Reasoning About Consistency of Relational Knowledge Bases

Tadeusz Pankowski
Institute of Control and Information Engineering
Poznań University of Technology
Poznań, Poland
Email: tadeusz.pankowski@put.poznan.pl

***Abstract*—We study a translation of relational database into an extended DL knowledge base that is both sound and complete with respect to semantics preservation. The issue is of special importance while creating Web of Data, where relational databases are exposed in the Web to be searched, integrated and exchanged without loss of information and semantics. Our approach considers all fundamental requirements of DL/OWL knowledge bases such as OWA and rejecting the UNA. We propose a system of axioms for representing database integrity constraints, which guarantees information and semantics preserving data-to-knowledge exchange.**

***Keywords*—knowledge bases; data integration; integrity constraints; data exchange; ontology-based data management.**

I. INTRODUCTION

In this paper, we study the problem of transforming a relational databases to *Description Logic* (DL) knowledge bases. We propose a method to realize such a transformation and study its correctness. The translation is correct if in the target knowledge base both information and semantics are preserved. The problem of representing a relational database in a DL knowledge base has gained in importance in recent years, mainly due to the emergence of the Semantic Web. It is expected that the Semantic Web enables web-wide integration of data coming from various sources, also referred to as Web of Data. The Web Of Data can be formally perceived as a giant DL knowledge base (or DL ontology) $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ is a set of axioms modeling the intensional knowledge (the TBox axioms), and $\mathcal{A}$ is a set of assertions forming the extensional knowledge (the ABox assertions) [1]. It is expected that in such a knowledge base the underlying source repositories are represented as precisely as possible and that distributed execution of queries over these repositories is possible allowing for so called *ontology-based data access* [2]. Most of the data exposed in the Web are originally stored in relational databases, and have to be mapped or translated to *Resource Description Framework* (RDF)/*Ontology Web Language* (OWL) representations. Sometimes we face an inverse problem – Web data are to be stored in a relational database, so the relational structures and constraints have to be discovered. Thus, the similarities and differences between databases and knowledge bases has been an important and attractive field of research since many years [3], [4], [5], [6].

Related work. Some recent results of representing relational databases in the Semantic Web are surveyed in [7]. There are two approaches to integrating relational databases with the Semantic Web: (1) *direct mapping* [8], [9] – consists in translating the database contents and its schema directly to representations in Semantic Web languages, i.e., in RDF and OWL, in result an ontology is created from the database; (2) *mapping to ontology* [2] – assumes the existence of a shared domain ontology and a language for mapping database schema to this ontology. Then, the *extensional* layer of the database is represented by RDF graph in a form of a set of semantic triples $(subject, predicate, object)$ [10]. The *intensional* layer is captured by means of axioms specified in OWL [11] or RDFS [10]. The direct mapping of relational database schema to OWL DL is the subject of the current draft of the W3C Group [12]. A formal basis for RDF(S) and OWL provides Description Logic (DL), that is a family of knowledge representation languages [13] being usually a decidable subset of first-order logic. A relationship between relational databases and DL knowledge bases are studied in [6]. There are three main differences between databases and knowledge bases making the translation between them difficult [5], [4]: (a) databases are based on the *Closed Word Assumption* (CWA) while knowledge bases follow usually the *Open World Assumption* (OWA); (b) databases make the *Unique Name Assumption* (UNA) while knowledge bases usually do not accept it; (c) integrity constraints in databases are interpreted as checks while in knowledge bases all rules are interpreted as deductive rules. It turns out that incorporating integrity constraints into knowledge bases is the most challenging issue. To deal with the problem, the following three approaches have been proposed: (a) introduction of epistemic operators **K** and **A** into the knowledge base, to capture the semantics of integrity constraints by restricting interpretation of some formulas to the "state" of the knowledge base, i.e., to *"what the knowledge base knows"* [14]; (b) knowledge bases are restricted only to those, which accept UNA [15], (c) defining an *extended DL knowledge base* (EKB), where the set $\mathcal{T}$ of TBox axioms is divided into *standard* TBox axioms, $\mathcal{S}$, and *integrity constraint* TBox axioms, $\mathcal{C}$; semantics of $\mathcal{S}$ and $\mathcal{C}$ are defined in different ways.

Contribution. In this paper we will use the notion of EKB to represent a relational database in DL. We define a *data-to-knowledge exchange* (*dk-exchange*) system that defines translation of relational database schema, its integrity constraints and instances into an extended DL knowledge base. The resulting extended DL knowledge base is referred to as a *relational knowledge base* (RKB). This approach provides an automatic translation and a relational schema is mechanically transformed into TBox axioms. However, for the approach to be meaningful, some assumptions must be made about the way

 ISBN: 978-1-61208-283-7

in which the underlying relational database has been created. To this order we assume that the relational database under consideration implements a conceptual model developed using ER (*Entity-Relationship*) or its extension EER (*Expanded Entity-Relationship*) approach [16], [17]. Additionally, we assume that all relationships in ER schema are binary and one-to-many. It means that any many-to-many relationship is replaced with two one-to-many relationships. Such the replacement is always possible and commonly applied to form UML class diagrams [18].

In Figure 1, the EER diagram describes students, courses and exams taken by students [9]. In particular, a student is a specialization of a person. Every entity set in Figure 1 (b), i.e., Person, Student, Exam, and Course, has a key (identifier). Other attributes will be specified below in SQL definition.

