ISSN XXXX XXXX © 2017 IJESC

Research Article Volume 7 Issue No.11

A Perspective Survey on Software and Database Reverse Engineering Methodologies

J.M. Dhayashankar[1], Dr. A.V. Ramani[2]
Assistant Professor[1], Associate Professor & HOD[2]
Department of Computer Applications[1], Department of Computer Science[2]
Sri Ramakrishna Mission Vidyalaya College of Arts and Science, Coimbatore, India

Abstract:
This paper focuses on maintenance of Legacy databases which are obviously valuable assets to many organizations. One solution to modernize the legacy databases is to migrate and transform their structures and corresponding contents to the new systems. Maintaining a legacy database is a difficult task especially when system documentation is poor written or even missing. To handle the legacy database maintenance efficiently the database reverse engineering methodology plays a key role. The problem of choosing a method for the reverse engineering of relational database systems is not trivial. Methods have different input requirements and each legacy system has its particular characteristics that restrict information availability. Hence there is few number of research works done in the field of software reverse engineering. This study analysis the existing techniques in the selection of reverse engineering of relational databases.

Keywords: legacy, reverse engineering, database, relational database, maintenance

1.1. INTRODUCTION

Information systems are accepted now as a source of competitive advantage for organizations. They have to incorporate fast changes resulting from the evolutions that characterize enterprises today.

Generally, in large organizations, the amount of saved and manipulated data is becoming more and more important; the information manipulated by automated procedures, have been designed and structured by several generations of analysts and database administrators.

As a consequence, there is a situation where a lot of redundant information is identified, undeclared dependencies and incoherent sources [4]. Giving order to data becomes a necessity. Instead of designing a new system from the beginning, a reverse engineering process allows to design a new database schema by using the existing system and the old conception efforts.

Legacy systems are old software systems that are crucial to the operation of a business. These systems are expected to have undergone changes in their lifetime due to changes in requirements, business conditions and technology. It is quite likely that such changes were made without proper regard to software engineering principles.

The result is often a deteriorated structure, which is unstable but cannot be discarded because it is costly to do so. Moreover, another reason for retaining these legacy systems is that they embed business knowledge which is not documented elsewhere.

Since it is often not feasible to discard a system and develop a new one, techniques must be employed to improve the structure of the existing system. An effective strategy for change must be devised; re-engineering is one such strategy. Re-engineering is a process that transforms or re-implements legacy systems to make them more maintainable.

The re-engineering option should be chosen when system quality has been degraded by regular change, but change is still required i.e. the system under consideration has low quality but a high business value, and the re-engineering effort is less risky and less costly than system replacement.

Reverse engineering of software refers to the process of discovering source code and system design from the available software.

Typically, database reverse engineering is the process of extracting design specifications from legacy systems and making the reverse transformation from logical to conceptual schema.

The evolution of databases has always been an active research area as indicated by the quantity of studies done in the past. Database reverse engineering (DRE) has become an important research field these last few years [1,2] even though the pioneering work started at the very beginning of the 83s [3].

Over the years, the research and publications in DRE by both communities has been mainly in three areas as illustrated in Figure 1.

