The Object Databases Transformation

Marek Tavač[*]
Department of Software Technologies
Faculty of Management Science and Informatics
University of Žilina
Univerzitná 8215/1, 010 26 Žilina, Slovakia
marek.tavac@fri.uniza.sk

Abstract

This article deals with the transformation of a relational database to an object database. It suggests a new solution based on database and model-driven engineering integration. In the first part it is discussed in general issues of relational database transformation into object one, some problematic areas are identified. Subsequently, depending on detailed analyze of the database reverse engineering processes, the whole process is defined as a sequence of models and transformation among them based on model driven architecture (MDA). Some of the models are based on generally accepted standards. The proposed approach comes also with its own models. The paper defines not only their exact structure but also abstracts its own algorithms used during their development. The final part contains an experimental verification of the proposed procedures with a particular example and summary of the results.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems—*object-oriented databases, relational databases*; D.2.0 [**Software Engineering**]: General—*standards*

Keywords

MDA, MDE, database reverse engineering, transformation

1. Introduction

There have been extensive changes and huge progress in the area of the information technology during the last decades. There are new requests for even more innovations, their improvements and integration of existing solutions there. The changes do not avoid to databases and databases applications either. Databases are an area where it is not always possible to replace an existing solution by a new technology. This is one of the reasons why object-oriented database systems are only gradually and slowly gaining their position in the world of information technology.

Relational database systems (RDBMS) provide a reliable, standardized and robust solution to ensure data persistence. But relational database weaknesses are rapidly demonstrated during attempts to use them for advanced applications such as CAD, CAM, knowledge, transport and multimedia systems. The implementation of difficult operations over complex objects shows the need for a new generation of database applications: object-oriented database systems that would meet their requirements better.

Object-oriented database systems (OODBMS) began to develop in the mid-eighties. They have been building up to more appropriate represent, model real world entities and to provide a richer data model than relational database systems.

Of course, as almost each system it has its disadvantages. One of the fundamental shortcomings is the lack of interoperability between OODBMS and RDBMS. Some technique must define how to migrate from relational to object-oriented systems, that allows coexistence and migration from existing products to new ones step by step.

The literature distinguishes two basic migration modes: a complete rewriting of the entire information system from the beginning or a strategy of the gradual migration in small steps. Only a few particularly large information systems can afford the luxury of a start from scratch. The gradual strategy allows controlling the whole process. Every migration process consists of three basic steps: schema transformation, data and applications migration.

The first step is to remap the existing relational schema to the equivalent object one. There are many articles published in this area, mainly engaged in the database reverse engineering. Based on the reverse engineering output it is necessary to create an object schema. Data are migrated in the second step. Ideally, this step is fully automated, but methods and ways of mapping individual items of data between the relational and object oriented scheme must be defined before. The last step is the process of the migration of all the application programs.

[*]Recommended by thesis supervisor: Prof. Karol Matiaško. Defended at Faculty of Management Science and Informatics, University of Žilina in Žilina on August 23, 2012.

Tavač, M. The Object Databases Transformation. Information Sciences and Technologies Bulletin of the ACM Slovakia, Vol. 5, No. 2 (2013) 1-10

As you can see, the database migration is a complex and difficult process. Therefore there is an effort for its full, or at least, partial automation there. A common primary source of information for its automation is a relational schema. This information can be extended also by other sources, such as existing applications, integrity constraints and SQL queries within reverse engineering. The resulting schema looks therefore rather as an object view to the relational database than a real object model.

The need to integrate new requirements in the systems complicates the whole database migration process even more. The main goal of my work is to create algorithms and support mechanisms that allow to automate it. At the same time, it will enable to create an object model that would meet all requirements for the proper object-oriented design and would take all advantages of OODBMS this way.

1.1 The Goal

The main objective of my work is to create a methodology of relational database migration to object one. In addition to the schema transformation, it is necessary for the whole migration process to allow also a subsequent and dependant automated data migration.

The goal of this work is not only to design the process itself, but also to ensure that it is formally correctly specified using standardized tools and protocols. The whole process will be inserted into the MDA architecture framework and so defined as a sequence of standardized models and transformations among them, organized into architecture of layers and transformations. The goal is their metamodel description, defining the relationship between domain concepts, their semantics and constraints.

2. State of the Art

There are several ways and approaches to the issue of transformation of a relational database. In terms of their data model representations they can be divided into two basic groups. In the first case, data persistence is managed by the relational database, and above this layer is formed a special layer: an object-oriented front-end, serving for transformations between both models. In the second case, the relational database is replaced by an OODB. We can talk about a full database migration.

2.1 Database Migration

It is necessary to replace most of the elements of a relational database for their corresponding object-oriented elements during migration. In addition to the relational schema transformation data themselves must be transformed. The disadvantage of this approach is the need of reworking the existing applications, the transition from relational database to OODB.

The semantic enrichment precedes the process of the schema transformation – additional not explicitly available information achievement from the RDBMS. The process is also known as the database reverse engineering. Hainaut has laid introduction to it in [9, 10, 12, 13]. The general systematic DBRE methodology and detailed description of each step can be found in [11].