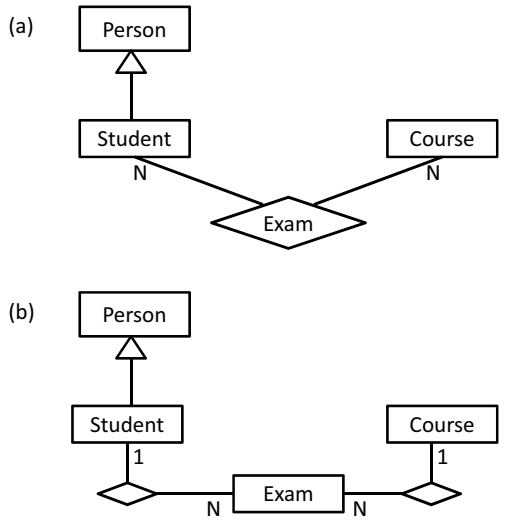

Fig. 1. EER diagrams of a university domain. (a) Exam is a many-to-many relationship, (b) Exam is a member in two one-to-many relationships [9].

A specification of the EER diagram in Figure 1 (b) using SQL notation is given in Figure 2.

```
create table Person(
PId int primary key,
Name char(20));

create table Course(
CId char(5) primary key);

create table Student(
SId int primary key foreign key references Person(PId),
Faculty char(20) not null);

create table Exam(
EId int primary key,
ESId int foreign key references Student(SId),
Course char(5) foreign key references Course(CId),
Grade char(1));
```

Fig. 2. SQL specification of the EER diagram in Figure 1 (b) [9].

We assume that a relational database is created with the following assumptions: (1) The ER methodology is followed, and all relationships are binary and one-to-many. Any primary key and any foreign key consists of exactly one column. (2) A single and partial specialization (inheritance) between entity sets is allowed. (3) The final database schema is defined in SQL with the following integrity constraints: unique, not-null, primary key, and foreign key. Some combinations of primary keys and foreign keys will be referred to as *inheritance constraints* (see the definition of SId column in Student table).

Analysis of relational database schema suggests a possible translation of this schema into an ontology. Some natural semantic interpretations of schema elements and integrity constraints and their translation into extended DL knowledge base (or an OWL ontology), are:

- Both relation names and attribute type names become atomic concepts (class names).
- Attributes become atomic roles (object property names) connecting individuals in classes corresponding to relation names, with individuals in classes corresponding to attribute type names.
- Integrity constraints become TBox axioms, divided properly between standard TBox axioms and TBox ic-axioms.

The main novelty of this paper is that we propose a system of standard TBox axioms and TBox ic-axioms, such that the final dk-exchange system is both sound and complete with respect to semantics preservation. We show how consistency of the resulting relational knowledge base can be verified.

The structure of the paper is as follows. In Section II, we review some definitions concerning relational databases (RDB). In Section III, we discuss a relational knowledge base and translation of RDB to RKB. Dk-exchange system and its fundamental property concerning semantics preservation is discussed in Section IV. Checking consistency of RKB is discussed in Section V. Section VI concludes the paper.

II. RELATIONAL DATABASES

A. Schemas and instances

A (relational) *database schema* (*db-schema*) is a pair $(\mathbf{R}, \mathbf{IC})$, where $\mathbf{R} = \{R_1, \ldots, R_n\}$ is a *relational schema* consisting of a set of *relation symbols*, and $\mathbf{IC}$ is a set of *integrity constraints* over $\mathbf{R}$. Each *relation symbol* $R \in \mathbf{R}$ has a *type*, which is a nonempty finite set U of *attributes*, denoted $att(R) = U$. Without loss of generality, we can assume that types of relation symbols are pairwise disjoint (this can be achieved, for example, by prefixing attribute names with relation symbols), $arity(R)$ denotes the cardinality of $att(R)$.

Let $\mathbf{Const}$ be a countable infinite set of *constants*, and NULL be a reserved symbol not in $\mathbf{Const}$.

Definition 2.1: An *instance* I of a relational schema $\mathbf{R}$ is a finite multiset of *facts* (or *ground atoms*) of the form $R(A_1 : c_1, \ldots, A_m : c_m)$, where $R \in \mathbf{R}$, $att(R) = \{A_1, \ldots, A_m\}$, and $c_i \in \mathbf{Const} \cup \{\mathtt{NULL}\}$, $1 \leq i \leq m$. By $Inst(\mathbf{R})$ is denoted the set of all instances of $\mathbf{R}$. The set of all values from $\mathbf{Const} \cup \{\mathtt{NULL}\}$ occurring in I is referred to as the *active domain* of I and is denoted $adom(I)$. □

An instance of R corresponds to the notion of Codd table [4]. Since an instance is a multiset, thus the concept of duplicates

within a table is captured (as in SQL databases). Because the NULL value can occur in instances, the three-valued logic is used in SQL databases, where besides TRUE and FALSE, the UNKNOWN logical value is also needed [4].

Definition 2.2: Let $\mathcal{DB} = (\mathbf{R}, \mathbf{IC}, I)$ be a (relational) *database* with a db-schema $(\mathbf{R}, \mathbf{IC})$ and an instance I of $\mathbf{R}$. $\mathcal{DB}$ is consistent, if I satisfies (is a model of) all integrity constraints, denoted $I \models \mathbf{IC}$. Otherwise we say that $\mathcal{DB}$ is inconsistent. □

B. Integrity constraints

Integrity constraints in databases play a dual role. On the one hand, they describe all possible worlds, and, on the other hand, they describe the allowed states of the database [4]. Integrity constraints can be used in data reasoning tasks, such as checking the correctness of database data, as well as in schema reasoning tasks, such as computing query containment (or subsumption).

