META-MODELS IN SUPPORT OF DATABASE MODEL TRANSFORMATIONS

Sonja Ristić, Slavica Aleksić, Milan Čeliković, Ivan Luković
University of Novi Sad, Faculty of Technical Sciences,
Trg Dositeja Obradovića 6
21000 Novi Sad, Serbia
(sdristic, slavica, milancel, ivan)@uns.ac.rs

ABSTRACT
Model-Driven Software Engineering (MDSE) aims to provide automated support for the development, maintenance and evolution of software by performing transformations on models. During these transformations model elements are traced from a more abstract model to a more concrete model and vice versa, achieved through meta-modeling. Software development process produces numerous models of complex application artifacts, such as application programs, databases, web sites or user interfaces. In the paper we focus on models related to databases. For these models we use a generic name database models. They may be created at several, usually different levels of abstraction. In order to specify and generate model transformations between these database models, theirs meta-models have to be defined. In the paper, we propose a classification of database models and meta-models that are involved in the database model transformations. Also, we present a meta-model of relational database schema specified by means of the Eclipse Modeling Framework (EMF) and based on the EMF Ecore meta-meta-model which is closely aligned with the Essential MOF (EMOF) specification.

Keywords: Model-driven Software Engineering, Meta-modeling, Database Reengineering, Intensional database meta-models.

1 INTRODUCTION
The emergence of large and complex software systems increases the interest in Model-Driven Software Engineering (MDSE), as a way to lower the cost of development and maintenance of software. Models allow us to hide irrelevant details, provide different model viewpoints, and isolate and modularize models of cross-cutting concerns of a system under study. Models, as first class entities [26], are used to specify, simulate, test, verify and generate code for the application to be built [8]. Many of these activities include the specification and execution of model-to-model (M2M) or model-to-text (M2T) transformations. During these transformations model elements are traced from a more abstract model to a more concrete model and vice versa, achieved through meta-modeling [1]. A meta-model defines the modeling language, i.e. the constructs that can be used to make a model and, consequently, defines a set of valid models [4]. In that way the execution of M2M transformations of a model conformant to a meta-model into another one conformant to a different meta-model is facilitated. The most mature formulation of MDSE paradigm currently is the OMG’s Model-Driven Architecture (MDA) which refers to a high-level description of an application as a Platform Independent Model (PIM) and a more concrete implementation-oriented description as a Platform Specific Model (PSM) [22]. The OMG’s Meta Object Facility (MOF) defines the metadata architecture that lies at the heart of MDA. MOF standard [20] offers a generic framework that combines both syntax and semantics of models and model transformations. MOF meta-modeling architecture is defined in a way that meta-models and models based on it can be linked together using a simple language. MOF is used to define semantics and structure of generic meta-models or domain specific ones. It provides a four-level hierarchy, with levels M0–M3. The concept of a model is specialized depending on the level, in which a model is located. Therefore it is: a model at M1 level, a meta-model at M2 level and a meta-meta-model at M3 level.

In MDSE generally, as well as in MDA in particular, models are not just designer artifacts, but they are included in production process meaning that a code for target platform may be generated from such models. These models differ in how much platform specific information they contain. A platform should not be seen just as an execution infrastructure. Atkinson and Kuhne in [5] a platform
view “as any system capable of supporting the
fulfillment of some goal with respect to a software
application”. They emphasize that platform
independency is not a binary property, and therefore
one can view several PSMs, with different degree of
platform independency. Therefore, a designer starts
with a high-level model, abstracting from all kinds of
platform issues. Through the chain of M2M
transformations, ending up with a M2T transformation, initial PIM iteratively transforms to a
series of PSMs with less independency degree,
introducing more and more platform specific
extensions.

Meta-modeling is widely spread area of research.
Since software development process produces
several models, going from abstract to concrete,
there is a broad space of problems involving the
design, integration and maintenance of complex
application artifacts, such as application programs,
databases, web sites and user interfaces (UI).
Engineers use tools to manipulate models of these
artifacts, such as class diagrams, interface definitions,
database schemas, web site layouts, XML schemas,
and UI form specifications.