In most concepts, the final conceptual schema is extracted in the form of ER or EER schema [4, 7]. They are usually

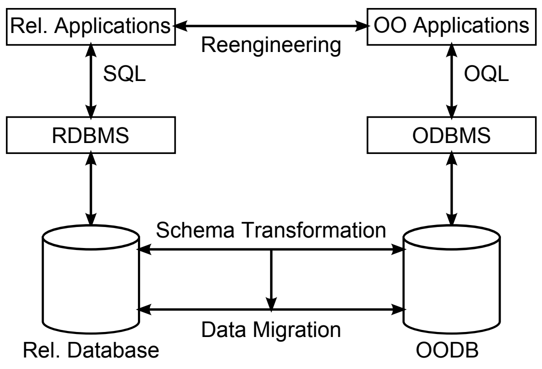

Figure 1: Database migration architecture.

based on keys analysis and from their resulting dependencies. Some of them take the actual data stored in relations into account. Some approaches do not expect the active user participation [19, 16] and are fully automated, which results to schemas graphically and semantically equivalent to the representations of the source relational schema only. Some conceptual structures can be represented by the relational schema in different ways. This statement is a true vice versa and it is not always possible to identify the correct representation without the user intervention.

Only a few concepts extract a resulting model directly into the object-oriented schema. On principle, they do not differ from the previous ones. [21] informally describes the sequence of steps to extract the OMT model.

More advanced approaches do not take into account only the structural elements, but as for example [23] also analyze SQL queries on a procedural level. Mature system based on fuzzy logic is described in [14].

There are many approaches dealing with reverse database engineering, but only few of them have dealt with this topic so complexly to allow the subsequent migration of data. [3, 8] deal with the transformation of the relational schema into object-oriented one and with the data migration methodology. [6] provides an interesting solution, that, in addition to the data migration, suggests also the algebra for the transformation of the generated OODB schema.

2.2 Model Driven Architecture

Architecture is a specification of components and system connection and interaction rules of these components using these connections [22]. From MDA point of view, connections and rules are represented by a set of interrelated models. It provides a set of rules how to structurally specify these models. Its approach is based on the previous widely accepted standardized specifications, and therefore, it provides a complex and interoperable framework for their definition. MDA is a way of organization and management of large architectures supported by automated tools and services, whether to model a definition or to facilitate transformations between different types of models.

Basic MDA principles fully reflect the needs of the proposed system of the relational database transformation into OODB. MDA does not come only with the principles and methodology for modeling and transformations between models, but, unlike many other approaches, it is based on clearly defined standards such as: CWM, UML, MOF and XMI.

UML is a widely accepted graphical modeling language (a detailed description can be found for example in [5]). One of the reasons why the UML is so widely accepted lies in its openness. It is not a closed language but it is possible to expand it using UML profiles – to define own model semantics.

The main objective of CWM is to allow an easy exchange of metadata between different tools, platforms, metadata repositories in a heterogeneous distributed environment [1]. It is based on three industry standards: UML – modeling standard, MOF – metamodelling and metadata standard and XMI – the standard for exchange of metadata. MOF standardizes the framework for the models definition. It provides tools with the programming interface for the metadata storage and an access to them. XMI enables the exchange of metadata in XML standard format. The CWM specification consists of metamodels definitions for different domains as for example: the object model, relational model, the actual CWM, XML, ...

Interactions between these three standards can be expressed as: XMI is used to exchange metadata based on CWM metamodel and CWM metamodel is specified using the MOF model.

2.3 OODB Modeling

There is an effort to standardize the OODB for many reasons. Unlike relational models, which have a clear common basic design, object-oriented concepts differ among existing programming languages (for example, support for a single or multiple inheritance). Several major OODBMS producers proposed the standard known as ODMG with the aim to mutual portability and interoperability. ODMG standard provides an object model, language specification and query language.

ODMG object model defines a common set of features of the object-oriented concept that can be used in different languages. It is the extension of the OMG object model. The basic modeling primitive is an object that is defined by its type.

ODMG specification language can be used to specify a schema and an object database data. The database schema is defined through the language called ODL. It is independent on the specific programming language. Second language specified by ODMG is the OIF. OIF is a language specification allowing to map OODB content into a serializable form (e.g. a file) and vice versa.

2.4 Object λ-calculus

One possibility to formalize the object-oriented programming concepts, and thus the object database is a λ-calculus [15]. λ-calculus is based on the notions of abstraction, function and function applications [20]. The thesis goal is to transform the database as a sequence of models and transformations among them. Although MDA provides a standard specification for models transformations – QVT, but it is not supported by current tools. λ-calculus is an appropriate form of formal specification of the transformations because of its better abstraction and enhanced independence.

