Computer Science and Information Systems 11(2):679–696 DOI:10.2298/CSIS140216037R

Generic and Standard Database Constraint Meta-Models

Sonja Ristić[1], Slavica Aleksić[2], Milan Čeliković[2] and Ivan Luković[2]

[1]University of Novi Sad, Faculty of Technical Sciences,
Department for Industrial Engineering and Management
Trg Dositeja Obradovića 6
21000 Novi Sad, Serbia
sdristic@uns.ac.rs
[2]University of Novi Sad, Faculty of Technical Sciences,
Department of Computing and Control
Trg Dositeja Obradovića 6
21000 Novi Sad, Serbia
{slavica, ivan, milancel}@uns.ac.rs

Abstract. Many software engineering activities entail dealing with legacy information systems. When these systems become too costly to maintain, or when new technologies need to be incorporated, they need to be replaced or somehow reengineered. This can be done with significantly reduced amount of effort and cost if the conceptual models of these systems are available. Reverse engineering is the process of analyzing a subject system to create representations of the system at a higher level of abstraction. Relational databases are a common source of reverse engineering. Starting from a physical database schema, that is recorded into relational database schema data repository, the conceptual database schema or logical database schema could be extracted. The extraction process may be seen as a chain of model-to-model transformations that trace model elements from a model at the lower level of abstraction to a model at the higher level of abstraction, achieved through meta-modeling. In the paper we present generic and standard database constraint meta-models, focusing on multi-relational database constraints captured in a legacy database. These meta-models are aimed at support of model transformations to create conceptual models, as a useful source for the system reengineering process.

Keywords: Model-driven Software Engineering, Meta-modeling, Inclusion Dependency, Database Reengineering.

1. Introduction

The evolution in organization procedures and objectives over the time significantly reduces the effectiveness of an information system implemented to fulfill organizational information requirements. Coupled with the technological development it becomes the major cause for a legacy information system replacement or any form of its reengineering. A new system can be redeveloped from scratch, but in that case the knowledge captured in the legacy system is lost. Legacy system replacement or reengineering can be done with significantly reduced amount of effort and cost if the conceptual models are reconstructed from them. Reverse engineering is the process of analyzing a subject system to create models of the system at a higher level of abstraction. It encompasses a broad set of methods and tools related to understanding and modifying information systems. Relational databases are at the core of most company information systems, hosting critical information for the day to day operation of the company. The knowledge captured in them can serve as an important resource in a legacy information system modernization project and they are a common source of reverse engineering processes. Starting from a physical database schema, that is recorded into the relational database schema data repository, the conceptual database schema or logical database schema may be extracted. All of these database schemas represent models at different levels of abstraction. An extraction process may be seen as a chain of model-to-model (M2M) transformations that trace model elements from a model at the lower level of abstraction to a model at the higher level of abstraction.

Models are widely used in engineering disciplines. In the model-driven approach to software engineering (MDSE) the idea of abstracting implementation details by focusing on models as first class entities is promoted in [30]. Models are used to specify, simulate, test, verify and generate code for the application to be built [9]. Each model is expressed by the concepts of a modeling language that is specified by means of a meta-model. A meta-model defines a set of valid models [5]. An M2M transformation is based on meta-models that are conformed by the source and target models of the transformation. These meta-models are said to be in support of M2M transformation.

In a forward engineering process, designers start with a high-level model, abstracting from all kinds of platform issues. Through a chain of M2M transformations, ending up with a model-to-text (M2T) transformation, the initial platform independent model transforms iteratively to a series of models with less degree of platform independency, introducing more and more platform specific extensions. Conversely, in a reverse engineering process, the abstraction level of models and degree of platform independency are increasing throughout the chain of transformations.

Here we present a part of our research efforts focused on meta-models relating to databases that we call database meta-models. These meta-models are in support of database M2M transformations. In [28] we proposed the classification of database meta-models as follows: i) data model (dm) meta-models; ii) generic database schema meta-models; iii) standard physical database schema meta-models; and iv) vendor-specific physical database schema meta-models.

In [29] we have proposed a meta-model of relational database schema concerning inclusion dependency (IND) constraints. Both meta-model of the relational database schema and meta-model of the Universal Relational Schema (URS) are presented there. In the context of forward engineering, these meta-models enable the platform independent specification of a broad class of INDs and the development of M2M and M2T transformations. The meta-model of INDs is important in the context of database and information system reverse engineering, too. A discovery of inclusion dependencies has attracted a lot of research interests together with methods for discovery of INDs. A formal specification of discovered INDs by means of proposed IND meta-model provides a better support of the automated reengineering and improvement of legacy databases.

Here we shift focus on M2M transformation of a physical database schema to a logical relational database schema. Therefore, we present one generic and one standard physical database schema meta-model. Generic database schema meta-models are based on theoretical foundations of a data model as it is, for example, relational data model. The relational data model is the focus of a continuous standardization process, and therefore we have extracted the standard physical database schema meta-models according to the specific SQL standard. We have developed a meta-model of relational database schema (RDSMM), which can be classified as a generic database schema meta-model. Also, we developed a meta-model of a standard physical database schema (SPMM). We selected these meta-models, because they are in support of database model transformations that capture logical database schemas based on RDSMM from legacy databases based on SPMM. In this way, the extraction and conceptualization of a database schema from a legacy database is provided. Then, it can be analyzed, restructured or improved and it becomes the input of a forward engineering process to get a modernized database schema.

To specify and manage RDSMM and SPMM, we used the Eclipse Modeling Framework (EMF) [16]. Both of these two meta-models are complex, and here we focus on their parts aimed at specifying multi-relational constraints, and particularly INDs. An overview of a complete meta-model of the relational database schema may be found in [28]. It comprises several modeling concepts, like: Attribute Constraint, Relation Scheme, Universal Relational Schema (URS) and Relational Database Schema.

Apart from Introduction and Conclusion the paper has five sections. Section 2 is devoted to the recall of basic notions. A generic database schema meta-model is presented in Section 3. A standard physical database schema meta-model is explained in Section 4. In Section 5 we present a case study of an M2M transformation. Related work is presented in Section 6.

2. Relational Database Schema

In this section, we briefly recall some basic notions of the relational data model used in the text to assist the reader in easier following the rest of the paper. They are borrowed from many sources, as well as [13] and [15], and are slightly adapted to the needs of our research.