We assume that $\mathbf{IC} = Unique \cup NotNull \cup PKey \cup FKey \cup Inherit$, where: $Unique$, $NotNull$, $PKey$, $FKey$, and $Inherit$ are sets of, respectively, *unique*, *not-null*, *primary key*, *foreign key*, and *inheritance* integrity constraints.

Let $\mathbf{R}$ be a relational schema, $R \in \mathbf{R}$ be a relation symbol of type $att(R) = \{A_1, \ldots, A_m\}$, and $A_k \in att(R)$.

1) **Unique integrity constraint:** A *unique integrity* constraint is an expression of the form $unique(R, A_k)$. An instance I of $\mathbf{R}$ is consistent with $unique(R, A_k)$, iff for every i, $0 \leq i \leq m$, I satisfies the formula

$$R(t_1) \wedge R(t_2) \wedge t_1.A_k = t_2.A_k \wedge t_1.A_i \neq \texttt{NULL} \wedge t_2.A_i \neq \texttt{NULL} \Rightarrow t_1.A_i = t_2.A_i. \quad (1)$$

Note that the occurrence of NULLs affects the interpretation of the satisfaction of $unique(R, A)$. An instance I with NULLs is consistent with $unique(R, A)$, if there is such a replacement of NULLs with constants (every occurrence of NULL can be replaced with different constant) that the result instance is consistent with $unique(R, A)$.

Example 2.3: Let $R(A, B)$ denote a relation symbol R of type $\{A, B\}$ and $unique(R, A)$ be a unique integrity constraint. Let I_1, I_2 (Figure 3) be instances of R. Then, I_1 is consistent with $unique(R, A)$, while I_2 is not.

I_1 :

A	B
a	b
a	b
a	NULL
NULL	c
NULL	NULL

I_2 :

A	B
a	b
a	c
NULL	b
NULL	c
NULL	NULL

Fig. 3. Consistent, I_1, and inconsistent, I_2, instances of $R(A, B)$ with respect to $unique(R, A)$

2) **Not-null integrity constraint:** A *not-null* integrity constraint is an expression of the form $notnull(R, A_k)$. An instance I of $\mathbf{R}$ is consistent with $notnull(R, A_k)$, if for any fact $R(t) \in I$, $t.A_k$ is a constant, i.e., iff I satisfies the formula

$$R(t) \Rightarrow t.A_k \neq \texttt{NULL}.$$

3) **Primary key integrity constraint:** A *primary key integrity constraint* is an expression of the form $pkey(R, A_k)$. An instance I of $\mathbf{R}$ is consistent with $pkey(R, A_k)$ iff it is consistent with $unique(R, A_k)$, and $notnull(R, A_k)$.

4) **Foreign key integrity constraint:** Let $R, R' \in \mathbf{R}$, $A \in att(R)$, and $A' \in att(R')$. A *foreign key integrity constraint* is an expression of the form $fkey(R, A, R', A')$. An instance I of $\mathbf{R}$ is consistent with $fkey(R, A, R', A')$ iff I satisfies $unique(R', A')$, and I satisfies the following existence constraint

$$R(t) \wedge t.A \neq \texttt{NULL} \Rightarrow \exists t'.(R'(t') \wedge t'.A' = t.A). \quad (2)$$

5) **Inheritance constraint:** A pair $(pkey(R, A), fkey(R, A, R', A'))$ is referred to as an *inheritance constraint*. It specifies that some entities in a class $C_{R'}$ corresponding to the table R', can be *specialized* in a class C_R corresponding to the table R (or that each entity in C_R *inherits* from, exactly one, entity in $C_{R'}$). An instance I of $\mathbf{R}$ is consistent with $(pkey(R, A), fkey(R, A, R', A'))$ iff it is consistent with $pkey(R, A)$ and $fkey(R, A, R', A')$.

For example, (see Figure 1), the inheritance constraint

$$(fkey(Student, SId), fkey(Student, SId, Person, PId))$$

specifies that a student is uniquely identified by SId, and simultaneously SId refers to a person identified by PId. It means, that description of a person is contained in part in the $Person$ table, and in part in the $Student$ table (we say that $Person$ objects are *specialized* as $Student$ objects). Then, the composition (s, p) of a tuple s describing a student and a corresponding tuple p describing a person (such that $s.SId = p.PId$) fully describes a person under consideration.

III. Relational knowledge bases

To carry out reasoning procedures about properties of relational databases (such as preservation of semantics during data exchange) it is useful to represent the relational database by means of a knowledge base. However, a traditional DL knowledge base understood as a pair $(\mathcal{T}, \mathcal{A})$ is unable to model integrity constraints [6]. The reason is two-fold: firstly, axioms in $\mathcal{T}$ are interpreted under the standard first-order semantics and are treated as deductive rules and not as checks, and secondly, the UNA (*Unique Name Assumption*) is not accepted in general in DL knowledge bases, it means that two different individual names can denote the same individual.