In the paper we focus on models relating to
databases. For these models we use the generic name
database models.

A database is a collection of related data stored
on some storage medium controlled by the database
management system (DBMS). The description of
database that is specified during database design is
called database schema. A data model provides the
means to achieve data abstraction and to express
database schema. According to Date and Darwen
definition [12] revised by Eessaar [13]: "A data
model is an abstract, self-contained, implementation-
independent definition of elements of a 4-tuple of sets
(T, S, O, C) that together make up the abstract
machine with which database users interact, where T
is a set of data types; S is a set of data structure
types; O is a set of data operation types; C is a set of
integrity constraint types.” Numerous data models
are proposed. Elmasri and Navathe classify data
models according to the types of concepts they use to
describe the database structure, as follows: i) high-
level or conceptual data models; ii) representational,
logical or implementation data models, and iii)
low-level or physical data models [15].

Some of the well-known data models are:
hierarchical, network, entity-relationship (ER),
extended ER (EER), relational, object-oriented and
object-relational (OR). Some of them are used
mostly for the conceptual database schema design
(like ER and EER data models), while the others are
used predominantly for logic and implementation
database design and database implementation (like
relational and OR data model). A database schema
has to conform to a data model. A database
management system (DBMS) is based on a data
model, too. Hence, there are relational DBMS
(RDBMS) and OR DBMS, e.g. The plethora of
models related to databases points out to the need
and importance of M2M and M2T transformations
between these database models. Thereby, the
abstraction level of target model of a transformation
may be the same, lower or higher comparing to the
abstraction level of source model.

An explicit representation of mappings specifies
how two models are related to each other. Some
mapping examples, according to Bernstein [7] are: i)
mapping between an entity-relationship (ER) model
and a SQL schema to navigate between a database
schema conceptual design and its implementation; ii)
mapping between class definitions and relational
schemas to generate object wrappers; iii) mapping
between data sources and a mediated schema to drive
heterogeneous data integration; iv) mapping between
a database schema and its next release to guide data
migration or view evolution, etc. Additionally, the
growth of eXtended Markup Language (XML)
technologies has led to the need to have object-
oriented (OO) wrappers for XML data and the
translation from nested XML documents into flat
relational databases and vice versa.

In a forward engineering process there is a chain
of M2M model transformations, ending up with M2T
transformation that transform a conceptual database
schema, via an implementation database schema and
a physical database schema, into an SQL script e.g.,
aimed at creating database under the vendor specific
RDBMS. The abstraction level of models is
decreasing throughout the chain of transformations.

In a reverse engineering process the abstraction
level of models is increasing throughout the chain of
transformations. Starting from a physical database
schema, recorded into RDBMS data repository e.g., a
logical database schema (based on the relational data
model) or a conceptual database schema (ER or EER
data schema) could be extracted. Both of them
are at the higher abstraction level than the physical
database schema.

To manage heterogeneous data, many applications
need to translate data and their descriptions from one
model (i.e. data model) to another. Even small
variations of models are often enough to create
difficulties. For example, Structured Query Language
(SQL) is currently available in most commercial and
open-source RDBMSs. It is also the focus of a
continuous standardization process, resulting in SQL
standards (the current revision is: SQL:2011,
ISO/IEC 9075:2011). However, issues of SQL code
portability between major RDBMS products still exist
due to a lack of full compliance with the standard and
proprietary vendor extensions. Therefore, even the
mapping between SQL database schemas extracted
from RDBMS data repository of different vendors
may be a serious problem.
Model transformations between database models we call database model transformations. These transformations are based on meta-models that are conformed by the source and target database models of the transformations. These meta-models are said to be in support of database model transformation. Due to the diversity of database models it is important to classify meta-models of database models and to distribute them across the abstraction level stack. In the paper we propose a classification of database meta-models. We have designed a meta-model of relational database schema based on the theoretical foundations of relational data model. It is presented in the paper to illustrate abundance and diversity of relational data model constructs, alongside with semantics that may be expressed in logical relational database schema.