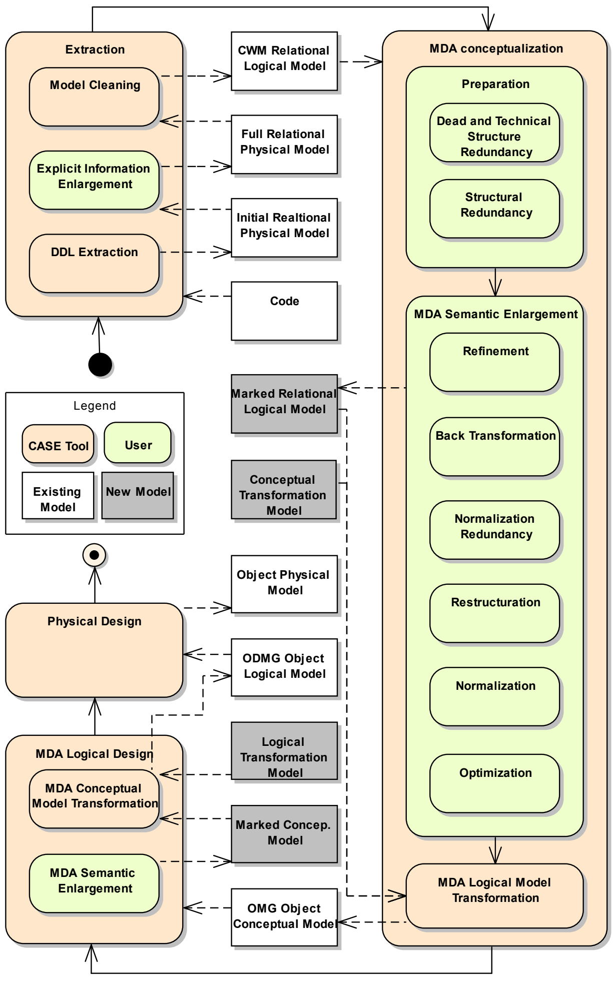
Extraction
Model Cleaning
Explicit Information Enlargement
DDL Extraction
CWM Relational Logical Model
Full Relational Physical Model
Initial Realtional Physical Model
Code
MDA conceptualization
Preparation
Dead and Technical Structure Redundancy
Structural Redundancy
MDA Semantic Enlargement
Refinement
Back Transformation
Normalization Redundancy
Restructuration
Normalization
Optimization
MDA Logical Model Transformation
Legend
CASE Tool
User
Existing Model
New Model
Marked Relational Logical Model
Conceptual Transformation Model
Object Physical Model
ODMG Object Logical Model
Logical Transformation Model
Marked Concep. Model
OMG Object Conceptual Model
Physical Design
MDA Logical Design
MDA Conceptual Model Transformation
MDA Semantic Enlargement

Figure 2: MDA Schema Transformation.

3. Database Transformation

MDA architecture is based on the principle of models and incremental and standardized transformations among them. Similarly, I will try to define a relational database transformation to OODB as a set of models and sequence of transformations among them.

It is not possible to realize the whole transformation process within one transformation step. Data achieved by the database reverse engineering are not sufficient enough for the correct transformation. The human factor intervention is necessary to semantically enlarge the basic data. That is why it is necessary to divide the transition process into several steps. Unlike traditional MDA process it is not enough to transform just models but it is also necessary to ensure the proper data migration.

The whole MDA database transformation process at the highest level consists of two separate but interdependent subprocesses: MDA schema transformation and MDA data migration.

3.1 MDA Schema Transformation

From a procedural perspective, it is possible to apply the MDA theory to DBRE. From the user point of view it will run as a sequence of transformations and models (see Figure 2). Some of them will be automated, but active user's interaction is expected with the others. The first model, a user comes into a contact is the system source code. This model, however, is not platform-independent, and it is not suitable from general and automatic processing point of view. It has to be transformed to platform-independent model – PIM (in terms of RDBMS). CWM Relational Logical Model is extracted through the DDL extraction, explicit information enlargement and model cleaning processes.

The next step is MDA conceptualization. It is the core of the proposed methodology, based on the marking process. Logical model created by the extraction process is necessary to transform to the conceptual model. From the logical model it can be removed the technical, dead structures and structural redundancies during the preparation process.

Relational schema, and thus the logical model extracted from it, does not contain enough information necessary to realize the subsequent transformation to the conceptual model. The type mapping is not sufficient. The solution is a user semantic enlargement of the source model to define the required mappings between elements of both models. The output is marked relational logical model – the original logical model plus user-defined marks applied on CWM relational model elements.

It is therefore necessary to define a group of marks based on the requirements of the MDA conceptualization subprocesses – their metamodel. They extend the conceptual metamodel. The conceptual metamodel extension is the set of defined stereotypes – transformation marks, their tagged values and constraints. Together they form the model, which I will entitle the extended conceptual metamodel.

Implementation width of the possibilities of the mentioned subprocesses and thus the resulting quality of the conceptual model is given by the width and quality of transformation marks design. However, marks are only a support tool for mapping rules application. MDA conceptualization mapping rules form a conceptual transformation model together. It is exactly defined how individual elements and also their structures and relations should be automated transformed from relational logical model to the conceptual one. MDA transformation rules are defined on a metamodel level: at the type level, in accordance to patterns of type usages or on the basis of the marks.

Mapping rules may overlap and combine miscellaneously. An option how to simplify the process of semantic enlargement is to classify rules: to implicit and explicit, or also based on their identical *applicability* – they are applied to the identical structure of the model (or part thereof).

A marked conceptual model and a conceptual transformation model are inputs of the final conceptualization subprocess – MDA logical model transformation. An object conceptual model is the output of a fully MDA automated transformation.

The MDA conceptualization process is followed by a MDA logical design process. The platform-independent conceptual model is the input. The platform-specific logical object model is the output. In contrast to the conceptualization process, it is not necessary to bypass such fundamental semantic and physical differences of the models during the process of the conceptual model transformations into the logical one. However, the semantic enlargement is required again. It is necessary to create the models responsible for the manner of marking and transformation: a marked conceptual model – together with the definition of the source model includes inserted marks of the explicit mapping rules. These marks together with their constraints are defined by the extended logical metamodel. The set of explicit and implicit rules for the conceptual model transformation into the logical object model compose a logical transformation model.

