Mapping Relational Database Constraints to SHACL

Ratan Bahadur Thapa and Martin Giese(✉)

Department of Informatics, University of Oslo, Oslo, Norway
{ratanbt,martingi}@ifi.uio.no

Abstract. Most structured data today is still stored in relational databases, which makes it important to provide a translation between relational and semantic data. A relational to RDF mapping, such as R2RML [13], provides a way to view existing relational data in the RDF data model through declarative mappings. While relational to RDF mapping translates relational instance data to RDF, it does not specify any translation of existing relational constraints such as primary and foreign key constraints. Since the introduction of R2RML, interest in RDF constraint languages has increased and SHACL [15] has been standardised. This raises the question of which SHACL constraints are guaranteed to be valid on a dataset produced by a relational to RDF mapping. For arbitrary SQL constraints and relational to RDF mappings, this is a hard problem, but we introduce a number of restrictions on the mappings that allow us to introduce a *constraint rewriting* for relational to RDF mappings that faithfully transfers SQL integrity constraints to SHACL constraints. We define and prove two fundamental properties, namely *maximal semantics preservation* and *monotonicity*.

1 Introduction

In relational database theory, one can restrict data to a set of relations that are considered to be useful to applications at hand by imposing relevant integrity constraints upon them, i.e., the semantics properties, also known as data dependencies, that the data in the database must obey. However, such integrity constraints of relational data are not explicit when mapped into RDF. A relational to RDF (R2R) mapping outputs an RDF graph that no longer contains the integrity constraints information. To overcome the problem, one can restore the semantic properties of R2R transformed data by using a semantics preserving constraint rewriting [7,23,26] that maps the integrity constraints of relational data into a well-behaved constraint formalism, which provides a closed-world description for the mapped RDF graph. The integrity constraints of the dataset that is being stored or represented in the RDF graph are a critical piece of information in practice, both to detect problems in the RDF dataset and provide data quality guarantees for RDF data exchange and interoperability.

In this paper, we study constraint rewriting for R2R mapping to make it more faithful by transforming the integrity constraints, such as primary and foreign keys, unique and not null integrity constraints as well as data types, from SQL database to RDF graph. In an attempt to transfer such integrity constraints of relational data, such as key constraints and functional dependency in *direct mapping* [2] to a larger perspective of relational constraints [1, Sect. 10] in more expressive *ontology-based mapping* [18] of relational data, into OWL DL axioms [20] as well as Epistemic DL axioms [14], the problem has recently been studied in [7,23] and [10,11,21] respectively. However, for our work, we follow the constraint rewriting technique proposed in [26] that explicitly transforms integrity constraints of SQL database into integrity constraints on the RDF graph, expressed in SHACL [15] as opposed to OWL/Epistemic DL axioms. Contrary to OWL, SHACL, the Shapes Constraint Language recommended by W3C since 2017, has a closed world semantics and uses the unique name assumption, which makes it a more suitable candidate than OWL for expressing as well as detecting the violations of integrity constraints on an RDF graph.

For arbitrary SQL constraints and relational to RDF mappings, constraint rewriting is a hard problem. For simplicity, we restrict ourselves to (a) the most common SQL constraints, namely keys, uniqueness and not null constraints, and (2) *simple* R2R mappings (Definition 4), which are restricted in such a way that the resulting RDF is structurally close enough to the source that it remains possible to analyse the propagation of source constraints to the target. Thus, once the SHACL descriptions of the mapped RDF graph are available, they can be used to validate that the facts in the graph are compatible with the constraints of the relational source and the mapping, using the SHACL validation engine. However, R2R mappings are also known for their mapping inconsistency and redundancy anomalies [9,16], thus one-to-one semantics correspondence such as *semantics preservation* proposed in [26, Defn. 6] and [23, Defn. 12] between the relational and the mapped RDF data can not be established in general [23,26, Prop. 1]. One of the prominent reasons behind such flaws is that R2R mappings often imply SHACL constraints that satisfy the mapped RDF graph with respect to database constraints even if the key constraints are violated in the source database, which can not be easily fixed as the mappings rely on the values of database keys to produce RDF terms [26, Exam. 4 and 5]. We can thus not hope for semantic *equivalence* between the SQL and SHACL constraints. In this work we instead define a notion of *maximal semantics preservation* to express that any additional SHACL constraints are either implied by the generated ones, or not implied by the SQL constraints.

