UDC 004.62 https://doi.org/10.15407/pp2021.01.056

I.S. Chystiakova

MAPPING OF THE DESCRIPTIVE LOGIC INTO RDF USING BINARY RELATIONAL DATA MODEL

This paper is dedicated to the data integration problem. The descriptive logic and the relational data model are at the heart of a study. They have been used to create a mapping method on the theoretical level. The previous studies are continued in this paper to prove on practice a mapping creation method between the descriptive logic and the binary relational data model, which is a part of a mapping method. The method uses the binary relational data model as an integrating model. The task to prove the theoretical mapping method on practice was formulated. A question how to map the binary relational data model into RDF-triples was considered. A brief overview of the R2R ML conversion tool was given. Triple maps were created to convert a conceptual information model of descriptive logic into RDF triplets with the help of R2R ML. Also, triples maps are described to convert basic mapping mechanisms into RDF with the help of R2R ML.
Key words: binary relational data model, description logic, mapping, RDF, DL, RM^2, ALC, OWL.

Introduction

A complex problem of data integration in the semantic web exists in the modern scientific field of research. An analysis of this problem can be found in the [1]. A task to establish an interaction between a descriptive logic (DL) and a relational data model (RDM) arises as a part of solution of the data integration problem. Such interaction is called mapping. To establish an interaction means to create mapping mechanisms between the DL and the RDM. A series of studies [1–7] is dedicated to the analysis and solution of the mapping creation problem. **A binary relational data model (RM^2)** [6] was created as a result of this series. The main task of RM^2 is to be an integrating model for creating mappings. How to map the descriptive logic ALC and its main components into RM^2 was described in the study [8] as well as how to map the classical RDM into RM^2. This approach was described purely on a theoretical level. Until now, the lack of any practical approbation was a significant drawback of the proposed results.

A method to test mappings between DL and RM^2 with the help of RDF graphs is proposed in this paper. The main idea is to map DL-to-RDM conversion formulas into RDF, and then to test them for workability within the unified RDF framework. The results of mapping DL expressions into RDF using OWL 2 were published in [9]. This study focuses on creation of mappings for RDM expressions into RDF using R2R ML.

Section 1 is dedicated to the problem statement. Section 2 provides a short overview of the R2R ML. Section 3 summarizes the main theoretical aspects of the DL-RDM mapping method. Section 4 describes the RM^2 to RDF mapping rules using R2R ML. Section 5 contains conclusions.

Problem statement

A theoretical approach how to describe mappings between DL and RDM is presented in a series of studies [1-9]. A binary relational data model (RM^2) is proposed as an integrating model. The approbation task of this approach is based on a number of facts.

Firstly, it is known that DL is the mathematical basis of any ontology description language. Thus, all the constructors of concepts and roles of the underlying DL are reflected in the toolbox of the corresponding language. OWL 2 is not an exception.

Secondly, only binary connections are allowed in RM^2. Both binary and n-ary connections are allowed in a classical RDM. A result of [10] seems to indicate that any n-ary relation can be represented by a set of binary ones. Thus, any classic RDM can be expressed with RM^2. A way how to convert RDM into RM^2 is described in [6].

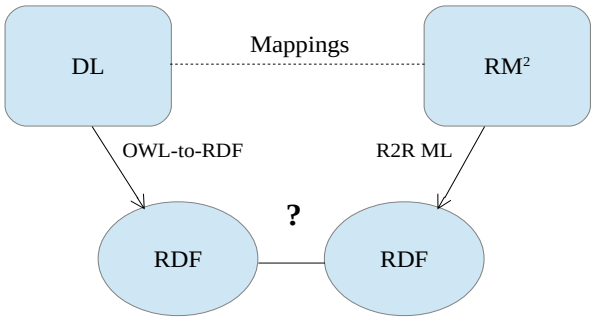
The idea of testing is not new. The statements of the theory being proved are transformed into statements of the established theory. The converted expressions are then checked for truth within the well-established

theoery using its own methods and properties. If the final expression is true in the existing theory, then the original expression is also true in the study area.

Thus, the problem statement ti test mappings between DL and RDM is formulated as follows. On the one hand, DL statements (expressed in OWL 2) are mapped to the RDF triplets using OWL 2-to-RDF conversion rules. On the other hand, relational database (RDB) expressions are mapped to the RDF triplets using R2R ML. The resulting RDF triplets constitute a set of RDF graphs. The resulting graphs are compared for equivalence.

The implementation idea is schematically shown in Figure 1.

Figure1. Approbation scheme of the mapping



method between DL and RM^2

OWL 2-to-RDF conversion rules have official W3C status [11]. R2R ML also has official W3C status [12].

The algorithm for testing mappings between DL and RM^2 is as follows:

Step 1. There is a DL expression. It is mapped into the RM2 statement.

Step 2. The statement is mapped intto an OWL 2-expression from the DL side.

Step 3. OWL 2-expression (step 2) is converted to RDF-triples that form an RDF-graph. OWL 2-to-RDF rules are used for mapping.

Step 4. The statement is formulated in terms of RDB from the RM2 side.

Step 5. RDB expression is converted to the RDF-triples that form an RDF-graph. R2R ML is used to create mappings.

Step 6. The RDF-graph (step 3) is compared for equivalence with another RDF-graph (step 5). If they are equivalent, then the DL-to-RM2 mapping formula is true.

The mappings from step 1 are described in [8] on the theoretical level. Also, they are briefly described in section 3. The mappings from steps 2 and 3 are described in article [9]. The current work will present a way to perform transformations from steps 4 and 5. The comparison from step 6 remains in the field of future research.

It is known that OWL 2 is based on the SROIQ descriptive logic. Thus, the OWL 2-to-RDF mapping area is limited to only those operations that are present in DL SROIQ. DL SROIQ syntax include the following: DL ALC syntax, DL axiomatics, numerical constraints, nominals, and inverse roles. The theoretical part of DL-to-RM^2 mappings has been worked out for several role constructors. The issue of mapping some of the role constructors in RDF (except the inverse role) remains open.

There are a number of features in R2R ML. It allows you to transform the structure and integrity constraints of an RDB into RDF triplets. However, there are some features of mapping the manipulative part of RDM into RDF. Any operation can be mapped only as part of an SQL query. Each SQL query is represented as a logical table within a triples map generated for such table. Thus, there are no mechanisms to transform directly each individual operation of relational algebra (RA) within R2R ML itself. A small overview of R2R ML is presented in section 2.

A key question of the approbation task is to prove the equivalence of graphs, obtained as a result of pairwise mapping of the DL and RM^2 statements. The results of [13] demonstrate that RDF-graph is a special case of a regular graph. This means that the question of equivalence is reduced to proving their isomorphism. In the work [13] RDF-graph is analyzed as a special case of a usual graph. Several criteria for graph isomorphism in the general case are also studied. Based on these criteria, three necessary and sufficient conditions for the RDF-graphs equivalence of are formulated. They are as follows:

1. *Equal number of vertices*. Both graphs must contain the same number of vertices, otherwise they are not isomorphic.

2. *Vertices equivalence*. Each vertex of one graph must have an equivalent in the other graph in a pairwise comparison. Otherwise, such graphs are not isomorphic.

3. *Ribs equivalence*. Each edge of one graph must have an equivalent in the other graph in a pairwise comparison. Otherwise, such graphs are not isomorphic.

Reducing a graph to a self-isomorphic remains the last question in the problem framework. As a result of mappings, the following situation may arise at the RDF level. The vertex of one graph will semantically correspond to a subgraph of the comparable graph. Such a subgraph can consist of several vertices connected by edges. This situation is possible, since a large number of anonymous (empty) nodes appears during the mapping process. Such nodes have their own semantic purpose. Thus, the question of reducing an RDF-graph to the self-isomorphic remains open.

Preliminaries: R2R ML Overview

Here is a brief overview of the R2R ML. It will be used in Section 4 to map RM^2 expressions into RDF.

The R2R ML has a development history. Tim Berners-Lee published an article [14] in 1998. It was entitled as “Relational Databases on the Semantic Web”. This paper discusses the concept of presenting any database in the semantic web. Its main idea is to use RDF as an ER model. The ER model establishes a correlation between relational database elements and RDF-triples. The author of the concept proposes to use an XML-format of serialization of RDF-triples. The paper also presents the "bottlenecks" of mapping of the RDB information into RDF.

Then, in 2007, the conference "W3C Workshop on RDF Access to Relational Databases" was held [15]. It was dedicated to the presentation of ordinary relational data in RDF format, as well as the use of RDF in RDB queries. The W3C RDB2RDF Incubator Group was established in 2009 and operated until 2012. It declared the following goals:

- to study and classify existing approaches of mapping the relational data in RDF;
- to determine the need for standardization of this area;
- to determine the mechanisms of generation the RDF-triples from one or more relational databases without the loss of information;
- to study the possibility of mapping OWL classes into relational data, taking into account the developed approach.

As a result, a document with official R2R ML recommendations was received and published. Figure 2 shows the chronology of R2R ML development [16].

Now it is necessary to consider the main standings of R2R ML. It`s important to understand how RDF-triples are formed from relational data. Let's turn to the official documentation [12].

R2R ML – is a language for expressing customized mappings from relational databases to RDF datasets. It defines the mapping of the relational database into the RDF. An R2RML mapping refers to *logical tables* to retrieve data from the *input database*. The input to an R2R ML mapping is called **the input database**. Figure 3 highlights a UML diagram of the R2R ML language overview. The picture is taken from the official R2R ML website.

An R2R ML mapping – is a structure that consists of one or more *triples maps*. **A triples map** is a rule that maps each row in the logical table to a number of RDF triples. The rule has two main parts: *a subject map* and multiple *predicate-object map*.

A subject map generates the subject of all RDF triples that will be generated from a logical table row. The subjects are often IRIs that are generated from the primary key column(s) of the table.

Predicate-object maps in turn consist of *predicate maps* and *object maps*. Sometimes predicate-object map additionally has *referencing object maps*.

