Excel-Database Converting System using Data Normalization Technique

Rohiza Ahmad[1], Prum Saknakosnak[2], Yew Kwang Hooi[3]

Department of Computer and Information Sciences, Universiti Teknologi PETRONAS
Bandar Seri Iskandar, 31750 Tronoh, Perak, Malaysia
{[1]rohiza_ahmad, [3]yewkwanghooi}@petronas.com.my, [2]nakprum@gmail.com

Abstract. This paper is intended to present an application system which can facilitate users in converting from Excel spreadsheet to database structure by using data normalization technique. It provides users with an automated system that can develop a database in its third normal form (3NF) from an Excel spreadsheet. There are two major problems of Excel spreadsheet that are emphasized in this project which are data redundancy and update anomaly. Prototyping-based methodology is used in order to develop this system. The system just creates the SQL codes for creating the database structure and populating the database; however, how the users make use of it such as interface for database interaction is not covered at this stage of project. Based on its functionalities, accuracy and performance, this system is able to bring more benefit and convenience for its users.

Keywords. SQL commands, Excel spreadsheet, Data normalization, Normal Forms, Functional dependency

1 Introduction

Data and information are important assets and valuable resources for individuals, companies, businesses and organizations. They can be used to help run a business as well as to make good decisions in achieving the goal of the business or organization. Therefore, proper data management is really needed.

There are a number of data management software that people always use to store their data and information. Among all of them, for certain reasons, Excel spreadsheet has been considered by many to be one of the most familiar software for users to key in their data. However, using Excel spreadsheet to store data and information has certain drawbacks. Among them, the spreadsheet will become unmanageable when the amount of data stored is large. Even more important than that are the problems of data redundancy and update anomaly.

Data redundancy refers to having information that is repeated in two or more places. In Excel, the same data will need to be entered in several different places if the data is needed there. As for update anomaly, in database terminology, it refers to problems which can occur when insertion, modification or deletion is done to the data

T. Herawan et al. (eds.), *Proceedings of the First International Conference on Advanced Data and Information Engineering (DaEng-2013)*, Lecture Notes in Electrical Engineering 285, DOI: 10.1007/978-981-4585-18-7_3,
© Springer Science+Business Media Singapore 2014

[1]. For example, due to the existence of multiple copies of the same data, any modification to one of the copies may lead to data inconsistency since some of them might be missed out during the update process.

Therefore, this research project intends to help Excel spreadsheet users in converting their stored data into a third normal form (3NF) database. According to Connolly and Begg [1], a 3NF database is minimal in redundancy and free from update anomaly. Hence, there will be two objectives to be achieved by the project which are to construct algorithms for automating the manual process of database normalization starting from First Normal Form (1NF) up until 3NF and also to develop an application system that can facilitate in converting an Excel spreadsheet to a database structure using the data normalization algorithms developed. As for the scope of the project, the output of the application system will only be SQL commands for the users to create and populate the database. How the users make use of the codes such as executing the codes in certain DBMS or querying the data once the database has been created is not covered at this stage of the project.

The remaining of this paper will present some related works in Section 2, the methodology that has been opted to carry out this project work in Section 3, some results from the project in Section 4 and conclusions in Section 5.

2 Related Work

Due to the importance of database normalization in refining and ensuring high quality of relational database, many research works have been conducted in this area. Most of these research works were focusing on approaches to find functional dependencies, which are the bases for normalization steps, among attributes. For example, Yao and Hamilton [2] proposed an efficient rule discovery algorithm called FD_Mine for mining functional dependency from data. FD_Mine searches for functional dependencies by using equivalence and nontrivial closure. Besides, identifying the problem of discovering functional dependencies in a relation, the authors have also provided some results obtained from a series of experiments conducted on FD_Mine. Using both synthetic and real data, the experiments have proven the effectiveness of FD_Mine.

Apart from the above, it was mentioned by Beaubouef et al. [3] that rough relational database model can be developed in order to manage the uncertainty in relational database. In their work, they concentrated on rough functional dependencies when conducting the normalization process. In other words, normal forms are constructed based on roughly identified functional dependencies.