A. Syntax of relational knowledge bases

In this section, we define a *relational knowledge base* (RKB) that is a DL knowledge base adequately representing a relational database. RKB is based on the concept of EKB [6]. We propose and discuss a system of TBox axioms, which properly represents a relational database defined in the previous section.

Definition 3.1: A *relational knowledge base* is a tuple $\mathcal{R} = (\mathcal{N}, \mathcal{S}, \mathcal{C}, \mathcal{A})$, where:

1) $\mathcal{N}$ is the *vocabulary* (or *signature*) of $\mathcal{R}$, consisting of a set $\mathcal{N}_{\mathrm{Ind}}$ of *individual names*, a set $\mathcal{N}_{\mathrm{Cl}}$ of *class*

names (or *atomic concepts*), a set $\mathcal{N}_{\mathrm{OP}}$ of *object property names* (or *atomic roles*).

2) $\mathcal{S}$ is a finite set of *standard* TBox axioms, which are treated as deductive rules and can infer new assertions (see Table I).
3) $\mathcal{C}$ is a finite set of *integrity constraint* TBox axioms. These axioms act as checks and are used to check whether a set of assertions implied by $\mathcal{A}$ and $\mathcal{S}$, satisfies conditions specified by $\mathcal{C}$ (see Table II). Axioms in $\mathcal{C}$ cannot imply new assertions.
4) $\mathcal{A}$ is a set of ABox *assertions*, which consists of factual knowledge about universe of discourse, i.e., properties of individual objects.

Creating vocabulary. Let $(\mathbf{R}, \mathbf{IC})$ be a db-schema, $\mathbf{Const}$ be a countably infinite set of constants, and Δ_{Var} be a countable infinite set of *labeled nulls* disjoint from the set of constants; labeled nulls, denoted $X, X_1, X_2, ...$, are used as "fresh" Skolem terms, which are placeholders for unknown values, and can thus be seen as *variables* [19]. Then the vocabulary $\mathcal{N} = \mathcal{N}_{\mathrm{Cl}} \cup \mathcal{N}_{\mathrm{OP}} \cup \mathcal{N}_{\mathrm{Ind}}$, is created as follows:

1) Distinguished class names Tuple, and Val, are inserted into $\mathcal{N}_{\mathrm{Cl}}$. These names are independent of a concrete db-schema: Tuple is a class of individual names called *tuple names* (or *tuples*), and Val is a class of individual names called *attribute value names* (or *attribute values*).
2) For each relational symbol $R \in \mathbf{R}$, there is a class name $C_R \in \mathcal{N}_{\mathrm{Cl}}$; this class is a subclass of Tuple, $C_R \sqsubseteq \mathsf{Tuple}$.
3) For each attribute $A \in att(R)$, $R \in \mathbf{R}$, there is a class name $C_A \in \mathcal{N}_{\mathrm{Cl}}$ being a subclass of Val ($C_A \sqsubseteq \mathsf{Val}$), and an object property name $P_A \in \mathcal{N}_{\mathrm{OP}}$; the object property P_A connects a tuple in C_R with an attribute value in C_A.
4) A set $\mathcal{N}_{\mathrm{Ind}}$ of individual names consists of the union of $\mathbf{Const}$ and Δ_{Var}. We assume that the UNA does not hold.

Creating standard TBox axioms. The set $\mathcal{S}$ of standard TBox axioms is given in Table I. Since each axiom in $\mathcal{S}$ is a deductive rule then the assertion on the right hand side of any FOL rule is inserted into the set $\mathcal{A} \cup \mathcal{S}$, i.e., into the closure of $\mathcal{A}$ with respect to $\mathcal{S}$.

TABLE I. STANDARD TBOX AXIOMS OF RELATIONAL KNOWLEDGE BASE

	Constraints of relational db	DL	FOL
S1	$R \in \mathbf{R}$	$C_R \sqsubseteq \mathsf{Tuple}$	$C_R(x) \Rightarrow \mathsf{Tuple}(x)$
S2	$A \in att(R), R \in \mathbf{R}$	$C_A \sqsubseteq \mathsf{Val}$	$C_A(v) \Rightarrow \mathsf{Val}(v)$
S3	range of P_A	$\exists P_A^- \sqsubseteq C_A$	$P_A(x,v) \Rightarrow C_A(v)$
S4	domain of P_A	$\exists P_A \sqsubseteq C_R$	$P_A(x,v) \Rightarrow C_R(x)$
S5	$unique(R, A)$ unique constraint	$(\mathtt{func}\ P_A^-)$	$P_A(x_1,v_1) \wedge P_A(x_2,v_2) \wedge v_1 = v_2 \Rightarrow x_1 = x_2$
S6	$(pkey(R, A), fkey(R, A, R', A'))$ inheritance constraint	$P_A \sqsubseteq P_{A'}$	$P_A(x,v) \Rightarrow P_{A'}(x,v)$

$(S1)$ and $(S2)$ belong to translation of facts that $R \in \mathbf{R}$ and $A \in att(R)$; they say that appropriate tuple names and attribute value names are to be inserted into, respectively, classes Tuple and Val. $(S3)$ and $(S4)$ belong to translation of the fact that $A \in att(R)$, where: $(S3)$ defines the range of the object property P_A corresponding to an attribute A of the relational symbol R, and $(S4)$ defines the domain of this object property. $(S5)$ is the result of translation of a unique constraint $unique(R, A)$, and enforces equality between x_1 and x_2, if hold all $P_A(x_1, v_1)$, $P_A(x_2, v_2)$, and $v_1 = v_2$ (A in R has the key property for not null values). $(S6)$ results of the translation of an inheritance constraint of the form $\{pkey(R, A), fkey(R, A, R', A')\}$, and says that extension of P_A is to be inserted into the extension of $P_{A'}$.