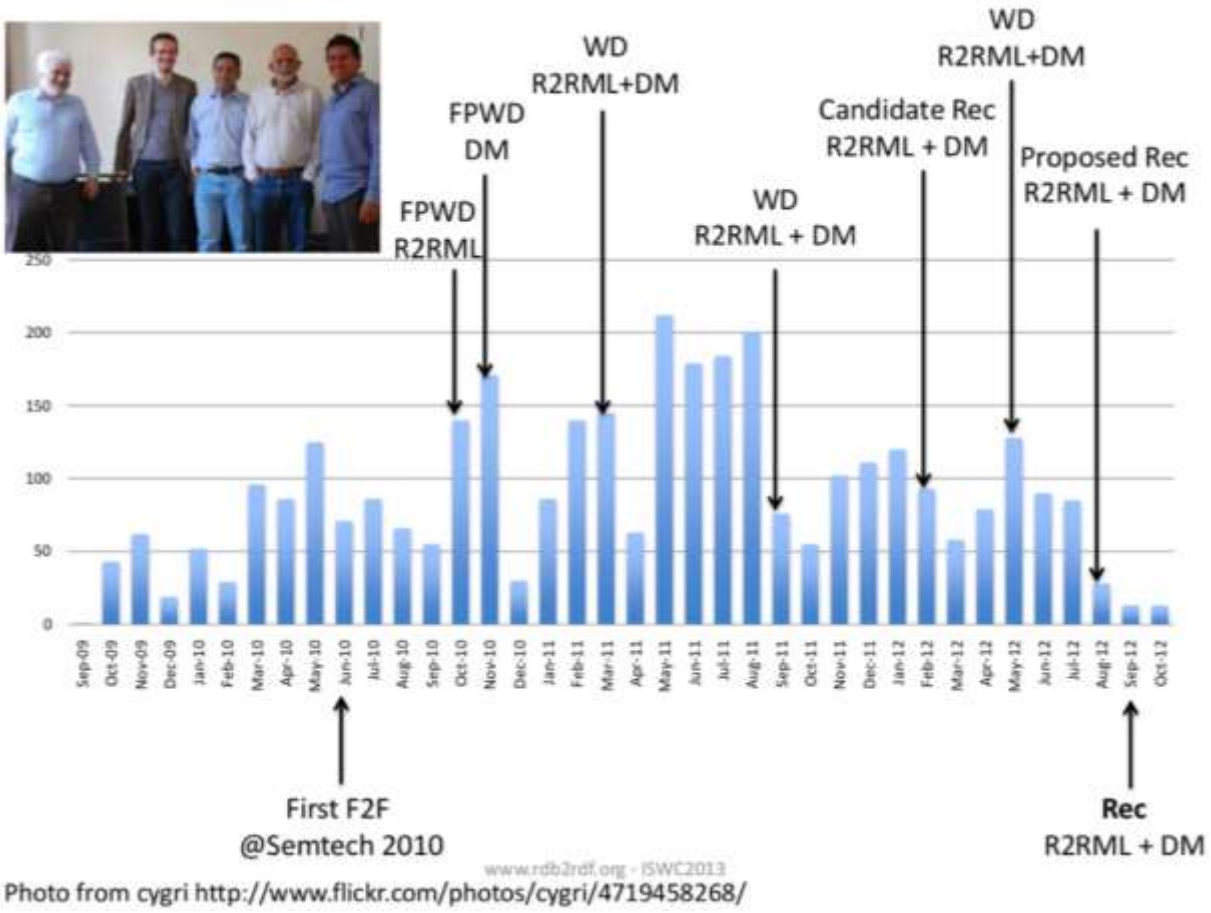


Figure 2. Chronology of the R2R ML development

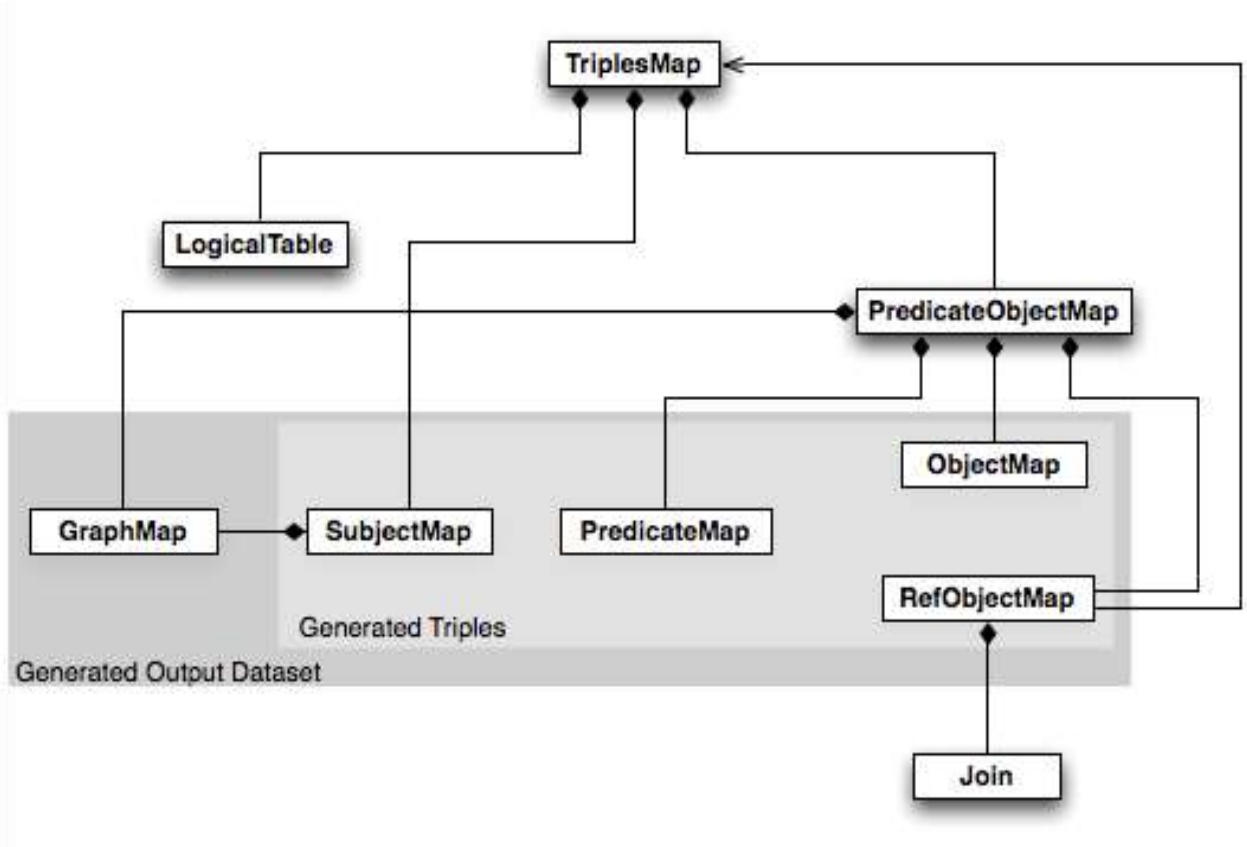


Figure 3. R2R ML overview

Triples are produced as follows. Subject map is combined with a predicate map and object map. These three are applied to each logical table row. By default, all RDF triples are in the default graph of the output dataset. A triples map can contain graph maps. Such graph maps place some or all of the triples into named graphs instead of the default graph.

Triples map (fig. 2) consists of the three main components: *a logical table*, *a subject map*, *a predicate-object map*. In detail each of them is as follows.

A logical table is a tabular SQL query result that is to be mapped to RDF triples. In simple words a logical table is **what** will be displayed. It can take one of the following three forms:

- SQL base table;
- SQL view;
- R2R ML view (a valid SQL query). It got its name because it only emulates a SQL view without modifying the database.

A **logical table row** is a row in a logical table. It can be a row of a base SQL table or SQL view. It can also be a row of an R2R ML view obtained with an SQL query.

A **column name** is the name of a column of a logical table. A column name must be a *valid SQL identifier*. For example, the name of a SQL object, such as a column, table, view, schema, or catalog. Column names do not include any qualifying table, view or schema names.

The logical table in the triples map is written using one of the properties:

- *rr:tableName* – specifies the table or view name of the base table or view. Its value must be a valid schema-qualified name. It is a sequence of one, two or three valid SQL identifiers, separated by the dot character (“.”). The three identifiers name, respectively, a catalog, a schema, and a table or view. If no catalog or schema is specified, then the default catalog and default schema of the SQL connection are assumed.

- *rr:sqlQuery* and *rr:sqlVersion* – defines R2R ML view and SQL query version. R2R ML view is a logical table whose contents are the result of executing a SQL query against the input database. If rr:sqlVersion property is absent, then the rr:sqlQuery property value conforms to Core SQL 2008.

Before considering the subject maps and the predicate-object maps, it’s necessary to give a number of definition.

An **RDF term** is either an IRI, or a blank node, or a literal.

A **term map** is a function that generates an *RDF term* from a *logical table row*. The result of that function is a generated RDF term. Term maps are used to generate the subjects, predicates and objects of the RDF triples. In turn, RDF triples are generated by a triples map. Consequently, there are several kinds of term maps, depending on where in the mapping they occur: subject maps, predicate maps, object maps and graph maps. The *referenced columns* of a term map are the set of column names referenced in the term map. They depend on the type of term map.

A **subject map** is a term map. It specifies a rule for generating the subjects of the RDF triples.

In the triples map, the subject map is specified as follows:

- *rr:subjectMap* is a property, which value must be a specific subject map;

- *rr:subject* is a shortcut constant property whose value is IRI.

The subject map can contain the *rr:class* property. Its value must be an IRI, which is called class IRI. In this case, the RDF expression generated by the subject map will look like this. For each subject, a triple is created with the *rdf:type* predicate and the *rr:class* property as object value. A subject map can contain several class IRIs at the same time. There are cases when the class IRI must be computed based on the contents of source database. In such situations, a predicate object map is used. The predicate value is indicated by rdf: type. The value of an object is set through a non-constant object map.

A **predicate-object map** is a function that creates one or more predicate-object pairs for each logical table row of a logical table. It is used in conjunction with a subject map to generate RDF triples in a triples map. A predicate-object map is represented by a resource that references: one or

more predicate maps and one or more object maps.

A **predicate map** is a term map. It can be defined in two ways:

- *rr:predicateMap* is a property, whose value must be a predicate map;
- *rr:predicate* is a constant shortcut property whose value is IRI.

An **object map** is a term map. It can be defined in two ways:

- *rr:objectMap* is a property, whose value must be either an object map, or a referencing object map;
- *rr:object* is a constant shortcut property whose value is IRI or literal.

A **referencing object map** allows using the subjects of another triples map as the objects generated by a predicate-object map. Since both triples maps may be based on different logical tables, this may require a join between the logical tables. However, the join condition (one or more joins) is optional.

A referencing object map is represented by the following resources:

- *rr:parentTriplesMap* is a property, whose value must be a triples map. Such triples map is known as the referencing object map's parent triples map. The value of object will be extracted exactly from the parent triples map.
- *rr:joinCondition* is a property whose values must be join conditions options.

A **join condition** is represented by a resource that has exactly **one value** for each of the following two properties:

- *rr:child* is a property, whose value is known as the join condition's child column. It must be a column name that exists in the logical table of the triples map (that contains the referencing object map).
- *rr:parent* is a property whose value is known as the join condition's parent column. It must be a column name that exists in the logical table of parent triples map (of the referencing object map's). The name of the parent triples map was specified in the rr:parentTriplesMap property.

Here is one of the examples given in the description of the R2R ML standard [12]. The following example database consists of two tables, EMP (table 1) and DEPT (table 2), with one row each:

Table 1

EMP

EMPNO Integer primary key	*EName Varchar (100)*	*Job Varchar (20)*	*DEPTNO Integer references DEPT (DEPTNO)*
7369	*Smith*	*Clerk*	*10*

Table 2

DEPT

DEPTNO Integer primary key	*DName Varchar (30)*	*Loc Varchar (100)*
10	*Appserver*	*New York*

The following R2R ML mapping document will produce the desired triples from the EMP table:

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix ex: <http://example.com/ns#>.
<#EmpMap>
rr:logicalTable [ rr:tableName "EMP" ];
rr:subjectMap [
rr:template "http://data.example.com/employee/{EMPNO}";
rr:class ex:Employee;
];
rr:predicateObjectMap [
rr:predicate ex:name;
rr:objectMap [ rr:column "EName" ];
];
rr:predicateObjectMap [
rr:predicate ex:job;
rr:objectMap [ rr:column "Job" ];
];
rr:predicateObjectMap [
rr:predicate ex:department;
rr:objectMap [
rr:parentTriplesMap <#DeptMap>;
rr:joinCondition [
```

rr:child "DEPTNO";
rr:parent "DEPTNO";
];
].

The definition of a triples map that generates the desired DEPT triples follows.

<#DeptTableView> rr:sqlQuery """
SELECT DEPTNO, DName, Loc,
(SELECT COUNT(*) FROM EMP WHERE EMP.DEPTNO=DEPT.DEPTNO) AS Staff
FROM DEPT;
""".
<#DeptMap>
rr:logicalTable <#DeptTableView>;
rr:subjectMap [
rr:template "http://data.example.com/department/{DEPTNO}";
rr:class ex:Department;
];
rr:predicateObjectMap [
rr:predicate ex:name;
rr:objectMap [rr:column "DName"];
];
rr:predicateObjectMap [
rr:predicate ex:location;
rr:objectMap [rr:column "Loc"];
];
rr:predicateObjectMap [
rr:predicate ex:staff;
rr:objectMap [rr:column "Staff"];
].

The desired RDF triples to be produced from this database are as follows:

<http://data.example.com/employee/7369> rdf:type ex:Employee.
<http://data.example.com/employee/7369> ex:name "Smith".
<http://data.example.com/employee/7369> ex:job "Clerk".
<http://data.example.com/employee/7369>ex:department<http://data.example.com/department/10>.
<http://data.example.com/department/10> rdf:type ex:Department.
<http://data.example.com/department/10> ex:name "Appserver".
<http://data.example.com/department/10> ex:location "New York".
<http://data.example.com/department/10> ex:staff 1.

Preliminaries: mapping DL into RM^2

The approbation task of testing mappings between DL and RDM is based on a series of theoretical studies [1–7]. Here is a brief summary of them. This summary will be used in Section 4 to create mappings from RM^2 to RDF.

Binary Relational Data Model (RM^2) [6] was developed to address the issue of establishing relations between DL and RDM. It has several advantages over the classic RDM by Codd [17]. For example, RM^2 contains support for the open world assumption, while classical RDM works according to epy closed world assumption. Unlike Codd's RDM, RM^2 supports the implementation of DL constructors and concepts and roles axioms. The RM^2 is described in detail in [6].

To describe mappings, the first step is to build a conceptual information model of descriptive logic. The main task of the conceptual information model of any subject area is to define the basic concepts and to describe their properties and relations. The ER language is one of the most used for this purpose. It operates with the concepts of entity, attribute and relation. The Barker dialect [18] of the ER language was used to describe the conceptual information model of descriptive logic (Fig. 4).

There are three basic entities in the model:

- Concept – to present DL concepts
- Role – to present DL roles
- CIndividual – to present DL individuals

Each entity has a single attribute that is called “Name”. This attribute uniquely identifies the entities. The rest of entities are relation entities. They represent binary relations between basic entities. Link entities do not have their own attributes and are uniquely identified only by their links.

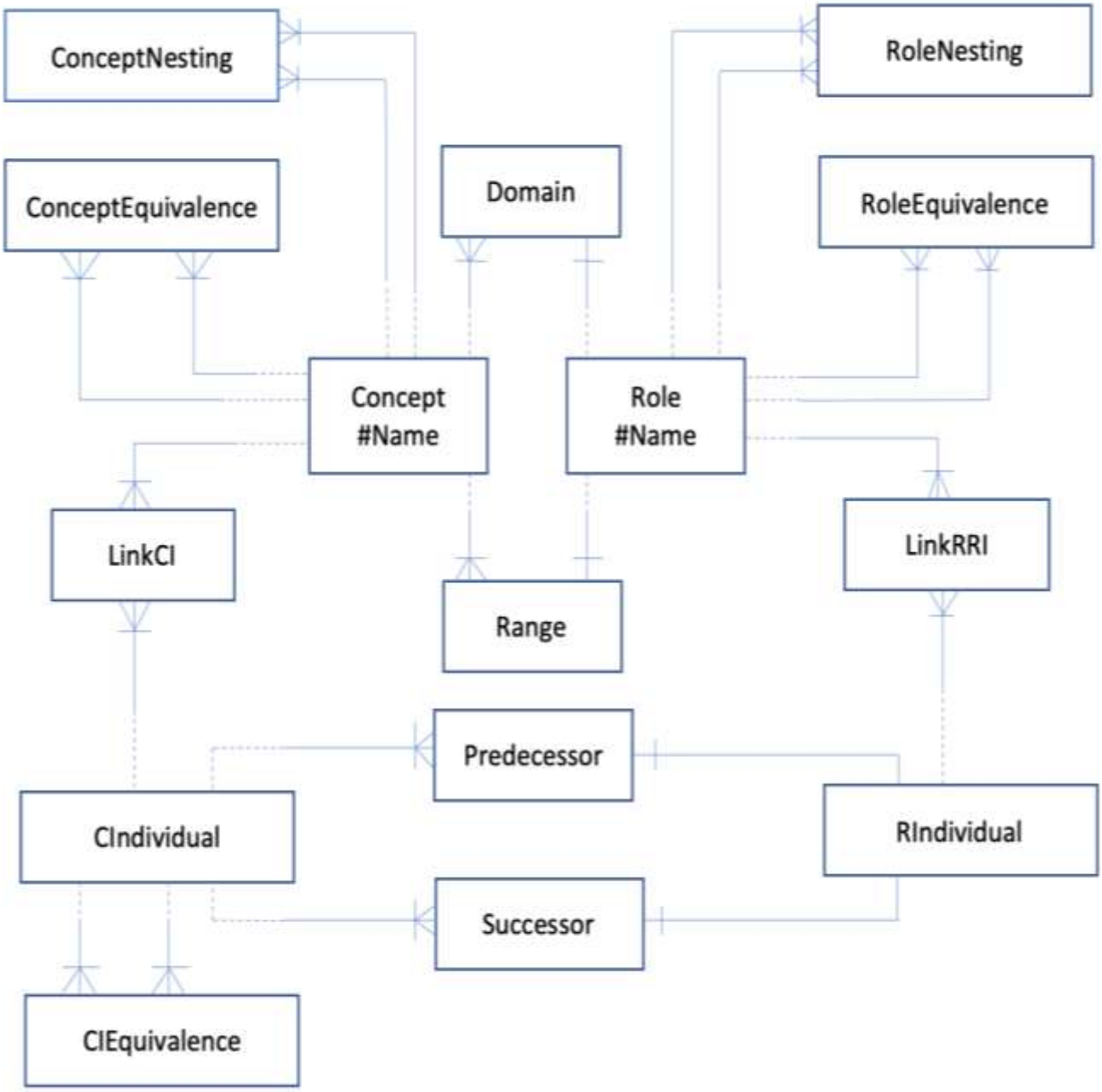


Figure 4. Conceptual information model of descriptive logic

The RM^2 scheme was built according to the given ER-model. The transformation algorithm described in [2, 6, 8] was used to construct the scheme. Additionally, the idea of placeholder attributes was used to represent the primary keys. This idea firstly was proposed by E. Codd [17], the founder of RDM. Each entity is represented as a RM^2 relationship, since the constructed ER-scheme fits the 3NF requirement. The name of the relationship is the same as the name of the entity. Table 3 highlights all the ER-model entities with their descriptions and the corresponding relationships with the list of attributes.

To describe mappings, the second step is to express DL ALC in RM^2. ALC is the simplest variant of DL. It is included in all widely used dialects of other DLs. It should be recalled, that description logics uses constructs that have semantics given in predicate logic. The ALC semantics is defined through the concept of interpretation.

Table 3

ER-model entities with their descriptions and corresponding relationships

Entity (relationship) name	**Entity description**	**Relationship attributes with description**
Concept	Presents concepts	CPK – primary key Name – concept name
Role	Present roles	RPK – primary key Name – role name IsTransitive – is role transitive
CIndividual	Present individuals	CIPK – primary key Name – individual name
RIndividual	Pesent role individuals	RIPK – primary key
Domain	Present role domain	CFK – foreign key on Concept RFK – foreign key on Role
Range	Present role range	CFK – foreign key on Concept RFK – foreign key on Role
LinkCI	Allows many-to-many relations between concepts and individuals	CFK – foreign key on Concept CIFK – foreign key on CIndividual
LinkRRI	Allows many-to-many relations between roles and their instances	RFK – foreign key on Concept RIFK – foreign key on RIndividual
Predecessor	Presents the first individual of a role individual	CIFK – foreign key on CIndividual RIFK – foreign key on RIndividual
Successor	Presents the second individual of a role individual	CIFK – foreign key on CIndividual RIFK – foreign key on RIndividual
Concept Nesting	Represents the concept inclusion (hierarchy) axiom	CInFK – foreign key on Concept («child») COutFK – foreign key on Concept («parent»)
Concept Equivalence	Represents the concept equivalence axiom	CForFK – foreign key on Concept («which is equal») CIsFK – foreign key on Concept («equals to which»)
Role Nesting	Represents the role inclusion (hierarchy) axiom	RInFK – foreign key on Role («child») ROutFK – foreign key on Role («parent»)
Role Equivalence	Represents the role equivalence axiom	RForFK – foreign key on Role («which is equal ») RIsFK – foreign key on Role («equals to which »)
CIEquivalence	Represents the individual equivalence axiom	CIForFK – foreign key on CIndividual («which is equal ») CIIsFK – foreign key on CIndividual («equals to which »)

Interpretation is a pair $I = (\Delta, \bullet^I)$, where

- Δ – a non-empty set called the domain of interpretation,
- $\bullet I$ – interpretation function.

Interpreter function assigns each atomic concept A a set $A^I \subseteq \Delta$, and each atomic role R a binary relationship $R^I \subseteq \Delta \times \Delta$.

Further, $C^E_{RM^2}$ will denote the RM^2 relationship extensional. This relationship corresponds to the interpretation of an arbitrary concept C.

Table 4 shows the DL ALC syntax and semantics, as well as their corresponding mapping formulas.

Table 4

ALC syntax and semantics and their corresponding mapping formulas

Syntax	**Semantics**	**Mapping**
1	**2**	**3**
ALC Syntax Concepts		
$\top$	$\top^I = \Delta$	$\top^E_{RM^2} = \pi_{Name}(CIndividual)$
$\bot$	$\bot^I = \emptyset$	Empty(Name)
C	$C^I \subseteq \Delta^I$	$C^E_{RM^2} = = \pi_{CIndividual.Name}(\sigma_{Concept.Name='C'} (CIndividual \bowtie_{CIPK=CIFK} (LinkCI \bowtie_{CFK=CPK} Concept)))$
R	$R^I \subseteq \Delta^I \times \Delta^I$	$R^E_{RM^2} = \pi_{First,Second}(\rho_{CIdividual.Name/Second}(CIndividual \bowtie_{CIPK=CFPK} (Successor \bowtie_{RIFK=RIPK} (\rho_{CIndividual.Name/First}(CIndividual \bowtie_{CIPK=CIFK} (Predecessor \bowtie_{RIFK=RIPK} (\sigma_{Role.Name='R'}(RIndividual \bowtie_{RIPK=RIFK} (LinkRRI \bowtie_{RFK=RPK} Role)))))))))$
$\neg C$	$\Delta^I \backslash C^I$	$(\neg C)^E_{RM^2} = \pi_{Name}(CIndividual) - C^E_{RM^2}$
$C \sqcap D$	$(C \sqcap D)^I = C^I \cap D^I$	$(C \sqcap D) = C^E_{RM^2} \cap D^E_{RM^2}$
$C \sqcup D$	$(C \sqcup D)^I = C^I \cup D^I$	$(C \sqcup D) = C^E_{RM^2} \cup D^E_{RM^2}$
$\exists R.C$	$\exists R.C = \{a \in \Delta \mid \exists b \in \Delta((a,b) \in R^I \wedge b \in C^I)\}$	$(\exists R.C)^E_{RM^2} = \pi_{First}(R^E_{RM^2} \bowtie_{Second=Name} C^E_{RM^2})$
$\forall R.C$	$\forall R.C = \{a \in \Delta \mid \forall b \in \Delta((a,b) \in R^I \rightarrow b \in C^I)\}$	$(\forall R.C)^E_{RM^2} = \pi_{First}(R^E_{RM^2}) - \pi_{First}(R^E_{RM^2} \cap (\pi_{First}(R^E_{RM^2}) \times (\pi_{Second}(R^E_{RM^2}) - \pi_{Name}(C^E_{RM^2}))))$
Number restrictions, nominals		
$(\geq nR)$	$(\geq nR)^I = \{e \in \Delta \mid \lvert R^I(e)\rvert \geq n\}$	$(\geq nR)^E_{RM^2} = \pi_{First}(\sigma_{1 \leq i < j \leq n} R_i.Second_i \neq R_j.Second_j(\ast_{1 \leq i \leq n} \rho_{R_i(First,Second_i)}(R^E_{RM^2})))$
$(\leq nR)$	$(\leq nR)^I = \{e \in \Delta \mid \lvert R^I(e)\rvert \leq n\}$	$(\leq nR)^E_{RM^2} = \pi_{Name}(Cindividual) - \pi_{First}(\sigma_{1 \leq i < j \leq n+1} R_i.Second_i \neq R_j.Second_j (\ast_{1 \leq i \leq n+1} \rho_{R_i(First,Second_i)}(R^E_{RM^2})))$
$(\geq nR.C)$	$(\geq nR.C)^I = \{e \in \Delta \mid \lvert R^I(e) \cap C^I\rvert \geq n\}$	$(\geq nR.C)^E_{RM^2} = \pi_{First}(\sigma_{1 \leq i < j \leq n} R_i.Second_i \neq R_j.Second_j(\ast_{1 \leq i \leq n} \rho_{R_i(First,Second_i)}(R^E_{RM^2} \bowtie_{Second=Name} C^2_{RM^2})))$
$(\leq nR.C)$	$(\leq nR.C)^I = \{e \in \Delta \mid \lvert R^I(e) \cap C^I\rvert \leq n\}$	$(\leq nR.C)^E_{RM^2} = \pi_{Name}(CIndividual) - \pi_{First}(\sigma_{1 \leq i < j \leq n} R_i.Second_i \neq R_j.Second_j (\ast_{1 \leq i \leq n} \rho_{R_i(First,Second_i)}(R^E_{RM^2} \bowtie_{Second=Name} C^E_{RM^2})))$
$\{a\}$	$\{a\}^I$	$\{a\}^E_{RM^2} = \{a^E_{RM^2}\}$
Role Constructors		
R^-	$(R^-)^I = \{(e,d) \in \Delta \times \Delta \mid (d,e) \in R^I\}$	$(R^-)^E_{RM^2} = (\rho_{R(Second,First)}(R^E_{RM^2}))$

1	2	3
¬R	$(\neg R)^I = \Delta \times \Delta \backslash R^I$	$(\neg R)^E_{RM^2} = (\rho_{Name/First}(\pi_{Name}(CIndividual)) \times \rho_{Name/Second}(\pi_{Name}(CIndividual))) - R^E_{RM^2}$
R ⊓ S	$(R \sqcap S)^I = R^I \cap S^I$	$(R \sqcap S)^E_{RM^2} = R^E_{RM^2} \cap S^E_{RM^2}$
R ⊔ S	$(R \sqcup S)^I = R^I \cup S^I$	$(R \sqcup S)^E_{RM^2} = R^E_{RM^2} \cup S^E_{RM^2}$
R ∘ S	$(R \circ S)^I = \{(e,d) \in \Delta \times \Delta \mid \exists c \in \Delta((e,c) \in R^I \wedge (c,d) \in S^I)\}$	$(R \circ S)^E_{RM^2} = \pi_{R.First,S.Second}(R^E_{RM^2} \bowtie_{R.Second=S.First} S^E_{RM^2})$
id(C)	$(id(C))^I = \{(e,e) \in \Delta \times \Delta \mid e \in C^I\}$	$(id(C))^E_{RM^2} = (\rho_{Name/First}(\pi_{Name} C^E_{RM^2})) \bowtie_{First=Second} (\rho_{Name/Second}(\pi_{C.Name} C^E_{RM^2}))$
R^+	$(R^+)^I = \bigcup_{n \geq 1} (R^I)^n$	$(R^+)^E_{RM^2} = (R^E_{RM^2})^+$
R^*	$(R^*)^I = \bigcup_{n \geq 0} (R^I)^n$	$(R^*)^E_{RM^2} = (\rho_{Name/First}(\pi_{Name} CIndividual)) \bowtie_{First=Second} (\rho_{Name/Second}(\pi_{Name} CIndividual)) \cup (R^+)^E_{RM^2}$

The DL axiom mapping has been shown in the conceptual ER scheme. Each axiom has its own entity. Each axiom has its own binary relationship in RM^2. Each relationship has two foreign keys. Each of the keys refers to concepts, roles or individuals about which the axiom is formulated.

ConceptNesting (CInFK, COutFK)

$C \sqsubseteq D$

ConceptEquivalence (CForFK, CIsFK)

$C \equiv D$

RoleNesting (RInFK, ROutFK)

$R \sqsubseteq S$

RoleEquivalence (RForFK, RIsFK)

$R \equiv S$

CIEquivalence (CIForFK, CIIsFK)

a = b

Role(RPK, Name **IsTransitive**)

TR(R)

Mapping RM^2 into RDF

RM^2 to RDF mappings can be meaningfully divided into several parts. Firstly, how to transform each RDB relationship of an RM^2 will be shown. All RDB relationships can be divided into the following groups: basic relationships (concepts, roles, individuals), connective relationships and axiom relationships. The mapping of all ALC constructs will be shown next. The description of mapping mechanisms for number restrictions, roles restrictions and nominals completes this section.

Since empty relationships map to an empty RDF graph, there are a number of rows in each relationship to be an example. These strings will be mapped to the RDF triples using R2R ML triple maps. These triple maps are the mechanism for mapping RDB relationships into RDF.

Examples of mapping are present for only one row of each logical table. This is done to save space and to emphasize the rules themselves, not just their use.

Turtle syntax was used to describe triple maps, as well as the following notation:

@prefix rr:<http://www.w3.org/ns/r2rml#>

@prefix ex: http://example.com/Ch#

1. Basic concepts

1.1. Concept

SQL table Concept

CPK	Name
1	C
2	D
3	E
4	F

R2R ML Triple Map

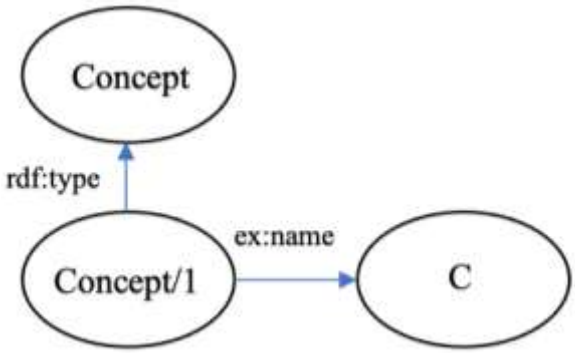
<#TriplesMap1>

rr:logicalTable [rr:tableName «Concept»];
r:subjectMap [
rr:template «http://example.com/Ch#/Concept/{CPK}»;
rr:class ex:Concept;
]
rr:predicateObjectMap [
rr:predicate ex:name;
rr:objectMap [rr:column «Name»];
]

RDF output example

<http://example.com/Ch#/Concept/1> rdf:type ex:Concept.
<http://example.com/Ch#/Concept/1>ex:name"C

RDF output example graph



1.2. Role

SQL table Role

RPK	Name	IsTransitive
56	R	No
67	S	No
89	T	No
34	Z	No
23	U	Yes

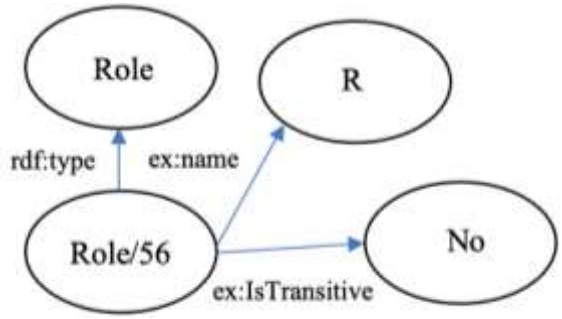
R2R ML Triple Map

<#TriplesMap2>
rr:logicalTable [rr:tableName «Role»];
r:subjectMap [
rr:template «http://example.com/Ch#/Role/{RPK}»;
rr:class ex:Role;
]
rr:predicateObjectMap [
rr:predicate ex:name;
rr:objectMap [rr:column «Name»];
]
rr:predicateObjectMap [
rr:predicate ex:IsTransitive;
rr:objectMap [rr:column «IsTransitive»];
]

RDF output example

<http://example.com/Ch#/Role/56> rdf:type ex:Role.
<http://example.com/Ch#/Role/56> ex:name "R".
<http://example.com/Ch#/Role/56> ex:IsTransitive "No".

RDF output example graph



1.3. CIndividual

SQL table CIndividual

CIPK	Name
100	abc
101	def
102	aaa
103	bbb
104	ccc
105	ddd

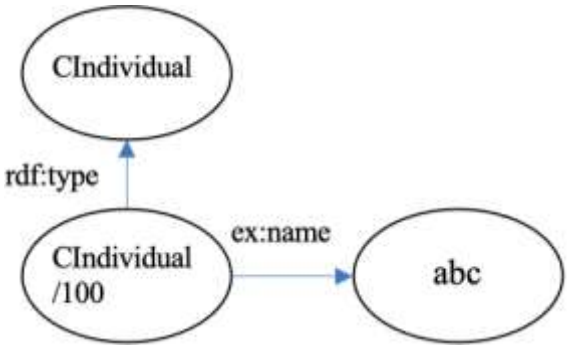
R2R ML Triple Map

<#TriplesMap3>
rr:logicalTable [rr:tableName «CIndividual»];
r:subjectMap [
rr:template «http://example.com/Ch#/CIndividual/{CIPK}»;
rr:class ex:CIndividual;
]
rr:predicateObjectMap [
rr:predicate ex:name;
rr:objectMap [rr:column «Name»];
]

RDF output example

<http://example.com/Ch#/CIndividual/100> rdf:type ex:CIndividual.
<http://example.com/Ch#/CIndividual/100> ex:name "abc".

RDF output example graph



1.4. RIndividual

SQL table RIndividual

RIPK
10
11
12
13
14
15

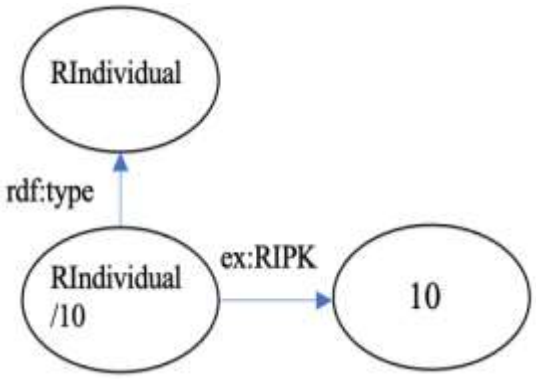
R2R ML Triple Map