In the purpose of specifying and managing our meta-model we use the Eclipse Modeling Framework (EMF) [14], a current MOF-like modeling environment. The EMF meta-modeling language is based on the Ecore meta-meta-model which is closely aligned with the Essential MOF (EMOF) specification [20].

Apart from the Introduction and Conclusion, the paper has four sections. In Section 2 a classification of database meta-models is presented. Section 3 is devoted to a relational database schema meta-model organized in several packages. An example aimed at better explanation of the concepts that are introduced in Section 3, is presented in Section 4. Related work is elaborated in Section 5.

2 A CLASSIFICATION OF DATABASE META-MODELS

The work we describe in this paper unifies two main research areas: database design and implementation and meta-modeling in the context of MDSE.

We identify different kinds of database meta-models (MMs) that are models of modeling languages used to express database models at certain abstraction level. Hereof, we distinguish:

- data model meta-models;
- generic database schema meta-models:
  - generic conceptual database schema meta-models,
  - generic logical database schema meta-models;
- standard physical database schema meta-models; and
- vendor-specific physical database schema meta-models.

In Table 1 we propose a classification and distribution of database meta-models and database models across the MOF level stack. System under study (SUS) is at the M0 level. An SUS is represented by a model at M1 level, which conforms to a meta-model at M2 level that is conformant with a meta-meta-model at M3 level. For example, in column (5) in Table 1, a database instance may be represented by an Oracle 10g database schema conformant with the Oracle 10g database schema meta-model conformant with EMOF.

Data model meta-models stand at the M2 level of MOF stack. For example, one may specify relational data model meta-model or ER data model meta-model, and they are containing constructs like data types, data structure types, constraint types, etc. They are specific for relational or ER data model, respectively. In a generic approach we can assume that besides well-known data models may emerge new data models and their meta-models may be included in this classification.

Some of the generic database schema meta-models describe conceptual database schemas, like ER or EER database schema MM, while others describe logical database schemas, like relational or OR database schema MM. In both cases they are based on theoretical foundations of ER/EER, relational or OR data model, respectively.

Relational data model is the focus of continues standardization process, and therefore we have extracted the standard physical database schema meta-models according to the specific SQL standard. But, the conformance of a vendor database management system with a SQL standard by the rule is not complete. That is the reason why we introduce class of vendor-specific physical database schema meta-models.

MOF stack presented in column (1) of Table 1 is specific in relation to the MOF stacks in other columns. SUS in column (1) is at the higher abstraction level then SUSs in other columns. For example, a SUS may be a generic, logical relational database schema of UniversityDb information system (UNIRDBS). This SUS is represented by generic relational database schema meta-model that is conformant with relational data model meta-model.

In columns (2), (3) and (4) SUS is the same—logical data structure of a database, that may be represented by a conceptual database schema, a logical database schema or a database schema based on an SQL standard, respectively. If it is, for example, represented by generic, logical relational database schema of UniversityDb information system (UNIRDBS), then UNIRDBS is at the M1 level of MOF stack in column (3), while it is at the M0 level of the MOF stack in column (1).

SUS in column (5) is a database instance that is represented by an Oracle 10g database schema of UniversityDb IS (UNIORA), e.g. UNIORA conforms to Oracle 10g database schema meta-model.
### Table 1: A classification of database meta-models