Example 1. *Consider the following database instance $\mathcal{D}$ with schemas that describes students and their enrollment in courses being offered by a university:*

create table `course` *(*`C_id` *varchar primary key,* `Title` *varchar unique);*
create table `student` *(*`S_id` *integer primary key,* `Name` *varchar,* `Code` *varchar not null foreign key references* `course(C_id)`*);*

S_id	Name	Code
011	Ida	CS40
012		CS20

FK

C_id	Title
CS40	Logic
CS20	Database
CS50	Data Eng

In general, an R2R mapping is an assertion of the form $Q \longrightarrow \psi$ that transforms a set of tuples projected by SQL query Q, called *source query*, over a relational source $\mathcal{D}$ into a set of RDF triples defined by graph triple patterns ψ. Assume an R2R mapping M to retrieve students and their enrollment in the university's courses,

$$\text{Select } \texttt{S\_id} \text{ from } \texttt{student} \longrightarrow \langle \texttt{iri}_1(\texttt{S\_id}), \texttt{rdf:type}, \texttt{Student}\rangle.$$

$$\text{Select } \texttt{C\_id} \text{ from } \texttt{course} \longrightarrow \langle \texttt{iri}_2(\texttt{C\_id}), \texttt{rdf:type}, \texttt{Course}\rangle.$$

$$\text{Select } \texttt{S\_id, C\_id} \text{ from } \texttt{student}, \texttt{course} \longrightarrow \langle \texttt{iri}_1(\texttt{S\_id}), \texttt{enrolledFor}, \texttt{iri}_2(\texttt{C\_id})\rangle.$$
$$\text{where } \texttt{student.Code} = \texttt{course.C\_id}$$

where $\mathtt{iri}_1$ and $\mathtt{iri}_2$ are injective functions that construct iris for students and courses from their respective id's. The mapping M yields the following RDF graph G (on the left) from the database instance $\mathcal{D}$:

$\langle \mathtt{iri}_1(011), \mathtt{rdf{:}type}, \mathtt{Student}\rangle.$
$\langle \mathtt{iri}_1(012), \mathtt{rdf{:}type}, \mathtt{Student}\rangle.$
$\langle \mathtt{iri}_2(CS40), \mathtt{rdf{:}type}, \mathtt{Course}\rangle.$
$\langle \mathtt{iri}_2(CS20), \mathtt{rdf{:}type}, \mathtt{Course}\rangle.$
$\langle \mathtt{iri}_2(CS50), \mathtt{rdf{:}type}, \mathtt{Course}\rangle.$
$\langle \mathtt{iri}_1(011), \mathtt{enrolledFor}, \mathtt{iri}_2(CS40)\rangle.$
$\langle \mathtt{iri}_1(012), \mathtt{enrolledFor}, \mathtt{iri}_2(CS20)\rangle.$

```
:Student a sh:NodeShape, rdfs:Class;
  sh:property [ sh:path :enrolledFor;
      sh:maxCount 1; sh:minCount 1;
      sh:nodeKind sh:IRI; sh:class :Course ].
:Course a sh:NodeShape, rdfs:Class;
  sh:property [ sh:path [sh:inversePath
      :enrolledFor];
     sh:nodeKind sh:IRI; sh:class :Student ].
```

Next, consider a SHACL document S (on the right), which consists of node shapes :Student and :Course with implicit target class[1] that define the constraints, intuitively, all students must be enrolled for exactly one course, and all courses must be enrolled by zero or more students. Now observe that the document S not only validates the graph G but also guarantee the validation of every RDF graphs that can be generated via mappings M from any valid instance $\mathcal{D}$ of the schemas in Example 1, i.e., semantics preservation. Moreover, any further restrictions on the property paths of S, such as all courses must be enrolled by at least one students, would easily be violated, meaning that a valid database instance $\mathcal{D}$ can be found such that mapped RDF graphs would not validate the document S. Thus, we say that S is a maximally implied set of SHACL shapes for the given relational source and the mappings M. For proof details, we refer the readers to the extended version [28].

Example 1 illustrates that an assessment of R2R mapping is necessary to guarantee whether the integrity constraints of relational data are maximally propagated via mappings to the RDF. We thus take the process of R2R transformation into account and define constraint rewriting as a function from constraints in SQL database to the sets of SHACL shapes over RDF graph. We first introduce two fundamental properties of constraint rewriting, namely maximal semantics preservation and monotonicity. Finally, we show that our proposed constraint rewriting is both maximal semantics preserving and monotone, even in the most general and practical scenario where relational databases contain null values. A constraint rewriting for R2R mappings is monotonic if it assures that the result of constraint rewriting that is already computed no longer requires alteration after the addition of new mappings.

2 Preliminaries

In this section, we fix notions and notations fundamental to the definition of R2R mapping, and SHACL constraints [15].