```
<#TriplesMap4>
rr:logicalTable [ rr:tableName «RIndividual»];
r:subjectMap [
        rr:template
«http://example.com/Ch#/RIndividual/{RIPK}»;
        rr:class ex:RIndividual;
]
rr:predicateObjectMap [
        rr:predicate      ex:RIPK;
        rr:objectMap     [rr:column «RIPK»];
]
```

RDF output example

<http://example.com/Ch#/RIndividual/10> rdf:type ex:RIndividual.
<http://example.com/Ch#/RIndividual/10> ex:RIPK 10.

RDF output example graph



2. Relationship-bundles

2.1. Domain

SQL table Domain

CFK	RFK
1	56
2	67

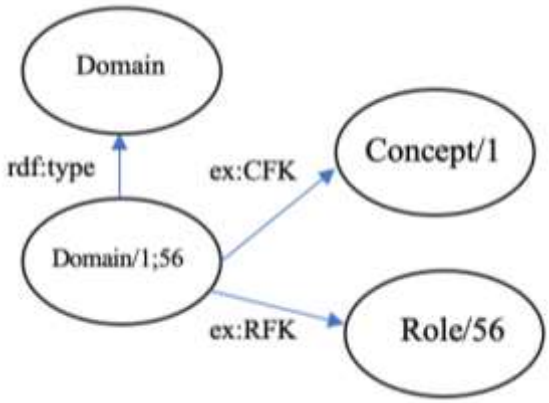
R2R ML Triple Map

```
<#TriplesMap5>
rr:logicalTable [ rr:tableName «Domain»];
r:subjectMap [
        rr:template
«http://example.com/Ch#//Domain/{CFK};{RFK
}»;
        rr:class ex:Domain;
]
rr:predicateObjectMap [
        rr:predicate      ex:CFK
        rr:objectMap     [
        a rr:RefObjectMap ;
rr:parentTriplesMap <#TriplesMap1>
        rr:joinCondition [
                rr:child «CFK»;
                rr:parent «CPK»;
]]]
rr:predicateObjectMap [
        rr:predicate      ex:RFK
        rr:objectMap     [
        a rr:RefObjectMap ;
rr:parentTriplesMap <#TriplesMap2>
        rr:joinCondition [
                rr:child «RFK»;
                rr:parent «RPK»;
]]]
```

RDF output example

<http://example.com/Ch#/Domain/1;56>rdf:type ex:Domain.
<http://example.com/Ch#/Domain/1;56>ex:CFK <http://example.com/Ch#/Concept/1>.
<http://example.com/Ch#/Domain/1;56>ex:RFK <http://example.com/Ch#/Role/56>.

RDF output example graph



2.2. Range

SQL table Range

CFK	RFK
2	56
1	67

R2R ML Triple Map