The MDA conceptual model transformation follows the logical model transformation. Its realization is fully equivalent to the process of the relation logical model transformation with the only difference in input and output models (marked conceptual model and logical transformation model on the input side and ODMG object logical model on the output side).

3.2 MDA Data Migration

The second major process of the relational database transformation to the OODB is the data migration. Its aim is to transform the original relational data into the object structures. It depends on the previous MDA schema transformation. I define MDA data migration as an automated transformation of the MDA data model.

The output of each MDA transformation, in addition to the target model, can be also a trace model. Source model elements are transformed into elements of the target model step by step. Their relation is captured by the trace model. Trace model created during the MDA relational schema transformation contains the required mappings (of the attributes specific to relational schema to specific properties of objects). This fact has important implications for the entire data migration process and allows its automation.

There is not just one but two transformations during the MDA schema transformation: relational logical and conceptual model transformations. I also propose two tracing models: the conceptual trace model – mapping CWM relational logical model to the OMG object conceptual model that contains relational schema elements mappings to elements of the conceptual model. The second is the logical trace model – mapping OMG object conceptual model to the ODMG object logical model, which contains conceptual model elements mappings to elements of the object schema.

One of the MDA principles is chaining models and their transformations. We can look at the data migration as a sequence of two successive transformations. MDA data migration, therefore, consists of two subprocesses: conceptual data migration – the source relational data model is transformed into the resulting OMG object data model and logical data migration – the output of the conceptual data migration is transformed into the final ODMG object data model.

The trace model defines what and how should be transformed. Still, it should be defined how. Data are changing during the transformation from one model to another. The data transformation model is responsible for correct

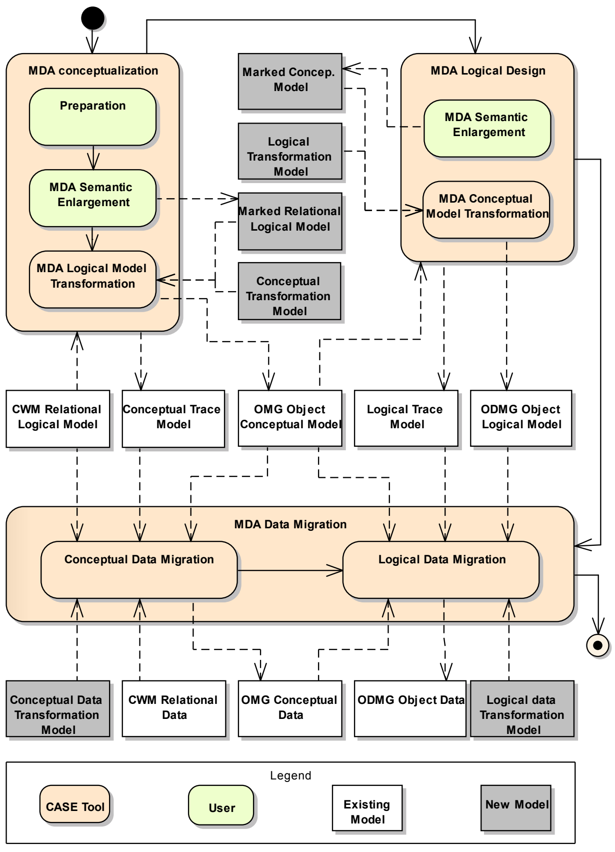
MDA conceptualization
Preparation
MDA Semantic Enlargement
MDA Logical Model Transformation
Marked Concep. Model
Logical Transformation Model
Marked Relational Logical Model
Conceptual Transformation Model
MDA Logical Design
MDA Semantic Enlargement
MDA Conceptual Model Transformation
CWM Relational Logical Model
Conceptual Trace Model
OMG Object Conceptual Model
Logical Trace Model
ODMG Object Logical Model
MDA Data Migration
Conceptual Data Migration
Logical Data Migration
Conceptual Data Transformation Model
CWM Relational Data
OMG Conceptual Data
ODMG Object Data
Logical data Transformation Model
Legend
CASE Tool
User
Existing Model
New Model

Figure 3: MDA Database Transformation.

data migration. Its core consists of mapping rules which are defined by the trace model. The conversion functions are its second part. They are responsible for the physical transformation of the data. An input format and type of data is given by the elements of the source model. Through a trace model, it is possible to find their corresponding target model elements, their types. Depending on the complexity of the source and target elements relation I propose to divide the conversion functions into two groups: standard conversion functions – providing data transformation only at the level of type changes and specific conversion functions – providing more complex transformations where the type transformation only is not sufficient.

The MDA database transformation process and the interconnection of the schema transformation and data migration processes are shown in the figure 3.

Algorithms for the analysis and design of the models used in transformations are generally identical and they can be abstracted. There are used the same procedures and methods during the design of the conceptual metamodel extension, conceptual transformation models of the schema and data on one hand and logical metamodel extension, logical transformation models of the schema and data on the other hand. That is why I have created a general algorithm for the design of transformation models. Applying it I have designed basic models for the transformation of logical and conceptual models and models required for the data migration itself.