Creating integrity constraint TBox axioms. The set $\mathcal{C}$ of TBox ic-axioms is given in Table II. Note that ic-axioms are checks, so we expect that the value of such an axiom is either TRUE or FALSE.

TABLE II. INTEGRITY CONSTRAINT TBOX AXIOMS OF RELATIONAL KNOWLEDGE BASE

	Constraints of relational db	DL	FOL
C1	disjointness	$\mathsf{Tuple} \sqsubseteq \neg\mathsf{Val}$	$\mathsf{Tuple}(x) \Rightarrow \neg\mathsf{Val}(x)$
C2	$A \in att(R), R \in \mathbf{R}$ functional property	$(\mathtt{func}\ P_A)$	$P_A(x_1,v_1) \wedge P_A(x_2,v_2) \wedge x_1 = x_2 \Rightarrow v_1 = v_2$
C3	$notnull(R, A)$ not-null property	$C_R \sqsubseteq \exists P_A$	$C_R(x) \Rightarrow \exists v.P_A(x,v)$
C4	$fkey(R, A, R', A')$ foreign key	$\exists P_A^- \sqsubseteq \exists P_{A'}^-$	$P_A(x_1,v) \Rightarrow \exists x_2.P_{A'}(x_2,v)$
C5	$(pkey(R, A), fkey(R, A, R', A'))$ inheritance constraint	$C_R \sqsubseteq C_{R'}$	$C_R(x) \Rightarrow C_{R'}(x)$

$(C1)$ tests disjointness of Tuple and Val. $(C2)$ belongs to translation of $A \in att(R)$ and checks if P_A has the functional property. $(C3)$ is the result of translation of $notnull(R, A)$ constraint and tests if any tuple name in C_R is connected via P_A with some value name. $(C4)$ is the result of translation of a foreign key constraint, $fkey(R, A, R', A')$, and tests the inclusion of the set of value names in the "second column" of P_A in the set value names in the "second column" of $P_{A'}$. $(C5)$ belongs to the result of the translation of $\{pkey(R, A), fkey(R, A, R', A')\}$, and tests the inclusion of C_R in $C_{R'}$.

Creating ABox assertions ABox assertions are expressions of the form: $C(a), P(a_1, a_2), a_1 = a_2$, where $C \in \mathcal{N}_{\mathrm{Cl}}$, $P \in \mathcal{N}_{\mathrm{OP}}$, and $a, a_1, a_2 \in \mathcal{N}_{\mathrm{Ind}}$. Translation of an instance I of $\mathbf{R}$ can be performed using Algorithm 1

Algorithm 1 Creating ABox assertions

Input: Instance I of $\mathbf{R}$, and an empty ABox $\mathcal{A}$.
Output: ABox assertions in $\mathcal{A}$ representing I.
for each $R(t) \in I$
  $U_{R,t} := \{A \in att(R) \mid t.A \neq \mathtt{NULL}\}$
  $X :=$ a fresh labeled null in Δ_{Var}
  if $U_{R,t} = \emptyset$ **then**
    $\mathcal{A} := \mathcal{A} \cup C_R(X)$
  else
   for each $A \in U_{R,t}$
    $\mathcal{A} := \mathcal{A} \cup P_A(X, t.A)$
end

B. Semantics of relational knowledge bases

The semantics of $\mathcal{R}$ is defined in the standard way [13] by means of the interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, where $\Delta^{\mathcal{I}}$ is a non-empty set of individuals, and $\cdot^{\mathcal{I}}$ is an interpretation function

assigning to every vocabulary element (i.e., to every individual name, class name or property name) elements in $\Delta^{\mathcal{I}}$, subsets of $\Delta^{\mathcal{I}}$, or binary relations over $\Delta^{\mathcal{I}}$, respectively. Semantics of complex expressions is defined inductively in a set-theoretical manner based on semantics of simpler expressions.

The difference between standard TBox axioms in $\mathcal{S}$ and TBox ic-axioms in $\mathcal{C}$, appears mainly in definition of consistency [6]. A relational knowledge base $\mathcal{R} = (\mathcal{N}, \mathcal{S}, \mathcal{C}, \mathcal{A})$ is *consistent*, if

1) $\mathcal{A} \cup \mathcal{S}$ has a model, i.e., there is at least one interpretation $\mathcal{I}$ in which all formulas belonging to $\mathcal{A} \cup \mathcal{S}$ are satisfied, written $\mathcal{I} \models \mathcal{A} \cup \mathcal{S}$; and
2) any minimal Herbrand model of $\mathcal{A} \cup \mathcal{S}$ is also a model of $\mathcal{C}$, i.e., the entailment $\mathcal{A} \cup \mathcal{S} \models_{mHm} \mathcal{C}$ holds.

Intuitively, (1) says that there is no any contradiction between assertions in $\mathcal{A}$ and the general knowledge in $\mathcal{S}$. In (2), a Herbrand model of $\mathcal{A} \cup \mathcal{S}$ contains in fact *all the knowledge base knows*. This model is a finite set of assertions since $\mathcal{A}$ is finite (see Algorithm 1), and axioms in $\mathcal{S}$ are existential-free and without functional symbols (see Table I).

