AN EMF ECORE BASED RELATIONAL DB SCHEMA META-MODEL

Sonja Ristić[*1], Slavica Aleksić [#2], Milan Čeliković [#3], Ivan Luković [#4]

[*] University of Novi Sad, Faculty of Technical Sciences,
Department for Industrial Engineering and Management
Trg Dositeja Obradovića 6
21000 Novi Sad, Serbia
[#] University of Novi Sad, Faculty of Technical Sciences,
Department of Computing and Control
Trg Dositeja Obradovića 6
21000 Novi Sad, Serbia
[1]sdristic@uns.ac.rs, [2]slavica@uns.ac.rs, [3]milancel@uns.ac.rs, [4]ivan@uns.ac.rs

Abstract

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 this paper, we propose a classification of database models and meta-models. 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 Re-engineering

1 INTRODUCTION

MDSE (Model-Driven Software Engineering) paradigm has promoted the idea of abstracting implementation details by focusing on models as first class entities [24]. They are used to specify, simulate, test, verify and generate code for the application to be built [9]. Many of these activities include the specification and execution of model-to-model (M2M) 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 model-to-code (M2C) 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, user interfaces (UI), etc. 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.

Here, we focus on models relating to databases (db). For these models we will use the generic name db models. Db models may exist at several, different, levels of abstraction: data model level, database schema level, implementation database schema level, physical database schema level. According to Date and Darwen definition [13] revised by Eessaar in [14]: „*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*." 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 logical and implementation db design and db 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 M2C transformations between these db models at different abstraction levels. An explicit representation of mappings specifies how two models are related to each other. Some mapping examples, according to Bernstein [6] are: i) mapping between an entity-relationship (ER) model and a SQL schema to navigate between a db 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.

One of the key concepts in software maintenance is re-engineering. It generally includes some form of reverse engineering followed by some form of forward engineering or restructuring. Relational or OR databases are a common source of reverse engineering. Starting from physical database schema, recorded into RDBMS data repository, the conceptual db schema (ER or EER db schema) or logical db schema (based on the relational data model) could be extracted. Structured Query Language (SQL) is currently available in most commercial and open-source DBMSs. 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 db schemas extracted from RDBMS data repository of different vendors may be a serious problem. One of the solutions is meta-modeling of db models. In this paper we propose a meta-model of relational database schema.

In the purpose of specifying and managing our meta-model we decided to use the Eclipse Modeling Framework (EMF) [15], 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 db 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 DB 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 db meta-models that describe db models at certain abstraction level. Hereof, we are distinguishing: i) data model (dm) meta-models; ii) generic db schema meta-models; iii) standard physical db schema meta-models; and iv) vendor-specific physical db schema meta-models.

Data model meta-models stands at the M2 level of MOF stack. For example, one may specify relational dm meta-model or ER dm 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, like relational, ER, EER or OR data models, we may specify specific data types, data structure types, constraint types for some SQL standard or vendor-specific data model. Therefore, as may be seen in Table 1 (MOF stack – Example 1), at M2 level appear corresponding meta-models.

The remaining three classes of meta-models are at the same abstraction level. They may be seen as the models at the M1 level if the MOF stack begins with a data model meta-model (Table 1, MOF stack – Example 1), or as the models (meta-models) at the M2 level (Table 1, MOF stack – Example 2). Some of the generic db schema meta-models describe conceptual db schemas, like ER or EER db schema MM, while others describe logical db schemas, like relational or OR db schema MM. Relational data model is the focus of a continues standardization process, and therefore we have extracted the standard physical db 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 introduced class of vendor-specific physical db schema meta-models.

Our classification and distribution of db models across the MOF level stack will enable systematic approach for mapping specification between different models/meta-models and development of appropriate M2M or M2C transformations.

In order to do that, corresponding meta-models have to be specified. In this paper we are presenting a relational db schema meta-model, according the theoretical definition of relational data model ([13], [21]).