Let R be a finite set of **attributes**. For each attribute $A \in R$, the set of all its possible values is called the domain of A. The domain associated with an attribute A is denoted by $Dom(A)$, and the set of possible values of attribute A (A-values) is denoted by $dom(A)$ [25]. A **domain constraint** restricts allowed values within a certain domain. A **tuple** t over $R = \{A_1, ..., A_m\}$ is a sequence of values $(a_1, ..., a_m)$ where: $(\forall i \in \{1, ..., m\})(a_i \in dom(A_i))$. A **relation** over R, denoted with $r(R)$, is a set of tuples over R.

A universal relational schema (URS) is a pair (R, UC), where R is a set that contains all the attributes of the Universe of Discourse (UoD) with associated domain constraints, and UC is a set of URS constraints that comprises a set of functional dependencies and non-trivial inclusion dependencies. R is called **universal attribute set** (UAS). A **universal relation** u(UAS) is a relation over the UAS. A f**unctional dependency** (FD) is a relationship that exists when each X-value uniquely determines a Y-value. Formally, given a set of attributes R, a functional dependency between attribute sets X and Y is represented as $X \rightarrow Y$, which specifies that Y is functionally dependent on X. A **non-trivial inclusion dependency** is a statement of the form $[X] \subseteq [Y]$, where X and Y are non-empty sequences of attributes from UAS. The cardinalities of X and Y have to be equal (unlike the cardinalities of attribute sets X and Y in FD), and the corresponding sequence elements from X and Y have to be domain compatible (again, unlike FD). A universal relation u is said to satisfy the non-trivial inclusion dependency if for each tuple $t \in u$ exists at least one tuple $s \in u$ such that $t[X] = s[Y]$, where $t[X]$ represents the projection of tuple t on X. There are different database design approaches ([15], [25]). One of them is based on the URS assumption. Using the set of FDs, the URS is decomposed into a set of relation schemes, resulting in a relational database schema.

Formally, a **relational database schema** is a pair (S, I), where S is a finite set of relation schemes and I a finite set of multiple relational constraints. A **relation scheme** is a named pair $N(R, C)$ where N is the name of relation scheme, R is a finite set of attributes (from UAS) and C a finite set of relational constraints. C contains **attribute value constraints** alongside with **null constraints**, **tuple check constraints**, **uniqueness constraints** and **key constraints**. A set of **multiple relational constraints** I, contains **extended tuple constraints** (an example can be seen in [28]) and **inclusion dependencies**. Here we give only the definition of inclusion dependency.

Let $N_l(R_l, C_l)$ and $N_r(R_r, C_r)$ be two relation schemes, where N_l and N_r are their names, R_l and R_r, their corresponding sets of attributes, and C_l and C_r their corresponding sets of relation scheme constraints. An **inclusion dependency (IND)** is a statement of the form $N_l[LHS] \subseteq N_r[RHS]$, where LHS and RHS are non-empty sequences of attributes from R_l and R_r respectively. Having the inclusion operator ($\subseteq$) orientated from the left to right we say that relation scheme N_l is on the left-hand side of the IND, while the relation scheme N_r is on its right-hand side. We use the indexes l and r, and the names of attribute sequences LHS and RHS, in order to indicate the left and right hand side of the IND, respectively. To define a validation rule of the IND we use the following notation: (i) the relation $r(N_l)$ is a set of tuples $u(R_l)$ (or just u) satisfying all constraints from the constraint set C_l; (ii) X-value is a projection of a tuple u on the set of attributes X; and (iii) according to the aforementioned orientation of the inclusion operator, $r(N_l)$ is called the referencing relation, while $r(N_r)$ is called the referenced relation. Informally, a database satisfies the inclusion dependency if the set of LHS-values in the referencing relation $r(N_l)$ is a subset of the set of RHS-values in the referenced relation $r(N_r)$.

There are two basic kinds of INDs: key-based INDs and non-key-based INDs. An IND is said to be **key-based** if the RHS is a key of the relation scheme N_r. Otherwise, it is a **non-key-based**. More often a key-based IND is called **referential integrity constraint (RIC)**. A non-key-based IND with a LHS that is a key of the relation scheme N_l, where a RIC $N_r[RHS] \subseteq N_l[LHS]$ is specified at the same time, is called **inverse referential integrity constraint (IRIC)**. The detailed explanation of these constraints may be found in [4].

In Fig. 1, a simplified part of a University database is given. The database satisfies inclusion dependencies:

$Ind1$: $Course[DepID] \subseteq Department[DepID]$

$Ind2$: $Department[DepID] \subseteq Course[DepID]$

$Ind3$: $Employed\_At[EID] \subseteq Employee[EID]$

$Ind4$: $Employee[EID] \subseteq Employed\_At[EID]$

$Ind5$: $Employed\_At[DepID] \subseteq Department[DepID]$

$Ind6$: $Taught\_By[DepID + CID] \subseteq Course[DepID + CID]$

$Ind7$: $Employee[SupervisorId] \subseteq Employee[EID]$

$Ind8$: $Taught\_By[EID + DepID] \subseteq \sigma_{Position = \text{'Prof.'} \text{ or } Position = \text{'Ass.'}} Employed\_At \rhd\lhd Employee[EID + DepID]$.

Ind1, *Ind3*, *Ind5*, *Ind6* and *Ind7* are the RICs since *DepID*, *EID*, *DepID*, *DepID* + *CID* and *EID* on the RHS of INDs are the keys of relation schemes *Department*, *Employee*, *Department*, *Course* and *Employee*, respectively. *Ind1*, *Ind3*, *Ind5* and *Ind7* are unary RICs, since the cardinality of the attribute sequence is 1, while *Ind6* is a binary RIC since the cardinality of the attribute sequence is 2. Generally, if the cardinality of attribute sequence is *n*, an IND is said to be ***n*-ary**. *Ind7* from the relational database schema is the consequence of a non-trivial IND from URS: $[SupervisorId] \subseteq [EID]$, while other RICs are the consequence of the decomposition of URS. *Ind2* and *Ind4* are the IRICs since: i) there are specified RICs *Ind1* and *Ind3*, respectively; and ii) *DepID* and *EID* on the LHS of *Ind2* and *Ind4* are the keys of relation schemes *Department* and *Employee*, respectively.

