Model-Driven Reengineering of Database

Hanzhe Wang
School of Software, Shanghai Jiaotong University, Shanghai 200240, China
jailywang@sjtu.edu.cn

Beijun Shen
School of Software, Shanghai Jiaotong University, Shanghai 200240, China
bjshen@sjtu.edu.cn

Cheng Chen
Wonders Information Co., Ltd., Shanghai 200230, China
chen_goodwin@yahoo.com.cn

Abstract

A lot of work has been done applying Model-Driven Approach to those business domain concerned software development. These researches mostly show how to transform business domain models to software application with different paradigms, rather than how to transform specific software artifacts generally regarding of business domain factor, such as database, the common infrastructure of nowadays software system. The later kind of work can make more contribution to general software development rather than some specific business domains. In this paper, we present a MDA based approach to perform database reengineering and also build a framework based on current framework (EMF, Operational-QVT).

1. Introduction

Database reengineering includes two stages just like software reengineering [1]: reverse engineering and forward engineering. The first stage's major goal is to obtain the design specification of the original system, more specifically, the data model of a database system. It concentrates on data model recovery of legacy systems and is also called database reverse engineering (DBRE) [2]. And the second stage's goal is to generate a new data model, or migrate to a new DBMS with the same data model. The second stage has been already discussed in many business domain concerned MDA approaches, as the database is the most common scheme to information storage in today's software system. But little attention has been paid to the first stage, which in fact is the central problem of database reengineering. In this paper, discussions focus on the DBRE in database reengineering.

There are already a wide range of DBRE methods as the results of those significant research works in the context of database reengineering in latest years. The most obvious problem about these approaches is that each of them requires its own specific inputs and assumptions, and each of them exhibits its own methodology in different ways. When applying these approaches to a specific legacy system, it's found that the particular characteristics of the system not only restrict the information availability but also make each single one of afore-mentioned approaches not adequate by their own [3]. In such case, a simple solution is applying multiple methods to the same single system which can take advantage of each single method and produce better results together. But there is one gap to across, which is also the most important factor preventing combining these methods together. It's the understandings between each method since they all have their own describing paradigm. We present a model-driven approach in this paper to solve such understanding problem, which allows different method corporate with other methods and apply them to the specific system.

The rest of the paper is organized as follows: Section 2 describes briefly the related work. Section 3 describes the model-driven approach. Section 4 introduces the framework built for the approach. And finally the conclusion and future work is given in section 5.

2. Related Work

In the field of database reengineering we can find some significant research works which all focus on recovering the conceptual schema used during initial development. And they are usually divided into two major steps [4]: (1) database structure extraction and (2) data structure conceptualization. This two stage process of DBRE is shown in Figure 1.

This research is partially supported by the National High-Tech Research Development Program of China (863 program) under Grant No. 2007AA01Z139.

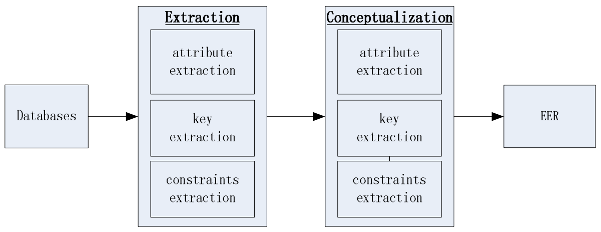

Figure 1. The DBRE General Process

In the extraction stage, complete database schemas are extracted from DBMS. And if there is a formal DDL description of the database, this stage could be greatly expedited. Otherwise, a fair amount of data analysis, program analysis, and form analysis need to be performed to get the database schemas which is also called the logic schemas. Fortunately, most modern DBMS maintain DDL information within the database. This stage is not the hotspot anymore.

In the conceptualization stage, logic schemas from the previous step are transformed into conceptual schemas. This is most important step of DBRE, and is the most significant difference between approaches. After Pedro de Jesus and Sousa [3] did their interesting study analyzing the characteristics of several proposals and telling selection of these methods according to particular characteristics of legacy system, more methodologies and tools have been designed to solve new arisen problems under certain software context. For example, it presented the way to generate three-tier application from relation database in [5] and how to generate competent-based web application in [6] from relation databases. In [7], the author discussed extracting generalization hierarchies form relational database. In higher abstraction, approach as DB-MAIN [8] and framework as MeDEA [9] have been designed to manage the database evolution. But all these approaches didn't get much share with each other. When a new method is introduced, it needs to build a whole new set of tools. Sometimes, the latter method only introduces little changes to the older one solving the same problem, but they develop their tools separately and do not share technical details between each other. This kind of duplication of effort wastes researchers too much time and prevents them focusing on what they really need to solve. More approaches and tools are developed, this problem becomes more serious.