Table 1. A Classification of db meta-models

<table>
<tr><th>MOF level</th><th>MOF stack – Example 1</th><th colspan="3">MOF stack – Example 2</th></tr>
<tr><td>M3</td><td>EMOF/CMOF/Ecore</td><td colspan="3">EMOF/CMOF/Ecore</td></tr>
<tr><td rowspan="2">M2</td><td rowspan="2">Relational dm MM,
ER dm MM,
SQL:2003 dm MM,
Oracle 10g dm MM,
MySQL dm MM,
dBase III+ dm MM …</td><td>Generic db schema MM</td><td>Standard Pdb MM</td><td>Vendor-specific Pdb MM</td></tr>
<tr><td>ER db schema MM,
Relational db schema MM,
OR db schema MM …</td><td>SQL:1999 db schema MM,
SQL:2003 db schema MM …</td><td>Oracle 10g db schema MM,
MySQL db schema,
dBase III+ db schema</td></tr>
<tr><td>M1</td><td>ER db schema MM,
Relational db schema MM, …
SQL:2003 db schema MM …
Oracle 10g db schema MM,
MySQL db schema, …</td><td>ER db schema 1,
ER db schema 2, …
Relational db schema 1,
Relational db schema 2, …
OR db schema 1 …</td><td>SQL:2003 db schema 1,
SQL:2003 db schema 2,
SQL:1999 db schema 1, …</td><td>Oracle 10g db schema 1,
MySQL db schema 1,
MySQL db schema 2, …</td></tr>
<tr><td>M0</td><td>ER db schema 2, …
Relational db schema 1, …
SQL:2003 db schema 2,
SQL:1999 db schema 1, …</td><td colspan="2">Logical data structure of a database</td><td>Database instance</td></tr>
</table>

	Oracle 10g db schema 1, MySQL db schema 1		

3 A RELATIONAL DB 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 db 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 db design approaches based on the URS assumption [21]. A db 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 db project is composed from URS and relational db schema. As can be seen in Fig. 2 db 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 db schema concept meta-model is presented in Subsection 3.2. Due to the space limits we do not give here detail explanations of presented concepts. In order to make some of them clearer in Section 4 we give an example of a relation db schema instantiating some of the concepts presented here.

