Research in Database Engineering at the University of Namur

Jean-Luc Hainaut
LIBD - Laboratory of Database Applications Engineering
University of Namur, Institut d'Informatique, B-5000 Namur, Belgium
jean-luc.hainaut@fundp.ac.be - http://www.info.fundp.ac.be/libd

1 Introduction

The Laboratory of Database Applications Engineering (LIBD) is devoted to the development of models, techniques, methods and tools to support all the engineering activities related to databases and their applications. It also develops material and activities to transfer database knowledge towards industry.
This report describes the main activities of the laboratory during the last ten years. It first discusses general resources and processes that form the baselines for the other research activities. The latter will be classified into reverse engineering, interoperability, advanced processes and CASE technology.

2 The baselines

Most projects and activities have been based on a few general paradigms that we describe in this section.

2.1 Generic specification model

The Generic Entity-relationship model (GER) is a wide-spectrum information/data structure specification model. Encompassing the main concepts and constructs of most popular modeling formalisms, be they value-based or object-based, it has been given a precise semantics via an extended version of the NF2 (non-first normal form, or nested) relational model [6]. Through a specialization mechanism, such usual models as ERA, UML class diagram and ORM can be rigourously specified and compared [7]. Similarly, the GER can be used to define standard data models such as the relational, object-relational, CODASYL, IMS, XML or plain file structure models.

2.2 Transformational technology

Most database engineering processes can be modelled by a chain of schema rewriting operators called transformations. For instance, deriving a relational schema from an ERA conceptual schema can be carried out by replacing each entity type with a table, each single-valued attribute with a column, each one-to-many relationship type with a foreign key, each multivalued attribute with a dependent table, etc.

Considering a formalism F, defined as a specialization of the GER, a schema S expressed in F, and an object O in S, a transformation Σ is defined by a couple of mappings <T,t>, where T - the structural mapping - is a rewriting rule that replaces O with T(O) in S, and t - the instance mapping - specifies how any instance o of O is transformed into t(o), a valid instance of T(O). An important aspect of transformations is to what extent they preserve the semantics of the source schema. Semantics-preserving transformation can be identified through a proof mechanism based of the algebra of the GER [12].

2.3 Database Design

Though widely described since the seventies, this process still deserves further development. Indeed, modern database building brings new challenges for which current methods appear too weak. In addition, understanding database design approaches, not only as they are described in textbooks, but also those that are actually practised by development teams, is at the core of database reverse engineering. We have developed a generic database analysis and design methodology, based on the transformational paradigm, that can be specialized to fit the current approaches and to address new emerging challenges.

2.4 CASE support

DB-MAIN is the experimental meta-CASE platform that has been built to enact and evaluate all the modeling and methodological results of the laboratory. It comprises a basic services layer that includes all the functions necessary to perform standard engineering activities and an extensibility layer, through which new models and processors can be developed (as described below). The DB-MAIN architecture has been based on the GER and includes a rich transformation toolset. Its history manager allows a formal trace of the engineering activities to be recorded for further processing as we will see later.

The Education edition of the tool is in use in an estimated 4,000 schools and universities. The industrial editions (currently version 6.5d) have been available since 1996.

3 Database Reverse Engineering

3.1 A Generic Approach

Recovering the abstract specifications of a legacy database, be it made up of a genuine database or of a set of standard files, have progressively proved much more complex than first thought, and that merely drawing the data structures declared in the DDL code generally lead to poor, incomplete and obscure schemas. Many unexpected problems were discovered in large systems. They lead to two challenges, namely (1) eliciting implicit (that is, undeclared) physical constructs and (2) interpreting the physical constructs into conceptual terms. From these observations, we have developed a comprehensive generic methodology in which each problem can be addressed and solved through specific reasoning and techniques. It comprises three main processes, namely project preparation, data structure extraction and data structure conceptualization.

The goal of the first process is to build the gross architecture of the system by identifying the major components: programs, libraries, files, databases, JCL scripts, high level data and control flows. The useful sources of information also are identified: documentation (if any), DDL code, data dictionaries, file description, program code, screen layout, etc.

The second process, data structure extraction, is intended to rebuild a complete physical schema of the database which includes both the declared constructs (structures and constraints) and the implicit constructs. Starting from the raw physical schema, that strictly expresses the constructs declared in the DDL code, the process enriches and refines it from the information extracted from various sources such as program source code, database contents, documentation fragments, screen layout, report definition, and the like. This information is translated into additional record type and field decomposition, uniqueness constraints, foreign keys, redundancies and functional dependencies, etc.

Through the third process, data structure extraction, a plausible conceptual schema is derived from the complete physical schema. The problems are here of a different nature. They consists in discovering the intention (or semantics) expressed by the physical constructs, taking into account the techniques and tricks used by the developers when the database was built.