Employee

EID	FName	LName	Position	SupervisorID
003	Iva	Ilic	Ass.	007
007	Aca	Jovic	Prof.	009
009	Ina	Ras	Prof.	
010	Mila	Kun	PR	009

Department

DepID	DName
D1	Comp.
D2	Mech.
D3	Art

Employed_At

EID	DepID	Percent
003	D1	100
007	D1	70
007	D3	30
009	D2	100
010	D1	100

Course

DepID	CID	CName
D1	001	Java
D1	002	Databases
D2	003	Robotics
D3	001	Painting

Taught_By

EID	DepID	CID	ClassPerWeek
003	D1	001	3
007	D1	002	2
007	D3	001	2
009	D2	003	4

Fig. 1. A part of University database

IND *Ind8* requires further explanation. This type of IND is called **extended IND** [28] due to the fact that at least on the one side of the IND there is the natural join of two or more relation schemes. In our example, the RHS of the IND contains the natural join of two relation schemes *Employed_At* $\rhd\lhd$ *Employee*. *Ind8* is also an example of **selective (conditional) IND** ([8], [17], [28]). An IND is said to be selective if there is a selection condition at least on the one side of the IND. The semantics of constraint *Ind8* is that a course can be taught only by an employee that is employed at the department that contains the course, and that the employee must be either professor or assistant. Therefore, a database with a relation *Taught_by* that would contain one of the tuples: (003, D3, 001, 1) or (010, D1, 002, 2) would not obey the constraint *Ind8* and would not be formally consistent. The first tuple insertion would fail due to the fact that employee with *EID* = 003 is not employed at department D3, and the second due to the fact that employee with *EID* = 010 has the position of PR, and is neither a professor nor an assistant.

Integrity has always been an important issue for database design and implementation. Its importance grows with increasing demands regarding the quality and reliability of data. Integrity constraint specifications are translated into constraint enforcing mechanisms provided by the Database Management System (DBMS) used to implement a database. Most of the commercial DBMSs offer efficient declarative support for the domain constraints, null value constraints, uniqueness constraints and RICs (by means of foreign key constraints). On the contrary, non-key-based INDs are completely disregarded by actual RDBMSs, obliging the users to manage them via custom procedures or triggers. That is the reason why these kinds of constraints are ignored by database designers in a way that they do not recognize, specify and implement them. In the paper we present two meta-models of relational database constraints: the Generic Constraint Meta-Model (GCMM) and the Standard Constraint Meta-Model (SCMM). In that way two abstract syntaxes of two modeling languages are defined to enable database schema specification. That is a prerequisite for the development of M2M transformations that would enable automated transformation of a physical database schema extracted from a database to a relational database schema based on theoretical foundations of the relational data model. These specifications may be transformed into declarative scripts, procedures or triggers for integrity constraint enforcement supported by a DBMS. These transformations are M2T transformations. In the following section the fundamental GCMM concepts are presented.

3. Fundamental Generic Constraint Meta-Modeling Concepts

There are two approaches to perform relational database design: top-down (design by analysis) and bottom-up (design by synthesis). Top-down design methodology involves conceptual schema design (e.g., using Entity-Relational data model) followed by its mapping into relational database schema that can be improved in the subsequent analysis process. Bottom-up approach presupposes that the set of UoD attributes and functional dependencies among them have been given as a URS. Several algorithms may be used to decompose URS into a relational database schema. Our meta-model comprises modeling concepts to specify both: URS and relational database schema. Our main motive to design a meta-model of URS was to support the database design approaches based on the URS assumption and appliance of a synthesis algorithm to generate relational database schema starting with a URS. In our ongoing research we develop M2M transformations of legacy relational database schema into URS. We use our IIS*Studio development environment (presented in [3] and [22]) aimed at relational database schema generation and integration, to reengineer relational database schema and to further generate application prototype.

Fundamental GCMM concepts are presented in Fig. 2. *Project* concept encompasses *URS* concept and *RelationDBSchema* concept.

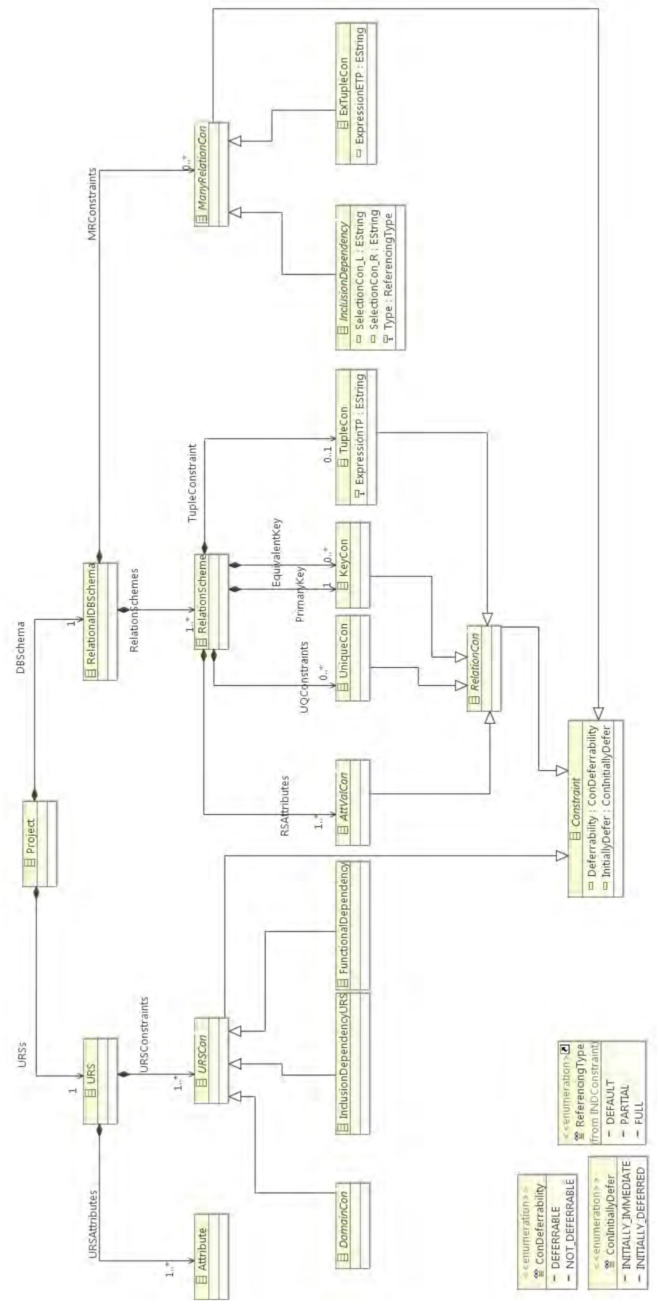


Fig. 2. Fundamental GCMM concepts

The *Constraint* is an abstract concept that has two properties: *Deferrability* and *InitiallyDefer*. *Deferrability* is aimed to specify whether or not constraint checking can be deferred until the end of the transaction. *InitiallyDefer* is used to specify the default checking behavior for constraints that are deferrable. The *Constraint* is specialized as: URS constraint (abstract concept *URSCon*), relation scheme constraint (abstract concept *RelationCon*) or multi relation constraint (abstract concept *ManyRelationCon*). There are three types of URS constraints: domain, functional dependency and non-trivial inclusion dependency. A detailed description of URS constraint meta-model may be found in [29]. Relation scheme constraints are specialized as: attribute value constraints (*AttValCon*), uniqueness constraints (*UniqueCon*), key constraints (*KeyCon*) and check constraints (*CheckCon*). Inclusion dependencies (*InclusionDependency*) and extended tuple constraint concept (*ExTupleCon*) specialize *ManyRelationCon* concept. Hereinafter we give a detailed description of inclusion dependency meta-model.