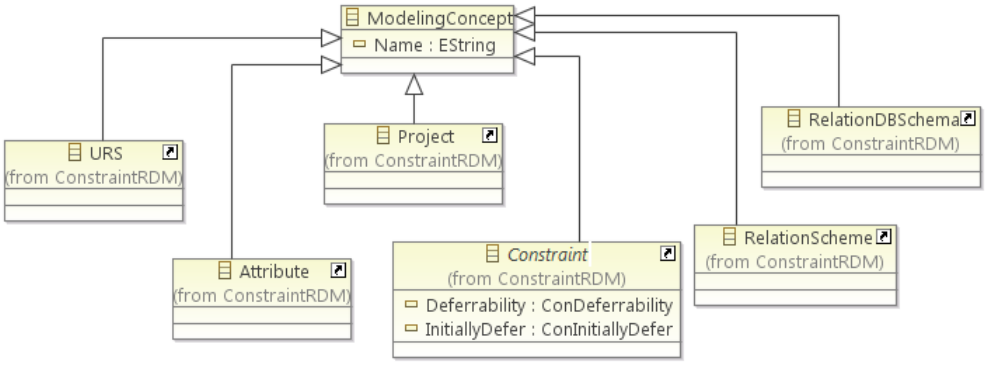

Fig. 1. The relational db schema modeling Concepts

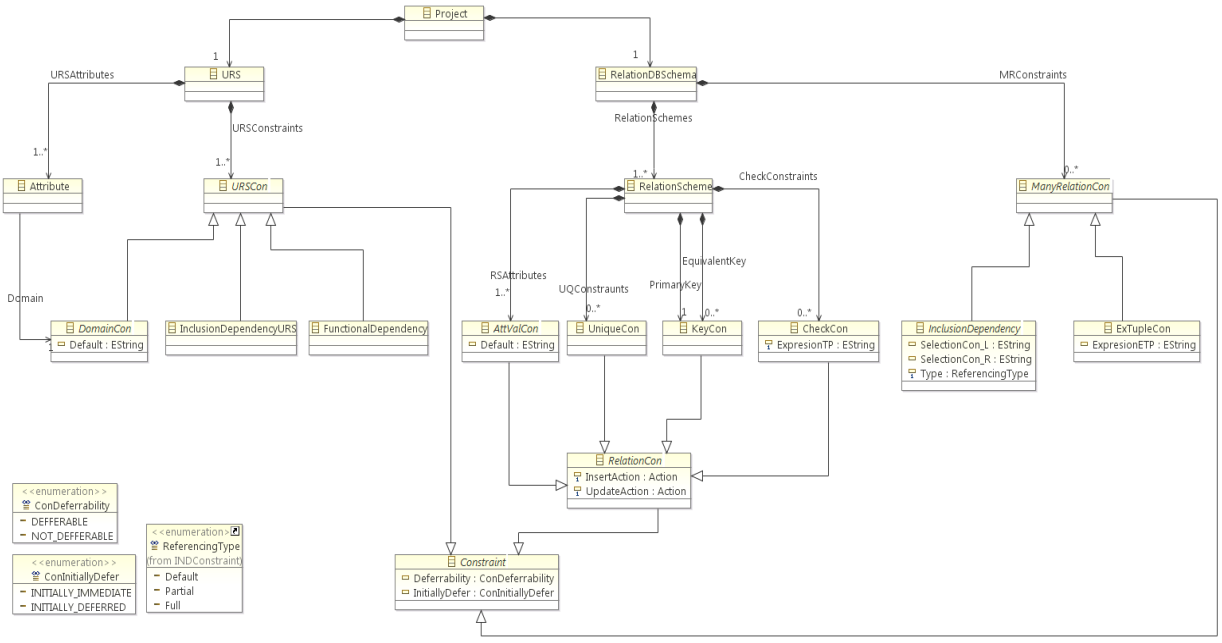

Fig. 2. A meta-model of project concept

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.

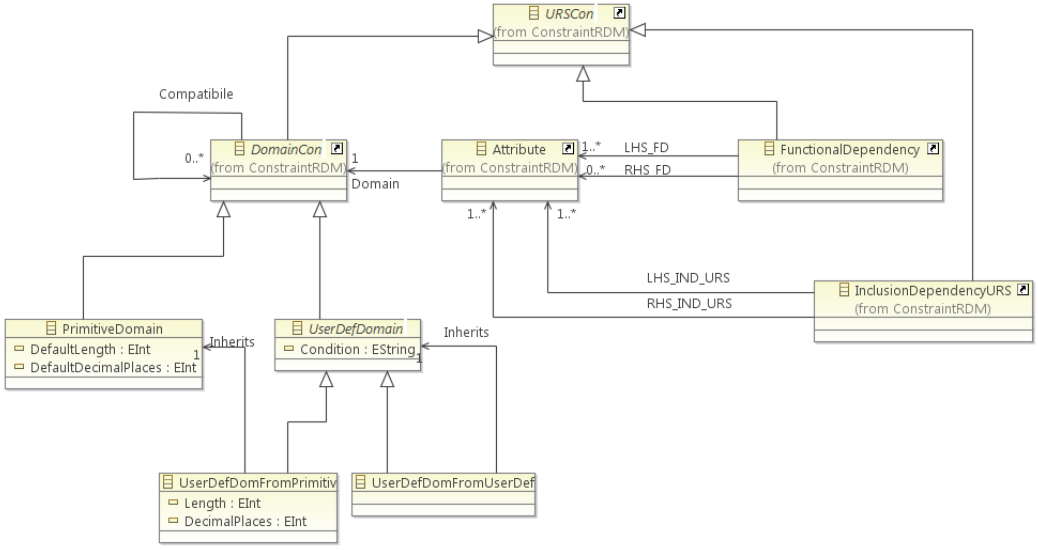

Fig. 3. A meta-model of URS concept

3.2 A meta-model of relational db schema concept

A relational db schema is composed of a set of relation schemes and a set of multi-relational constraints (Fig. 2). 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).