Chen et al. [4] on the other hand, demonstrated the deficiencies of normal form definitions based on functional dependency. According to them, the traditional way of solely defining normal forms based on functional dependency is not effective enough in removing common data redundancies and data anomalies. Instead, normalization process can yield better database designs with the addition of functional independency. For that, the authors have also introduced a new normal form concept which is based on functional independency. The new normal form was shown to be able to deliver additional criteria to further remove data redundancies and data anomalies.

Regarding algorithms to automate the process of data normalization, few works can also be found in the literature. Among them is the work done by Bahmani et al. [5]. The work presents a new complete automatic relational database normalization method which covers the data normalization technique up to Boyce-Codd Normal Form (BCNF). The approach uses dependency matrix and directed graph matrix to represent dependencies.

Much similar to the above approach is the work of Verma [6]. Verma has conducted a comparative study between manual and automatic normalization techniques using sequential as well as parallel algorithms. The technique for automatic normalization used was also depended upon the generation of dependency matrix, directed graph matrix, as well as determinant key transitive dependency matrix. From the study, it was found that the main benefit of the automatic approach as compared to the traditional one is the primary key. The key is automatically identified for each final table generated.

Another approach of normalizing relational database schemas is by using Methematica modules [7]. According to Yazici and Karakaya [7], Methematica provides a straightforward platform as compared to Prolog based tools which require complex data structures. The Methematica modules are put together into a window based system called JMath-Norm so that it can provide interaction between the user and Methematica. Apart from this, the interactive tool of the prototype is also a handy tool for teaching normalization theory in database management course.

3 Methodology

This project has adopted Prototyping-based methodology for the development of algorithms and the application system. As shown in Fig. 1, the methodology covers planning until system stages. Analysis, design and implementation phases are repeatedly executed until the prototype has been completed with enough functionalities. Once completed, the prototype can be implemented as a working system.

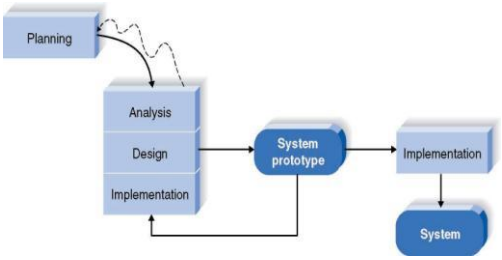

Fig. 1. Prototyping Development Methodology [8]

In this project, for the development of the prototype, the hardware used was a notebook, i.e., ACER Aspire 4736, and the software were Netbeans IDE 7.3 and Wamp-

Server 2.2D which includes Apache 2.2.21, MySQL 5.5.20 and phpMyAdmin 3.4.10.1. As for the programming languages, Java and SQL (MySQL) were chosen.

4 Results and Discussions

4.1 Data Gathering - Interview

Before initiating this project, internal interviews had been conducted to five staffs from the organization the authors are in. The staffs were from different departments and they use Microsoft Excel Spreadsheet in their everyday work. The main purpose for the interview was to get information about the experience of the real Microsoft Excel spreadsheet's users in doing their tasks in relation to the problem statement of the project. From the interview feedbacks, it can be concluded that the users did not have much knowledge about database. They could not differentiate between database and normal data storage such as Excel spreadsheet. From their points of view, as long as the software can be used to store certain amount of data, then it will become their database. However, regarding data redundancy and update anomaly, they did agree that these did occur in their data storage and a solution which can help them overcome the problems will be much welcomed.

4.2 System Flow

Fig. 2 shows the flow of operations for the system which has been developed. First, users need to import an Excel spreadsheet file into the system. The content of the spreadsheet will be displayed on the interface of the system. Next, the users will be guided to identify one functional dependency at a time. Each functional dependency identified will invoke one of the various forms of data normalization including 1NF, 2NF and 3NF. For easier understanding, 1NF refers to database table which has single data in all of its cells. As for 2NF, it refers to database tables which have no partial dependencies. In other words, the whole primary key is needed in order to tell the content of non-key attributes. 3NF on the other hand, refers to database tables which have no transitive dependencies. A 3NF table's non-key attributes can be identified solely based on the table's key, not from other non-key attribute. Upon completion of the normalization process, the system will create SQL codes according to the final data of tables from normalization. After that, the users just need to run the SQL codes in any DBMS selected. In this project, phpMyAdmin was used to show example of how to create database by importing the SQL file which has been created by the system.