```
<#TriplesMap6>
rr:logicalTable [ rr:tableName «Range»];
r:subjectMap [
        rr:template
«http://example.com/Ch#/Range/{CFK};{RFK}»;
        rr:class ex:Range;
]
rr:predicateObjectMap [
        rr:predicate      ex:CFK
        rr:objectMap    [
        a rr:RefObjectMap ;
rr:parentTriplesMap <#TriplesMap1>
        rr:joinCondition [
                rr:child «CFK»;
                rr:parent «CPK»;
]]]
rr:predicateObjectMap [
        rr:predicate      ex:RFK
        rr:objectMap    [
        a rr:RefObjectMap ;
rr:parentTriplesMap <#TriplesMap2>
        rr:joinCondition [
                rr:child «RFK»;
                rr:parent «RPK»;
]]]
```

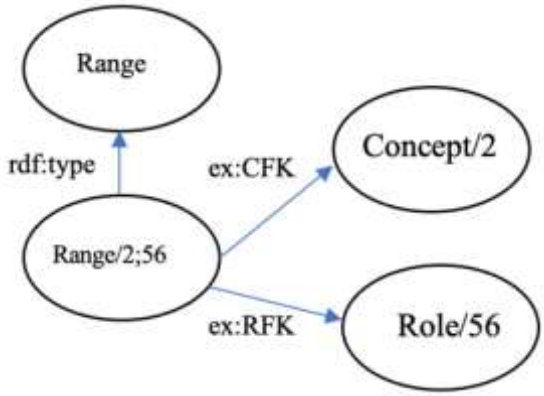
RDF output example

<http://example.com/Ch#/Range/2;67>rdf:type ex:Range.

<http://example.com/Ch#/Range/2;67>ex:CFK <http://example.com/Ch#/Concept/2>.

<http://example.com/Ch#/Range/2;67>ex:RFK <http://example.com/Ch#/Role/67>.

RDF output example graph



2.3. LinkCI

SQL table LinkCI

CFK	CIFK
1	100
1	101
2	102
2	103
2	104
2	101

R2R ML Triple Map

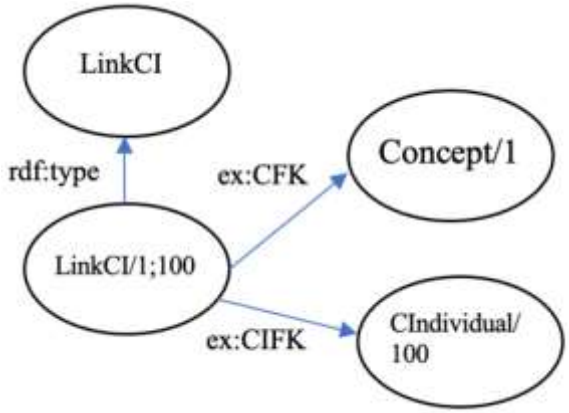
```
<#TriplesMap7>
rr:logicalTable [ rr:tableName «LinkCI»];
r:subjectMap [
        rr:template
«http://example.com/Ch#/LinkCI/{CFK};{CIFK}
»;
        rr:class ex:LinkCI;
        ]
rr:predicateObjectMap [
        rr:predicate      ex:CFK
        rr:objectMap    [
        a rr:RefObjectMap ;
rr:parentTriplesMap <#TriplesMap1>
        rr:joinCondition [
                rr:child «CFK»;
                rr:parent «CPK»;
]]]
rr:predicateObjectMap [
        rr:predicate      ex:CIFK
        rr:objectMap    [
```

a rr:RefObjectMap ;
rr:parentTriplesMap <#TriplesMap3>
rr:joinCondition [
rr:child «CIFK»;
rr:parent «CIPK»;
]]]

RDF output example

<http://example.com/Ch#/LinkCI/1;100>rdf:type ex:LinkCI.
<http://example.com/Ch#/LinkCI/1;100>ex:CFK <http://example.com/Ch#/Concept/1>.
<http://example.com/Ch#/LinkCI/1;100>ex:CIFK <http://example.com/Ch#/CIndividual/100>.

RDF output example graph



2.4. LinkRRI

SQL table LinkRRI

RFK	RIFK
56	10
56	11
67	12
67	13
67	14
56	15

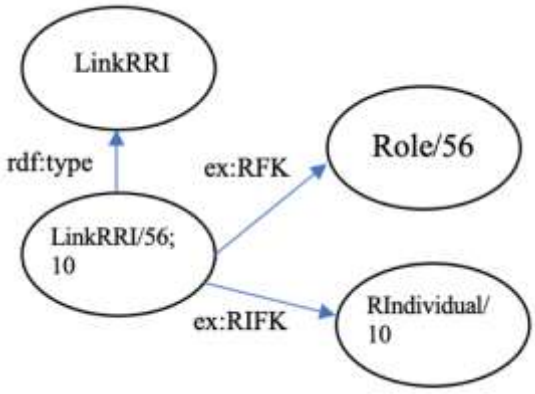
R2R ML Triple Map

<#TriplesMap8>
rr:logicalTable [rr:tableName «LinkRRI»];
r:subjectMap [
rr:template
«http://example.com/Ch#/LinkRRI/{RFK};{RIFK}»;
rr:class ex:LinkRRI;
]
rr:predicateObjectMap [
rr:predicate ex:RFK
rr:objectMap [
a rr:RefObjectMap ;
rr:parentTriplesMap <#TriplesMap2>
rr:joinCondition [
rr:child «RFK»;
rr:parent «RPK»;
]]]
rr:predicateObjectMap [
rr:predicate ex:RIFK
rr:objectMap [
a rr:RefObjectMap ;
rr:parentTriplesMap <#TriplesMap4>
rr:joinCondition [
rr:child «RIFK»;
rr:parent «RIPK»;
]]]

RDF output example

<http://example.com/Ch#/LinkRRI/56;10> rdf:type ex:LinkRRI.
<http://example.com/Ch#/LinkRRI/56;10> ex:RFK <http://example.com/Ch#/Role/56>.
<http://example.com/Ch#/LinkRRI/56;10> ex:RIFK <http://example.com/Ch#/RIndividual/10>.

RDF output example graph



2.5. Predecessor

SQL table Predecessor

CIFK	RIFK
100	10
101	11
100	15
102	12
103	13
104	14

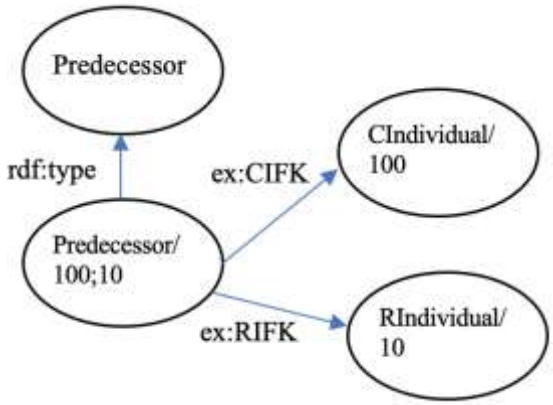
R2R ML Triple Map

```
<#TriplesMap9>
rr:logicalTable [ rr:tableName «Predecessor»];
r:subjectMap [
        rr:template
«http://example.com/Ch#/Predecessor/{CIFK};{R
IFK}»;
        rr:class ex:Predecessor;
]
rr:predicateObjectMap [
        rr:predicate      ex:CIFK
        rr:objectMap    [
        a rr:RefObjectMap ;
rr:parentTriplesMap <#TriplesMap3>
        rr:joinCondition [
                rr:child «CIFK»;
                rr:parent «CIPK»;
]]]
rr:predicateObjectMap [
        rr:predicate      ex:RIFK
        rr:objectMap    [
        a rr:RefObjectMap ;
rr:parentTriplesMap <#TriplesMap4>
        rr:joinCondition [
                rr:child «RIFK»;
                rr:parent «RIPK»;
]]]
```

RDF output example

<http://example.com/Ch#/Predecessor/100;10>
rdf:type ex:Predecessor.
<http://example.com/Ch#/Predecessor/100;10>ex
:CIFK
<http://example.com/Ch#/CIndividual/100>.
<http://example.com/Ch#/Predecessor/100;10>ex
:RIFK
<http://example.com/Ch#/RIndividual/10>.

RDF output example graph



2.6. Successor

SQL table Successor

CIFK	RIFK
102	10
103	11
104	15
100	12
101	13
101	14

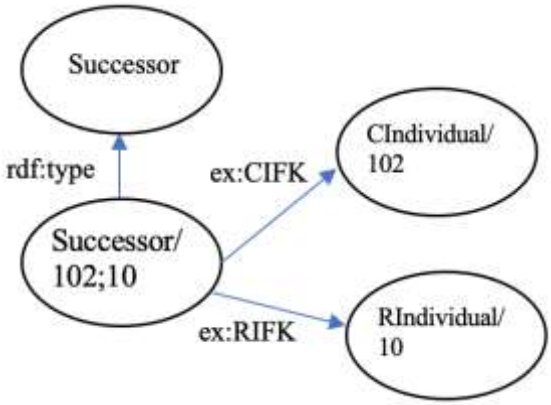
R2R ML Triple Map

```
<#TriplesMap10>
rr:logicalTable [ rr:tableName «Successor»];
r:subjectMap [
        rr:template
«http://example.com/Ch#/Successor/{CIFK};{RI
FK}»;
        rr:class ex:Successor;
]
rr:predicateObjectMap [
        rr:predicate      ex:CIFK
        rr:objectMap    [
        a rr:RefObjectMap ;
rr:parentTriplesMap <#TriplesMap3>
        rr:joinCondition [
                rr:child «CIFK»;
                rr:parent «CIPK»;
]]]
rr:predicateObjectMap [
        rr:predicate      ex:RIFK
        rr:objectMap    [
        a rr:RefObjectMap ;
rr:parentTriplesMap <#TriplesMap4>
        rr:joinCondition [
                rr:child «RIFK»;
                rr:parent «RIPK»;
]]]
```

RDF output example

<http://example.com/Ch#/Successor/102;10>
rdf:type ex:Successor.
<http://example.com/Ch#/Successor/102;10>
ex:CIFK
<http://example.com/Ch#/CIndividual/102>.
<http://example.com/Ch#/Successor /102;10>
ex:RIFK
<http://example.com/Ch#/RIndividual/10>.

RDF output example graph



3. Axioms

3.1. ConceptEquivalence

SQL table ConceptEquivalence

CForFK	CIsFK
3	4

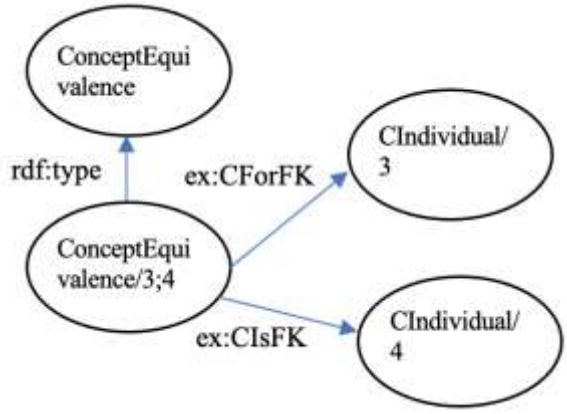
R2R ML Triple Map