By carrying out a series of reverse engineering projects, we made two surprising observations. The first one is that most constructs are implicit, so that the DDL code provides a very partial view of the physical schema. The second observation is that the way constructs are hidden is for the largest part model-independent. For instance, while standard files naturally include many implicit foreign keys, the latter have been found in most hierarchical and network databases as well and even in recent relational databases, where that could have been declared. In other words, designing dedicated reverse engineering methodologies for IMS, CODASYL DBTG or COBOL would be meaningless, since they would be almost the same, but for the preliminary DDL code parsing.

The generic methodology we have developed relies on the GER model. Indeed, database schemas as they are built and used in reverse engineering are, be nature, heterogeneous. On the one hand, since real databases generally consist of a federation of physical databases, a final physical schema often results from the integration of several schemas in different technologies. Typically relational or IMS schemas include dozens of COBOL record types. On the other hand, the conceptual process basically is continuous, so that a schema being reverse engineered still includes unprocessed physical and logical (model-specific) constructs, while it already includes final conceptual constructs.

The methodology also relies on the transformational paradigm. In particular, the conceptualization process has been modelled as a chain of transformation that are the inverse of logical design techniques.

The baselines of the database reverse engineering methodology have been presented in [8, 9, 14].

3.2 Specialized methodologies

We have developed specific techniques and rules to help solving complex problems. We describe shortly three of them.

Program understanding techniques. Many implicit constructs can be identified by examing how the application programs use the data. For example, the code for inter-record navigation, field processing and validation before writing data in a file generally gives much information on the structure and properties of the data. Hence the use of program understanding techniques, that have been adapted for data structure elicitation. In particular, we have developed variants of regular pattern matching, dependency graph analysis, data flow analysis and program slicing [15].

Foreign key interpretation. Foreign keys form one of the most important structure to recover. However, interpreting them poses another challenge. Besides the standard "*foreign key* into *many-to-one relationship type*" transformation rule, we have identified and solved about thirty non standard patterns of which we have found no trace in the literature.

Subtype hierarchies rebuilding. Many techniques exist to implement is-a relations in non object-oriented technologies. One of the most popular of them is through downward inheritance, which consists in recursively distributing the attributes and other constructs of supertypes among their subtypes, then discarding these supertypes. To rebuild the subtype hierarchy from such data structures, we have derived an efficient and intuitive algorithm from the Galois lattice.

3.3 Web site reengineering

In many companies, the official web site is an integral part of the information system, in that, much information of the web pages has no counterpart in the corporate databases. Reengineering this information by splitting these pages into interpreted data and presentation information has become vital. Though this process requires specific techniques, it shares much with pure database reverse engineering. The project WebReverse addresses this problem by developing specialized techniques and tools [5]. This project takes place at the CETIC (www.cetic.be), a inter-university research center in which the LIBD is involved.

3.4 CASE support

Reverse engineering requires the precise analysis of huge documents such as million-LOC program code and schemas that include thousands of tables. It also requires repeatedly applying complex rules on thousands of patterns. Hence the need for specialized reverse engineering tools. We have included in the DB-MAIN CASE environment tools that support many of the reasoning and techniques we have discussed above. See [11, 17] for instance.

4 Database Interoperability

Developing models, techniques and tools for building links between existing databases is another stream of activity of the LIBD. Here below, we describe three of the main results.

4.1 Scalable Federated Database Architecture

Since 1995, we have been exploring architectural issues in federated databases within the wrapper/mediator framework. We have proposed a new scheme of sharing responsibilities such that wrappers assume a greater part of the federation mapping in order to alleviate the tasks of the mediators. A methodology that combines reverse and forward approaches has been developed to specify and automatically build the components of a federation while taking into account the current requirements of the organization [13, 21]

4.2 Wrapper design and development

This line of activity is linked with the former, since it concentrates on the development of a specific type of components of a federated database. However, due to its wider scope, we discuss it independently. Generally a database wrapper is a piece of software that translates queries and data between an abstract external model and the physical model of an existing database. Considering the poor quality of legacy databases, we have been lead to extend this definition as follows. We have shown that the raw physical schema, merely derived from the DDL code, most often is incomplete, and must be enriched with hidden constructs made explicit. The wrappers we intended to build had to enjoy two advanced properties, (1) they provide both extract and update facilities, (2) they control both the explicit and explicit constructs of the source database. For instance, these "intelligent" wrappers manage the implicit compound and multivalued fields found in relational databases and the foreign keys that implicitly hold in COBOL files. The DB-MAIN environment has been extended with a plugin that allows wrappers for SQL databases and COBOL files to be designed and automatically generated with two interfaces, namely OQL and Java objects. These results strongly rely on the database reverse engineering methodology and on the history processing facility of DB-MAIN [20].