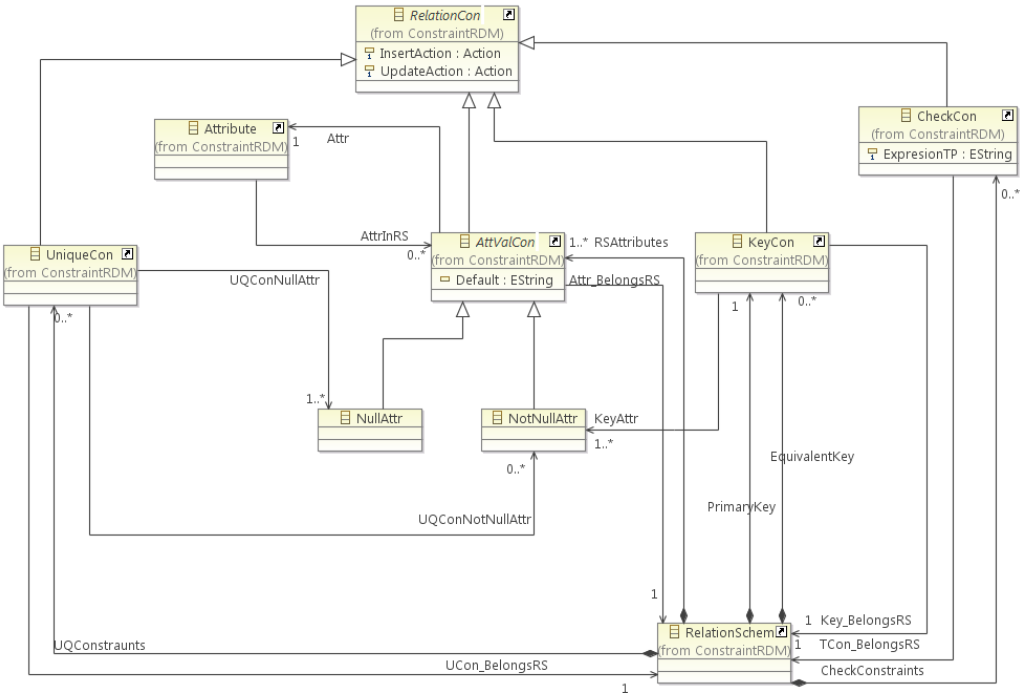

Fig. 4. A meta-model of relation scheme concept

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 (IRIC, 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.