IV. Data-to-knowledge exchange systems

A *data-to-knowledge exchange* (*dk-exchange system*) systems we define as a mapping $\mathcal{M} = (\tau, \Sigma)$, from relational databases to relational knowledge bases, such that for each db-schema $(\mathbf{R}, \mathbf{IC})$ and every instance I of $\mathbf{R}$,

$$\mathcal{M}(\mathbf{R}, \mathbf{IC}, I) = (\tau(\mathbf{R}, \mathbf{IC}), \Sigma(I)) = (\mathcal{N}, \mathcal{S}, \mathcal{C}, \mathcal{A}),$$

where $\tau(\mathbf{R}, \mathbf{IC}) = (\mathcal{N}, \mathcal{S}, \mathcal{C})$, and $\Sigma(I) = \mathcal{A}$.

One of the most challenging issues in dk-exchange is to show that the semantics of the source data is not lost by the transformation into a knowledge base. The preservation of semantics of a dk-exchange system $\mathcal{M} = (\tau, \Sigma)$ can be understood in two ways:

1. *Soundness of dk-exchange systems with respect to semantics preservation*. $\mathcal{M} = (\tau, \Sigma)$ is *sound* w.r.t. *semantics preservation* if every consistent database $(\mathbf{R}, \mathbf{IC}, I)$ is transformed into a consistent relational knowledge base $(\mathcal{N}, \mathcal{S}, \mathcal{C}, \mathcal{A})$, i.e.,

$$I \models \mathbf{IC} \wedge \tau(\mathbf{R}, \mathbf{IC}) = (\mathcal{N}, \mathcal{S}, \mathcal{C}) \wedge \Sigma(I) = \mathcal{A} \Rightarrow \mathcal{A} \cup \mathcal{S} \models_{mHm} \mathcal{C}.$$

2. *Completeness of dk-exchange systems with respect to semantics preservation*. $\mathcal{M} = (\tau, \Sigma)$ is *complete* w.r.t. *semantics preservation* if every inconsistent database $(\mathbf{R}, \mathbf{IC}, I)$ is transformed into an inconsistent relational knowledge base $(\mathcal{N}, \mathcal{S}, \mathcal{C}, \mathcal{A})$, i.e.,

$$I \not\models \mathbf{IC} \wedge \tau(\mathbf{R}, \mathbf{IC}) = (\mathcal{N}, \mathcal{S}, \mathcal{C}) \wedge \Sigma(I) = \mathcal{A} \Rightarrow \mathcal{A} \cup \mathcal{S} \not\models_{mHm} \mathcal{C}.$$

We can show that a dk-exchange system $\mathcal{M} = (\tau, \Sigma)$, where: (1) τ creates a vocabulary $\mathcal{N}$, a set of standard TBox axioms $\mathcal{S}$, a TBox ic-axioms $\mathcal{C}$; and (2) Σ creates ABox $\mathcal{A}$, in the way described in Section III-A, is both sound and complete w.r.t. semantics preservation ([9]).

Let I be an instance of $\mathbf{R}$ (satisfying or not integrity constraints in $\mathbf{IC}$). Let $\mathcal{M}(\mathbf{R}, \mathbf{IC}, I) = (\mathcal{N}, \mathcal{S}, \mathcal{C}, \mathcal{A})$. Let us denote $\mathcal{S}_0 = \{S1, S2, S3\}$, and $\mathcal{C}_0 = \{C1\}$. Then the following properties can be shown:

1) $\mathcal{M}$ such that $\mathcal{S} = \mathcal{S}_0 \cup \{S5\}$ and $\mathcal{C} = \mathcal{C}_0 \cup \{C2\}$ is sound and complete w.r.t. unique integrity constraint preservation.
2) $\mathcal{M}$ such that $\mathcal{S} = \mathcal{S}_0 \cup \{S4\}$ and $\mathcal{C} = \mathcal{C}_0 \cup \{C3\}$ is sound and complete w.r.t. not-null integrity constraint preservation.
3) $\mathcal{M}$ such that $\mathcal{S} = \mathcal{S}_0 \cup \{S4, S5\}$ and $\mathcal{C} = \mathcal{C}_0 \cup \{C2, C3\}$ is sound and complete w.r.t. primary key integrity constraint preservation.
4) $\mathcal{M}$ such that $\mathcal{S} = \mathcal{S}_0 \cup \{S5\}$ and $\mathcal{C} = \mathcal{C}_0 \cup \{C2, C4\}$ is sound and complete w.r.t. foreign key integrity constraint preservation.
5) $\mathcal{M}$ such that $\mathcal{S} = \mathcal{S}_0 \cup \{S4, S5, S6\}$ and $\mathcal{C} = \mathcal{C}_0 \cup \{C2, C3, C4, C5\}$ is sound and complete w.r.t. inheritance integrity constraint preservation.