```
<#TriplesMap11>
rr:logicalTable [ rr:tableName «ConceptEquivalence»];
r:subjectMap [
        rr:template «http://data.example.com/ConceptEquivalence/{CForFK};{CIsFK}»;
        rr:class ex:ConceptEquivalence;
]
rr:predicateObjectMap [
        rr:predicate    ex:CForFK
        rr:objectMap    [
        a rr:RefObjectMap ;
rr:parentTriplesMap <#TriplesMap1>
        rr:joinCondition [
                rr:child «CForFK»;
                rr:parent «CPK»;
]]]
rr:predicateObjectMap [
        rr:predicate    ex:CIsFK
        rr:objectMap    [
        a rr:RefObjectMap ;
rr:parentTriplesMap <#TriplesMap1>
        rr:joinCondition [
                rr:child «CIsFK»;
                rr:parent «CPK»;
]]]
```

RDF output example

<http://example.com/Ch#/ConceptEquivalence/3;4> rdf:type ex:ConceptEquivalence.
<http://example.com/Ch#/ConceptEquivalence/3;4>ex:CForFK <http://example.com/Ch#/Concept/3>.
<http://example.com/Ch#/ConceptEquivalence/3;4>ex:CIsFK <http://example.com/Ch#/Concept/4>

RDF output example graph



3.2. ConceptNesting

SQL table ConceptNesting

CInFK	COutFK
3	1

R2R ML Triple Map

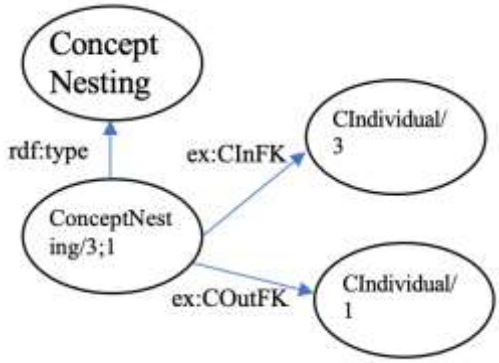
```
<#TriplesMap12>
rr:logicalTable [ rr:tableName «ConceptNesting»];
r:subjectMap [
        rr:template «http://data.example.com/ConceptNesting/{CInFK};{COutFK}»;
        rr:class ex:ConceptNesting;
]
rr:predicateObjectMap [
        rr:predicate    ex:CInFK
        rr:objectMap    [
        a rr:RefObjectMap ;
rr:parentTriplesMap <#TriplesMap1>
        rr:joinCondition [
                rr:child «CInFK»;
                rr:parent «CPK»;
]]]
rr:predicateObjectMap [
        rr:predicate    ex:COutFK
        rr:objectMap    [
        a rr:RefObjectMap ;
```

rr:parentTriplesMap <#TriplesMap1>
rr:joinCondition [
rr:child «COutFK»;
rr:parent «CPK»;
]]]

RDF output example

<http://example.com/Ch#/ConceptNesting/3;1> rdf:type ex:ConceptNesting.
<http://example.com/Ch#/ConceptNesting/3;1>ex :CInFK <http://example.com/Ch#/Concept/1>.
<http://example.com/Ch#/ConceptNesting/3;1>ex :COutFK <http://example.com/Ch#/Concept/1>

RDF output example graph



3.3. RoleNesting

SQL table RoleNesting

RInFK	ROutFK
89	56

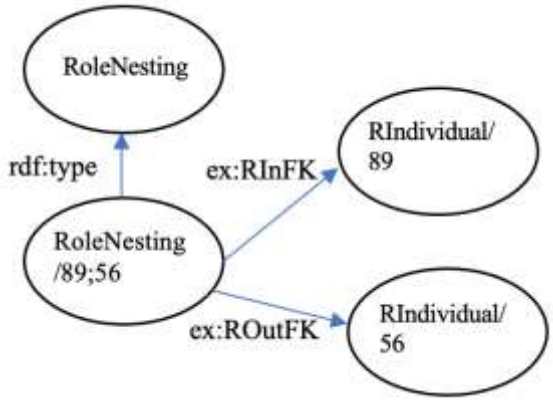
R2R ML Triple Map

<#TriplesMap13>
rr:logicalTable [rr:tableName «RoleNesting»];
r:subjectMap [
rr:template «http://example.com/Ch#/RoleNesting/{RInFK};{ROutFK}»;
rr:class ex:RoleNesting;
]
rr:predicateObjectMap [
rr:predicate ex:RInFK
rr:objectMap [
a rr:RefObjectMap ;
rr:parentTriplesMap <#TriplesMap2>
rr:joinCondition [
rr:child «RInFK»;
rr:parent «RPK»;
]]]
rr:predicateObjectMap [
rr:predicate ex:ROutFK
rr:objectMap [
a rr:RefObjectMap ;
rr:parentTriplesMap <#TriplesMap2>
rr:joinCondition [
rr:child «ROutFK»;
rr:parent «RPK»;
]]]

RDF output example

<http://example.com/Ch#/RoleNesting/89;56> rdf:type ex:RoleNesting.
<http://example.com/Ch#/RoleNesting/89;56> ex:RInFK <http://example.com/Ch#/RoleNesting/89>
<http://example.com/Ch#/RoleNesting/89;56> ex:ROutFK <http://example.com/Ch#/RoleNesting/56>

RDF output example graph



3.4. RoleEquivalence

SQL table RoleEquivalence

RForFK	RIsFK
89	34

R2R ML Triple Map

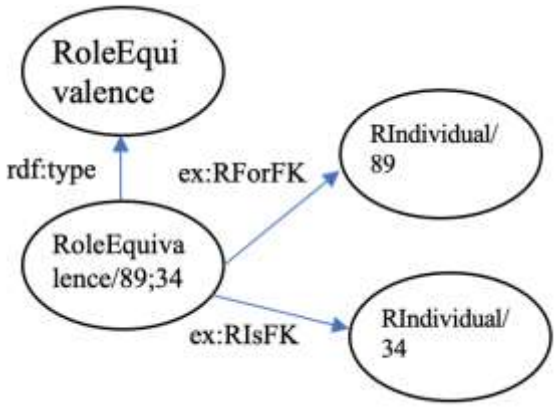
<#TriplesMap14>
rr:logicalTable [rr:tableName «RoleEquivalence»];
r:subjectMap [
rr:template «http://example.com/Ch#/RoleEquivalence/{RForFK};{RIsFK}»;
rr:class ex:RoleNesting;
]

```
rr:predicateObjectMap [
        rr:predicate      ex:RForFK
        rr:objectMap    [
        a rr:RefObjectMap ;
rr:parentTriplesMap <#TriplesMap2>
        rr:joinCondition [
                rr:child «RForFK»;
                rr:parent «RPK»;
]]]
rr:predicateObjectMap [
        rr:predicate      ex:RIsFK
        rr:objectMap    [
        a rr:RefObjectMap ;
rr:parentTriplesMap <#TriplesMap2>
        rr:joinCondition [
                rr:child «RIsFK»;
                rr:parent «RPK»;
]]]
```

RDF output example

<http://example.com/Ch#/RoleEquivalence/89;34> rdf:type ex:RoleEquivalence.
<http://example.com/Ch#/RoleEquivalence/89;34>ex:RForFK
<http://example.com/Ch#/RoleEquivalence/89>
<http://example.com/Ch#/RoleEquivalence/89;34>ex:RIsFK
<http://example.com/Ch#/RoleEquivalence/56>

RDF output example graph



3.5. Transitive role

SQL table Role

RPK	Name	IsTransitive
56	R	No
67	S	No
89	T	No
34	Z	No
23	U	Yes

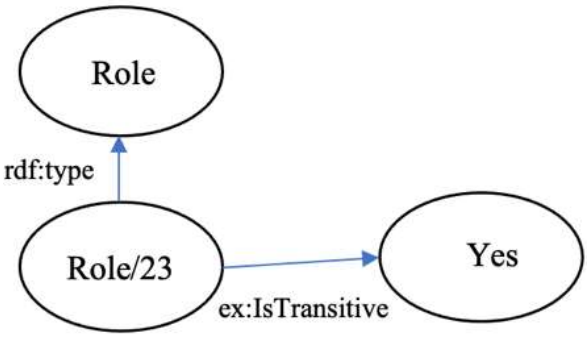
R2R ML Triple Map