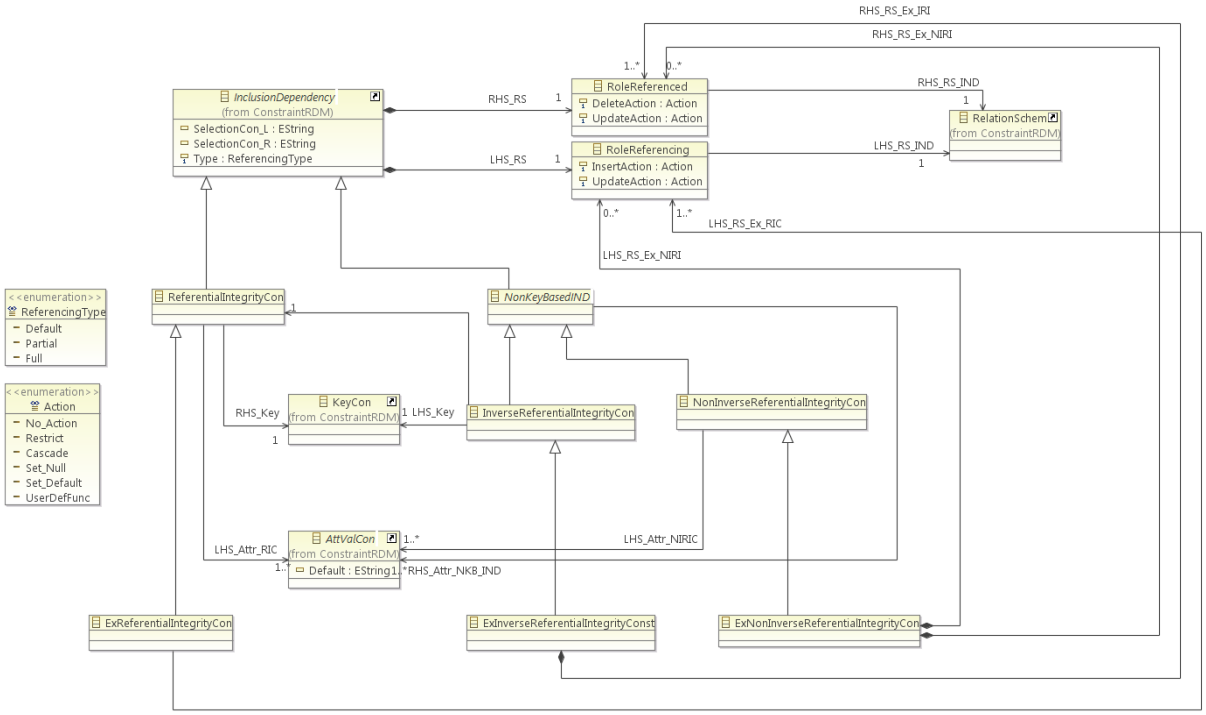

Fig. 5. A meta-model of inclusion dependency concept

Finally, a meta-model of extended tuple constraint is presented in Fig. 1.

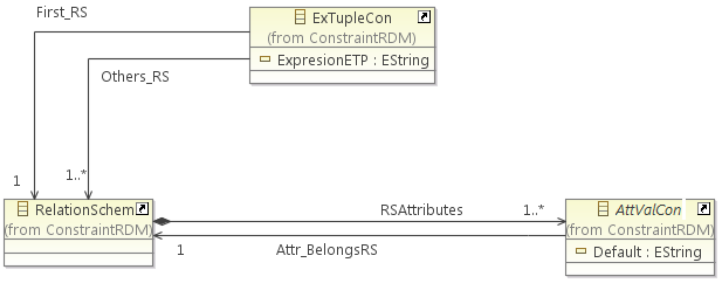

Fig. 6. A meta-model of extended tuple constraint concept

4 AN EXAMPLE OF RELATIONAL DB 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 db designers in a way that they don't 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 db schema meta-model comprising all kinds of constraints according to theoretically defined relational data model. Here we use the example of University db schema to explain some kinds of constraints that are not broadly accepted within db designers' community. In Fig. 7 the conceptual db schema of University database is visually represented by means of UML class diagram to facilitate better understanding of db constructs and relationships between them. The relational db 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 $N_l(R_l, C_l)$, where N_l is the relation scheme name, R_l set of attributes, and C_l set of relation scheme constraints. An inclusion dependency is a statement of the form $N_l[LHS] \subseteq N_r[RHS]$, where *LHS* and *RHS* are non-empty arrays 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. In the following text we enumerate relation schemes and multi-relation constraints of University db schema and give the explanation of specified constraints.