4.3 Data conversion and migration

This a natural extension of the results described above. When the relation between data structures S1 and S2 can be described by the chain of structural transformations Tn*…*T1, with $N \geq 1$, then converting any instance s1 of S1 into a valid instance of S2 consists in computing tn*…*t1(s1). This principle has been applied to DB-to-DB, DB-to-XML and XML-to-XML conversion. When the schemas and the transformations are under the responsibility of DB-MAIN, the chain Tn*…*T1 is derived from the history and tn*…*t1 is automatically built from the known instance mappings of T1 to Tn. Then, DB-MAIN automatically generates a procedural expression of the latter in the form of an ETL component. According to the context, this component is expressed in COBOL, in Java, or as an XSLT sheet. A simplified migration architecture uses wrappers so that the migrator interfaces with the source and target databases through a unique abstract model. For further details see [2].

5 Special Processes

This section describes the results of other R&D activities that do not fit into the first three categories but that can be located at their upstream or downstream.

5.1 Active Databases Engineering

The theme focuses on the systematic generation of the set of triggers that implement abstract behaviour specifications. The main results are (1) a methodology for translating conceptual integrity constraints into triggers, (2) a graphical extension of DB-MAIN to show the network of a trigger system (as well as the potential critical sections) and (3) a parametric Oracle DDL generator, integrated to DB-MAIN, that can generate efficient data structures that completely implement complex integrity constraints through various techniques. The latter is also the only (as far as we know) SQL generator that completely and correctly generates arbitrary multiple is-a hierarchies for Oracle and InterBase, including mutation operators (see [1] for technical details).

5.2 Temporal Database Engineering

The goal of this research was to bring research results in temporal databases to practitioners. It has resulted into three products: (1) a complete methodology with which a temporal conceptual schema can be progressively transformed into an SQL2 relational schema, (2) a sophisticated Oracle generator that produces an active database according to several data distribution strategies for transaction-time, valid-time, bitemporal and monotonic data, (3) an ODBC-like API allowing programmers to use such a temporal database without worrying about the technical aspects. The latter offers a reduced version of TSQL2 through which temporal projections, joins and agregations can be carried out. These products, that have been developed as extensions of DB-MAIN, are described in [3].

5.3 XML engineering

The baselines of the LIDB activities, namely the GER model, the transformational paradigm, the formalized DB design approach and the DB-MAIN platform are quite fitted to reasoning about, and to process, XML structures. Indeed, the DTD and XML schema models can be completely specified as specialization of the GER. Therefore, all the processes that are relevant to database engineering also apply to XML structure manipulations. In particular, we have developed a complete methodology for expressing any conceptual schema into a DTD or an XML schema (full generators included). We have also built the required techniques and tools to reverse engineer and to transform XML structures. As described above, XML-to-* and *-to-XML data migrators are available. All these tools have been developed in the DB-MAIN environment.

5.4 Database Evolution

This research addresses one of the most complex database engineering problems, that is, how to adapt a complex legacy database to changing users requirements. It explores several strategies, the most challenging of which being the automatic propagation of conceptual changes down to the physical schema and to the database contents. A comprehensive methodology has been designed and is supported by a plugin of DB-MAIN. The approach and the tools are described in some details in [10, 18].

5.5 Database Applications Reengineering

This activity copes with the third, and still widely unsolved, aspect of database evolution, namely automatically propagating requirements changes to the application program that access a legacy database. We have so far identified and analyzed six strategies through their application to a small but representative case study [16]. One of them seems particularly attractive in platform migration problems. It allows legacy programs to run on top of a modernized database (for instance the normalized SQL translation of a COBOL database) while requiring little program rewriting. It relies on the database reverse engineering methodology and on intelligent wrapper generation.

6 CASE technology

6.1 Meta-development environment

The DB-MAIN platform provides an efficient support for the development of additional specialized components, or plugins. Besides the possibility to dynamically associate meta-properties with any object class of the repository, the main component of the meta-layer of DB-MAIN is the Voyager language with which most extensions described in this report have been developed. This complete procedural language includes a predicative and navigational interface to the repository, a direct access to the kernel basic functions, a powerful list processor, a tokenizer for parser development, I/O functions and a coordination mechanism with external independent processes. Some of its features have been described in [4].

6.2 Engineering process control

The genericity of the GER, of the transformational technology and of DB-MAIN itself provides what can be called *methodology-neutral* engineering resources. This research addressed the problem of specializing them for specific engineering tasks such as database design, database reverse engineering, database evolution ot XML database development. It has yielded three products, that are described in [19] as well as in the technical documents of the CASE tool: (1) a comprehensive model of engineering processes and products, (2) a method description language (MDL) and its development environment, comprising an specification editor, a graphical viewer and a compiler, (3) the method engine of DB-MAIN, which enacts rigorously the current method through a graphical view of the method and of the history of the processes.