[1] https://www.w3.org/TR/shacl/#implicit-targetClass.

Databases. Let Δ be a countably infinite set of constants, including the reserved symbol $\texttt{null}$. A *relational schema* $\mathcal{R}$ is a finite set of relation names, known as *relation schemas*. We associate with each relation schema $R \in \mathcal{R}$ a finite, non-empty *set of named attributes*, denoted by $\mathsf{att}(R)$. An *instance* $\mathcal{D}$ of $\mathcal{R}$ assigns each relation schema $R \in \mathcal{R}$ a finite set of tuples $R^{\mathcal{D}}$, where each *tuple* $t \in R^{\mathcal{D}}$ is a function that assigns to each attribute in $\mathsf{att}(R)$ a value from domain Δ.

We write X as shorthand for a non-empty set $\{x_1, \ldots, x_n\}$ of attributes for $n \geq 1$, and $x \in X$ to say that x is one of the elements of the set. $|X| = n$ denotes the cardinality of the set. We further write $X \lhd R$ to denote that X is a non-empty subset of $\mathsf{att}(R)$. We write $t(x)$ to denote the restriction of a tuple $t \in R^{\mathcal{D}}$ to an attribute $x \in \mathsf{att}(R)$, which can be extended to a set $X \lhd R$, i.e., $t(X)$. Finally, we define a *relational database* as a pair of $\mathcal{R}$ and $\mathcal{D}$, where $\mathcal{R}$ is a relational schema and $\mathcal{D}$ is a database instance of $\mathcal{R}$. The *active domain* $\Gamma_{\mathcal{D}}$ of a database is the set of constants appearing in $\mathcal{D}$, i.e., $\Gamma_{\mathcal{D}} \subseteq \Delta \setminus \{\texttt{null}\}$.

SQL Constraints. We consider declarations of the SQL: (a) *primary* (PK) and *foreign* (FK) keys, (b) *not null* (NN) and *unique* (UNQ) integrity, and (c) *data types*, constraints on the relational schema $\mathcal{R}$. We write Σ for the set of SQL constraints. NN, UNQ and PK constraints on a relational schema $\mathcal{R}$ are expressions of the form $\texttt{NN}(X, R)$, $\texttt{UNQ}(X, R)$ and $\texttt{PK}(X, R)$, resp., for any $X \lhd R$ such that $R \in \mathcal{R}$. An instance $\mathcal{D}$ of $\mathcal{R}$ satisfies:

- ☐ $\texttt{NN}(X, R)$ if for every $t \in R^{\mathcal{D}}$ and $x \in X$, $t(x) \neq \texttt{null}$.
- ☐ $\texttt{UNQ}(X, R)$ if for every $t, t' \in R^{\mathcal{D}}$, if $t(x) = t'(x) \neq \texttt{null}$ for every $x \in X$ then $t = t'$.
- ☐ $\texttt{PK}(X, R)$ if: (a) for every $t \in R^{\mathcal{D}}$ and $x \in X$, $t(x) \neq \texttt{null}$, and (b) for every $t, t' \in R^{\mathcal{D}}$, if $t(X) = t'(X)$ then $t = t'$.

An FK constraint on $\mathcal{R}$ is an expression of the form $\texttt{FK}(X, R, Y, S)$ for any $X \lhd R$ and $Y \lhd S$ with $|X| = |Y|$ and $R, S \in \mathcal{R}$. An instance $\mathcal{D}$ of $\mathcal{R}$ satisfies $\texttt{FK}(X, R, Y, S)$ if for every $t \in R^{\mathcal{D}}$: either (a) $t(x) = \texttt{null}$ for some $x \in X$, or (b) there exists a tuple $t' \in S^{\mathcal{D}}$ such that $t(X) = t'(Y)$. Next, to handle SQL data types, let the domain of an SQL data type v be a subset $\Delta_v \subseteq \Delta$. An SQL data type declaration on $\mathcal{R}$ is an expression of the form $\texttt{Type}(x, v, R)$ for every $x \in \mathsf{att}(R)$ such that $R \in \mathcal{R}$, where v is an SQL data type. An instance $\mathcal{D}$ of $\mathcal{R}$ satisfies $\texttt{Type}(x, v, R)$ for an attribute $x \in \mathsf{att}(R)$, if $t(x) \in \Delta_v$ for every $t \in R^{\mathcal{D}}$.