The models obtained by one algorithm proposed in one paper can be sometimes richer than others, but only from a merely conceptual point of view. No matter how the author claims that the approach is better than others, it just makes sense in particular situations. And using them together seems is still the best way. But the approaches are so different from each other, that using one across with another will bring more trouble than convenience. Not only can their outputs be very different in content and expression, but also their algorithm are implemented in different technologies what prevent them work together smoothly. It's very necessarily to propose an approach to integrate these algorithms, not only for achieving good results from DBRE, but also making it easier to develop new approaches on them.

3. Model-Driven DBRE

MDA approach was proposed by the OMG years ago. The three primary goals of MDA are portability, interoperability and reusability through architectural separation of concerns [12]. In this section, the MDA approach is applied to DBRE to increase the interoperability and reusability of different methods and tools.

3.1. Model Transformations

Models are tagged as two kinds in MDA [12], Platform Independent Models (PIM) and Platform Specific Model (PSM). The central problem in MDA is the transformation between them. Those software assets resilient to changes in the technologies take the form of PIMs, that is, models that do not contain implementation specific details. These models may be transformed to other models that includes information specific to implementation technologies (PSMs). PSMs then can be used for code generation.

Consider DBRE process as a serious of model transformations, the extraction step gets the logic model (PSM) from DBMS. And the conceptualization step get conceptual model (PIM) from logic model (PSM). Replace the extraction and conceptualization steps with corresponding transformations in Figure 1, it comes out the Model Driven DBRE process as shown in figure 2.

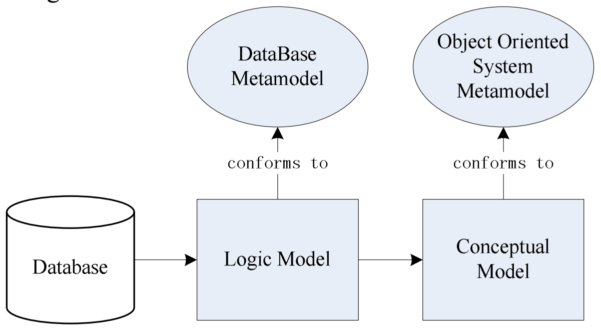

Figure 2. DBRE Transformations

The outputs of aforementioned DBRE process are conceptual models presented in EER. Such outputs are often translated into a class diagram in order to take advantage of object-oriented constructions and tools for the next forward stage. And object-oriented conceptual schemas can be improved with formal notations, as object constraint language (OCL) [10]. In our approach, the conceptual models extracted from databases are object-oriented.

3.2. Transformation Rules

Although the existed DBRE approaches follow different paradigms in their description of the rules used to get the conceptual model schemas from logic model. They all follow the same simple pattern: select one subset of content according to some specific statement of the element in the logic model, and then create corresponding elements in the conceptual model. Based on these rules, tools are built in different languages and using different technologies.

The rule pattern covers two important aspects in a transformation. First, the query criteria, it tells how to select element from the source model. Second, the mapping, it tells what element in the target model to create. Rules following the pattern can be viewed as a two-tuples:

R = (Q, M), where Q is the set of criteria contained in the rules and M is the set of mapping in the rules.

An approach is a combination of rules, and can be viewed as a set of rules:

$A = \{r_i \mid r_i \text{ is a rule}\}$.

3.3. Approach Using QVT

QVT [11] describes the model transformation in MDA and provides an operational mappings language. This language is specified as a standard way of providing imperative implementations. It provides OCL extensions with side effects that allow a more procedural style, and a concrete syntax that looks familiar to imperative programmers.

According to [11], Mapping Operations invoking other Mapping Operations always involves a Relation for the purposes of creating a trace between model elements, but this can be implicit, and an entire transformation can be written in this language in the imperative style. A transformation entirely written using Mapping Operations is called an operational transformation.