<table>
<thead>
<tr>
<th>MOF level</th>
<th>MOF Architecture</th>
<th>Conceptual database schema MM</th>
<th>Implementation database schema MM</th>
<th>Physical dbS MM</th>
</tr>
</thead>
<tbody>
<tr>
<td>M3</td>
<td>EMOF/CMOF/Ecore</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M2</td>
<td>Data model meta-model (MM)</td>
<td>Conceptual database schema MM</td>
<td>Logical database schema MM</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Standard database schema (dbS) MM</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Vendor-specific Physical dbS MM</td>
<td></td>
</tr>
<tr>
<td>(1)</td>
<td>(2)</td>
<td>(3)</td>
<td>(4)</td>
<td>(5)</td>
</tr>
<tr>
<td></td>
<td>(1)</td>
<td>(2)</td>
<td>(3)</td>
<td>(4)</td>
</tr>
<tr>
<td></td>
<td>Relational database schema MM, SQL:2003 database schema MM …</td>
<td>Oracle 10g database schema MM, MySQL database schema, dBase III+ database schema</td>
<td>Oracle 10g database schema of UniversityDb IS, MySQL database schema 1, Oracle10g database schema 1, MySQL database schema 1</td>
<td></td>
</tr>
<tr>
<td>M1</td>
<td>ER database schema MM, …</td>
<td>ER database schema 1, ER database schema 2, …</td>
<td>Relational database schema of UniversityDb IS, Relational database schema 2, …</td>
<td>Oracle 10g database schema of UniversityDb IS, MySQL database schema 1, Oracle10g database schema 1, MySQL database schema 1</td>
</tr>
<tr>
<td>M0</td>
<td>ER database schema 2, …</td>
<td>Logical data structure of a database</td>
<td>Database instance</td>
<td></td>
</tr>
</tbody>
</table>

The UNIORA2UNIRDBS transformation, which is aimed at transforming an Oracle 10g physical database schema to a logical, relational database schema, is based on Oracle 10g database schema meta-model and generic relational database schema meta-model. Therefore, these two meta-models are in support of database model transformation UNIORA2UNIRDBS.

If Table 1 is viewed by columns in accordance with the OMG’s MDA model classification it can be concluded that:

- models from column (5) are fully PSM, since they are specific for a platform and its vendor;
- models from column (4) are PSM, but with the lower level of platform specificity then the models from column (5);
- models from columns (1) and (3) can be seen as PSM or as PIM depending on platform context—UNIRDBS is PSM since it conforms relational data model, but it is PIM since it can represent different vendor specific physical database schemas e.g.; and
- models from column (2) are PIMs.

Our classification and distribution of database models across the MOF level stack will enable systematic approach for mapping specification between different models/meta-models and development of appropriate M2M or M2T transformations.

In order to do that, corresponding meta-models have to be specified. In this paper we are presenting a relational database schema meta-model, according the theoretical definition of relational data model, ([12], [15], [21]).
3 A RELATIONAL DATABASE SCHEMA META-MODEL

Proposed meta-model is fairly huge and complex, so we use packages to organize the meta-model.

Modeling concepts in the relational database schema meta-model (RDBSMM) are: attribute, constraint, relation scheme, Universal Relational Schema (URS), relational database schema and project (Fig. 1). In our approach we want to support different database design approaches, and therefore we include URS to support database design approaches based on the URS assumption [21]. A database design methodology based on such approach extracts relational database schema from:

- set of attributes associated with domains,
- set of functional dependencies and
- set of non-trivial inclusion dependencies.

A database project is composed from URS and relational database schema. As can be seen in Fig. 2 database constraints may be specialized as: URS constraints, relational constraints and multi-relational constraints. The package representing URS meta-model is presented in Subsection 3.1, and the package representing the relational database schema concept meta-model is presented in Subsection 3.2.

In order to make some of the meta-model concepts clearer in Section 4 we give an example of a relations database schema instantiating some of the concepts presented here.

![Figure 1: The relational database schema modeling concepts](image1)

![Figure 2: A meta-model of project concept](image2)

3.1 A URS meta-model

Basic constructs of URS meta-model are: attribute and three possible kinds of URS constraints: domain, functional dependency and non-trivial inclusion dependency (see Fig. 3). Domain (DomainCon) can be primitive (predefined) domain (PrimitiveDomain) or user defined domain.
(UserDefDomain) that can inherit primitive domain (UserDefDomainFromPrimitiv) or previously defined user defined domain (UserDefDomainFromUserDef). Each attribute is associated with one and only one domain. For a functional dependency (fd) the sets of attributes on the left-hand and the right-hand sides of fd are specified. The set of attributes on the right-hand side of the fd may be empty. Unlike fd, both the left-hand and the right-hand side attribute sets of an inclusion dependency (InclusionDependencyURS) are non-empty.