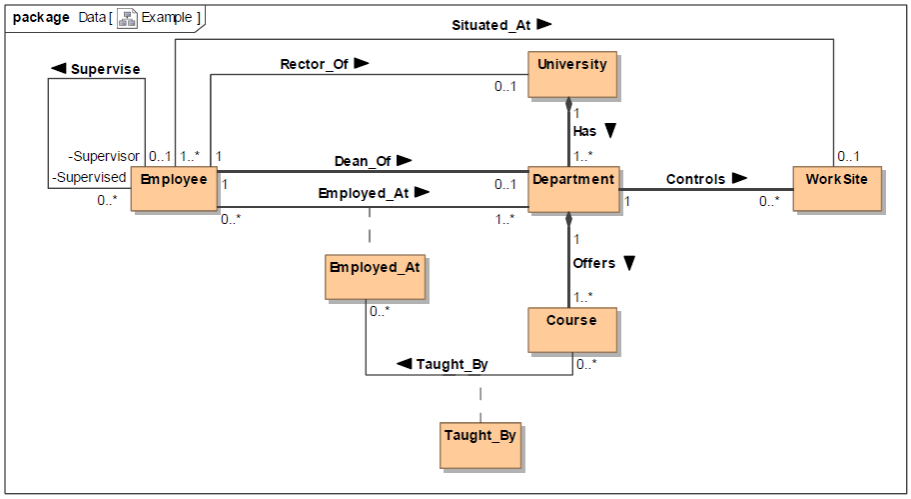

Fig. 7. The conceptual db schema of University database

***Employee*({*EmpId*, *EmpFName*, *EmpLName*, *EmpBirthD*, *EmpSSN*, *EmpPosition*, *PasportNo*,**
***SupervisorId*, *WSId*}, {PrimaryKey(*EmpId*), EquivalentKey(*EmpSSN*), Unique(*PasportNo*),**
CheckCon((*EmpPosition* = 'Prof' $\vee$ *EmpPosition* = 'Assistent') $\Rightarrow$ *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 *PasportNo* 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*, *PartTimePercent*}, {PrimaryKey(*EmpId* + *UniId* + *DepId*)})**
***Taught_By*({*UniId*, *DepId*, *CourseId*, *EmpId*, *ClassesPerWeek*},**
{PrimaryKey(*UniId* + *DepId* + *CourseId* + *EmpId*)})

1. *Employee*[*SupervisorId*] $\subseteq$ *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*] $\subseteq$ [*EmpId*], that is represented by *InclusionDependecyURS* concept in RDBSMM (Fig. 3).
2. *Employee*[*WSId*] $\subseteq$ *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. *WorkSite*[*WSId*] $\subseteq$ *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 *InverseReferentialIntegrityCon* concept (Fig. 5).
4. *University* [*RectorId*] $\subseteq \sigma_{EmpPosition = \text{'Prof}}$*Employee*[*EmpId*]

