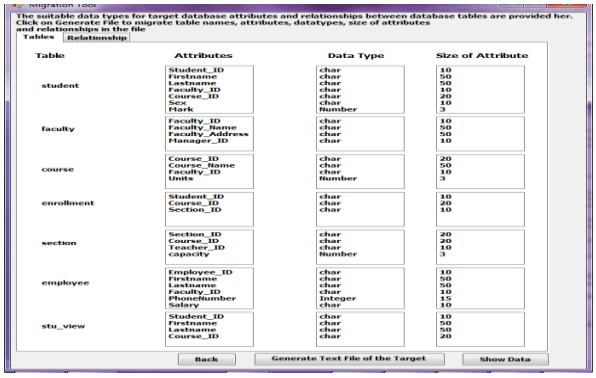

Figure 14 - Suitable data types for Oracle database

f) Creating SQL text file to migrate source data and physical model of the database into the target database. The following figures present the SQL text file, first figure shows data contained in database tables, in second figure there are CREATE commands of database tables, in third figure there are ALTER commands to add primary and foreign keys, and the last figure there are INSERT commands to insert data in the database tables.

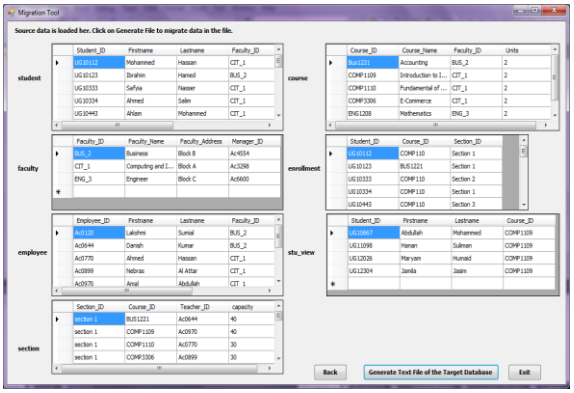

Figure 15 - Data contained in database

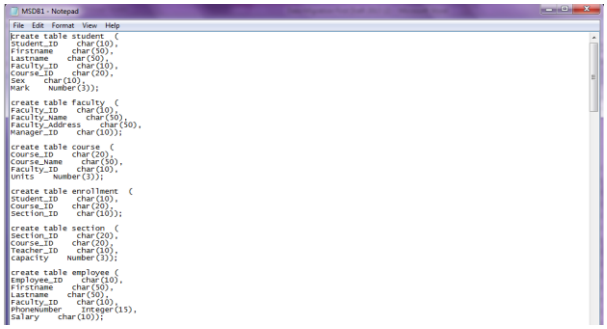

Figure 16- SQL text file contained CREATE commands

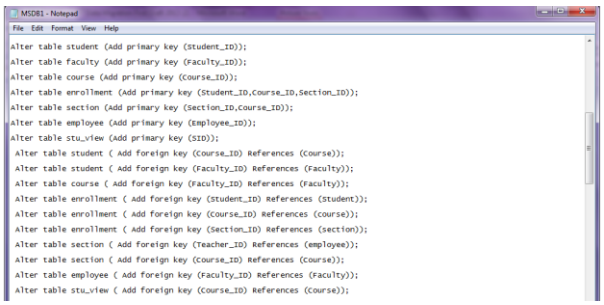

Figure 17 - SQL text file contained ALTER commands

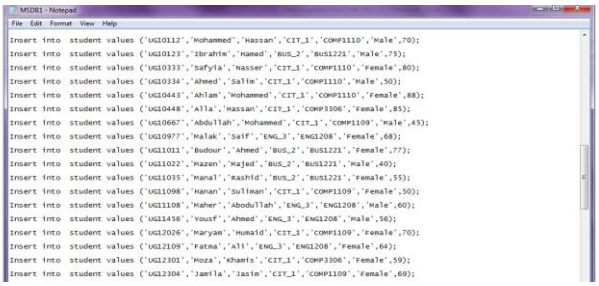

Figure 18 - SQL text file contained ALTER commands

6. CONCLUSION AND FUTURE RESEARCH DIRECTIONS

This paper identifies the disadvantages of current tools for data migration between different types of RDBMS. Then it proposed a framework to improve data quality, which can be affected during data migration process for databases. Therefore, proposed framework opens the door for organizations to cope with technological progress and enhance database performance without any fear about problems of data quality. However, proposed framework can be improved by enhancing its ability to migrate more than one RDBMS, and make process of data cleaning focuses on solving the multi-source problems in order to improve data quality.

7. REFERENCES

[1] Q. Wang, N. Liu and C. Yu, "Research and Practice of university database Migration," 2012 International Symposium on Information Technology in Medicine and Education (ITME), vol. 1, pp. 31- 34, 2012.

[2] F. Matthes, C. Schulz and K. Haller, "Testing & Quality Assurance in Data Migration Projects," in 27th IEEE International Conference on Software Maintenance (ICSM'11), Williamsburg, 2011.

[3] B. Walek and C. Klimes, "A methodology for Data Migration between Different Database Management Systems," International Journal of Computer and Information Engineering, p. 6, 2012.

[4] B. Wei and T. X. Chen, "Criteria for Evaluating General Database Migration Tools," October 2012. [Online]. Available: http://dx.doi.org/10.3768/rtipress.2012.op.0009.1210. [Accessed 25 April 2013].

[5] P. Paygude and P. R. Devale, "Automated Data Validation Testing Tool for Data Migration uality Assurance," International Journal of Modern Engineering Research (IJMER), vol. 3, no. 1, pp. 599-603, February 2013.

[6] B. Walek and C. Klimes, "Expert system for data migration between different database management systems," Advances in Data Networks, Communications, Computers and Materials, p. 6, 2012.

[7] S. A. Vitthal, T. V. Baban, R. Warade and K. Chaudhari, "Data Migration System in Heterogeneous Database," International Journal of Engineering Science and Innovative Technology (IJESIT), vol. 2, no. 2, p. 5, March 2013.

[8] M. Rehman and V. Esichaikul, "Duplicate Record Detection for Database Cleansing," in 2009 Second International Conference on Machine Vision, Dubai, 2009.

[9] L. He, Z. Zhang, Y. Tan and M. Liao, "An Efficient Data Cleaning Algorithm Based on Attributes Selection," 2011 6th International Conference on Computer Sciences and Convergence Information Technology (ICCIT), pp. 375 - 379, december 2011.

[10] J. T. J. and S. V., "Token-based method of blocking records for large data warehouse," Advances in Information Mining, vol. 2, no. 2, pp. 5-10, 2010.

[11] P. Pahwa and R. Chhabra, "Domain Dependent and Independent Data Cleansing Techniques," International Journal of Computer Science and Technology, vol. 2, no. 3, p. 3, September 2011.

[12] A. E. Monge and C. P. Elkan, "An efficient domain-independent algorithm for detecting approximately duplicate database records," 1997.

[13] P. Cong and Z. Xiaoyi, "Research and Design of Interactive Data Transformation and Migration System for Heterogeneous Data Sources," in 2009 WASE International Conference on Information Engineering, 2009.