3.1. Inclusion Dependency Meta-Model

The *InclusionDependency* modeling concept is abstract and it generalizes concepts for modeling several kinds of INDs listed in Section 2. As can be seen in Fig. 3, *InclusionDependency* is first specialized with *ReferentialIntegrityCon* and *NonKeyBased IND*. The first is a concrete modeling concept aimed at modeling RICs. The second concept is abstract and is further specialized with concrete modeling concepts: *InverseReferentialIntegrityCon* and *NonInverseReferentialIntegrityCon* aimed at modeling IRICs and other INDs (that are neither RIC nor IRIC), respectively. Each of these three concrete concepts is further specialized with concrete concepts aimed at modeling: extended RICs, extended IRICs and others extended INDs. The *InclusionDependency* modeling concept has two properties: *SelectionCon_L* and *SelectionCon_R* that are used to specify selection conditions on the left or right side of IND. These properties are optional and if at least one of them is specified that implies that the modeled constraint is selective (conditional) IND. The third property *ReferencingType* is used to specify whether the referencing is default, partial or full, for *n*-ary INDs. For unary INDs there are no differences between these referencing types.

The relation schemes specified in an IND may have two roles: referencing, if it is on the LHS of the IND and referenced, if it is on the RHS of the IND. In Fig. 3, roles are modeled with concrete concepts *RoleReferenced* and *RoleReferencing*. For the referenced role, critical database operations that may violate the IND constraint are deletes and updates, and for the referencing role, critical operations are inserts and updates. The *RoleReferenced* and *RoleReferencing* concepts contain properties aimed at specifying the actions that would take place to preserve database from violation of an IND constraint in case when a critical operation occurs.

For an IND it has to be specified at least one relation scheme in each of the roles. If an IND is an extended RIC, than at least one more relation scheme must be specified as the referencing role of the IND. In the case of extended IRICs at least one more relation scheme must be specified as the referenced role of IND. For other extended INDs at least one more relation schema has to be specified as the referencing or as the referenced role of the IND.

For each IND, the attribute sequences on the LHS and RHS of the IND must be specified. In particular, for a RIC on the RHS is specified a key of a referenced relation scheme and for an IRIC on the LHS is specified a key of a referencing relation scheme, instead of an arbitrary attribute sequence.

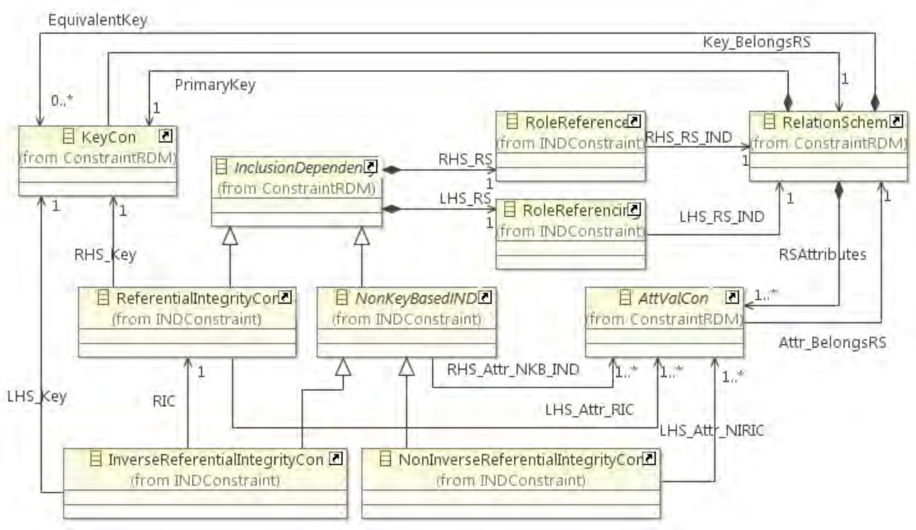

EquivalentKey
Key_BelongsRS
PrimaryKey
KeyCon (from ConstraintRDM)
InclusionDependen (from ConstraintRDM)
RoleReference (from INDConstraint)
RoleReferencin (from INDConstraint)
RelationSchem (from ConstraintRDM)
RHS_RS
LHS_RS
RHS_RS_IND
LHS_RS_IND
RSAttributes
RHS_Key
LHS_Key
ReferentialIntegrityCon (from INDConstraint)
NonKeyBasedIND (from INDConstraint)
AttValCon (from ConstraintRDM)
Attr_BelongsRS
RHS_Attr_NKB_IND
LHS_Attr_RIC
LHS_Attr_NIRIC
RIC
InverseReferentialIntegrityCon (from INDConstraint)
NonInverseReferentialIntegrityCon (from INDConstraint)
0..*
1
1..*

Fig. 3. Meta-model of Inclusion dependency concept

3.2. OCL Invariants to Improve Constraint Meta-Model Semantics

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 [5]. Meta-models presented in Fig. 2 and Fig. 3 comprise a lot of knowledge about the real system (relational database schema) that would be modeled by means of it. But, it is still not enough. In order to improve the semantics of our meta-model we use Object Constraint Language (OCL) to specify constraints concerning modeling concepts that can not be expressed solely by means of UML class diagram concepts. Here we present some OCL constraint examples.

In Fig. 4, an OCL invariant is presented that enables checking if the sequence of attributes on the LHS of a RIC belongs to the set of attributes of the relation scheme that is specified as the referencing in the RIC. In addition, it checks if the key specified at the RHS of a RIC is a key of the relation scheme specified as referenced in the RIC.

A similar OCL invariant for the extended RIC can be seen in Fig. 5. The difference is in the fact that attributes for attribute sequence must belong to at least one of the relation schemes specified as referencing in the RIC.