Here we have an example of selective RIC, since there is a selection condition *EmpPosition* = 'Prof' on the right-hand side of IND. The selection condition can be specified using the feature *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. *Department* [*UniId*] $\subseteq$ *University* [*UniId*] (RIC)
6. *University* [*UniId*] $\subseteq$ *Department*[*UniId*] (IRIC)
7. *Department* [*DeanId*] $\subseteq \sigma_{EmpPosition = \text{'Prof'}}$ *Employee* [*EmpId*]
   This is another selective IND modeling a business rule that the dean of a department may be only an employed professor.
8. *WorkSite* [*UniId* + *DepId*] $\subseteq$ *Department*[*UniId* + *DepId*] (RIC)
9. *WorkSite* $\triangleright\triangleleft$ *Employee*[*EmpId* + *UniId* + *DepId* + *WSid*] $\subseteq$
   Employed_At $\triangleright\triangleleft$ *Employee*[*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 *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 *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. *Course*[*UniId* + *DepId*] $\subseteq$ *Department*[*UniId* + *DepId*] (RIC)
11. *Department*[*UniId* + *DepId*] $\subseteq$ *Course*[*UniId* + *DepId*] (IRIC)
12. *Employed_At*[*EmpId*] $\subseteq$ *Employee*[*EmpId*] (RIC)
13. *Employee*[*EmpId*] $\subseteq$ *Employed_At*[*EmpId*] (IRIC)
14. *Employed_At*[*UniId* + *DepId*] $\subseteq$ *Department*[*UniId* + *DepId*] (RIC)
15. *Taught_By*[*UniId* + *DepId* + *CourseId*] $\subseteq$ *Course*[*UniId* + *DepId* + *CourseId*] (RIC)
16. *Taught_By*[*EmpId* + *UniId* + *DepId*] $\subseteq$
    $\sigma_{EmpPosition = \text{'Prof' or } EmpPosition = \text{'Assistent'}}$ *Employed_At* $\triangleright\triangleleft$ *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 *SelectionCon_R* of *InclusionDependency* concept of RDBSMM (Fig. 5).
17. ($\forall t \in$ *Employee* $\triangleright\triangleleft$ *Taught_By* $\triangleright\triangleleft$ *Course*)
    ((t[*EmpPosition*] = 'Prof' $\Rightarrow$ t[*ClassesPerWeek*] <= t[*LectureClassesPW*]) $\wedge$
    (t[*EmpPosition*] = 'Assistent' $\Rightarrow$ t[*ClassesPerWeek*] <= t[*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 *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 then the number of lecture 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 then the number of laboratory classes for the course per week.

5 RELATED WORK

Mapping of object/OR/ER/EER models to relational DB 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 db 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.

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 db schema meta-model that we presented in this paper is an intensional meta-model. Our future research has to consider extensional db 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 [26], 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 [25], the authors propose through a case study supported by a tool, a model-driven development of OR db schemas. To that end, Vara et. al. have implemented an ATL model transformation that generates an OR db model from a conceptual data model and a MOFScript model to text transformation that generates the SQL code for the modeled db schema. As part of the proposal they have defined a MOF-based Domain Specific Language (DSL) for OR db modeling as well as a graphical editor for such DSL. They presented Oracle 10g meta-model that can be classified as vendor-specific physical db schema meta-model according to the classification presented in our paper.

Lano and Kolahdouz-Rahimi in [18] and [19] presented case study of UML to relational database model transformation. In the context of relational db schema meta-model presented in our paper the relational db meta-model presented in [18] and [19] is rather simplified and does not differentiate between standard and vendor specific constructs.

In [12] 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 db schema meta-model according to the classification presented in our paper.

Calero et. al. in [11] have introduced an ontology for increasing the understandability of the SQL:2003 concepts. Their SQL:2003 meta-model can be seen as a standard db schema meta-model.

Cabot and Teniente in [10] 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 [9] 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 db 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 *trans*ML, 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 db re-engineering process.

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 with other data models.

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.

6 CONCLUSION

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 db meta-models. However, to the best of our knowledge, we could not find any systematical overview of db meta-models at different abstraction levels.

The main contribution of our paper may be two folded: through the db meta-model classification presented in Section 2 and through detailed relational db 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 plan to use presented meta-model 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 will enable reverse engineering, while IIS*Studio tool will be used for forward engineering and generating executable application prototypes.

ACKNOWLEDGMENT

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. (2007). 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.

[2] Atzeni, P., Gianforme, G., Cappellari, P.(2009). A universal meta-model and its dictionary. T. Large-Scale Data and Knowledge-Centered Systems 1, 38–62.

[3] Aleksic, S., Ristic, S., Lukovic, I., Celikovic, M. (2013). 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 (accepted for publishing).