V. Checking consistency of relational knowledge bases

Now, we will consider the problem of verifying whether a given relational knowledge base is consistent. The problem is significant if data from many databases is transformed into a repository in a form of DL assertions, and we are interested if this repository poses a "relational nature", i.e., whether satisfies a set of standard- and ic-axioms being result of some dk-exchange. We will use the *chase procedure* that was developed as a method for investigating data dependencies [4] and data exchange [19]. This will be done in two steps: (1) We chase a minimal Herbrand model $\mathcal{H}$ of $\mathcal{A} \cup \mathcal{S}$. Since $\mathcal{S}$ is existential-free (thus it is also *weakly acyclic*), and with no function symbols – we can obtain a finite solution in a polynomial time. (2) We will test satisfaction of $\mathcal{C}$ in $\mathcal{H}$.

The chase producing $\mathcal{H}$ from $\mathcal{A}$ is defined by Algorithm 2. By h we denote a *homomorphism* that maps constants to themselves and variables into the set $\mathbf{Const} \cup \Delta_{\mathrm{Var}}$.

Algorithm 2 Chasing a minimal Herbrand model of $\mathcal{A} \cup \mathcal{S}$

Input: $\mathcal{A}$ – a set of assertions;
$\mathcal{S}$ – a set of standard TBox axioms.
Output: $\mathcal{H}$ – a minimal Herbrand model of $\mathcal{A} \cup \mathcal{S}$.
(1) $\mathcal{H}_0 := \mathcal{A}$; $i := 0$;
(2) **for each** $\xi := (\varphi(\mathbf{x}, \mathbf{v}) \Rightarrow \psi(\mathbf{x}, \mathbf{v})) \in \mathcal{S}$
(3) **while** $\mathcal{H}_i \not\models \xi$ **do**
(4) **for each** homomorphism h over $(\mathbf{x}, \mathbf{v})$
(5) **if** $\mathcal{H}_i \models_h \varphi(\mathbf{x}, \mathbf{v})$
(6) **do** $\mathcal{H}_i \xrightarrow{\xi, h} \mathcal{H}_{i+1}$; $i := i + 1$;
end
end
end
(7) $\mathcal{H} := \mathcal{H}_{i+1}$

Chase steps – comments to Algorithm 2

1) $\mathcal{H}_0$ is initiated with $\mathcal{A}$ – line (1).
2) Let $\xi \in \mathcal{S}$ be an axiom of the form $\varphi(\mathbf{x}, \mathbf{v}) \Rightarrow \psi(\mathbf{x}, \mathbf{v})$ – line (2).
3) If ξ is not satisfied by $\mathcal{H}_i$, then there is a homomorphism h over $(\mathbf{x}, \mathbf{v})$ mapping variables in $\mathbf{x}$ into Δ_{Var} and variables in $\mathbf{v}$ into $\mathbf{Const}$, such that

 ISBN: 978-1-61208-283-7

$\mathcal{H}_i \models h(\varphi(\mathbf{x}, \mathbf{v}))$ and $\mathcal{H}_i \not\models h(\psi(\mathbf{x}, \mathbf{v}))$. Then, we say that ξ is *applicable* to $\mathcal{H}_i$ with h – lines (3–5).

4) The new state of $\mathcal{H}$ is equal to $\mathcal{H}_i \cup h(\psi(\mathbf{x}, \mathbf{v}))$, and the result of *applying* ξ *to* $\mathcal{H}_i$ *with* h is then denoted as $\mathcal{H}_i \xrightarrow{\xi,h} \mathcal{H}_{i+1}$ – line (6).

5) The final result is denoted by $\mathcal{H}$ – line (7).

The dk-chasing described by Algorithm 2 produces $\mathcal{A} \cup \mathcal{S}$ as a closure of $\mathcal{A}$ w.r.t. $\mathcal{S}$, and is denoted by $\mathcal{A} \xrightarrow{\mathcal{S}} \mathcal{H}$, meaning that all applicable axioms in $\mathcal{S}$ have been applied to $\mathcal{H}$.

Example 5.1: Given a set of assertions

$$\mathcal{A} = \{PId(X_1, 1), Name(X_1, john), PId(X_2, 2), Name(X_2, ann), PId(X_3, 3), Name(X_3, eva), Course(X_4, db), SId(X_5, 1), Faculty(X_5, math), SId(X_6, 3), EId(X_7, 1), ESId(X_7, 3), Course(X_7, db), Grade(X_7, A)\},$$

verify if it is a representation of an instance of a university database in Figure 1. For this database schema we have

$$\mathbf{R} = \{P(erson), C(ourse), S(tudent), E(xam)\},$$
$$\mathbf{IC} = \{pkey(P, PId), pkey(C, CId), pkey(S, SId), pkey(E, EId), notnull(S, Faculty), fkey(S, SId, P, PId), fkey(E, ESId, S, SId), fkey(E, Course, C, CId)\},$$

and implicitly in $\mathbf{IC}$ are integrity constraints implied by primary keys and foreign keys (i.e., *unique*s and *not-null*s).

In result of translation, we have $\tau(\mathbf{R}, \mathbf{IC}) = (\mathcal{N}, \mathcal{S}, \mathcal{C})$, where $\mathcal{S}$ is a set of standard TBox axioms of the form given in Table I, and $\mathcal{C}$ is a set of TBox ic-axioms of the form given in Table II. We will use names, respectively, R and A to denote a class C_R and an object property P_A. In particular, $(S4)$, $(S5)$ and $(S6)$ generate the following axioms:

(a) $PId(x, v) \Rightarrow P(x)$,

(b) $PId(x_1, v_1) \wedge PId(x_2, v_2) \wedge v_1 = v_2 \Rightarrow x_1 = x_2$,

(c) $SId(x, v) \Rightarrow PId(x, v)$.