3.2 A meta-model of relational database schema concept

A relational database schema is composed of a set of relation schemes and a set of multi-relational constraints (Fig. 4). A relation scheme is composed of a set of attribute value constraints (AttValCon), a set of unique constraints (UniqueCon), a set of key constraints (KeyCon) and a set of check (tuple) constraints (CheckCon) (see Fig. 4). All of these constraints are specializations of relational scheme constraint concept (RelationCon in Fig. 4).

Figure 3: A meta-model of URS concept

Figure 4: A meta-model of relation scheme concept

In Fig. 5 a meta-model of inclusion dependency (IND) concept (InclusionDependency) is presented. The IND concept may be specialized as key-based IND (referential integrity constraint,
RIC, meta-model concept *ReferentialIntegrityCon* or as non-key-based IND (*NonKeyBasedIND*). The non-key-based IND concept is further specialized as inverse referential integrity constraint (RIC, meta-model concept *InverseReferentialIntegrityCon*) and as non-IRIC (*NonInverseReferentialIntegrityCon*). Each of RIC, IRIC and non-IRIC concepts may be further specialized as extended RIC (*ExReferentialIntegrityCon*), extended IRIC (*ExInverseReferentialIntegrityCon*) and extended non-IRIC (*ExNonInverseReferentialIntegrityCon*), respectively. Detailed description of RICs and IRICs may be found in [3]. In Section 4 may be found some instances of aforementioned IND concept and its specializations.

Finally, a meta-model of extended tuple constraint is presented in Fig. 6.

![Figure 5: A meta-model of inclusion dependency concept](image)

![Figure 6: A meta-model of extended tuple constraint concept](image)

4 AN EXAMPLE OF RELATIONAL DATABASE SCHEMA

Some kinds of constraints meta-modeled in previous section are well-known and can be implemented by the declarative DBMS mechanisms (like key constraint and RIC). However, some kinds of constraints are not recognized by contemporary DBMSs and have to be implemented through the procedural mechanisms. Very often these kinds of constraints are ignored by database designers in a way that they do not recognize, specify and implement them (like IRIC, selective IND and extended IND). We believe that all kinds of constraints are important to be specified and implemented to achieve the best possible database consistency. That is the reason why we decide to create relational database schema meta-model comprising all kinds of constraints according to theoretically defined relational data model. Here we use the example of University database schema to
explain some kinds of constraints that are not broadly accepted within database designers’ community. In Fig. 7 the conceptual database schema of University database is visually represented by means of UML class diagram to facilitate better understanding of database constructs and relationships between them. The relational database schema University contains the set of relation schemes: Employee, University, Department, WorkSite, Course, EmployedAt and Taught_By, accompanied with the set of multi-relation constraints. A relation scheme is specified as named pair $N(R_i, C_i)$, where $N_i$ is the relation scheme name, $R_i$ set of attributes, and $C_i$ set of relation scheme constraints. An inclusion dependency is a statement of the form $N_i[LHS] \subseteq N_j[RHS]$, where LHS and RHS are non-empty arrays of attributes from $R_i$ and $R_j$ respectively. Having the inclusion operator (⊆) orientated from the left to right we say that relation scheme $N_i$ is on the left-hand side of the IND, while the relation scheme $N_j$ is on its right-hand side. In the following text we enumerate relation schemes and multi-relation constraints of University database schema and give the explanation of specified constraints.

Figure 7: The conceptual database schema of University database

```
Employee({EmpId, EmpFName, EmpLName, EmpBirthD, EmpSSN, EmpPosition, PassportNo, SupervisorId, WSId},
{PrimaryKey(EmpId), EquivalentKey(EmpSSN), Unique(PassportNo)},
CheckCon((EmpPosition = 'Prof' ∨ EmpPosition = 'Assistant') ⇒ WSId IS NOT NULL))
```