[4] Assmann, U., Zchaler and S., Wagner, G. (2006). Ontologies, Meta-Models, and the Model-Driven Paradigm. In: Calero, C., Ruiz, F., Piattini, M. (eds.) Ontologies for Software Engineering and Software Technology

[5] Atkinson, C. and Kuhne, T., (2005). A Generalized Notion of Platforms for Model-Driven Development, In *Model-Driven Software Development*, ed. Sami Beydeda, Matthias Book and Volker Gruhn, Berlin Heidelberg: Springer, 119−136.

[6] Bernstein, P. (2003). Applying Model Management to Classical Meta Data Problems. Paper presented at the Conference on Innovative Database Research (CIDR), Asilomar, CA, January, 5.

[7] Bezivin, J. (2005). On the unification power of models. *Software and Systems Modeling* 4(2), 171–188.

[8] Boronat, A. and Meseguer, J.. (2008). An algebraic semantics for MOF. In *Proceedings of the Theory and practice of software, 11th international conference on Fundamental approaches to software engineering* (FASE'08/ETAPS'08), Fiadeiro, L. and Inverardi, P. (Eds.). Springer-Verlag, Berlin, Heidelberg, 377-391.

[9] Cabot, J., Clarisó, R., Guerra, E., and De Lara, J. (2010). Verification and validation of declarative model-to-model transformations through invariants. *Journal of Systems and Software*, *83*(2), 283-302.

[10] Cabot, J. and E. Teniente. (2009). Incremental integrity checking of uml/ocl conceptual schemas. *Journal of Systems and Software*, 82(9), 1459–1478.

[11] Calero, C., Ruiz, F., Baroni, A., Brito e Abreu, F. , Piattini, M. (2006). An ontological approach to describe the SQL:2003 object-relational features. Computer Standards & Interfaces, Volume 28, Issue 6, September 2006, Pages 695–713.

[12] del Castillo, R.P., García-Rodríguez, I., and Caballero, I. (2009). 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.

[13] Date, C.J. and Darwen, H. (2006). Types and the Relational Model. The Third Manifesto, 3rd ed. Addison Wesley, Reading.

[14] Eessaar, E., (2007). 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] Eclipse Modeling Framework, [Online] Available: http://www.eclipse.org/modeling/emf/. (retrieved January, 05, 2013).

[16] Guerra, E., de Lara, J., Kolovos, D., Paige, R., dos Santos, O. (2011). Engineering model transformations with transML. Software and Systems Modeling. Springer-Verlag.

[17] Gogolla, M., Lindow, A., Richters, M. and Ziemann, P. (2002). Meta-model transformation of data models. Position paper. *WISME at the UML 2002*.

[18] Lano, K., and Kolahdouz-Rahimi, S. (2011). Model-driven development of model transformations. *Theory and Practice of Model Transformations*, 47-61.

[19] Lano, K., and Kolahdouz-Rahimi, S. (2013). Constraint-based specification of model transformations Journal of Systems and Software, Volume 86, Issue 2, Pages 412–436.

[20] Meta-Object Facility, [Online] Available: http://www.omg.org/mof/. (retrieved January, 05, 2013).

[21] Mogin, P., Luković, I., Govedarica, M., (2004). *Database Design Principles*, University of Novi Sad, Faculty of Technical Sciences & MP "Stylos", Novi Sad, Serbia.

[22] 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 January, 05, 2013).

[23] Polo, M., Garcia-Rodriguez, I., and Piattini, M. (2007). An MDA-based approach for database re-engineering. *J. Softw. Maint. Evol.* 19, 6 (November 2007), 383-417.

[24] Stahl T, Völter M, Bettin J, Haase A, Helsen S. (2006). Model Driven Software Development: Technology, Engineering, Management. John Wiley & Sons, Ltd.

[25] Vara, J., Vela, B., Bollati, V.A. and Marcos, E. (2009). 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.

[26] Wang, H., Shen, B. and Chen, C.(2009). Model-Driven Reengineering of Database. *Software Engineering, 2009. WCSE '09. WRI World Congress on* , vol.3, no., pp.113-117.