In addition to checking of conformance between attribute sets and specified relation schemes, in Fig. 6 is presented an OCL invariant that checks if the appropriate RIC is specified for an IRIC.

```
invariant CheckLHSAttributeRIC:
LHS_Attr_RIC->forAll(a : ConstraintRDM::AttValCon |
                                        a.Attr_BelongsRS = self.LHS_RS.LHS_RS_IND);
invariant CheckRHSAttributeRIC:
RHS_Key->forAll(a : ConstraintRDM::KeyCon |
                                        a.Key_BelongsRS = self.RHS_RS.RHS_RS_IND);
```

Fig. 4. OCL invariant for RIC attribute checking

```
invariant CheckLHSAttributeExRIC:
LHS_Attr_RIC->forAll(a : ConstraintRDM::AttValCon |
                                    a.Attr_BelongsRS = self.LHS_RS.LHS_RS_IND or
                                    a.Attr_BelongsRS = LHS_RS_Ex_RIC.LHS_RS_IND);
invariant CheckRHSAttributeExRIC:
RHS_Key->forAll(a : ConstraintRDM::KeyCon |
                                        a.Key_BelongsRS = self.RHS_RS.RHS_RS_IND);
```

Fig. 5. OCL invariant for extended RIC attributes checking

```
invariant CheckRIC:
LHS_Key = RIC.RHS_Key and RHS_Attr_NKB_IND = RIC.LHS_Attr_RIC;
invariant CheckRHSAttributeIRIC:
RHS_Attr_NKB_IND->forAll(a : ConstraintRDM::AttValCon |
                                        a.Attr_BelongsRS = self.RHS_RS.RHS_RS_IND);
invariant CheckLHSAttributeIRIC:
LHS_Key->forAll(a : ConstraintRDM::KeyCon |
                                         a.Key_BelongsRS = self.LHS_RS.LHS_RS_IND);
```

Fig. 6. OCL invariant for the RIC – IRIC pair existence checking

4. Fundamental Standard Constraint Meta-Model Concepts

The relational data model is a superior logical data model and the acceptance of RDBMSs is widespread, too. 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). RDBMS products more or less comply with an SQL standard. Here we propose a standard physical database schema meta-model (Fig. 7). Instead of attribute and relation scheme concepts from GCMM, in SCMM there are column (*Column*) and table (*Table*) concepts, respectively. The *Constraint* abstract concept has the same properties, as the corresponding concept in GCMM. There are four concepts to specialize the *Constraint* concept: check constraint (*CheckCon*), primary key constraint (*PrimaryKeyCon*), uniqueness constraint (*UniqueCon*) and foreign key constraint (*ForeignKey*). The *UniqueCon* and *PrimaryKeyCon* concepts indirectly specialize the *Constraint* concept, via their generalization (*PKeyUnique*). A detailed description of SCMM, alongside with appropriate OCL invariants may be found in [3].

In the next section an illustration of an M2M transformation is presented. The transformation is based on SCMM and GCMM.

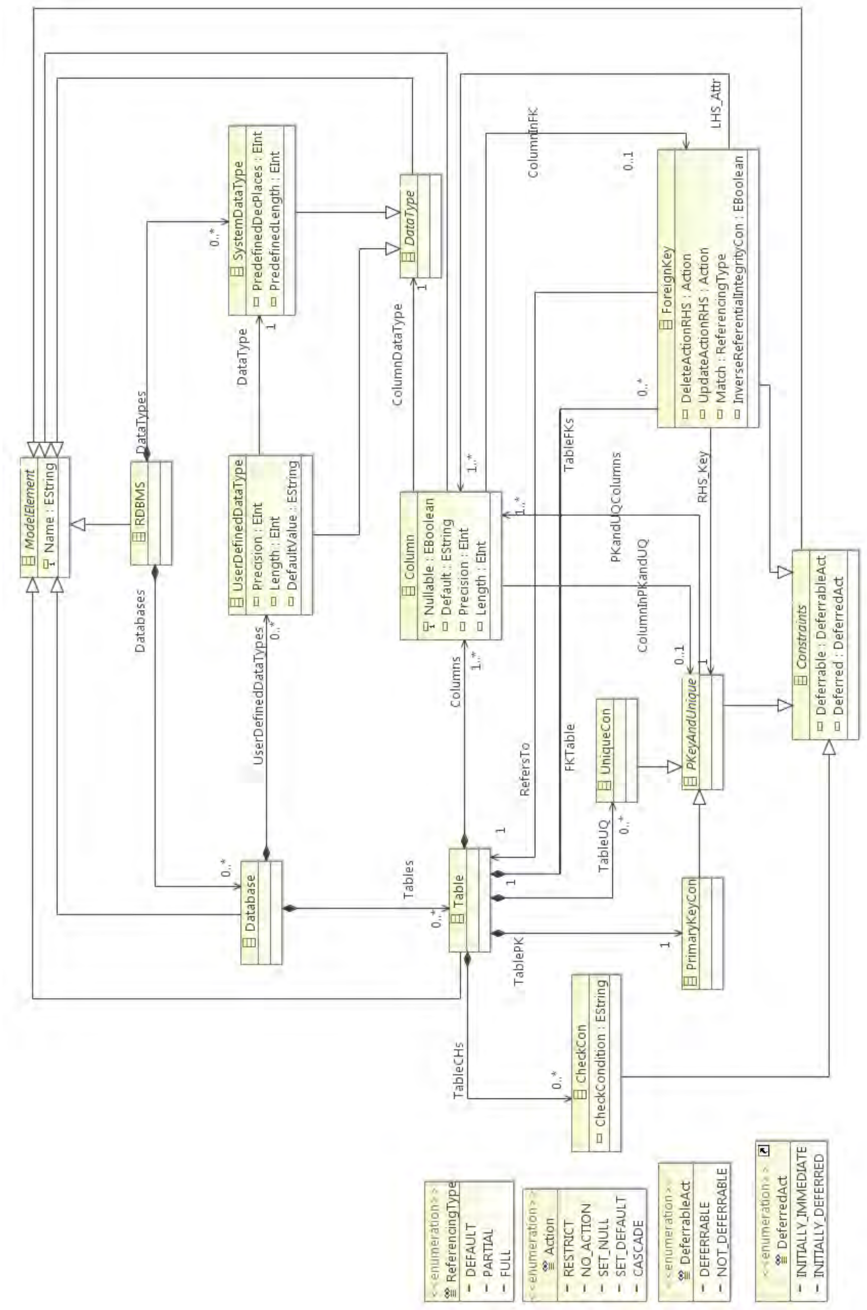


Fig. 7. Fundamental SCMM concepts

5. Case Study – Model-To-Model Transformation

To specify and manage presented meta-models we used the Eclipse Modeling Framework (EMF), Eclipse Juno 4.2.1. and OCL 3.2.1. After specifying the meta-model, a fully functional Eclipse editor can be generated for it. The editor guides a database schema specification process and ensures the conformance of the database schema with the proposed meta-model. By means of such an editor, we have specified a relational database scheme of the University database whose part is presented in Section 2. In Fig. 8, the conceptual database schema of the University database is visually represented by means of a 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-relational constraints. The detailed specification of the aforementioned relational database schema may be found in [28].