Relation schema Employee has two keys (key constraints over the set of attributes) EmpId and EmpSSN. These constraints are represented by the KeyCon concept of RDBSMM (Fig. 4). One of them (EmpId) is primary key. The other one is equivalent key. Unique constraint is specified for attribute PassportNo since it is nullify attribute in Employee and therefore can not be the part of any key of Employee, but if it has value it must be unique within the relation over relation scheme Employee. It is represented by the UniqueCon concept of RDBSMM (Fig. 4). Specified check constraint models a business rule that University professors and assistants must have worksite (office), while other employees need not. In RDBSMM it is represented by CheckCon concept (Fig. 4).

```
University({UniId, UniName, UniCity, RectorId},
{PrimaryKey(UniId)})
Department({UniId, DepId, DepName, DeanId },
{PrimaryKey(UniId + DepId)})
WorkSite({WSId, WSLoc, UniId, DepId},
{PrimaryKey(WSId)})
Course({UniId, DepId, CourseId, Semester, LectureClassesPW, LabClassesPW}
{PrimaryKey(UniId + DepId + CourseId)})
Employed_At({EmpId, UniId, DepId, PartTimePct},
{PrimaryKey(EmpId + UnId + DepId)})
Taught_By({UniId, DepId, CourseId, EmpId, ClassesPerWeek},
PrimaryKey(UniId+DepId+CourseId+EmpId)})
```

1. Employee[SupervisorId] ⊆ Employee[EmpId] This is an example of the RIC, since that EmpId is an equivalent key of relation scheme Employee at the right-hand side of RIC. In RDBSMM it is represented by ReferentialIntegrityCon concept (Fig. 5). The RIC is the consequence of URS IND [SupervisorId] ⊆ [EmpId], that is represented...
by InclusionDependencyURS concept in RDBSMM (Fig. 3).

2. \( \text{Employee[WSId]} \subseteq \text{WorkSite[WSId]} \)
   This is an example of the RIC, since that WSId is the primary key of relation scheme WorkSite that is on the right-hand side of the IND.

3. \( \text{WorkSite[WSId]} \subseteq \text{Employee[WSId]} \)
   The specified constraint is an IRIC, since there is specified RIC (item 2), and WSId is the primary key of relation scheme WorkSite that is on the left-hand side of the IRIC presented in this item (item 3). In RDBSMM it is represented by \( \text{InverseReferentialIntegrityCon} \) concept (Fig. 5).

4. \( \text{University[RectorId]} = \sigma_{\text{EmpPosition} = \text{'Prof'}} \text{Employee[EmpId]} \)
   Here we have an example of selective RIC, since there is a selection condition \( \text{EmpPosition} = \text{'Prof'} \) on the right-hand side of IND. The selection condition can be specified using the feature \( \text{SelectionCon}_R \) of InclusionDependency concept from the meta-model in Fig. 5. This constraint models a business rule that the rector of the university may be only an employed professor.

5. \( \text{Department[UniId]} \subseteq \text{University[UniId]} \) (RIC)

6. \( \text{University[UniId]} \subseteq \text{Department[UniId]} \) (IRIC)

7. \( \text{Department[DeanId]} \subseteq \sigma_{\text{EmpPosition} = \text{'Prof'}} \text{Employee[EmpId]} \)
   This is another selective IND modeling a business rule that the dean of a department may be only an employed professor.

8. \( \text{WorkSite[UniId + DepId]} \subseteq \text{Department[UniId + DepId]} \) (RIC)

9. \( \text{WorkSite} \bowtie \text{Employee[EmpId + UniId + DepId + WSId]} \subseteq \text{Employed_At[EmpId + UniId + DepId + WSId]} \)
   Here we have an example of extended non-IRIC. It is extended for the fact that on the one of the IND sides (here on the both of them) there is a join of at least two relations. It is non-IRIC since the array of attributes \( \text{EmpId + UniId + DepId + WSId} \) is not the equivalent key neither for the relation scheme on the left-hand side nor for the relation scheme on the right-hand side of the IND. In RDBSMM it is represented by \( \text{ExNonInverseReferentialIntegrityCon} \) concept (Fig. 5). The constraint models a business rule that an employee can have only one office and that office has to be located in the worksite that is under control of a department that is one of the departments in which the employee is employed.