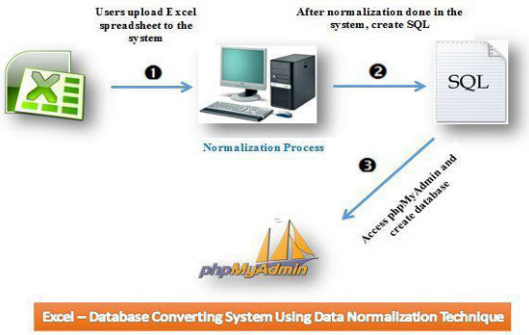

Fig. 2. System Flow for System

4.3 Normalization Algorithms

Based on the flow of the system, the normalization process basically begins with the submission of the Excel data to the conversion system. Hence, at this point of time, the data is not in any normal form or in other words, it is in the zero normal form (0NF). In order to normalize the data two steps have been taken: normalizing 0NF to 1NF and normalizing 1NF to 2NF and 3NF.

Zero Normal Form to First Normal Form - For this step, two functions were coded. The first is called generateInput(). The function returns an array in Java containing the data from the imported Excel file. The second, the outputArray() function, is used to generate array of 1NF by removing repeating groups from the original array. Fig. 3 shows the algorithms for both functions.

```
generateInput():
Begin
  SourceArray = data imported from Excel file
  Return SourceArray
End
outputArray():
Begin
  Generate two arrays namely tempArray and firstArray
  For all the entries in a row
    If any entry is not empty then
      tempArray = firstArray
    Else
      Fill with the nearest upper row of the same column
    Endif
Copy each row of tempArray into firstArray
Return firstArray
End
```

Fig. 3. Algorithms for generateInput() and outputArray()

From 1NF to 2NF and 3NF - In this part, the firstArray will become the main input in order to process to 2NF and 3NF. Two functions are used at this stage which are arrangeCompositeTable() and eliminateRedundancy(). The algorithm for the first function is given in Fig. 4 and the working of the second function is presented in Fig. 5. Basically, the first function is used to select the whole data columns from firstArray which have been selected by users as representing a functional dependency.

```
arrangeCompositeTable():
Begin
  If length of key is less or equal to zero
  Then return null
  End if
  For all the entries of firstArray
     Select the entries whose index is the same as the index
     of key and dependency selected by user
     Store these entries in output array
  End loop
  Return output
End
```

Fig. 4. Algorithm for arrangeCompositeTable()

As for the second function, it is used to eliminate data redundancy by generating a matrix and removing repeated data in each table set. As shown below, first, the same set of data result from the arrangeCompositeTable() will be compared. (Note: A to F are data, e.g., A is a course code CSC101, B is a lecturer named John, etc.)

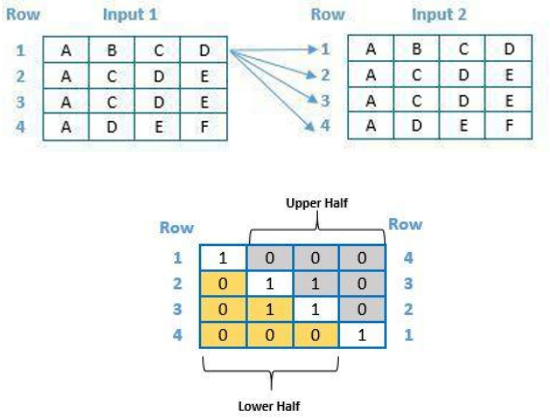

Fig. 5. Working of eliminateRedundancy()

Each row of input1 is compared to each row of input2 row by row. If the rows do not contain similar data, 0 will be placed in the resulting matrix underneath; otherwise, it will be 1. For example, in the above diagram, row 1 of input1 contains ABCD, while

row 2 of input2 contains ACDE. Comparing the two will result in 0 being placed in cell "row 2, column 1"of the resulting matrix. After the resulting matrix has been fully filled up, the next step will be removing the rows in the matrix which appear to have number 1 (excluding the diagonal since this algorithm is to compare the same set of data. By default, the diagonal of the matrix will appear as number 1). There are two ways of removing the row either though upper half of the diagonal or lower half of the diagonal. Removing through the upper half will make the last row of the matrix as row number 1 in relation to data from arrangeCompositeTable(). On the other hand, removing through the lower half will end up with the first row being row number 1 in relation to data from arrangeCompositeTable(). However, both methods work accurately the same.