Table 1: Basic mappings of the conceptual transformation model

Mapping Name	Source element	Target element
SchToPac	Schema	Package
TabToClas	Table	Class
ColToAtr	Column	Attribute
TypToTyp	SQLDataType	DataType
FkToAsoc	ForeignKey	Association

3.3 Conceptual Transformation Model

From the MDA database transformation process point of view, the first transformation is the logical relational model transformation to a conceptual object model. It is necessary to define a conceptual transformation model for its realization. It consists of two parts: mapping rules – the conceptual transformation model and mapping marks – extended conceptual metamodel.

The first step is to analyze the source and target domain model – their metamodels. The source model of the transformation is the CWM Relational model. The target model is the OMG object model. Both models belong among the MDA standards. The exact specification can be found in [1].

The second step is to identify the required mappings of source CWM Relational elements, their relationships, properties and patterns into the target OMG object elements. The mappings are based on the general DBRE theory.

Table 1 captures the basic mappings between elements of the relational metamodel and the object one. Each row corresponds to one type of mapping (e.g. schema will be transformed into a package). Column Mapping Name defines the abbreviation for the type of mapping. The next two columns contain the element type in the source and target metamodel.

The advanced mapping group includes not only the mappings based on patterns of type usages, but each mapping which relationship cardinality of the source element – the target element is not 1:1. For example *TabToAsoc* is the inverse mapping of an ORM M:N association mapping to a decomposition table. Mapping *FkToInh* is similarly an inverse ORM mapping of the inheritance relationship using pattern the *Table per class* ...

In the third step, it is necessary to define the applicability – *A(M)* for each mapping – *M*. I have formalized a new operator $//_T$ – *type selection* to simplify expressions. A new collection, containing only instances of defined class, is created from the original collection based on that defined class type:

$$\mathit{collection}\ //_T\ \mathit{Class} \equiv \mathit{collection}\ //\ \big(\lambda\ p \bigm| \xi(p) = \mathit{Class}\big)$$

Table 2 defines the applicability of the basic mappings. Similarly, I have defined the applicability of the selected advanced mappings.

Already during the determination of applicability, a semantic enlargement was necessary several times – adding additional information. Semantic enlargement is defined as an addition of the missing semantic information needed for a correct transformation, for example, of a logical model to a conceptual one. I have identified two main

Mapping – M	Type	Applicability – $A(M)$
SchToPac	Schema	$Catalog \lhd ownedElement\ //_T\ Schema$
TabToClas	Table	$Schema \lhd ownedElement\ //_T\ Table$
ColToAtr	Column	$Table \lhd feature\ //_T\ Column$
TypToTyp	SQLDataType	$Column \lhd type$
FkToAsoc	ForeignKey	$Table \lhd ownedElement\ //_T\ ForeignKey$

Table 2: Applicability of the basic mappings of the conceptual transformation model

reasons for its needs: the missing information (such as the type of association) and mapping identification in case of their ambiguity.

Missing (additional) information can be divided into two groups: obligatory – required data that can not be automatically reconstructed in the transformation. The second group contains optional data that can be restored with default value. Optional additional information is not necessary for the transformation, but it improves the quality of the resulting model. For example, additional value *name* of the mappings *M(SchToPac)*, *M (TabToClas)*, *M (ColToAtr)* allows to specify another name for the final element than the source one.

The last step before the extended conceptual metamodel creation is to specify implicit and explicit mappings. It is necessary to define stereotypes for all explicit mappings. They will be used to mark the source model element, and this way, explicitly specify concrete mapping to apply.

Each mapping, which allows additional information, is marked as explicit. Mappings without the mandatory additional information but allowing a standard transformation (without having to enter any supplementary information) are marked as implicit. Ambiguous applicability is another reason for signing the mapping as explicit. It is necessary to divide all mappings to mapping rules families according to the type of applicability and to sign selected mappings as explicit in the case of ambigue applicability.

We have all the necessary information to define an extended conceptual model at this moment. There are the stereotypes identifying corresponding transformation mapping rules defined in this metamodel and providing space for the missing information supplementation. I have created metamodel of the extended conceptual model based on splitting mapping rules to explicit and implicit and additional information (see Figure 4).

Extended conceptual metamodel is only a part of the conceptual transformation model. Transformation rules are its second part. They define a fully automated transformation of the source elements to target elements. Each transformation rule consists of two operations: the selection of a homogeneous set of the source model elements and following collection operation, which transforms them into a set of new target elements. According to the size of this model I present only an example of the mapping *M(SchToPac)*. Transformation *T(SchToPac)* for *M(SchToPac)* can be written as:

$$A(SchToPac) >> (\lambda\ s \mid Package\ [name{:}\ (s \lhd isStereotype{:}$$

$$Renamed\ ?\ s \lhd tag{:}\ newName \diamond s \lhd name)\ ])$$

A(SchToPac) is the applicability already defined in Table 2, that will return the collection of all schemas. A new element Package is created for each schema. Package name is set either to the name of the schema itself or tagged value defined in the stereotype (*newName*) depending on the assignment of the Renamed stereotype in the source schema. There is the explicit and implicit schema mapping realized in the single transformation.

Transformation rules form together the core of the conceptual transformation model. Applying them to the source relational conceptual model it is fully automated transformed into a conceptual object model.

3.4 Conceptual Data Transformation Model

The conceptual transformation model determines the transformation of the logical relational schema into conceptual object model. Similarly, the conceptual data transformation model (CDTM) defines a way to migrate relational data to the object one. The data migration consists of two subprocesses: conceptual and logical data migration. CDTM is the transformation model of the conceptual data migration process.