A rule of approach could be defined by set of queries and mappings in QVT. Then the approach is an operational transformation in QVT. Consider two approaches T and S:

$T = (Q_t, M_t)$ and $S = (Q_s, M_s)$.

Using T and S together will be a new operational transformation in QVT, including all the queries and mappings in T and S. Call it N:

$N = (Q_n, M_n)$, where $Q_n = Q_t \cup Q_s$ and $M_n = M_t \cup M_s$.

Each rule can be adapted to QVT schemas, and each approach can be turned to a Transformation in QVT. With the same model and transformation language they refer, any approach can smoothly mix with others.

3.4. RDBMS Metamodel

The rational database is the most case when we refer database. The metamodel of RDBMS is simple, and useful in our approach, which is the base of understanding between methods. Basic elements including table, key constraints and their relations are shown in Figure 3:

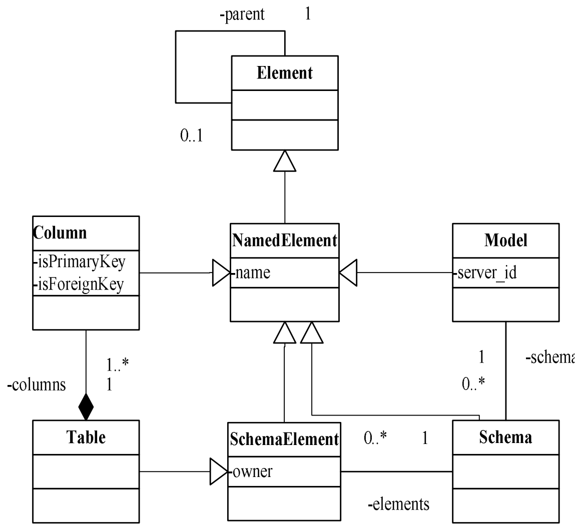

Figure 3. RDBMS Metamodel

The RDBMS model should reflect the extract physical database structure. The most important elements in the model are Table and Column. Table combines columns, and Column holds two main properties indicting whether it is a primary key or a foreign key of belonged table. Other elements like Schema and Model are used to organize them to correspond to complete databases. The NameElement and SchemaElement are for extension in future. For example, procedures in database may included to the model by extends the SchemaElement. This model covers the most common RDBMS structure and is enough to explain the approach in this paper.

3.5. An Example

Take the afore-mentioned RDBMS model as input model and the UML model as output model. Let's see how a rule is expressed in QVT language. Consider the simple rule below:

Each column in a table should refer an attribute in corresponding class, and the attribute name is same with the column's name.

In this rule, it only exists one mapping:

```
Mapping RDB::Column::column2attribute() :
UML::Property
{
   name := self.name;
   type := object
   UML::PrimitiveType{ name :=rdbPrimitive2umlPrimitive(
   self.type.name)};
}
```

There is a query in this mapping, although there seems no query in the rule.

```
Query rdbPrimitive2umlPrimitive(in name : String) : String
{
   if name = 'varchar' then 'String' else
      if name = 'int' then 'Integer' else
         name
      endif
   endif
}
```

This a very simple rule, but it shows how other rules are described in operation QVT language.

4. Proposed Framework

EMF is an open source Eclipse Project, and its metametamodel- ecore is quite popular and be able to construct powerful MDA approach. For example, the Operational QVT is an implementation of QVT specification in MDA in Eclipse, and is built based on EMF. We propose a framework called RDBREF, and Figure 4 shows the structure of the framework.

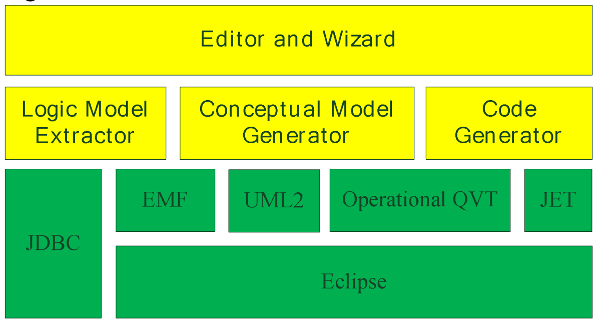

Figure 4. RDBERF Framework

The Logic Mode Extractor uses JDBC to get the DDL specification from DBMS. The Conceptual Model generator uses selected approaches' QVT schema to generate the conceptual models. There is also a Code Generator in the framework which is responsible of generating database schemas and well known CRUD operations and building a web application.