• DRE translation and methodologies algorithms

• DRE tools

• The DRE of specific applications and experiences in DRE

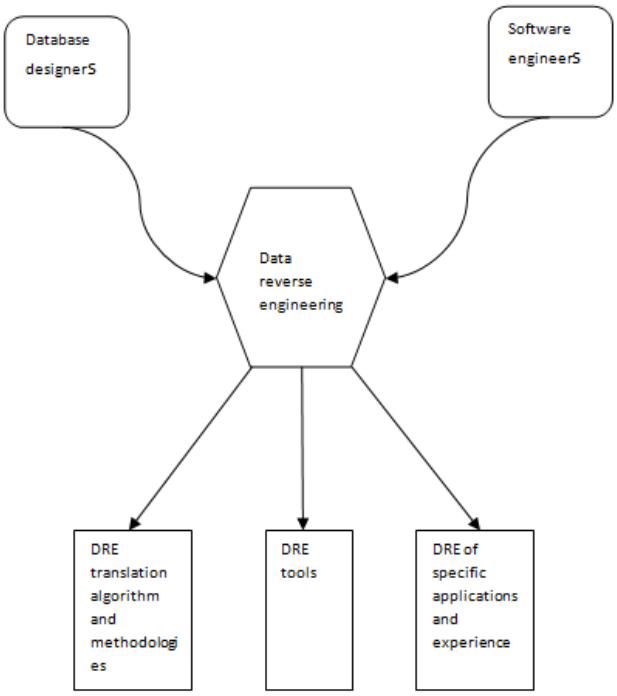

Figure.1. Data Reverse Engineering Publications come from Database Designers and Software Engineers

Various reasons can motivate a Database Reverse Engineering Process (DREP) and imply various approaches. It can be necessary for moving the database implementation from a Database Management System (DBMS) to another one. More frequently, it consists in modifying the underlying logical data model, for instance changing a hierarchical database into a relational database or a relational database into an object oriented database. For example, most molecular biology database are implemented in relational DBMSs. But because relational databases are limited in their modeling capabilities of sophisticated scientific structures, maintenance tasks are tedious. DRE methods tend to design specifications by understanding existing database semantics. A DRE aims at producing a new description of the stored data by studying the database implemented in a specific DBMS. Various information sources can be used. Our strategy is based on the hypothesis that essential semantic knowledge resides in queries expressed by expert users on the original database. They propose to enhance classical DRE methods by taking into account the semantics about data and structures which is implicitly included in these queries. This semantics is considered as knowledge and extracted by a data mining tool. The re-engineering effort starts with gaining an understanding of the software system, a process known as reverse engineering. Gaining system understanding is difficult because documentation for the system is often not available and source code files are the only means of information regarding the system. The understanding gained allows suggestions for making subsequent changes and optimizations to the source code for better maintainability. In recent years, there has been growing interest in the application of data mining techniques to gain better understanding of software systems. Researchers have applied data mining techniques in different contexts e.g. to recover the architecture of software legacy systems [5] - [7], to discover patterns for re-using software library components [8], [9], and to support software system maintenance [10], [11].

2. NEED OF DATA REVERSE ENGINEERING

The need for knowledge acquisition in the system development process is prevalent during the reverse engineering of legacy systems. As it is known that, legacy systems are usually poorly documented jumbles of unintelligible code that really must be understood in order to proceed correctly with new systems development. As a part of the reverse engineering process the data, be it flat files or a database, must be mastered. DRE can greatly assist in this process no matter the reason for the knowledge acquisition whether it be system maintenance, reengineering, extension, migration, or integration. Another nice thing that DRE assists us in performing is quality assessment. It is possible to use the techniques involved to determine the quality of vendor databases [34]. DRE can also assist us in converting from any current data environment (flat files, databases, etc.) to new database technology. This is desperately needed with the rapid changes in database technology. The last two feature of the list of reasons to perform DRE, allow companies to correctly and efficiently manage their data. While using the data more and more as information, there is a need to efficiently manage them and must be able to perform data administration and component reuse easily and pragmatically.
DRE was very prevalent during the Y2K work that was done at the end of the millennium. Currently, DRE is assisting in various areas

- evaluation of packages [34]
- test planning [35]

3. LITERATURE SURVEY ON SOFTWARE AND DATABASE REVERSE ENGINEERING

Maqbool et al [12] explored the use of data mining for software reverse engineering i.e. given the source files of a software system; they use association rule mining algorithms and tools to gain insight about the software. Most traditional reverse engineering tools focus on abstraction and analysis of source code, presenting a visual representation of the software architecture. This approach can be both helpful and cost effective in software maintenance tasks. However, where large software teams are concerned, with moderate levels of employee turnover, traditional reverse engineering tools can be inadequate. To address this issue, Xiaomin et al [13] examined the use of software process data, such as software artifact change history and developer activities. The application of this data confers additional information developers need to better understand, maintain and develop software in large team settings. The knowledge based approach used in [14, 15] requires, as *input*, the data instances and relation-schemes, including primary keys. Optionally, the user may insert some inclusion dependencies. The *assumptions* made are 3NF, consistent naming of attributes and no error on key attributes values. The *output* given is an Extended-Entity-Relationship (EER) model. The *methodology* has three major steps: (i) classification of relations and attributes, based on the relation-schemas and its primary keys, (ii) generation and verification, against data instances, of inclusion dependencies using heuristics based on the classified relations and attributes; (iii) identification of EER components using a list of rules. The *main contributions* of this work are the generation of inclusion dependencies and the complete justification of the