The MDA transformation model defines transformation rules. They are defined by the MDA conceptualization output trace model in the case of CDTM. Automatically generated trace model defines what has to be transformed and how to be transformed to. There is also a conversion of physical data during data transformation from one model to another. The conversion functions are, therefore, a necessary part of CDTM responsible for the physical data conversion. I have divided them into two groups: standard and specific in section 3.2.

The standard conversion functions perform simple transfer of the source element data type to the target type. It is necessary to determine the appropriate types of source and target metamodel in the process of pairing models types. For each identified pair, whether implicit or explicit, there is defined a conversion function in CDTM:

$$conversionFunctionName{:}\ SQLDataType \rightarrow DataType$$

For example, to convert data of type *Character* to *String*:

$$toString{:}\ CHARACTER \rightarrow String$$

Their implementation is very dependent on the environment in which the data migration will be implemented and executed.

Specific conversion functions are the second group of conversion functions solving complex data conversions. These functions may not differ only in their implementation, but even the declaration. They depend on a transformed relational model. It is not possible to generalize them. All conversion functions compose CDTM and together with the conceptual trace model they determine a way how the

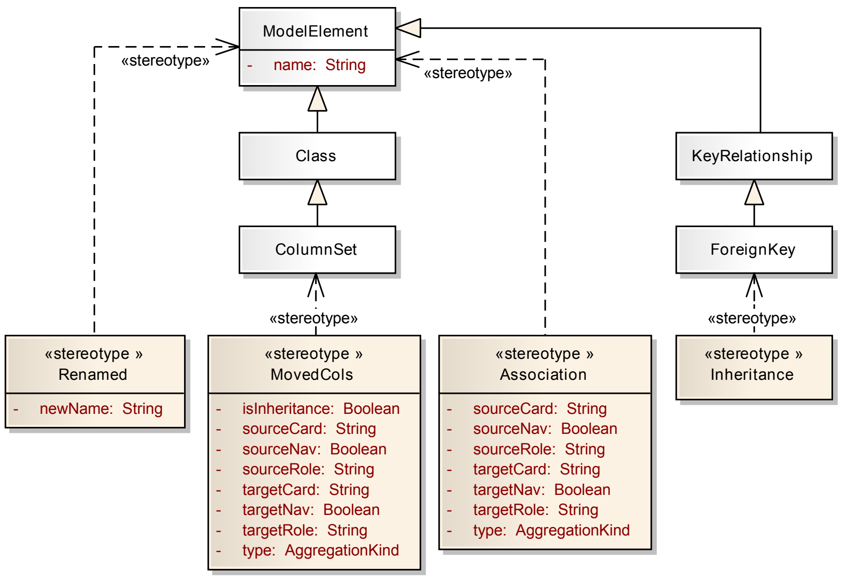

Figure 4: Extended conceptual metamodel.

relational data are migrated to the conceptual ones. Conceptual data are then transformed into the ODMG object data. First, it is necessary to define a logical transformation model.

3.5 Logical Transformation Model

The second major MDA step of the MDA schema transformation is the logical design. The input object conceptual model is marked and then fully automated transformed into the ODMG object logical model. The method and quality of this transformation is given by a logical transformation model: by transformation rules and mapping marks of the extended ODMG logical metamodel.

The process is based on the same principles as in the creation of the conceptual transformation model – based on the general algorithm of the transformation models design. MDA conceptual model transformation does not have to overcome such a fundamental mismatch of the source and target model as in the logical relational model transformation to object one. ODMG object metamodel is just an extension of the OMG Object metamodel. The whole process same as the output model (Figure. 5) is considerably simpler.

Although ODMG is a superset of the OMG metamodel, they are two different metamodels. Basic mappings arrange the transformation of elements from one metamodel to another.

We get a fully automated resulting logical object model applying these transformation rules on a marked conceptual model. One thing that left is to specify the logical data transformation model for a complete description of the MDA relational-object database transformation.

3.6 Logical Data Transformation Model

A logical data transformation model (LDTM) defines how to transform conceptual data into ODMG object data. Transformation rules of this model are one of the automated outputs of the MDA logic design (its trace model). Output data are in the format specified by the OIF standard. The original relational data using CDTM and subsequently LDTM are transformed into OIF data that can be imported into most of the OODBMS.

4. Evaluation

With regard to the solving area, the best way how to verify suggested methodology is the experimental verification. I have applied the designed models to a sample relational model, which I have transformed using a CASE tool to an object model.

The construction of the conceptual and logical transformation models is also a partial verification of the proposed procedures. The next step was their application to a specific input model. I have verified:

- The correctness of transformation rules – whether the source model elements are transformed into the target elements in accordance with the defined mappings.
- The accuracy of the MDA database transformation process – if the proposed process as a sequence of models and transformations between them leads to the desired output – the relational schema transformation into object one with the subsequent fully automated data migration.

The MDA specification is very large and detailed. There is no tool with its full implementation on the market. Usually it is possible only to transform the models. There is a proprietary language to define custom transformation models at best. Borland Together is the exception [2]. The main reason I have decided for it is its almost full QVT implementation.

However there is a problem with the other standards. I have not found any single tool, which would have its UML metamodel based on MDA standards. I solved this problem by adapting transformation models to Borland Together metamodels. So it is possible to adjust the methodology to various also not standardized tools.

