Recovery of Data Dependencies

Hee Beng Kuan Tan
Nanyang Technological University, Singapore

Yuan Zhao
Nanyang Technological University, Singapore

INTRODUCTION

Today, many companies have to deal with problems in maintaining legacy database applications, which were developed on old database technology. These applications are getting harder and harder to maintain. Reengineering is an important means to address the problems and to upgrade the applications to newer technology (Hainaut, Englebert, Henrard, Hick, J.-M., & Roland, 1995). However, much of the design of legacy databases including data dependencies is buried in the transactions, which update the databases. They are not explicitly stated anywhere else. The recovery of data dependencies designed from transactions is essential to both the reengineering of database applications and frequently encountered maintenance tasks. Without an automated approach, the recovery is difficult and time-consuming. This issue is important in data mining, which entails mining the relationships between data from program source codes. However, until recently, no such approach was proposed in the literature.

Recently, Hee Beng Kuan Tan proposed an approach based on program path patterns identified in transactions for the implementation of the most commonly used methods to enforce each common data dependency. The approach is feasible for automation to infer data dependencies designed from the identification of these patterns through program analysis (Muchnick & Jones, 1981; Wilhelm & Maurer, 1995).

BACKGROUND

Data dependencies play an important role in database design (Maiser, 1986; Piatetsky-Shapiro & Frawley, 1991). Many legacy database applications were developed on old generation database management systems and conventional file systems. As a result, most of the data dependencies in legacy databases are not enforced in the database management systems. As such, they are not explicitly defined in database schema and are enforced in the transactions, which update the databases. Finding out the data dependencies designed manually during the maintenance and reengineering of database applications is very difficult and time-consuming. In software engineering, program analysis has long been developed and proven as a useful aid in many areas. This article reports the research on the use of program analysis for the recovery of common data dependencies, that is, functional dependencies, key constraints, inclusion dependencies, referential constraints, and sum dependencies, designed in a database from the behavior of transactions.

RECOVERY OF DATA DEPENDENCIES FROM PROGRAM SOURCE CODES

Tan (Tan & Zhao, 2004) has presented a novel approach for the inference of functional dependencies, key constraints, inclusion dependencies, referential constraints, and sum dependencies designed in a database from the analysis of the source codes of the transactions, which update the database. The approach is based on the program path patterns for implementing the most commonly used methods for enforcing data dependencies. We believe that the approach should be able to recover majority of data dependencies designed in database applications. A prototype system has been implemented for the proposed approach in UNIX by using Lex and Yacc.

Many of the world's database applications are built on old generation DBMSs. Due to the nature of system development, many data dependencies are not discovered in the initial system development; they are only discovered during the system maintenance stage. Although keys can be used to implement functional dependencies in old generation DBMSs, due to the effort in restructuring databases during the system maintenance stage, many of these dependencies are not defined explicitly as keys in the databases. They are enforced in transactions. Most of the conventional files and relational databases allow only the definition of one key. As such, most of the candidate keys are enforced in transactions. The feature for implementing inclusion dependencies and referential constraints in a database is only available in some of the latest generations of DBMSs.

As a result, most of the inclusion dependencies and referential constraints in legacy databases are also not defined explicitly in the databases and are enforced in transactions. To avoid repeated retrieval of related records for the computation of a total in query and reporting programs, the required total is usually maintained and stored by some transactions that update the database such that other programs can retrieve them directly from the database. As such, many sum dependencies are maintained by transactions in database applications. In summary, much of the functional dependencies, key constraints, inclusion dependencies, referential constraints, and sum dependencies in existing database applications are enforced in transactions. Therefore, transactions are the only source that can accurately reflect them. The proposed approach can be used to automatically recover these data dependencies designed in database applications during the reverse engineering and system maintenance stages. These dependencies constitute the majority of data dependencies in database applications.

In the case that data dependencies are jointly enforced from schema, transactions, and their GUI (graphical user interface) definitions, the approach is still applicable. The data dependencies defined explicitly in database schema can be found from the schema without much effort. The GUI definition for a transaction can be interpreted as part of the transaction and analysed to recover data dependencies designed. All the recovered data dependencies designed form the design of data dependencies for the whole database application.

Extensive works have been carried out in database integrity constraints that include data dependencies. However, these works mainly concern enforcing integrity constraints separately in a data management system (Blakeley, Coburn, & Larson,1989; Orman, 1998; Sheard & Stemple, 1989) and the discovery of data dependencies hold in the current database (Agrawal, Imielinski, & Swami, 1993; Andersson, 1994; Anwar, Beck, & Navathe, 1992; Kantola, Mannila, Raiha, & Siirtola, 1992; Petit, Kouloumdjian, Boulicaut, & Toumani,1994; Piatatsky-Shapiro & Frawley, 1991; Signore, Loffredo, Gregori, & Cima, 1994; Tsur, 1990). No direct relationship exists between the former work and the proposed approach. The distinct difference between Tan's work and the latter work is that the proposed approach recovers data dependencies designed in a database, whereas the latter work discovers data dependencies hold in the current database. A data dependency that is designed in a database may not hold in the current database, due to the update by the transactions that were developed wrongly during the earlier stage, or to the update by the query utility without any validation.

FUTURE TRENDS

We believe that the integrated analysis of information in databases and programs will be a fruitful direction for establishing useful techniques in order to verify the quality and accuracy of database applications. The information can comprise both formally and empirically based characteristics of programs and databases.

CONCLUSION

We have presented a novel approach for the recovery of data dependencies in databases from program source codes. The proposed approach establishes a bridge for integrating information in databases and the source codes of programs that update the databases. As a final remark, we would like to highlight that as far as we can identify common methods for enforcing an integrity constraint and the resulting program path patterns for these method, a similar approach can be developed to recover the integrity constraint designed in a database. This research could be interesting to explore further.

REFERENCES

Agrawal, R., Imielinski, T., & Swami A. (1993). Database mining: A performance perspective. *IEEE Transactions on Knowledge and Data Engineering, 5*(6), 914-925.

Andersson, M. (1994). Extracting an entity relationship schema from a relational database through reverse engineering. *Proceedings of the 13th International Conference on ERA* (pp. 403-419).

Anwar, T. M., Beck, H. W., & Navathe, S. B. (1992). Knowledge mining by imprecise querying: A classification-based approach. *Proceedings of the IEEE Eighth International Conference on Data Engineering,* USA.

Blakeley, J. A., Coburn, N., & Larson, P. (1989). Updating derived relations: Detecting irrelevant and autonomously computable updates. *ACM Transaction on Database Systems, 14*(3), 369-400.

Hainaut, J.-L., Englebert, V., Henrard, J., Hick, J.-M., & Roland, D. (1995). Requirements for information system reverse engineering support. *Proceedings of the IEEE Working Conference on Reverse Engineering* (pp. 136-145).

Kantola, M., Mannila, H., Raiha, K., & Siirtola, H. (1992). Discovery functional and inclusion dependencies in rela-