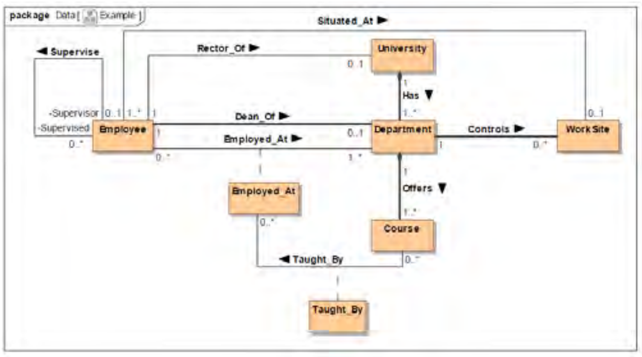
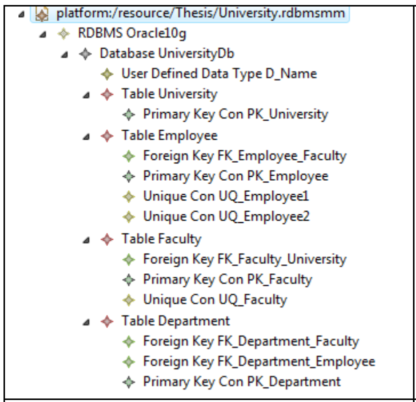


Fig. 8. The conceptual database schema of University database

A part of *University_PDBS* database schema modeled by means of the Eclipse editor generated from presented SCMM is shown in Fig. 9 (a). In this example it is captured from database repository.

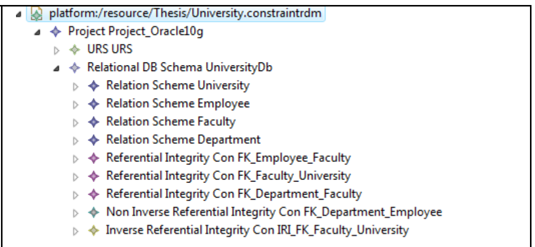
In Fig. 9 (b) a part of the *University_SDBS* is presented. It is the output of the *University_PDBS*2*University_SDBS* model-to-model transformation that is based on SCMM and GCMM. The transformation will transform a vendor specific physical database schema that conforms to the SQL standard database schema, into generic, relational database schema. The Atlas Transformation Language (ATL) is used to implement *University_PDBS*2*University_SDBS* transformation.

In Fig. 10 (a), ATL rules for mapping the *ForeignKey* concept onto *ReferentialIntegrity Con* are presented. Rules for mapping the *ForeignKey* concept onto the *NonInvReferentialIntegrityCon* concept are shown in Fig. 10 (b).



```
platform:/resource/Thesis/University.rdbmsmm
  RDBMS Oracle10g
    Database UniversityDb
      User Defined Data Type D_Name
      Table University
        Primary Key Con PK_University
      Table Employee
        Foreign Key FK_Employee_Faculty
        Primary Key Con PK_Employee
        Unique Con UQ_Employee1
        Unique Con UQ_Employee2
      Table Faculty
        Foreign Key FK_Faculty_University
        Primary Key Con PK_Faculty
        Unique Con UQ_Faculty
      Table Department
        Foreign Key FK_Department_Faculty
        Foreign Key FK_Department_Employee
        Primary Key Con PK_Department
```

a) Model of *University_PDBS* conformant with SCMM



```
platform:/resource/Thesis/University.constraintrdm
  Project Project_Oracle10g
    URS URS
    Relational DB Schema UniversityDb
      Relation Scheme University
      Relation Scheme Employee
      Relation Scheme Faculty
      Relation Scheme Department
      Referential Integrity Con FK_Employee_Faculty
      Referential Integrity Con FK_Faculty_University
      Referential Integrity Con FK_Department_Faculty
      Non Inverse Referential Integrity Con FK_Department_Employee
      Inverse Referential Integrity Con IRI_FK_Faculty_University
```

b) Model of *University_RDBS* conformant with GCMM

Fig. 9. Models of University_PDBS and University_RDBS

```
rule FK2RIC{
from
  fk: RDBMS!ForeignKey (fk.RHS_Key.isKey())
to
  out: RM!ReferentialIntegrityCon (
     Name <- fk.Name,
     LHS_Attr_RIC <- fk.LHS_Attr,
     RHS_Key <- fk.RHS_Key,
     RHS_RS <- roleRHS,
     LHS_RS <- roleLHS,
     Type <- fk.Match,
     Deferrability <- fk.Deferrable,
     InitiallyDefer <- fk.Deferred
  ),
  roleRHS: RM!RoleReferenced (
     RHS_RS_IND <- fk.RefersTo,
     DeleteAction <- fk.DeleteActionRHS,
     UpdateAction <- fk.UpdateActionRHS
  ),
  roleLHS: RM!RoleReferencing (
     LHS_RS_IND <- fk.FKTable,
     InsertAction <- 'NO_ACTION',
     UpdateAction <- 'NO_ACTION'   )}
```

a)ATL rules to map *ForeignKey* concept onto *ReferentialIntegrityCon* concept

```
rule FK2NonIRIC{
from
  fk: RDBMS!ForeignKey(not fk.RHS_Key.isKey())
to
  out: RM!NonInverseReferentialIntegrityCon (
     Name <- fk.Name,
     LHS_Attr_NIRIC <- fk.LHS_Attr,
     RHS_Attr_NKB_IND <- fk.RHS_Key.PKandUQColumns,
     RHS_RS <- roleRHS,
     LHS_RS <- roleLHS,
     Type <- fk.Match,
     Deferrability <- fk.Deferrable,
     InitiallyDefer <- fk.Deferred
  ),
  roleRHS: RM!RoleReferenced (
     RHS_RS_IND <- fk.RefersTo,
     DeleteAction <- fk.DeleteActionRHS,
     UpdateAction <- fk.UpdateActionRHS
  ),
  roleLHS: RM!RoleReferencing (
     LHS_RS_IND <- fk.FKTable,
     InsertAction <- 'NO_ACTION',
     UpdateAction <- 'NO_ACTION'
  )
}
```

b) ATL rules to map *ForeignKey* concept onto *NonInvReferentialIntegrityCon* concept

Fig. 10. ATL transformation rules

6. Related Work

Meta-modeling is widely spread area of research. OMG's Model-Driven Architecture (MDA) [26] currently is the most mature formulation of the MDSE paradigm. It 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) [26]. The OMG's Meta Object Facility (MOF) defines the metadata architecture that lies at the heart of MDA [24]. MOF is used to define semantics and structure of generic meta-models or domain specific ones. There are numerous references covering MOF based meta-models.