transformations applied to the existing database to produce the resulting EER model. The detailed description of the method in knowledge based reverse engineering in relational databases [14] highly contributes to consider its application in actual DBRE cases. It should be noted that, in each step, the cases in which human input is required are clearly identified for constructing reverse engineering using EER model [15]. The concept of transforming relational schemas into conceptual schemas [16] requires as *input*, the relation-schemes, functional dependencies and inclusion dependencies. Relations are *assumed* to be in 3NF. *The output* is a conceptual schema described as a pair containing a language L and a set IC of typing, mapping and generalization constraints. The *methodology* has four steps: (i) splitting of relation schemes that correspond to more than one object; (ii) addition of extra relation schemes to handle the occurrence of certain types of inclusion dependencies; (iii) collapsing of the schemes of the relations that correspond to the same object type; (iv) application of the schema mapping that converts a relational schema into a conceptual schema. This method is based on the well-established concepts of relational database theory. It is very complete, in terms of the description of the reverse engineering steps, but with the drawback of needing all keys and inclusion dependencies. Note also that at beginning of step (iv), when the mapping is ready to be done, the input information is already "clean and neat". So the mapping process is simple and automatic, with each relation scheme giving rise to an object type. The extended entity relationship object structure approach proposed in [17] requires, as *input*, the relation schemes, key dependencies and key-based inclusion dependencies, i.e. referential integrity constraints. Relations are *assumed* to be in Boyce-Codd normal form (BCNF), considering only the "relevant" functional dependencies (i.e. allowing hidden-objects). So, in our classification, this method is classified as not assuming 3NF. The *output* is an EER model. The *methodology* has five steps: (i) transformation of relational schemas into a form appropriate for identifying EER object structures; (ii) examination of the relation-schemes, functional dependencies and inclusion dependencies obtained after the transformations in order to detect whether they satisfy a set of properties; (iii) determination of the type of object-interaction for each inclusion dependency; (iv) derivation of a candidate EER schema using some mapping rules; (v) examination of the quality of the generated EER schema. The approach is very demanding on the input. In spite of not requiring that relation-schemes represent single-object sensor names consistency, it requires all key functional dependencies and key-based inclusion dependencies. Dependencies are not even a common type of input. They consider that the *main contribution* of this work is the formalization of the mappings between schemas. The entity relationship approach for designing conceptual database in [18] only requires as *input* the relation schemes, although with several *assumptions*: 3NF or BCNF relations are required; coherency on attribute names: there are no ambiguities in foreign keys and there are no homonyms; and all candidate keys must be specified. The *output* is an EER model and the used *methodology* has the following steps: (i) relations are processed and classified, with human intervention, so that assumptions are satisfied; (ii) the classified relations are mapped based on its classification and the key attributes; (iv) the special cases of non-classified relations are handled on a case-by-case basis. The method's goal is to resolve the most common situations rather than to claim exhaustiveness. This method is driven by the detection of relationships, which are discovered by observing the primary and candidate key attributes, referential constraints and inclusion dependencies. The method reveals the drawback of requiring, earlier in the process, semantic input. The dependency based approach in [19] requires, as *input*, the relation schemes, with unique and not null constraints, data instances and code. The *assumptions* identified in our framework are not mandatory. The *output* is an EER model. The *methodology* followed (i) proposes inclusion dependencies using relation-schemes, database instances and equip-join queries from the code; (ii) proposes relevant non-key functional dependencies using relation-schemes, candidate keys, and the inclusion dependencies not rejected by the user; (iii) then it uses a decomposition normalization algorithm to obtain a 3NF schema; (iv) the mapping from the relational schema into an EER one is done. To be noted that, among the analyzed methods, this is the only one that includes the normalization of the relational schema. However, hidden objects, by option of the user or not revealed by equijoins, may persist. Object oriented modeling was used in [20, 21] requires, as *input*, the relation schemes and data. The *assumptions* identified in our framework are not mandatory. The *output* is an Object Modeling Technique (OMT) model. The *methodology*, has the following steps: (i) prepares an initial object model representing each relation as a tentative class; then (ii) the user should look for candidate keys using some clues described; then (iii) the user should determine foreign key groups using, again, some clues described; (iv) the refinement of the OMT schema is progressively done by the user based on given guidelines that include querying data. This method is characterized by allowing only the automation of small steps, due to the high level of human input required. It is intended to process some tricky representations, providing guidelines for coping with design optimizations and unfortunate implementation decisions. This method gives several clues on how/where the user should look for the types of information needed. The relational schema abstracted using procedural patterns was developed in [22] requires, as *input*, the relation-schemes and code. The *assumptions* identified in our framework are not mandatory. The *output* is an ER model. The *methodology* has three phases: (i) identification of primary keys: at the end of this phase, each relation must have a primary key or, at least, a hypothesis for the primary key and possible candidate keys; (ii) detection of indicators of synonyms and referential key constraints using SQL instructions in the code; (iii) conceptualization: using the four types of indicators found - schema, primary key, SQL and procedural indicators - the conceptual model is derived. Notice that this method is based on clues. The clues are adopted to cope with unusual implementation techniques, optimization choices, poor Data Definition Language and code errors, among others. Since the introduction of a famous association technique known as Apriori algorithm and its variants are used in [23,24], there have long been immense attempts to integrate this technique to improve database design, consistency checking, and querying. Han et al. [25] improved the DB Miner system to work with relational databases and data warehouses. DB Miner can do many data mining tasks such as classification, prediction and association. Sreenath et al [26] adopted Apriori algorithm to work with relational database system. They created Fast Update algorithm to search association data when the system has new transaction.