10. \( \text{Course[UniId + DepId]} \subseteq \text{Department[UniId + DepId]} \) (RIC)

11. \( \text{Department[UniId + DepId]} \subseteq \text{Course[UniId + DepId]} \) (IRIC)

12. \( \text{Employed_At[EmpId]} \subseteq \text{Employee[EmpId]} \) (RIC)

13. \( \text{Employee[EmpId]} \subseteq \text{Employed_At[EmpId]} \) (IRIC)

14. \( \text{Employed_At[UniId + DepId]} \subseteq \text{Department[UniId + DepId]} \) (RIC)

15. \( \text{Taught_By[UniId + DepId + CourseId]} \subseteq \text{Course[UniId + DepId + CourseId]} \) (RIC)

16. \( \text{Taught_By[EmpId + UniId + DepId]} \subseteq \sigma_{\text{EmpPosition} = \text{'Prof'}} \text{Employee[EmpId + UniId + DepId]} \)
   This is an example of selective extended IND that models businesses rule that only a professor or an assistant that is employed at the department that offers a course may be engaged as a teacher of the course. This constraint is represented by ExNonInverseReferentialIntegrityCon concept alongside with the feature \( \text{SelectionCon}_R \) of InclusionDependency concept of RDBSMM (Fig. 5).

17. \( (\forall t \in \text{Employee} \bowtie \text{Taught_By} \bowtie \text{Course}) ((\{t[\text{EmpPosition}] = \text{'Prof'} \Rightarrow t[\text{ClassesPerWeek}] \leq t[\text{LectureClassesPW}]) \land (\{t[\text{EmpPosition}] = \text{'Assistant'} \Rightarrow t[\text{ClassesPerWeek}] \leq t[\text{LabClassesPW}]) \))
   Here is an example of extended tuple constraint. It is extended since it mutually constraints values of the attributes from different relations, but it is tuple constraint since these values are from only one tuple that belongs to a join of at least two relations. In RDBSMM it is represented by \( \text{ExTupleCon} \) concept (Fig. 6). This constraint models the business rule that a professor may teach only lecture classes, and therefore, classes per week that he/she has for that course has to be less or equal than the number of laboratory classes for the course per week. Besides, an assistant may teach only laboratory classes, and therefore, classes per week that he/she has for that course has to be less or equal than the number of laboratory classes for the course per week.

5 RELATED WORK

Mapping of object/OR/ER/EER models to relational database schemas and vice versa has been widely used as a case study to present new model transformations proposals.

Atzeni, Cappellari and Gianforme in [1] propose a framework focused on schema mappings. The proposal is based on a relational database formal basis, but the usage of a new meta-meta-model (known-as Supermodel), different from MOF, makes it hard to develop bridges towards the universe of MOF-compliant proposals.

The importance of generic models is also emphasized by Atzeni, Gianforme and Cappellari in...
[2]. They have shown how a meta-model approach can be the basis for numerous model-generic and model-aware techniques. A dictionary to store their schemas and models, a specific supermodel (a data model that generalizes all models of interest) is presented, too. They presented a classification of data model constructs and their distribution beyond six data models.

Gogolla et al. in [17] have sketched how syntax and semantics of the ER and relational data model and their transformation can be understood as platform independent and platform specific models. Presented ER and relational meta-models are very simple and can not be classified according meta-model classification presented in our work. This paper is interesting in another context: it presents the intensional and extensional ER/relational meta-models. The relational database schema meta-model that we presented in this paper is an intensional meta-model. Our future research has to consider extensional database meta-models, too.

Polo, Garcia-Rodriguez and Piattini in [23] present the technical and functional descriptions of a tool specifically designed for database re-engineering. In the case study they propose simplified relational and object-oriented meta-model. Both of them are to simple to be classified according to meta-model classification presented in our paper.

The similar, simplified RDBMS meta-model is presented in [28], where Wang, Shen and Chen emphasize that the assumption that the DDL statements can be extracted easily through DBMS is not always true.