In the paper [14], 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 between other data models. Polo, Garcia-Rodriguez and Piattini in [27] propose a very simplified relational and object-oriented meta-model. A similar, simplified RDBMS meta-model is presented in [32], by Wang, Shen and Chen. Vara et al. in [31] presented Oracle 10g meta-model that can be classified as vendor-specific physical database schema meta-model. Lano and Kolahdouz-Rahimi in [20] and [21] propose a rather simplified relational database model not discerning standard and vendor specific constructs. SQL standard meta-models can be found in Calero et al. [11] and del Castillo et al. [12]. The importance of generic models is emphasized by Atzeni, Gianforme and Cappellari in [1] and [2]. Cabot and Teniente in [10] and Cabot et al. in [9] present an OCL meta-model and a case study where they used a simplified UML class meta-model, that can be classified as a generic database schema meta-model. The paper of Gogolla et al. [18] is interesting in another context: it presents the intensional and extensional ER/relational meta-models. The relational database schema meta-models that we presented in this paper are intensional meta-models. Our future research has to consider extensional database meta-models, too.

Most of the presented database meta-models can be classified as standard physical database schema meta-models or vendor-specific physical database schema meta-models. On the contrary, our relational database schema meta-model is generic database schema meta-model and it comprises a broader set of concepts with more properties then the aforementioned database meta-models. Such a meta-model will enable the creation of semantically rich models of relational database schemas. Such models can be further transformed through the chain of M2M and M2T transformations, ending up with automatically generated executable program code for the implementation of all specified database constraints.

There is one more issue that is important in the context of results presented in this paper. Inclusion dependency discovery has attracted a lot of research interests from the communities of database design, machine learning and knowledge discovery. Bauckmann, Leser, and Naumann in [6], Koeller and Rundensteiner in [19] and De Marchi, Lopes and Petit in [23] propose different techniques to discover INDs. In recent years, some studies to extend traditional inclusion dependencies have been made, like Bravo, Fan and Ma in [8] and Fan in [17]. The methods for conditional IND discovery are suggested, too. Bi and Shan in [7] notify new interest in dependencies to extend traditional dependencies, such as conditional functional dependencies and conditional inclusion dependencies. They state that data dependencies play an important role in data repair, too. Once discovered, INDs would be properly specified by means of a meta-model that is semantically rich enough. We have developed our meta-models keeping that in mind.

7. Conclusion

Some kinds of relational database constraints are well-known and can be implemented by the declarative DBMS mechanisms, like the key constraint and the referential integrity constraint. 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. The striking examples are some kinds of inclusion dependency (IND) constraints, like: inverse referential integrity constraint, conditional IND and extended IND. In the paper we present a part of our research efforts focused on meta-models relating to databases. We developed a generic meta-model of relational database schema and a standard physical database schema meta-model. Here we deal with their parts concerning multi-relational constraints: inclusion dependencies in the generic meta-model and foreign key constraint in the standard meta-model. We consider all kinds of constraints as important to be specified and implemented. In the context of forward engineering, the proposed generic database meta-model will enable platform independent specification of a broad class of INDs and development of further M2M and M2T transformations ending up with executable program code. On the other side, a meta-model of INDs is important in the context of database and information system reengineering, too. Considering growing interest in IND discovery in legacy databases, we plan to integrate these results with our IIS*Studio development environment. That would enable the reconstruction of a relational database schema and its improvement through the mechanisms implemented in IIS*Studio.

Acknowledgements. 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*.

References

1. Atzeni, P., Cappellari, P., and Gianforme, G.: MIDST: model independent schema and data translation. In Proceedings of the 2007 ACM SIGMOD international Conference on Management of Data (Beijing, China, June 11 - 14, 2007). SIGMOD '07. ACM, New York, NY, 1134-1136. (2007)
2. Atzeni, P., Gianforme, G., Cappellari, P.: A universal meta-model and its dictionary. T. Large-Scale Data and Knowledge-Centered Systems 1, 38–62. (2009)
3. Aleksic, S. Methods of Database Schema Transformations in Support of the Information System Reengineering Process, Ph.D thesis, University of Novi Sad, Faculty of Technical Sciences (2013).
4. Aleksic, S., Ristic, S., Lukovic, I., Celikovic, M.: A Design Specification and a Server Implementation of the Inverse Referential Integrity Constraints. Computer Science and Information Systems (ComSIS), Consortium of Faculties of Serbia and Montenegro, Belgrade, Serbia, ISSN: 1820-0214, Vol. 10, No.1, pp. 283-320. (2013)
5. Assmann, U., Zchaler and S., Wagner, G.: Ontologies, Meta-Models, and the Model-Driven Paradigm. In: Calero, C., Ruiz, F., Piattini, M. (eds.) Ontologies for Software Engineering and Software Technology (2006)
6. Bauckmann, J., Leser, U., and Naumann, F.: Efficiently Computing Inclusion Dependencies for Schema Discovery. Proc. Second Int'l Workshop Database Interoperability. (2006)
7. Bi, Z., and Shan, M.: Review of Data Dependencies in Data Repair. Journal of Information & Computational Science 9: 15 4623–4630. (2012)