Tsechansky et al [27] applied Apriori to find association data from many relations in the database. Berzalet al [28] used Tree-Based Association Rule mining (TBAR) to find association data in relational database. They kept large item set in tree structure format to reduce time cost in association process. Hipp, Güntzer and Grimmer [29] implemented Apriori algorithm with C++ programming language to work on DB2 database system. They used the program to find association data in Daimler-Chrysler Company database. In parallel to the attempts of applying learning techniques to existing large databases, researchers in the area of database reverse engineering have proposed some means of extracting conceptual schema. Lee and Yoo [30] proposed a method to derive a conceptual model from object-oriented databases. The derivation process is based on forms including business forms and forms for database interaction in the user interface. The final products of their method are the object model and the scenario diagram describing a sequence of operations. The work of Perez et al. [31] emphasized on relational object-oriented conceptual schema extraction. Their reverse engineering technique is based on a formal method of term rewriting. They use terms to represent relational and object-oriented schemas. Term rewriting rules are then generated to represent the correspondences between relational and object-oriented elements. Output of the system is the source code to migrate legacy database to the new system. Recent work in database reverse engineering has not concentrated on a broad objective of system migration. Researchers rather focus their study on a particular issue of semantic understanding. Lammari et al. [32] proposed a reverse engineering method to discover inter-relational constraints and inheritances embedded in a relational database. Chen et al. [33] also based their study on entity relationship model. They proposed to apply association rule mining to discover new concepts leading to a proper design of relational database schema. They employed the concept of fuzziness to deal with uncertainty inherited with the association mining process. Our work is also in the line of association mining technique application to the database design. But our main purpose is for the understanding of legacy databases and our method deals with uncertainty by means of heuristic in the step of rule filtering.