The framework should handle two kinds of inputs. In most cases, it takes databases as input, and outputs the conceptual models. But when a new DBRE approach is proposed and is considered to be including in the framework, the framework should take the approach in the form of operational as inputs. Figure5 shows the two cases.

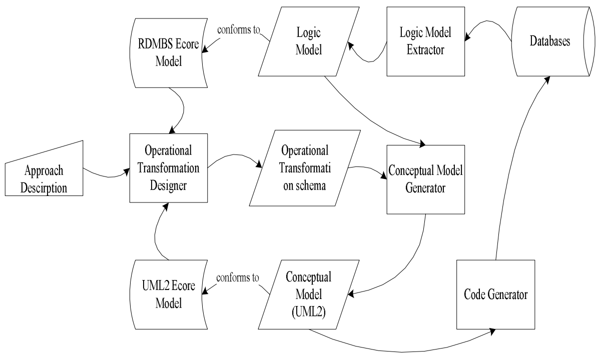

Figure 5. Flow under Different Inputs

Once the approach has been inputted, it becomes one candidate of methods stored in the framework.

5. Conclusions And Future Work

An important contribution of our work has been to take explicitly into account the common part of different DBRE methods and discover the shared pattern of rules in these methods. Instead of building a lot of different tools to implement different methods, and making a chose between these tools when dealing with a practical system, we break down each method to single rules, and break down a rule to queries and mappings according to the discovered rule pattern aforementioned, and combine these rules together to form a specific method. Not only can these rules be combined as transformation implemented the corresponding method, but also can these queries and mappings be selected to find new rules and new method.

Starting from there, the second contributions presented in this paper has been to provide the ability of implementing tools of new method at the same time while the new methods complete its rules taking the form of QVT language, thanks to the ability of MDA.

Although adaptation of existed methods to the framework benefits, little adaptation works have been done and there is no automated way to do this. All works need to be done manually. The competence of models used has not been proved yet, there is no guarantee that the models can serve well to all the methods. Besides, the assumption that the DDL can be gotten easily through DBMS is not always true. In those cases that doesn't fulfil the assumption, the extraction of logic model is still a noisy work since the framework covers little about this activity. How to break this limit, together with model enhancement and adaptation of existed work are the main jobs in the future.

6. References

[1] Arnold, R.S., Software Reengineering, IEEE Press, 1992.
[2] Aiken, P.H., Data Reverse Engineering, McGraw-Hill, 1996.
[3] L. Pedro de Jesus, P. Sousa, Selection of Reverse Engineering Methods for Relational Databases, Proceedings of the Third European Conference on Software Maintenance, IEEE Computer Society, Los Alamitos, CA, 1998, pp. 194-197.
[4] Davis, K.H., "Lessons Learned in Data Reverse Engineering", Proceedings of the 8th Working Conference on Reverse Engineering, October, 2001, Suttgart, Germany, pp.232-327.
[5] Polo, M., et al., Generating three-tier applications from relation databases: a formal and practical approach, Information & Software Technology, 2002, pp. 923-941.
[6] Ignacio, G.R de G., Polo, M., Mario P., Automated Generation of Component-Based Web Applications from Database, Software Engineering Research and Practice, 2004.
[7] N. L., I. CW., Jacky A., Extracting generalization hierarchies from relational databases: A reverse engineering approach, Data & Knowledge Engineering 63, 2007, pp.568-589
[8] Hick, J.M., J.L. Hainaut, Strategy for Database Application Evolution: The DB-MAIN Approach. LNCS 2813, 2003, pp.291-306
[9] E. Dominguez, Jorge L., Angel L.R., MeDEA: A database evolution architecture with traceability, Data & Knowledge Engineering 65, 2008, pp.419-441.
[10] J. Warmer, A. Kleppe, The Object Constraint Language, Addison-Wesley, Reading, MA, 1998.
[11] MOF QVT Final Adopted Specification, http://www.omg.org/docs/ptc/05-11-01.pdf, 2005.
[12] Joaquin Miller, Jishnu Mukerji, MDA Guide version 1.0.1, http://www.omg.org/docs/omg/03-06-01.pdf, 2003.