```
<#TriplesMap15>
rr:logicalTable [ rr:tableName «Role»];
r:subjectMap [
        rr:template
«http://example.com/Ch#/Role/{RPK}»;
        rr:class ex:Role;
]
rr:predicateObjectMap [
        rr:predicate ex:IsTransitive;
        rr:objectMap [rr:column «IsTransitive»];
]
```

RDF output example

<http://example.com/Ch#/Role/23> rdf:type ex:Role.
<http://example.com/Ch#/Role/23> ex:IsTransitive "Yes".

RDF output example graph



3.6. CIEquivalence

SQL table CIEquivalence

CIForFK	CIIsFK
105	104

R2R ML Triple Map

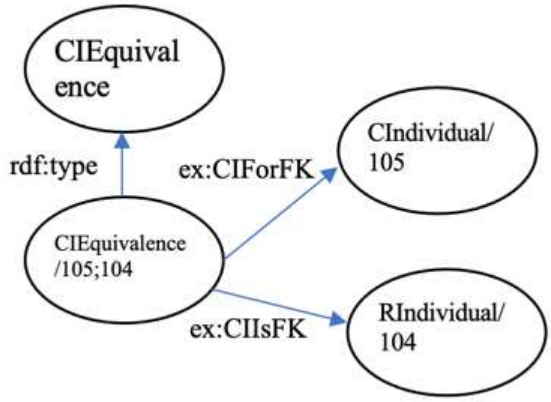
```
<#TriplesMap16>
rr:logicalTable [ rr:tableName «CIEquivalence»];
r:subjectMap [
        rr:template
«http://example.com/Ch#/CIEquivalence/{CIForFK};{CIIsFK}»;
        rr:class ex:RoleNesting;
]
rr:predicateObjectMap [
        rr:predicate      ex:CIForFK
        rr:objectMap    [
        a rr:RefObjectMap ;
```

rr:parentTriplesMap <#TriplesMap3>
rr:joinCondition [
rr:child «CIForFK»;
rr:parent «CIPK»;
]]]
rr:predicateObjectMap [
rr:predicate ex:CIIsFK
rr:objectMap [
a rr:RefObjectMap ;
rr:parentTriplesMap <#TriplesMap3>
rr:joinCondition [
rr:child «CIIsFK»;
rr:parent «CIPK»;
]]]

RDF output example

<http://example.com/Ch#/CIEquivalence/105;104 > rdf:type ex:CIEquivalence
<http://example.com/Ch#/CIEquivalence/105;104 >ex:CIForFK
<http://example.com/Ch#/CIndividual/105>.
<http://example.com/Ch#/CIEquivalence/105;104 >ex:CIIsFK
<http://example.com/Ch#/CIndividual/104>.

RDF output example graph



4. ALC syntax mapping

4.1. Concept

RA^2 term

$$C^{E}_{RM^2} = \pi_{CIndividual.Name}(\sigma_{Concept.Name='C'}(CIndividual \bowtie_{CIPK=CIFK} (LinkCI \bowtie_{CFK=CPK} Concept)))$$

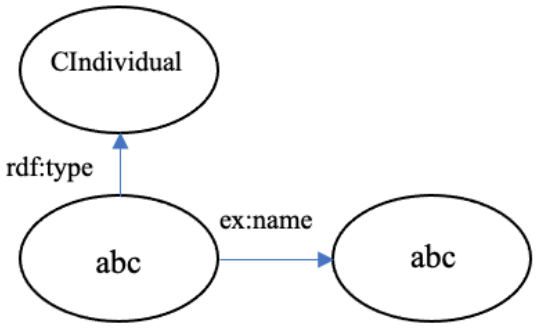
R2R ML Triple Map

<#TriplesMap17>
rr:logicalTable [rr:sqlQuery
“”” SELECT ci.Name
FROM CIndividual ci, Concept c, LinkCI lci,
WHERE ci.CIPK = lci.CIFK. AND lci.CFK = c.CPK AND c.Name = ‘C’ “”””];
r:subjectMap [
rr:template «http://example.com/Ch#/{Name}»;
rr:class ex:CIndividual;
]
rr:predicateObjectMap [
rr:predicate ex:name
rr:objectMap [rr:column: Name;]
]

RDF output example

<http://example.com/Ch#/CIndividual/abc> rdf:type ex:CIndividual.
<http://example.com/Ch#/CIndividual/abc> ex:name “abc”.

RDF output example graph



4.2. Role

RA^2 term

$$R^{E}_{RM^2} = \pi_{First,Second}(\rho_{CIdividual.Name.Second}(CIndividual \bowtie_{CIPK=CFPK} (Successor \bowtie_{RIFK=RIPK} (\rho_{CIndividual.Name.First}(CIndividual \bowtie_{CIPK=CIFK} (Predecessor \bowtie_{RIFK=RIPK} (\sigma_{Role.Name='R'}(RIndividual \bowtie_{RIPK=RIFK} (LinkRRI \bowtie_{RFK=RPK} Role)))))))))$$

R2R ML Triple Map

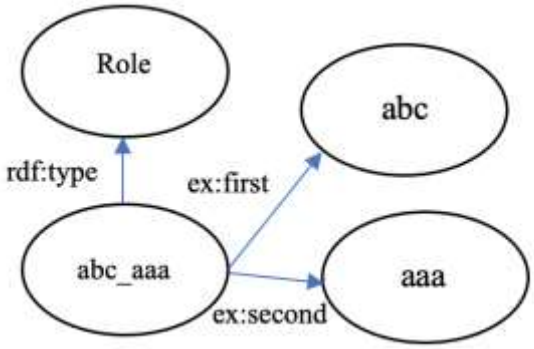
<#TriplesMap18>
rr:logicalTable [rr:sqlQuery
“”” SELECT first.Name AS First, second,Name AS Second
FROM Role r, RIndividual ri, LinkRRI lrri , Predecessor p, Successor s
CIndividual first, CIndividual.second

WHERE r.RPK = lrri.RFK AND lrri.RIFK = ri.RIPK AND ri.RIPK = p.RIFK AND
p.CIFK = first.CIPK AND ri.RIPK = s.RIFK AND s.CIFK = second. CIPK AND r.Name='R" """];
r:subjectMap [
rr:template
«http://example.com/Ch#/{First}_{Second}»;
rr:class ex:RIndividual;
]
rr:predicateObjectMap [
rr:predicate ex:first
rr:objectMap [rr:column: First;]
]
rr:predicateObjectMap [
rr:predicate ex:second
rr:objectMap [rr:column: Second;]
]

RDF output example

<http://example.com/Ch#/abc_aaa>rdf:type ex:Role;
<http://example.com/Ch#/abc_aaa>ex:first "abc"
<http://example.com/Ch#/abc_aaa>ex:second "aaa"

RDF output example graph



4.3. Concept negation

RA^2 term

$$(\neg C)^{E}_{RM^2} = \pi_{Name}(CIndividual) - C^{E}_{RM^2}$$

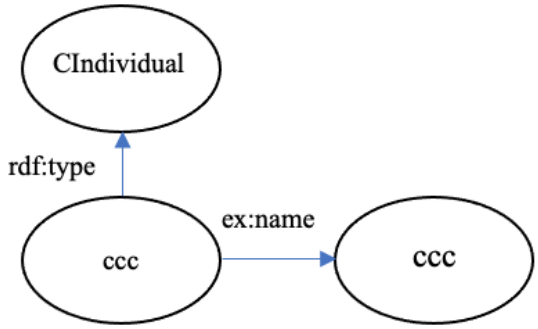
R2R ML Triple Map

<#TriplesMap19>
rr:logicalTable [rr:sqlQuery
""" SELECT ci.Name
FROM CIndividual ci
EXCEPT
SELECT ci.Name
FROM CIndividual ci, Concept c, LinkCI lci,
WHERE ci.CIPK = lci.CIFK. AND lci.CFK = c.CPK AND c.Name = 'C' """];
r:subjectMap [
rr:template
«http://example.com/Ch#/{Name}»;
rr:class ex:CIndividual;
]
rr:predicateObjectMap [
rr:predicate ex:name
rr:objectMap [rr:column: Name;]
]

RDF output example

<http://example.com/Ch#/CIndividual/ccc> rdf:type ex:CIndividual.
<http://example.com/Ch#/CIndividual/ccc> ex:name "ccc".

RDF output example graph



4.4. Concept union

RA^2 term

$$(C \sqcup D) = C^{E}_{RM^2} \cup D^{E}_{RM^2}$$

R2R ML Triple Map

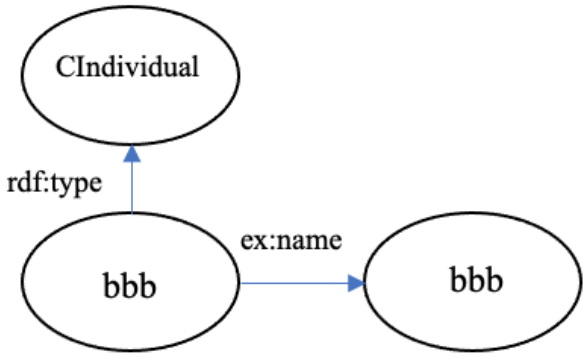
<#TriplesMap20>
rr:logicalTable [rr:sqlQuery
""" SELECT ci.Name
FROM CIndividual ci, Concept c, LinkCI lci,
WHERE ci.CIPK = lci.CIFK. AND lci.CFK = c.CPK AND c.Name = 'C'
UNION
SELECT ci.Name
FROM CIndividual ci, Concept c, LinkCI lci,
WHERE ci.CIPK = lci.CIFK. AND lci.CFK = c.CPK AND c.Name = 'D' """];
r:subjectMap [
rr:template

«http://example.com/Ch#/{Name}»;
rr:class ex:CIndividual;
]
rr:predicateObjectMap [
rr:predicate ex:name
rr:objectMap [rr:column: Name;]
]

RDF output example

<http://example.com/Ch#/CIndividual/bbb> rdf:type ex:CIndividual.
<http://example.com/Ch#/CIndividual/bbb> ex:name "bbb".

RDF output graph



4.5. Concept intersection

RA^2 term

$$(C \sqcap D) = C^{E}_{RM^2} \cap D^{E}_{RM^2}$$

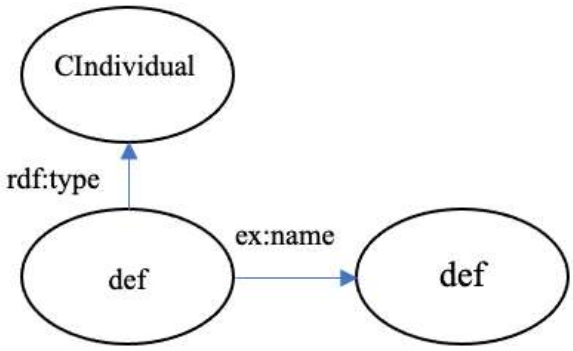
R2R ML Triple Map

<#TriplesMap21>
rr:logicalTable [rr:sqlQuery
""""
SELECT ci.Name
FROM CIndividual ci, Concept c, LinkCI lci,
WHERE ci.CIPK = lci.CIFK. AND lci.CFK = c.CPK AND c.Name = 'C'
INTERSECT
SELECT ci.Name
FROM CIndividual ci, Concept c, LinkCI lci,
WHERE ci.CIPK = lci.CIFK. AND lci.CFK = c.CPK AND c.Name = 'D'
""""];
r:subjectMap [
rr:template
«http://example.com/Ch#/{Name}»;
rr:class ex:CIndividual;
]
rr:predicateObjectMap [
rr:predicate ex:name
rr:objectMap [rr:column: Name;]
]

RDF output example

<http://example.com/Ch#/CIndividual/def> rdf:type ex:CIndividual.
<http://example.com/Ch#/CIndividual/def> ex:name "def".

RDF output example graph



The following notation is introduced:

- a table RE(First, Second) was get after mapping $R^{E}_{RM^2}$ (<#TriplesMap18>);
- a table CE(Name) was get after mapping $C^{E}_{RM^2}$(<#TriplesMap17>).

4.6. Existential quantification

RA^2 term

$$(\exists R.\, C)^{E}_{RM^2} = \pi_{First}(R^{E}_{RM^2} \bowtie_{Second=Name} C^{E}_{RM^2})$$

R2R ML Triple Map

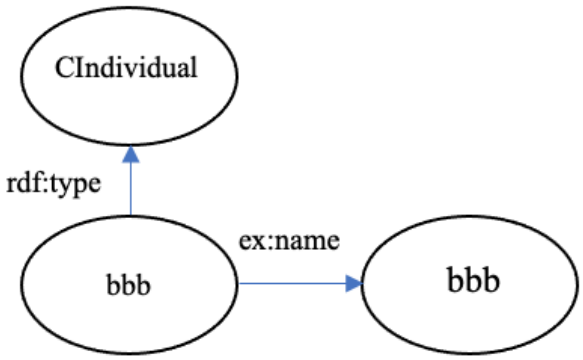
<#TriplesMap22>
rr:logicalTable [rr:sqlQuery
"""" SELECT RE.First
FROM RE, CE
WHERE RE.Second = CE.Name """"];
r:subjectMap [
rr:template
«http://example.com/Ch#/{First}»;
rr:class ex:CIndividual;
]
rr:predicateObjectMap [
rr:predicate ex:first

rr:objectMap [rr:column: First;]
]

RDF output example

<http://example.com/Ch#/CIndividual/bbb> rdf:type ex:CIndividual.
<http://example.com/Ch#/CIndividual/bbb> ex:name "bbb".

RDF output graph



4.7. Value restriction

RA^2 term

$$(\forall R.C)^{E}_{RM^2} = \pi_{First}R^{E}_{RM^2} - \pi_{First}(R^{E}_{RM^2} \cap (\pi_{First}R^{E}_{RM^2} \times (\pi_{Second}R^{E}_{RM^2} - \pi_{Name}C^{E}_{RM^2})))$$

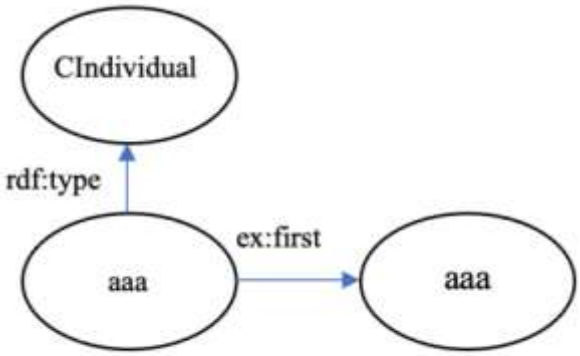
R2R ML Triple Map

```
<#TriplesMap23>
rr:logicalTable [ rr:sqlQuery
“””SELECT RE.First
FROM RE
EXCEPT
SELECT First
FROM (SELECT
        FROM RE
        INTERSECT
        (SELECT *
        FROM (SELECT RE.First
                FROM RE),
                (SELECT RE.Second
                FROM RE
                EXCEPT
                SELECT CE.Name
                FROM CE))) “””];
r:subjectMap [
rr:template «http://example.com/Ch#/{First}»;
rr:class ex:CIndividual;
]
rr:predicateObjectMap [
        rr:predicate    ex:first
        rr:objectMap    [ rr:column: First;]
]
```

RDF output example

<http://example.com/Ch#/CIndividual/aaa> rdf:type ex:CIndividual.
<http://example.com/Ch#/CIndividual/aaa> ex:first "aaa".

RDF output graph



5. ALC extensions

The mappings to only several number restrictions are shown in the paper. Mappings for the rest of extensions uses the recursive SQL.

5.1. Functional restrictions

RA^2 term

$$(\geq 2R)^{E}_{RM^2} = \pi_{First}(\sigma_{Second1 \neq Second2} (\rho_{Second/Second1}(R^{E}_{RM^2}) \bowtie_{First=First} (\rho_{Second/Second2}(R^{E}_{RM^2}))))$$

R2R ML Triple Map

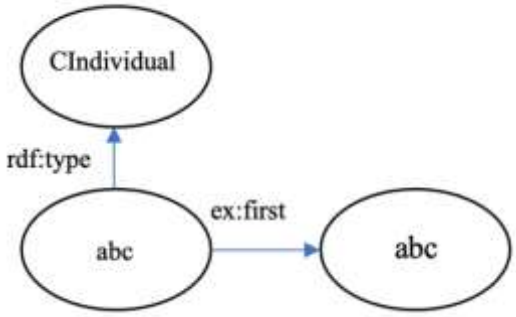
```
<#TriplesMap24>
rr:logicalTable [ rr:sqlQuery
“””SELECT one.First
FROM RE one, RE two
WHERE one.First = two.First
AND one.Second <> two.Second “””];
r:subjectMap [
rr:template «http://example.com/Ch#/{First}»;
```

```
rr:class ex:CIndividual;
]
rr:predicateObjectMap [
        rr:predicate      ex:first
        rr:objectMap      [ rr:column: First;]
]
```

RDF output example

<http://example.com/Ch#/CIndividual/abc> rdf:type ex:CIndividual.
<http://example.com/Ch#/CIndividual/abc> ex:name "abc".

RDF output graph



RA^2 term

$$(\leq 1R)^{E}_{RM^2} = \pi_{Name}(CIndividual) - (\geq 2R)^{E}_{RM^2}$$

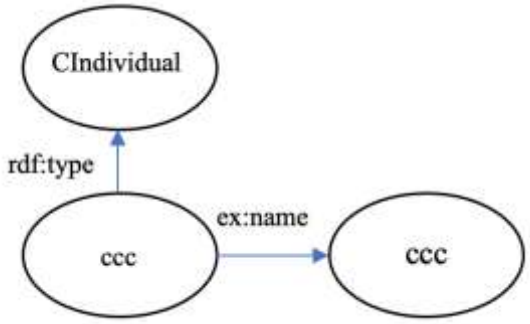
R2R ML Triple Map

```
<#TriplesMap25>
rr:logicalTable [ rr:sqlQuery
""""SELECT ci.Name
FROM CIndividual ci
EXCEPT
SELECT one.First
FROM RE one, RE two
        WHERE one.First = two.First
        AND one.Second <> two.Second """"];
r:subjectMap [
rr:template «http://example.com/Ch#/{Name}»;
rr:class ex:CIndividual;
]
rr:predicateObjectMap [
        rr:predicate      ex:name
        rr:objectMap      [ rr:column: Name;]
]
```

RDF output example

<http://example.com/Ch#/CIndividual/ccc> rdf:type ex:CIndividual.
<http://example.com/Ch#/CIndividual/ccc> ex:name "ccc".

RDF output graph



5.2. Several quality restrictions

RA^2 term

$$(\geq 2\mathrm{R}.\mathrm{C})^{\mathrm{E}}_{\mathrm{RM}^2} = \pi_{\mathrm{First}}(\sigma_{\mathrm{Second}_1 \neq \mathrm{Second}_2}(\rho_{\mathrm{Second}/\mathrm{Second}_1}(\mathrm{R}^{\mathrm{E}}_{\mathrm{RM}^2}) \bowtie_{\mathrm{Second=Name}}(\rho_{\mathrm{Second}/\mathrm{Second}_2}(\mathrm{R}^{\mathrm{E}}_{\mathrm{RM}^2} \bowtie_{\mathrm{Second=Name}} \mathrm{C}^{2}_{\mathrm{RM}^2}))))$$

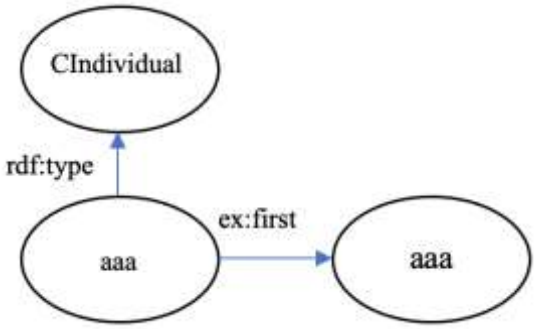
R2R ML Triple Map