4.CONCLUSION

The area of DRE continues to grow rapidly. The Y2K problem assisted in the software community gaining the knowledge that the data within organizations are valuable and within legacy systems essentially not understood. Correctly done, DRE can and does benefit organizations in the understanding of their current legacy systems, in the migration of their systems to new technology, and in the quality assessment of new software system. Recent work in database reverse engineering has not concentrated on a broad objective of system migration. Researchers rather focus their study on a particular issue of semantic understanding.

In legacy systems that design documents are incomplete or even missing, the system maintenance or modification is a difficult task due to the lack of knowledge regarding high level design of the system. To tackle this problem, a database reverse engineering approach is essential. Legacy systems have their own intrinsic characteristics and it is not easy to know all the relational concepts: relation schemes, primary keys, alternate keys, functional dependencies and inclusion dependencies. On the other hand, existing DBRE methods, or contributions to some particular problem, cover different cases of required input. The future work of this study aims at developing an optimal database and software reverse engineering process using the data mining techniques which provides hidden information on understanding and construction efficient reverse engineering process.

5.REFERENCES

[1]. C. Batini, S. Ceri and S. Navathe. Conceptual Database Design: an Entity-Relationship Approach. Benjamin Cummings, 1992.

[2]. R.H.L Chiang, T-M Barron and V.C Storey. Reverse Engineering of Relational Databases : Extraction of an EER Model from a Relational Database. Data and Knowledge Engineering, 12, pp. 107-142 (1994).

[3]. M.A. Casanova and J.E.A. de Sa. Designing Entity-Relationship Schemas for Conventional Information Systems. In Proc. Of the 3td Int. Conf. On the ER Approach to Software Engineering , pages 265-277, Anaheim, California , 1983. Elsevier Science Publishers.

[4]. M. Blaha. On Reverse Engineering of Vendor Databases. 5th Working Conference on Reverse Engineering, Honolulu, Hawaii, October 1998.

[5] C. Tjortjis, L. Sinos,P. Layzell, "Facilitating Program Comprehension by Mining Association Rules from Source Code", 11th IEEE International Workshop on Program Comprehension (IWPC'03), May 2003.

[6]. C. Montes de Oca, D.L. Carver, "A Visual Representation Model for Software Subsystem Decomposition", Working Conference on Reverse Engineering (WCRE'98), October 1998.

[7]. K. Sartipi, K. Kontogiannis, F. Mavaddat, "Architectural Design Recovery Using Data Mining Techniques", Conference on Software Maintenance and Reengineering(CSMR'00), February 2000.

[8]. A. Michail, "Data Mining Library Reuse Patterns in User-Selected Applications", 14th IEEE International Conference on Automated Software Engineering, October 1999.

[9]. A. Michail, "Data mining Library Reuse Patterns Using Generalized Association Rules", Proceedings of the 22nd International Conference on Software Engineering, June 2000.

[10]. J.S. Shirabad, T.C. Lethbridge, S. Matwin, "Supporting Maintenance of Legacy Software with Data Mining Techniques", Proceedings of the 2000 Conference of the Centre for Advanced Studies on Collaborative Research, November, 2000.

[11]. J.S. Shirabad, T.C. Lethbridge, S. Matwin, "Supporting Software Maintenance by Mining Software Update Records",

International Conference on Software Maintenance, (ICSM'01), November 2001.

[12]. O.Maqbool, A.Karim, H.A. Babri, and M.Sarwar, Reverse Engineering Using Association Rules

[13]. Xiaomin Wu, Adam Murray, Margaret-Anne Storey, A Reverse Engineering Approach to Support Software Maintenance: Version Control Knowledge Extraction Proceedings of the 11th Working Conference on Reverse Engineering (WCRE'04) 1095-1350/04 $20.00 © 2004 IEEE.

[14]. R. Chiang, T. Barron, and V. Storey. Reverse engineering of relational databases: Extraction of an EER model from a relational database .*Data & Knowledge Engineering*, 12:107–142, 1994.

[15]. R. Chiang. A knowledge-based system for performing reverse engineering of relational databases. *Decision Support Systems*, 13:295–312, 1995.