A *relational schema* $\mathcal{R}$ *with source constraints* Σ consists of the relational schema $\mathcal{R}$ and a set Σ of SQL constraints on $\mathcal{R}$, such that $\texttt{UNQ}(Y, R) \in \Sigma$ for all $\texttt{FK}(X, R, Y, S) \in \Sigma$, as usual in all SQL implementations. W.l.o.g., we also assume that for every $X \lhd R$: (a) if $\texttt{PK}(X, R) \in \Sigma$, then $\texttt{UNQ}(X, R) \in \Sigma$ and $\texttt{NN}(X, R) \in \Sigma$, (b) if $\texttt{NN}(X, R) \in \Sigma$, then $\texttt{NN}(x, R) \in \Sigma$ for every $x \in X$ and (c) if $\texttt{NN}(x, R) \in \Sigma$ for every $x \in X$, then $\texttt{NN}(X, R) \in \Sigma$. Finally, given a relational schema $\mathcal{R}$ with constraints Σ, and an instance $\mathcal{D}$ of $\mathcal{R}$, we call $\mathcal{D}$ a *legal instance* of $\mathcal{R}$ with Σ, denoted by $\mathcal{D} \models \Sigma$, if $\mathcal{D}$ satisfies all constraints in Σ.

Queries. Assume relational algebra with Selection $\sigma_{\neg\mathsf{isNull}}$, Projection π, Equi Join $\bowtie_{\texttt{equality}}$, Right Outer Join $\rtimes_{\texttt{equality}}$, Left Outer Join $\ltimes_{\texttt{equality}}$ and Full Outer Join $⟗_{\texttt{equality}}$ operations as query language that corresponds to a sub-class of *basic fragment of SQL* standard. We use notation $\sigma_{\neg\texttt{isNull}}$ for the select condition '`IS NOT NULL`' over an attribute as in SQL, which can be extended to a set of attributes. Assume that $\mathcal{R}$ is a relational schema, $\mathcal{D}$ is an instance of $\mathcal{R}$ and Q is a relational algebra expression over $\mathcal{R}$. Then $\mathsf{att}(Q)$, the set of attributes of Q, is recursively defined as follows, where we write $X \triangleleft Q$ to denote that X is a non-empty subset of $\mathsf{att}(Q)$:

1. If $Q = R$ such that $R \in \mathcal{R}$, then $\mathsf{att}(Q) = \mathsf{att}(R)$.
2. If Q' is a relational algebra expression over $\mathcal{R}$, $X \triangleleft Q'$ and $Q = \sigma_{\neg\mathtt{isNull}(X)}(Q')$, i.e., $\sigma_{\neg\mathtt{isNull}(x_1)\wedge\ldots\wedge\neg\mathtt{isNull}(x_n)}(Q')$, then $\mathsf{att}(Q) = \mathsf{att}(Q')$.
3. If Q' is a relational algebra expression over $\mathcal{R}$, $X \triangleleft Q'$ and $Q = \pi_X(Q')$, then $\mathsf{att}(Q) = X$.
4. Let Q_1, Q_2 be relational algebra expressions over $\mathcal{R}$ such that $X \triangleleft Q_1$ and $Y \triangleleft Q_2$ have compatible data types. If $Q = Q_1\, \mathsf{OP}_{X=Y}\, Q_2$ s.t.$\mathsf{OP} \in \{⋈, ⟖, ⟕, ⟗\}$, then $\mathsf{att}(Q) = \mathsf{att}(Q_1) \cup \mathsf{att}(Q_2)$.

The evaluation of Q over $\mathcal{D}$, a set of tuples denoted by $Q^{\mathcal{D}}$, is recursively defined as follows,