In paper [27], the authors propose through a case study supported by a tool, a model-driven development of OR database schemas. To that end, Vara et al. have implemented an ATL model transformation that generates an OR database model from a conceptual data model and a MOFScript model to text transformation that generates the SQL code for the modeled database schema. As part of the proposal they have defined a MOF-based Domain Specific Language (DSL) for OR database modeling as well as a graphical editor for such DSL. They presented Oracle 10g meta-model that can be classified as vendor-specific physical database schema meta-model according to the classification presented in our paper.


In [11] a process is proposed to automatically generate Web Services from relational databases. SQL-92 meta-model has been used to represent the database model, that can be classified as standard physical database schema meta-model according to the classification presented in our paper.

Calero et al. in [10] have introduced ontology for increasing the understandability of the SQL:2003 concepts. Their SQL:2003 meta-model can be seen as a standard database schema meta-model.

Cabot and Teniente in [9] presents an OCL meta-model that defines a set of techniques and a method of their integration, for the efficient checking of OCL integrity constraints specified in a UML conceptual schema. Cabot et al. in [8] present a new method for the analysis of declarative M2M transformations based on the automatic extraction of OCL invariants implicit in the transformation definition. In a case study, they used simplified UML class meta-model, that can be classified as a generic database schema meta-model according to the classification presented in our paper.

Guerra et al. in [16] stress that model transformations should be engineered, not hacked. For this purpose, they have presented transML, a family of languages to help building transformations using well-founded engineering principles. They presented platform meta-model, meta-model of the specification languages and mapping meta-model. They are not in the direct correlation with the results presented in our paper, but may be interesting in our further research of the database re-engineering process.

In the paper [13] Eessaar explained why it is advantageous to create meta-model of a data model. He demonstrated that a meta-model could be used in order to find similarities and differences with other data models.

Beggar, Bousetta and Gadi [6] propose a reverse engineering process based on MDSE that presents a solution to provide a normalized relational database which includes the integrity constraints extracted from legacy data. They extract entirely the description of legacy data from only source code and physical files. COBOL file section meta-model is proposed, that can be classified as generic file schema meta-model.

6 CONCLUSION

The value of models and abstractions in software development is substantial in order to master system complexity. MDSE has become a commonly used approach in software engineering. It promotes using models as primary artefacts and proposes methods for transforming them to desired software products. Model transformations are defined in order to bridge different modeling languages or to map between representations in the same language. Complex data mapping tasks often arise in MDSE. A number of data models are in common use and each data model provides slightly
different modeling structures. One important advantage of having multiple data models is that developers can select the data model that offers the most convenient representation for their particular needs. However, the use of multiple data models introduces the possibility of many kinds of structural heterogeneity. Therefore, the transformations between different database models are very important. That was our motivation to focus on database models and meta-models they are conformant with. Meta-modeling is widely spread area of research and there is a huge number of references covering MOF based meta-models. It is easy to conclude, based on the literature review in the previous section, that a lot of authors use or propose different database meta-models. However, to the best of our knowledge, we could not find any systematical overview of database meta-models at different abstraction levels.

The main contribution of our paper may be two folded: through the database meta-model classification presented in Section 2 and through detailed relational database schema meta-model, as proposed in Section 3. We believe that both contributions will enable our future efforts directed towards automating database and information system re-engineering process based on MDSE principles. We have used presented meta-model of relational database schema to develop a chain of M2M transformations starting with a legacy relational database schema and to integrate them with our IIS*Studio development environment [3]. The chain of these transformations enables reverse engineering, while IIS*Studio tool is used for forward engineering and generating executable application prototypes.

Category theory ([24], [25]) provides a kind of common language and tool, in which a sketch is a specification based on graphs as the formal structure. We plan to investigate a possible usage of category theory for PIM specification of model transformations in order to automate the process of database model transformations generation.

ACKNOWLEDGEMENT

Research presented in this paper was supported by Ministry of Science and Technological Development of Republic of Serbia, Grant III-44010, Title: Intelligent Systems for Software Product Development and Business Support based on Models.

7 REFERENCES