7 Further information

More detailed information can be found on the site of the laboratory: http://www.info.fundp.ac.be/libd. The site includes the text of many of the 60 articles published in the last ten years as well as about 4,000 pages of technical and didactic material on database engineering. The Education edition of the DB-MAIN CASE tool can be downloaded free of charge together with tutorials, sample projects and case studies. It includes all the basic functions as well as some advanced processors but is limited to small-size projects that nevertheless are quite comfortable for educational use.

References

[1] Brogneaux, A-F., *Code generator for Oracle databases*, v6.5, LIBD Technical manual, March 2002, http://www.info.fundp.ac.be/~dbm/publication/2002/SQL-generator.pdf

[2] Delcroix, C., Thiran, Ph., Hainaut, J-L., Approche transformationnelle de la réingénierie des données, Ingénierie des Systèmes d'Information, 6(1), Hermès, Paris, 2001

[3] Detienne, V., Hainaut, J-L., CASE Tool Support for Temporal Database Design, in *Proc. of 20th Int. Conf. on Conceptual modeling (ER 2001)*, Springer Verlag LNCS 2224, 2001

[4] Englebert, V., Hainaut, J-L., DB-MAIN: A Next Generation Meta-CASE, in *Information Systems Journal*, Special issue on meta-modelling and methodology engineering, 24(2) Pergamon, June 1999

[5] Estiévenart, F., François, A., Henrard, J., Hainaut, J., Web Site Engineering, in *Proc. of the 5th International Workshop on Web Site Evolution*, Amsterdam, Sept. 2003, IEEE CS Press, 2003

[6] Hainaut, J.-L., A Generic Entity-Relationship Model, in Proc. of the IFIP WG 8.1 Conf. on *Information System Concepts: an in-depth analysis*, North-Holland, 1989

[7] Hainaut, J-L., Entity-Relationship models : formal specification and comparison, in *Proc. of the 9th Int. conf. on ER Approach : the Core of Conceptual Modelling*, North-Holland, 1991

[8] Hainaut, J-L, Database Reverse Engineering, Models, Techniques and Strategies, in *Proc. of the 10th Conf. on ER Approach*, San Mateo (CA), E/R Institute Publish., 1991

[9] Hainaut, J-L., Chandelon M., Tonneau C., Joris M., Contribution to a Theory of Database Reverse Engineering, in *Proc. of the IEEE WCRE*, Baltimore, May 1993, IEEE CS Press, 1993

[10] Hainaut, J-L., Englebert, V., Henrard, J., Hick, J-M., Roland, D., Database Evolution - the DB-MAIN Approach, in *Proc. of the 13th Int. Conf. on ER Approach*, Manchester, Dec. 1994, Springer-Verlag, LNCS 881, 1994

[11] Hainaut, J-L., Englebert, V., Henrard, J., Hick, J-M., Roland, D., Database Reverse Engineering : from Requirements to CARE tools, *Journal of Automated Software Engineering*, 3(1), 1996, Kluwer Academic Press

[12] Hainaut, J-L., Specification preservation in schema transformations - Application to semantics and statistics, *Data & Knowledge Engineering*, 16(1), 1996, Elsevier Science Publish

[13] Hainaut, J-L., Thiran, Ph., Hick, J-M., Bodard, S., Deflorenne, A., Methodology and CASE tools for the development of federated databases, *The International Journal of Cooperative Information Systems*, 8(2-3), pp. 169-194, World Scientific, June & Sept. 1999

[14] Hainaut, J-L., *Introduction to Database Reverse Engineering*, LIBD lecture notes, {http://www.info.fundp.ac.be/~dbm/publication/2002/DBRE-2002.pdf}

[15] Henrard, J., Roland, D., Englebert, V., Hick, J-M., Hainaut, J-L., Program Understanding in Databases Reverse Engineering, in *Proc. of the 9th Int. Conf. on DEXA,* Vienna, June 1998, Springer-Verlag, 1998

[16] Henrard, J., Hick, J-M. Thiran, Ph., Hainaut, J-L., Strategies for Data Reengineering, in *Proc. of WCRE'02*, IEEE CS Press, 2002

[17] Henrard, J., *Program comprehension in database reverse engineering*, PHD Thesis, Sept. 2003

[18] Hick, J-M., Hainaut, J-L., Strategy for database application evolution: the DB-MAIN approach, in *Proc. ER'2003 conference*, Chicago, Oct. 2003, LNCS Springer-Verlag, 2003

[19] Roland, D., *Database engineering process modelling*, PHD Thesis, June 2003

[20] Thiran, Ph., Hainaut, J-L., Wrapper Development for Legacy Data Reuse, in *Proc. of WCRE'01*, IEEE CS Press, 2001

[21] Thiran, P., *Interoperability of legacy databases. A combined top-down and bottom-up approach*, PHD Thesis, Oct. 2003