8. Bravo, L., Fan, W., and Ma, S. Extending dependencies with conditions. In Proceedings of the 33rd international conference on Very large data bases (VLDB '07). VLDB Endowment 243-254. (2007)
9. Cabot, J., Clarisó, R., Guerra, E., and De Lara, J.: Verification and validation of declarative model-to-model transformations through invariants. Journal of Systems and Software, 83(2), 283-302. (2010)
10. Cabot, J. and Teniente, E.: Incremental integrity checking of uml/ocl conceptual schemas. Journal of Systems and Software, 82(9), 1459–1478. (2009)
11. Calero, C., Ruiz, F., Baroni, A., Brito e Abreu, F. , Piattini, M.: An ontological approach to describe the SQL:2003 object-relational features. Computer Standards & Interfaces, Volume 28, Issue 6, September 2006, Pages 695–713. (2006)
12. del Castillo, R.P., García-Rodríguez, I., and Caballero, I.: PRECISO: a reengineering process and a tool for database modernization through web services. In: Jacobson Jr., M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 2126–2133. Springer, Heidelberg. (2009)
13. Date, C.J. and Darwen, H.: Types and the Relational Model. The Third Manifesto, 3rd ed. Addison Wesley, Reading. (2006)
14. Eessaar, E.: Using Meta-modeling in order to Evaluate Data Models. In Proceedings of the 6th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases, Corfu Island, Greece, February 16-19. (2007)
15. Elmasri R., Navathe B. S.: Database Systems: Models, Languages, Design and Application Programming, Sixth Edition, Pearson Global Edition, ISBN 978-0-13-214498-8. (2011)
16. Eclipse Modeling Framework, [Online] Available: http://www.eclipse.org/modeling/emf/. (retrieved February, 2014).
17. Fan, W.: Extending dependencies with conditions for data cleaning, Computer and Information Technology, 2008. CIT 2008. 8th IEEE International Conference on, pp.185-190. (2008)
18. Gogolla, M., Lindow, A., Richters, M. and Ziemann, P.: Meta-model transformation of data models. Position paper. WISME at the UML (2002)
19. Koeller, A., and Rundensteiner, E.A.: Heuristic Strategies for Inclusion Dependency Discovery, On the Move to Meaningful Internet Systems 2004: Proc. Int'l Conf. CoopIS, DOA, and ODBASE, pp. 891-908. (2004)
20. Lano, K., and Kolahdouz-Rahimi, S.: Model-driven development of model transformations. Theory and Practice of Model Transformations, 47-61. (2011)
21. Lano, K., and Kolahdouz-Rahimi, S.: Constraint-based specification of model transformations Journal of Systems and Software, Volume 86, Issue 2, Pages 412–436. (2013)
22. Luković, I., Mogin, P., Pavićević, J. and Ristić, S.: An Approach to Developing Complex Database Schemas Using Form Types, Software: Practice and Experience, John Wiley & Sons Inc, Hoboken, USA, ISSN: 0038-0644, DOI: 10.1002/spe.820 Vol. 37, No. 15, pp. 1621-1656. (2007)
23. De Marchi, F., Lopes, S., and Petit, J.-M.: Unary and N-Ary Inclusion Dependency Discovery in Relational Databases, J. Intelligent Information Systems, vol. 32, no. 1, pp. 53-73. (2009)
24. Meta-Object Facility, [Online] Available: http://www.omg.org/mof/. (retrieved February, 2014)
25. Mogin, P., Luković, I., Govedarica, M.. Database Design Principles, University of Novi Sad, Faculty of Technical Sciences & MP "Stylos", Novi Sad, Serbia. (2004)
26. Mukerji, J. and Miller, J., 2003. MDA Guide Version 1.0.1, document omg/03-06-01 (MDA Guide V1.0.1), http://www.omg.org/, (retrieved February, 2014)
27. Polo, M., Garcia-Rodriguez, I., and Piattini, M.: An MDA-based approach for database reengineering. J. Softw. Maint. Evol. 19, 6 (November 2007), 383-417. (2007)

28. Ristic, S., Aleksic, S., Celikovic, M., and Lukovic, I.: An EMF Ecore based relational dB schema meta-model. In Proceedings of the 6th International Conference on Information Technology ICIT 2013. Amman, Jordan. (2013).
29. Ristic, S., Aleksic, S., Celikovic, M., and Luković, I. Meta-modeling of inclusion dependency constraints. In Proceedings of the 6th Balkan Conference in Informatics (BCI '13). ACM, New York, NY, USA, 114-121. (2013)
30. Stahl T, Völter M, Bettin J, Haase A, Helsen S.: Model Driven Software Development: Technology, Engineering, Management. John Wiley & Sons, Ltd. (2006)
31. Vara, J., Vela, B., Bollati, V.A. and Marcos, E.: Supporting model-driven development of object-relational database schemas: a case study, in: R. Paige (Ed.), Theory and Practice of Model Transformations, Heidelberg, Springer Berlin, pp. 181–196. (2009)
32. Wang, H., Shen, B. and Chen, C.: Model-Driven Reengineering of Database. Software Engineering, 2009. WCSE '09. WRI World Congress on , vol.3, no., pp.113-117. (2009)

Sonja Ristić works as an associate professor at the University of Novi Sad, Faculty of Technical Sciences, Serbia. She received two bachelor degrees with honors from University of Novi Sad (UNS), one in Mathematics, Faculty of Science in 1983, and the other in Economics from Faculty of Economics, in 1989. She received her Mr (2 year) and Ph.D. degrees in Informatics, both from Faculty of Economics (UNS), in 1994 and 2003. From 1984 till 1990 she worked with the Novi Sad Cable Company NOVKABEL–Factory of Electronic Computers. From 1990 till 2006 she was with High School of Business Studies – Novi Sad, and since 2006 she has been with the Faculty of Technical Sciences (UNS). Her research interests are related to Database Systems, Information Systems and Software Engineering.

Slavica Aleksić received her M.Sc. degree from the Faculty of Technical Sciences, at University of Novi Sad in 1998. She received her Mr (2 year) degree in 2006 and Ph.D. degree in 2013, at the University of Novi Sad, Faculty of Technical Sciences. Currently, she works as a assistant professor at the Faculty of Technical Sciences at the University of Novi Sad, where she lectures in several Computer Science and Informatics courses. Her research interests are related to Information Systems, Database Systems and Software Engineering.

Milan Čeliković graduated in 2009 at the Faculty of Technical Sciences, Novi Sad, at the Department of Computing and Control. Since 2009 he has worked as a teaching assistant at the Faculty of Technical Sciences, Novi Sad, at the Chair for Applied Computer Science. In 2010, he started his Ph.D. studies at the Faculty of Technical Sciences, Novi Sad. His main research interests are focused on: Domain specific modeling, Domain specific languages, Databases and Database management systems. At the moment, he is involved in the projects concerning application of DSLs in the field of software engineering.

Ivan Luković received his M.Sc. degree in Informatics from the Faculty of Military and Technical Sciences in Zagreb in 1990. He completed his Mr (2 year) degree at the University of Belgrade, Faculty of Electrical Engineering in 1993, and his Ph.D. at the University of Novi Sad, Faculty of Technical Sciences in 1996. Currently, he works as a Full Professor at the Faculty of Technical Sciences at the University of Novi Sad, where he lectures in several Computer Science and Informatics courses. His research interests are related to Database Systems and Software Engineering. He is the author or co-author of over 100 papers, 4 books, and 30 industry projects and software solutions in the area.

Received: *February 16*, *2014*, *Accepted*: *May 28*, *2014*.