4.4 User Interface

For the convenient of users, an interface was developed for the system. The interface has three main functions for the users to interact with. The functions are for importing the Excel file, normalizing the data and creating SQL code. In case some users may have limited knowledge in database, the system also provides a HELP button which displays various steps for users to follow when identifying functional dependency.

4.5 Testing

The developed system was tested for validation. Regarding load testing, the system was tested with both versions of Excel 2003 and 2007 with different increasing number of columns and rows. According to the result, it shows that regardless of number of columns and rows of the Excel table, the system is still capable of performing the intended functionalities effectively. However, in terms of efficiency, slight increase in processing speed was observed when the number of rows was increased. As for usability testing, three testers with varying knowledge of database were asked to evaluate the system. They were found to be satisfied with the system's ability to maintain the consistency between the data stored in the Excel file in both versions and the data read to the application. According to them, overall, all functions are very good and processing of data is very fast.

In order to enhance the system, they also gave some suggestions for improvement such as the system should be able to determine functional dependency without input from users as well as to import Excel files with many sheets. Based on the feedbacks, future plan will be to improve the system if feasible.

5 Conclusion

In conclusion, Excel – Database Converting System Using Data Normalization Technique has been developed successfully as a prototype that covers the scopes which have been determined for the project. Basically, the prototype is able to import and process both versions of Excel file – 2003 and 2007 version into the system. It is also

able to automatically normalize the data up to 3NF by minimizing data redundancy and update anomaly. The system is also capable of creating SQL file which will be used to create database in phpMyAdmin.

As for future enhancement, even though the system has been completely developed according to its scope and objectives, there are also certain additional features that can be enhanced. One of them is the identification of functional dependency which is currently identified by the users manually. Even though a HELP button is provided to guide the users, there are still possibilities that the users will not be able to identify the correct functional dependency based on the rule of normalization. Another possible enhancement is in terms of the interface for database interaction. The interface will be more useful and convenient for the users if it has update, edit and delete functionalities added. And lastly, future enhancement can also be done to the normalization process. Higher Normal Form such as BCNF, Forth Normal Form (4NF) and Fifth Normal Form (5NF) can be implemented to see if the solution can further reduce redundancy and update anomaly.

Acknowledgments. The authors would like to extend our gratitude and appreciation to Mr. Pan Sophoana for his strong support and technical assistance throughout the project.

References

1. Connolly, T., Begg, C.: Database Systems – A Practical Approach to Design, Implementation, and Management, 5th ed. Pearson Addison-Wesley, MA, USA (2010)
2. Yao, H., Hamilton, H.: Mining Functional Dependencies From Data. J. Data Mining and Knowledge Discovery, vol. 16, no. 2, pp. 197-219. Kluwer Academic Publishers Hingham, MA, USA (2008)
3. Beaubouef, T., Petry, F., Ladner, R.: Normalization in a Rough Relational Database. In: Slezak, D. et al. (eds.) RSFDGrC 2005. LNAI, vol. 3641, pp. 275-282. Springer, Heidelberg (2005)
4. Chen, T., Liu, S., Meyer, M., Gotterbarn, D.: An Introduction to Functional Independency in Relational Database Normalization. In: ACM-SE Proceedings of the 45th annual southeast regional conference, pp. 221-225. ACM Press, New York (2007).
5. Bahmani, A. H., Naghibzadeh, M., Bahmani, B.: Automatic Database Normalization and Primary Key Generation. In: Proceedings of Canadian Conference on Electrical and Computer Engineering 2008 (CCECE 2008), pp.11-16. IEEE Press, New York (2008)
6. Verma, S.: Comparing Manual and Automatic Normalization Techniques for Relational Database. International Journal of Research in Engineering & Applied Sciences (IJREAS), vol. 2, no. 2, pp. 59-67 (2012)
7. Yazici, A., Karakaya, Z.: Normalizing Relational Database Schemas Using Mathematica. In: V.N. Alexandrov et al. (eds.) ICCS 2006. Part II, LNCS, vol. 3992, pp. 375-382. Springer, Heidelberg (2006)
8. Dennis, A., Wixom, B., Tegarden, D.: Systems Analysis and Design with UML Version 2.0.- An Object-Oriented Approach, 2nd Ed. John Wiley & Sons, USA (2005).