Then, in $\mathcal{H}$, being the result of the chase procedure, the following set of derived assertions is included:

$$\mathcal{A}' = \{P(X_1), P(X_2), P(X_3), C(X_4), S(X_5), S(X_6), E(X_7), PId(X_5, 1), PId(X_6, 3), X_1 = X_5, X_3 = X_6)\}.$$

Now, it is easily seen that all ic-axioms generated by $(C4)$ and $(C5)$ are satisfied in $\mathcal{H}$. However, the ic-axiom

(d) $S(x) \Rightarrow \exists v.Faculty(x, v)$,

corresponding to $notnull(S, Faculty) \in \mathbf{IC}$, is not satisfied. Thus, $(\mathcal{N}, \mathcal{S}, \mathcal{C}, \mathcal{A})$ do not correspond to any consistent instance of the database in Figure 1. □

VI. Conclusion

In this paper, we studied the problem of representing relational databases in extended DL knowledge bases in a way that satisfies both information and semantics preservation. We mainly focused on semantics preservation meaning that a consistent database is mapped into consistent knowledge base (the soundness), and any inconsistent database is mapped into inconsistent knowledge base (the completeness). The importance of the problem is met while creating Web of Data integrating data coming from many distributed and heterogeneous sources [20], [21], [22]. Then, the knowledge base (ontological) layer is used to reason about the data, the metadata and queries in data integration and data exchange procedures [1].

References

[1] T. Pankowski, "Data Exchange Between Relational Knowledge Bases In The Web Of Data," in 15th International Conference on Enterprise Information Systems (ICEIS), July 04 - 07, 2013 - Angers, France. INSTICC Press, 2013, pp. 1–6.

[2] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati, "Linking data to ontologies," in Journal on Data Semantics X. Springer-Verlag, 2008, pp. 133–173.

[3] J. D. Ullman, Principles of Database and Knowledge-Base Systems, Vol. I/II. Computer Science Press, 1988/1989.

[4] S. Abiteboul, R. Hull, and V. Vianu, Foundations of Databases. Reading, Massachusetts: Addison-Wesley, 1995.

[5] R. Reiter, "Towards a logical reconstruction of relational database theory," in On Conceptual Modelling. Perspectives from Artificial Intelligence, Databases, and programming Languages, 1982, pp. 191–233.

[6] B. Motik, I. Horrocks, and U. Sattler, "Bridging the gap between OWL and relational databases," Journal of Web Semantics, vol. 7, no. 2, 2009, pp. 74–89.

[7] J. Sequeda, S. H. Tirmizi, Ó. Corcho, and D. P. Miranker, "Survey of directly mapping SQL databases to the Semantic Web," Knowledge Eng. Review, vol. 26, no. 4, 2011, pp. 445–486.

[8] J. Sequeda, M. Arenas, and D. P. Miranker, "On Directly Mapping Relational Databases to RDF and OWL (Extended Version)," CoRR, vol. abs/1202.3667, 2012, pp. 1–17.

[9] T. Pankowski, "Semantics Preservation In Data-To-Knowledge Exchange From Relational Databases To Knowledge Bases." (submitted), 2013.

[10] RDF Vocabulary Description Language 1.0: RDF Schema, 2004, www.w3.org/TR/rdf-schema/ .

[11] OWL 2 Web Ontology Language. Structural Specification and Functional-Style Syntax, 2009, www.w3.org/TR/owl-syntax.

[12] M. Arenas, A. Bertails, E. Prud'hommeaux, and J. Sequeda, "A Direct Mapping of Relational Data to RDF," 2012, http://www.w3.org/TR/rdb-direct-mapping.

[13] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, Eds., The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, 2003.

[14] F. M. Donini, M. Lenzerini, D. Nardi, W. Nutt, and A. Schaerf, "An epistemic operator for description logics," Artif. Intell., vol. 100, no. 1-2, 1998, pp. 225–274.

[15] D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati, "Data complexity of query answering in description logics," Artif. Intell., vol. 195, 2013, pp. 335–360.

[16] P. P. Chen, "The entity-relationship model - toward a unified view of data," ACM Transactions on Database Systtems, vol. 1, no. 1, 1976, pp. 9–36.

[17] R. Elmasri and S. B. Navathe, Fundamentals of Database Systems. Redwood City: The Benjamin/Cummings, 1994.

[18] Unified Modeling Language. Resource Page, 2005, www.uml.org/.

[19] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa, "Data exchange: semantics and query answering," Theor. Comput. Sci, vol. 336, no. 1, 2005, pp. 89–124.

[20] T. Pankowski, "Using Data-to-Knowledge Exchange for Transforming Relational Databases to Knowledge Bases," in RuleML 2012, LNCS 7438. Springer, 2012, pp. 256–263.

[21] G. Brzykcy, J. Bartoszek, and T. Pankowski, "Schema Mappings and Agents' Actions in P2P Data Integration System," Journal of Universal Computer Science, vol. 14, no. 7, 2008, pp. 1048–1060.

[22] T. Pankowski, "Semantics preservation in schema mappings within data exchange systems," in Knowledge Engineering, Machine Learning and Lattice Computing with Applications, Revised Selected Papers, KES 2012, LNAI 7828. Springer, 2013, pp. 88–97.

 ISBN: 978-1-61208-283-7