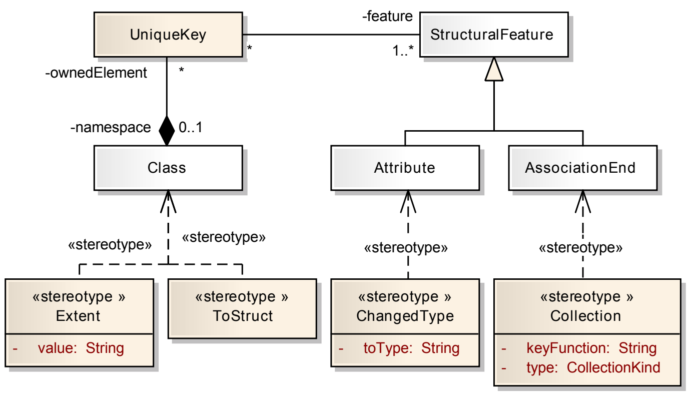

Figure 5: Extended logical metamodel.

I have chosen a relational schema of the GTN system [17, 18] as the input relational model. It is a software system developed at the Department of Software Technology in cooperation with the company AZD Praha. GTN is a graphical and technology extension of the stations electronic signaling equipment. Currently the system is used in several versions by Czech, Slovak and Belarusian railways on more than 2000 km of railway tracks. It is, therefore, a real system successfully operating in practice.

4.1 MDA Schema Transformation

The relational schema transformation into the object one is the first step of the database transformation.

I have fully automated imported relational schema directly from Oracle RDBMS into the CASE tool in the extraction process. The GTN relational model is complete (foreign keys, cardinality relations, . . .) and that is why it was not necessary to perform explicit information enlargement. The created model is based on Borland Together relational metamodel – erPhysical and it is the primary input of the MDA schema transformation.

The key activity from user's point of view is the semantic enlargement. Step by step I have marked the source model by stereotypes defined in the extended conceptual metamodel and inserted required tagged values. The marked relational logical model was created.

This marked relational model was fully automated transformed into the object conceptual model within the MDA logical model transformation.

Once again the conceptual model was marked with stereotypes defined in the extended logical metamodel within the logical design process. This time the marked conceptual model was constructed.

Using the logical transformation model I have transformed this marked conceptual model into the ODMG object logical model fully automated. The result of this M2T transformation was the ODL code specifying the object database schema.

The final ODMG object model was created from the original relational model by gradual models transformations based on the described MDA relational schema transformation methodology. During this process, within the semantic enlargement of the input models (relational or conceptual), I have defined the desired mapping method and appended missing semantic information. The source model elements were transformed into a target one based on the required mappings in a subsequent automated transformation by applying the conceptual and logical transformation models.

4.2 MDA Data Migration

The data migration is the second major process of a relational database transformation into an OODB. Its aim is to transform the original relational data to the object one. It consists of two subprocesses – interlocked data transformations: conceptual and logical data migration.

It depends on the previous MDA schema transformation process. The relational schema transformation precisely defines the relational data mappings to objects. In addition to the output model, the trace log is also generated during the transformation. Trace log created during the MDA transformation contains the required mapping data, which allows automation of the data migration.

The trace log does not contain all the necessary data for the data migration. For example, there is information needed on specific conversion functions. The definition is part of the data transformation model. A user enters to which data items they should be applied during a semantic enlargement in the form of tagged values. They are, therefore, a part of the source schema model. The source model is also one of the data migration input parameters. Based on the data trace log, it is possible to identify the source model element and then access its detailed information in the source schema. So, there is an access to the stereotypes and tagged values there. For more complex transformations it is possible even to access its dependent elements (for example, linked through a foreign key).

It is possible to transform the source relational data into the target object one based on the source schema model containing detailed metamodel of the migrated data, on source elements mappings to the target in a form of the trace log and on data transformation model containing the data conversion functions. Although the data transformation model depends on the transformed model (specific conversion functions depend on the required transformations of the input schema model), the proposed solution simplifies the entire process and it allows automated relational data migration to object one after the completion of the necessary data.

5. Conclusion

In my work, I dealt with the possibility of the relational databases transformation to the object one using MDA. To achieve it, it was necessary to meet several sub-tasks:

- to analyze the database reverse engineering, relational and object concepts, their mutual dependences and problems,
- to survey the current state of the problematic,
- to analyze possibilities of MDA applicability for the forward and reverse database engineering,
- to design MDA processes and models of the schema transformation and data migration,
- to design algorithms how to create transformation models,
- to design conceptual and logical transformation models,
- to evaluate the applicability of the methodology.

I have defined an entirely new methodology of the relational database transformation to the object one based on the sequence of models and transformations among them. Formal underpin of their models description through metamodels, their meaningful integration and transformation among them is essential for the tools creation. The proposed use of industry standards enables the methodology openness.

I have used existing standardized protocols, minimized and simplified essential human interactions during model definitions. Automated transitions from one model to another are realized in accordance with MDA and defined by transformation models through metamodels. A user can visually apply various marks and verify the correctness of their use. The support for graphical display of models using UML modeling tools is one of the advantages of integrating the whole methodology within the MDA framework. A significant advantage of the methodology is its scalability. Separated metamodel specifications of the transformation rules and metamodel extensions into separate models allow defining new rules and marks and make it easier to modify existing ones. Its design based on MDA standards simplifies the acquisition and enlargement.