[16]. P. Johannesson, A method for transforming relational schemas into conceptual schemas, In Rusinkiewicz, editor,*Proc. of the 10th Int. Conf. on Data Engineering*, pages 115–122, Houston, 1994. IEEE Press.

[17]. V. Markowitz and J. Makowsk. Identifying extended entityrelationship object structures in relational schemas. *IEEE Transactions on Software Engineering*, 16(8), Aug. 1990.

[18]. C. Batini, S. Ceri, and N. Shamkant.*Conceptual Database Design - An Entity-Relationship Approach*. Benjamin/ Cummings, 1992.

[19]. J.-M. Petit, F. Toumani, J.-F.Boulicaut, and J. Kouloumdjian.Towards the reverse engineering of denormalized relational databases. In *Proc. of the 12th Int. Conf. on Data Engineering*, New Orleans, USA, Feb. 1996. IEEE Press.

[20]. W. Premerlani and M. Blaha. An approach for reverse engineering of relational databases. *Communications of the ACM*, 37(5), May 1994.

[21]. M. Blaha and W. Premerlani. *Object-Oriented Modeling and Design for Database Applications*. Prentice-Hall, 1998.

[22]. O. Signore, M. Loffredo, M. Gregori, and M. Cima. Using procedural patterns in abstracting relational schemata. In *Proc. of the 13th Int. Conf. on Entity-Relationship Approach*, volume 881 of LNCS, Dec. 1994.

[23]. R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between set of items in large databases", in Proceedings of ACM SIGMOD International Conference on Management of Data, 1993, pp. 207-216.

[24]. R. Agrawal, and R. Srikant, "Fast algorithms for mining association rules in large database", in Proceedings of 20th International Conference on Very Large Data Base, 1994, pp.487-499.

[25]. J. Han, et al., "DB Miner: System for data mining in relational databases and data warehouses", in Proceedings of CASCON'97: Meeting of Minds, 1997, pp. 249-260.

[26]. S. T. Sreenath, S. Bodogala, K. Alsabti, and S. Ranka, "An efficient algorithm for the incremental updating of association rules in large databases", in Proceedings of 3rd International Conference on KDD and Data Mining, 1997, pp.263-266.

[27]. S. Tsechansky, N. Pliskin, G. Rabinowitz, and A. Porath,"Mining relational patterns from multiple relational tables", Decision Support Systems, Vol.27, No.1-2, 1999, pp. 179-195.

[28]. F. Berzal, J. Cubero, N. Marin, and J. Serrano, "TBAR: An efficient method for association rule mining in relational databases", Data & Knowledge Engineering, Vol.37, No.1, 2001, pp. 47-64.

[29]. J. Hipp, U. Guntzer, and U. Grimmer, "Integrating association rule mining algorithms with relational database systems", in Proceedings of 3rd International Conference on Enterprise Information Systems, 2001, pp. 130-137.

[30]. H. Lee, and C. Yoo, "A form driven object-oriented reverse engineering methodology", Information Systems, Vol.25,No.3, 2000, pp. 235-259.

[31]. J. Perez, I. Ramos, V. Anaya, J. M. Cubel, F. Dominguez, A.Boronat, and J. A. Carsi, "Data reverse engineering for legacy databases to object oriented conceptual schemas",Electronic Notes in Theoretical Computer Science, Vol.72,No.4, 2003, pp. 7-19.

[32]. N. Lammari, I. Comyn-Wattiau, and J. Akoka, "Extracting generalization hierarchies from relational databases: A reverse engineering approach", Data & Knowledge Engineering, Vol.63, 2007, pp. 568-589.

[33]. G. Chen, M. Ren, P. Yan, and X. Guo, "Enriching the ER model based on discovered association rules", Information Sciences, Vol.177, 2007, pp. 1558-1556.

[34]. Blaha, Michael, "On Reverse Engineering of Vendor Databases", Proceedings of the Fifth Working Conference on Reverse Engineering, October 1998, pages 183 -190.

[35]. Chikofsky, Elliot and Kathi Hogshead Davis, Preface to the proceedings of the Workshop on Data Reverse Engineering, Zurich, Switzerland, March 2000.