```
<#TriplesMap26>
rr:logicalTable [ rr:sqlQuery
""""SELECT one.First
FROM
        (SELECT First, Second
        FROM RE, CE
        WHERE RE.Second = CE.Name) one,
        (SELECT First, Second
        FROM RE, CE
        WHERE RE.Second = CE.Name) two
WHERE one,First = two.First
AND one.Second <> two.Second""""];
r:subjectMap [
rr:template «http://example.com/Ch#/{First}»;
rr:class ex:CIndividual;
]
rr:predicateObjectMap [
        rr:predicate      ex:first
        rr:objectMap      [ rr:column: First;]
]
```

RDF output example

<http://example.com/Ch#/CIndividual/aaa> rdf:type ex:CIndividual.
<http://example.com/Ch#/CIndividual/aaa> ex:first "aaa".

RDF output graph



RA^2 term

$$(\leq 1R)^{E}_{RM^2} = \pi_{Name}(CIndividual) - (\geq 2R.\,\mathrm{C})^{E}_{RM^2}$$

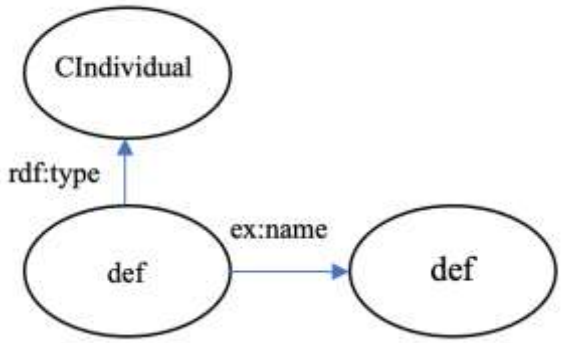
R2R ML Triple Map

<#TriplesMap27>
rr:logicalTable [rr:sqlQuery
“””SELECT ci.Name
FROM CIndividual ci
EXCEPT
SELECT one.First
FROM
(SELECT First, Second
FROM RE, CE
WHERE RE.Second = CE.Name) one,
(SELECT First, Second
FROM RE, CE
WHERE RE.Second = CE.Name) two
WHERE one,First = two.First
AND one.Second <> two.Second “””];
r:subjectMap [
rr:template «http://example.com/Ch#/{Name}»;
rr:class ex:CIndividual;
]
rr:predicateObjectMap [
rr:predicate ex:name
rr:objectMap [rr:column: Name;]
]

RDF output example

<http://example.com/Ch#/CIndividual/def> rdf:type ex:CIndividual.
<http://example.com/Ch#/CIndividual/def> ex:name "def".

RDF output graph



6. Role restrictions

Despite the number of role restrictions only role inverse mapping into RDF is considered in the paper. As known [19] the OWL 2 is based on the DL SROIQ. This logic includes only role inverse through all the role restrictions amount. So, the mapping for other role restrictions is out of scope of this research.

6.1. Role inverse

RA^2 term

$$(\mathrm{R}^{-})^{\mathrm{E}}_{\mathrm{RM}^2} = (\rho_{\mathrm{R(Second,First)}}(\mathrm{R}^{\mathrm{E}}_{\mathrm{RM}^2}))$$

R2R ML Triple Map

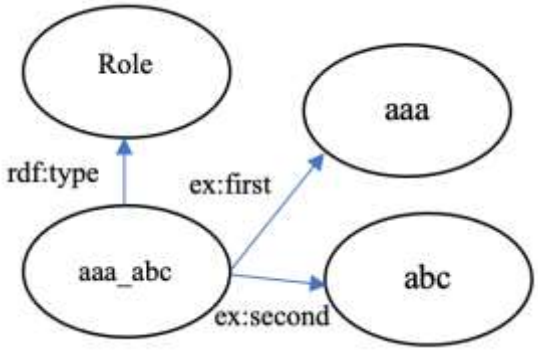
<#TriplesMap28>
rr:logicalTable [rr:sqlQuery
“””SELECT second.Name AS First, first.Name AS Second
FROM Role r, RIndividual ri, LinkRRI lrri, Predecessor p, Successor s
CIndividual first, CIndividual.second
WHERE r.RPK = lrri.RFK AND lrri.RIFK = ri.RIPK AND ri.RIPK = p.RIFK AND p.CIFK = first.CIPK AND ri.RIPK = s.RIFK AND s.CIFK = second. CIPK AND r.Name=’R” “””];
r:subjectMap [

```
rr:template «http://example.com/Ch#/{Name}»;
rr:class ex:Role;
]
rr:predicateObjectMap [
        rr:predicate      ex:name
        rr:objectMap      [ rr:column: Name;]
]
```

RDF output example

<http://example.com/Ch#/aaa_abc>rdf:type ex:Role;
<http://example.com/Ch#/aaa_abc>ex:first "aaa"
<http://example.com/Ch#/aaa_abc>ex:second "abc"

RDF output example graph



Conclusions

The article outlines a method how to check mappings between the descriptive logic and the binary relational data model using mappings into RDF. The task is set. The description of the theoretical aspects is presented. The publication provides mappings for the binary relational data model into an RDF triples using the R2R ML language. The paper also outlines the rules for converting the DL-to-RM^2 mapping formulas into RDF.

The issue of converting a number of role constructors into RDF remains open.

References

1. Chystiakova, I. Ontology-oriented data integration on the Semantic Web. *Problems in Programming.* 2014. N 2-3. P. 188–196.
2. Reznichenko, V. and Chystiakova, I. Mapping of the Description Logics ALC into the Binary Relational Data Structure. *Problems in Programming.* 2015. N 4. P. 13–30.
3. Reznichenko, V. and Chystiakova, I. Integration of the family of extended description logics with relational data model. *Problems in Programming*. (2016). N 2-3. P. 38–47.
4. Chystiakova, I. Integration of the description logics with extensions into relational data model. *Problems in Programming*. 2016. N 4. P. 58–65.
5. Chystiakova, I. Integration of the Descriptive Logic Axiomatics with the Relational Data Model. *Problems in programming*. 2017. N 1. P. 51–58.
6. Reznichenko, V. and Chystiakova, I. Binary Relational Data Model. *Problems in Programming*. 2017. N 2 (4). P. 96–105.
7. Chystiakova I.S. (2018). Mapping of the Relational Algebra into Descriptive Logic. *Problems in Programming*. 2017. N 2–3. P. 214 – 225.
8. Andon P. and Reznichenko V. and Chystiakova I. Mapping of Description Logic to the Relational Data Model. Cybernetics and Systems Analysis. 2017. N 53 (6). P. 963–978.
9. Chystiakova I.S. Implementation of mappings between the description logic and the binary relational data model on the RDF level. *Problems in programming.* 2020. N 4. P. 41 – 54.
10. Heath I.J. Unacceptable file operations in a relational data base. SIGFIDET '71. 1971. P. 19–33.
11. OWL 2 Web Ontology Language. Mapping to RDF Graphs (Second Edition). [Online] December 2012. Available from: https://www.w3.org/TR/owl2-mapping-to-rdf/#Translation_of_Axioms_without_Annotations. [Accessed: 20 February 2021].
12. R2RML: RDB to RDF Mapping Language. [Online] September 2012. Available from: https://www.w3.org/TR/r2rml/. (last access 20 February 2021).
13. Caroll J.J. Matching RDF Graphs I. Horrocks and J. Hendler (Eds.): ISWC. 2002. LNCS 2342. P. 5–15.
14. Berners-Lee, T. Relational Databases on the Semantic Web [Online] September 1998. Available from: https://www.w3.org/DesignIssues/RDB-RDF.html. (last access 20 February 2021).
15. W3C Workshop on RDF Access to Relational Databases URL:

https://www.w3.org/2007/03/RdfRDB/ (last access 20 February 2021)
16. RDB2RDF Tutorial (R2RML and Direct Mapping) at ISWC 2013 URL: https://www.slideshare.net/juansequeda/rdb2-rdf-tutorial-iswc2013 (last access 20 February 2021)
17. Codd E.F. Extending the database relational model to capture more meaning. ACM Transactions on Database Systems (TODS). 1979. Vol. 4. Issue 4. P. 397–434.
18. Barker R. Case* method: entity relationship modelling. Addison-Wesley. 1990. P. 240.
19. The description logic handbook: theory, implementation, and applications. Baader F., Calvanese D., McGuinness D., Nardi D., Patel-Schneider P. (Eds.) Cambridge University Press. 2003. 555 p.

Література

1. Чистякова І.С. Онтолого-ориентированная интеграция данных в семантическом вебе. *Проблеми програмування*. 2014. № 2-3. С. 188–196.
2. Резниченко В.А., Чистякова И.С. Отображение дескриптивной логики ALC в бинарную реляционную структуру данных. Проблеми програмування. 2015. № 4. С. 13–30.
3. Резниченко В.А., Чистякова И.С. Интеграция семейства расширенных дескриптивных логик с реляционной моделью данных. *Проблеми програмування*. 2016. № 2-3. С. 38–47.
4. Чистякова И.С. Интеграция логик с операциями над ролями с реляционной моделью данных. *Проблеми програмування*. 2016. № 4. С. 58–65.
5. Чистякова И.С. Интеграция аксиоматики дескриптивных логик с реляционной моделью данных. *Проблеми програмування*. 2017. № 1. С. 51 – 58.
6. Резниченко В.А., Чистякова И.С. Бинарная реляционная модель данных. *Проблеми програмування.* 2017. № 2. С. 96 – 105.
7. Чистякова И.С. Отображение реляционной алгебры в дескриптивную логику. *Проблеми програмування.* 2018. № 2–3. С. 214 – 225.
8. Andon P. and Reznichenko V. and Chystiakova I. Mapping of Description Logic to the Relational Data Model. Cybernetics and Systems Analysis. 2017. N 53 (6). P. 963–978.
9. Chystiakova I.S. Implementation of mappings between the description logic and the binary relational data model on the RDF level. *Problems in programming.* 2020. N 4. P. 41 – 54.
10. Heath I.J. Unacceptable file operations in a relational data base. SIGFIDET '71. 1971. P. 19–33.
11. OWL 2 Web Ontology Language. Mapping to RDF Graphs (Second Edition). [Online] December 2012. Available from: https://www.w3.org/TR/owl2-mapping-to-rdf/#Translation_of_Axioms_without_Annotations. [Accessed: 20 February 2021].
12. R2RML: RDB to RDF Mapping Language. [Online] September 2012. Available from: https://www.w3.org/TR/r2rml/. (last access 20 February 2021).
13. Caroll J.J. Matching RDF Graphs I. Horrocks and J. Hendler (Eds.): ISWC. 2002. LNCS 2342. P. 5–15.
14. Berners-Lee, T. Relational Databases on the Semantic Web [Online] September 1998. Available from: https://www.w3.org/DesignIssues/RDB-RDF.html. (last access 20 February 2021).
15. W3C Workshop on RDF Access to Relational Databases URL: https://www.w3.org/2007/03/RdfRDB/ (last access 20 February 2021)
16. RDB2RDF Tutorial (R2RML and Direct Mapping) at ISWC 2013 URL: https://www.slideshare.net/juansequeda/rdb2-rdf-tutorial-iswc2013 (last access 20 February 2021)
17. Codd E.F. Extending the database relational model to capture more meaning. ACM Transactions on Database Systems (TODS). 1979. Vol. 4. Issue 4. P. 397–434.
18. Barker R. Case* method: entity relationship modelling. Addison-Wesley. 1990. P. 240.
19. The description logic handbook: theory, implementation, and applications. Baader F., Calvanese D., McGuinness D., Nardi D., Patel-Schneider P. (Eds.) Cambridge University Press. 2003. 555 p.

Received 04.02.2021

About the author:

Inna Chystiakova,
junior researcher at the Institute of software systems of NASU.
The number of publications in Ukrainian journals is 11.
The number of publications in foreign journals is 1.
Hirsh index is 5.
https://orcid.org/0000-0001-7946-3611.

Affiliation:

Institute of software systems of NASU
03187, Kyiv,
pr. Glushkova, 40, build 5.
Tel.: +38(066)8477784.
E-mail: inna_islyamova@ukr.net.