The result of the proposed methodology is the possibility of an automated transformation of RDBMS to OODBMS. Such created object model meets the requirements for the correct object-oriented design and it is using its advantages, at least influenced by the original relational database and even does not prevent the incorporation of new requirements of the emerging object-oriented database system.

5.1 The Scientific Contribution of the Work

- Identification of problems of the current relational databases transformation solutions.
- The design of the new MDA relational database transformation methodology.
- The design of methods for creating transformation models.
- The design of new types of models.
- The definition of a reverse MDA process.
- Demonstration of the MDA applicability also to other areas than the creation of new software systems.

5.2 The Next Research

The proposed methodology allows the relational database transformation into OODB. I would like to focus on further minimization of the user interaction – a source of potential problems. The main information source of the proposed methodology is a relational schema. I would like to extend the methodology and integrate also other input sources such as stored procedures, views, triggers, the data itself, . . . An interesting source of transformation input data can be application code, especially the DAL layer usually containing the explicit specification of the relational schema mapping to memory object structures.

I would like to pay a special attention to the further improvements of the data migration process. For example, to find a way how to implement and realize it directly in the CASE tool, to specify data migration models in a more detailed way, to change the two-phase data transformation to a single-one, . . .

References

[1] Common warehouse metamodel (cwm) specification, 2003.

[2] Together, visual modeling for software architecture design, 2012.

[3] S. Amer/Yahia, S. Cluet, and C.Delobel. Bulk loading techniques for object databases and an application to relational data, 1998.

[4] M. Anderson. Extracting an entity relationship schema from a relational database thorugh reverse engineering, 1994.

[5] J. Arlow and L. Neustadt. *UML2 and the Unified Process: Practical Object-oriented Analysis and Design*. Addison Wesley, 2005.

[6] A. Behm. *Migrating Relational Databases to Object Technology*. PhD thesis, Wirtschaftswissenschaftlichen Fakultat der Universitat Zurich, Zurich, 2001. PhD thesis.

[7] R. Chiang, T. Barron, and V. Storey. Reverse engineering of relational databases: Extraction of an err model from a relational database, 1994.

[8] J. Fong. *Converting relational to object-oriented databases*, pages 53–58. SIGMOD Record, 1997.

[9] J. Hainaut, M. Chandelon, C. Tonneau, and M. Joris. Contribution to a theory of database reverse engineering, 1993.

[10] J. Hainaut, V. Englebert, J. Henrard, J. Hick, and D.Roland. Requirements for information system reverse engineering support, 1995.

[11] J.-L. Hainaut. Introduction to database reverse engineering, 2002.

[12] J.-L. Hainaut, C. Tonneau, M. Joris, and M. Chandelon. Schema transformation techniques for database reverse engineering, 1993.

[13] J. Henrard, V. Englebert, J.-M. Hick, D. Roland, and J.-L. Hainaut. *Program understanding in databases reverse engineering*. Verso report, INDRIA, 1998.

[14] J. Jahnke, W. Schäfer, and A. Zündorf. Generic fuzzy reasoning nets as a basis for reverse engineering relational database applications, 1997.

[15] J. Janech. Lambda kalkulus: využitie na dotazovanie v distribuovaných databázach, 2012.

[16] W. Ji. An algorithm converting relational schemas to nested entity relationship schemas, 1991.

[17] E. Kršák, H. Bachratý, and V. Polach. Gtn - information system supporting the dispatcher and remote tracks control, 2010.

[18] E. Kršák, H. Bachratý, and V. Tavač. Gtn - information system for railway control on international corridor tracks, 2007.

[19] V. Markowitz and J. Makowsky. Identifying extended entity-relationship object structeres in relational schemes, 1990.

[20] V. Merunka. *Objektové modelování*. Alfa Nakladatelství, Praha, 2008.

[21] W. Premerlani and M. Blaha. An approach for revrse engineering of relational databases, 1994.

[22] M. Shaw and D. Garlan. *Sotware Architecture, Perspectives on an Emerging Discipline*. Prentice Hall, 1996.

[23] O. Signore, M. Loffredo, M. Gregori, and M. Cima. Reconstruction of er schema from database applications: a cognitive approach, 1994.

Selected Papers by the Author

J. Janech, T. Bača, M. Tavač. Distributed database system in ad-hoc networks. In *Datakon*, Ostrava : Ostravská univerzita v Ostravě, 2010.

M. Kardoš, Z. Bizoňová, M. Tavač. Notation for computational independent model in model driven architecture. In *TRANSCOM – 8-th European conference of young research and science workers*, Žilina, Slovak Republic : University of Žilina, 2009.

M. Tavač. MDA based object database transformation. In *TRANSCOM – 7-th European conference of young research and science workers*, Žilina, Slovak Republic : University of Žilina, 2007.

M. Tavač, M. Kardoš, M. Murín. Object-relational mapping and MDA. In *Datakon*, Praha : Vysoká škola ekonomická v Praze, 2009.

M. Tavač, V. Tavač. MDA relational database transformation into OODB. database transformation. In *Objekty*, Ostrava : VŠB - Technická univerzita, 2007.

M. Tavač, V. Tavač. DBRE and MDA integration. database transformation. In *Objekty 2011 : proceedings of the 16th international conference on object-oriented technologies.*, Žilina : University of Žilina, Faculty of Management Science and Informatics, 2011.