1. If $Q = R$ such that $R \in \mathcal{R}$, then $Q^{\mathcal{D}} = R^{\mathcal{D}}$.
2. If Q' is a relational algebra expression over $\mathcal{R}$, $X \triangleleft Q'$ and $Q = \sigma_{\neg\mathtt{isNull}(X)}(Q')$, then $Q^{\mathcal{D}} = \{t \in Q'^{\mathcal{D}} \mid t(x) \neq \mathtt{null}$ for every $x \in X\}$.
3. If Q' is a relational algebra expression over $\mathcal{R}$, $X \triangleleft Q'$ and $Q = \pi_X(Q')$ then, for every $t \in Q^{\mathcal{D}}$ there exists $t' \in Q'^{\mathcal{D}}$ such that $t(X) = t'(X)$.
4. Let Q_1, Q_2 be relational algebra expressions over $\mathcal{R}$ such that $X \triangleleft Q_1$ and $Y \triangleleft Q_2$ have compatible data types.
   a. If $Q = Q_1 ⋈_{X=Y} Q_2$ then for every $t \in Q^{\mathcal{D}}$: (i) there exist $t_1 \in Q_1{}^{\mathcal{D}}$ and $t_2 \in Q_2{}^{\mathcal{D}}$ s.t. $t(x) = t_1(x) = t_2(y) \neq \mathtt{null}$ for every $x \in X$ and $y \in Y$, (ii) $t(u) = t_1(u)$ for every $u \in (\mathsf{att}(Q_1)\backslash\mathsf{att}(Q_2))$, and (iii) $t(v) = t_2(v)$ for every $v \in (\mathsf{att}(Q_2)\backslash\mathsf{att}(Q_1))$.
   b. If $Q = Q_1 ⟕_{X=Y} Q_2$ then for every $t \in Q^{\mathcal{D}}$: either (i) there exist $t_1 \in Q_1{}^{\mathcal{D}}$ and $t_2 \in Q_2{}^{\mathcal{D}}$ s.t. $t(x) = t_1(x) = t_2(y) \neq \mathtt{null}$ for every $x \in X$ and $y \in Y$, $t(u) = t_1(u)$ for every $u \in (\mathsf{att}(Q_1) \setminus \mathsf{att}(Q_2))$ and $t(v) = t_2(v)$ for every $v \in (\mathsf{att}(Q_2) \setminus \mathsf{att}(Q_1))$, or (ii) there exist $t_1 \in Q_1{}^{\mathcal{D}}$ s.t. $t(u) = t_1(u)$ for every $u \in (\mathsf{att}(Q_1) \setminus \mathsf{att}(Q_2))$ and $t(v) = \mathtt{null}$ for every $v \in (\mathsf{att}(Q_2) \setminus \mathsf{att}(Q_1))$.
   c. If $Q = Q_1 ⟖_{X=Y} Q_2$ then for every $t \in Q^{\mathcal{D}}$: either (i) there exist $t_1 \in Q_1{}^{\mathcal{D}}$ and $t_2 \in Q_2{}^{\mathcal{D}}$ s.t. $t(x) = t_1(x) = t_2(y) \neq \mathtt{null}$ for every $x \in X$ and $y \in Y$, $t(u) = t_1(u)$ for every $u \in (\mathsf{att}(Q_1) \setminus \mathsf{att}(Q_2))$ and $t(v) = t_2(v)$ for every $v \in (\mathsf{att}(Q_2) \setminus \mathsf{att}(Q_1))$, or (ii) there exist $t_2 \in Q_2{}^{\mathcal{D}}$ s.t. $t(v) = t_2(v)$ for every $v \in (\mathsf{att}(Q_2) \setminus \mathsf{att}(Q_1))$ and $t(u) = \mathtt{null}$ for every $u \in (\mathsf{att}(Q_1) \setminus \mathsf{att}(Q_2))$.
   d. If $Q = Q_1 ⟗_{X=Y} Q_2$ then $Q^{\mathcal{D}} = Q_a{}^{\mathcal{D}} \cup Q_b{}^{\mathcal{D}}$ s.t.$Q_a = Q_1 ⟕_{X=Y} Q_2$ and $Q_b = Q_1 ⟖_{X=Y} Q_2$.

Henceforth, we denote by `SP` the relational expression containing only select-project relational operations, and `SPJ` the relational expression containing select-project-(outer)join relational operations, respectively.

Definition 1. *Let Q be a relational expression over a relational schema $\mathcal{R}$. Then, we say that the Q is a* `valid query` *if and only if there exist foreign key references between every two sets of attributes participating in an equality join condition in the Q.*

RDF Graphs. Assume that $\mathcal{I}, \mathcal{B}$ and $\mathcal{L}$ are countably infinite disjoint sets of *Internationalized Resource Identifiers* (IRIs), *Blank nodes* and *Literals*, respectively. The set of

RDF terms $\mathcal{T}$ is $\mathcal{I} \cup \mathcal{L} \cup \mathcal{B}$. A *well-defined RDF triple* is defined as a triple $\langle s, p, o\rangle$ where $s \in \mathcal{I} \cup \mathcal{B}$ is called the subject, $p \in \mathcal{I}$ is called the predicate and $o \in \mathcal{T}$ is called the object. An RDF graph $G \subseteq (\mathcal{I} \cup \mathcal{B}) \times \mathcal{I} \times \mathcal{T}$ is a finite subset of RDF triples.

Definition 2. *The set of nodes of an RDF graph G is the set of subjects and objects of triples in the graph, i.e.,* $\{s, o \mid \langle s, p, o\rangle \in G\}$.

Assume a countably infinite set $\mathcal{V}$ of variables disjoint from $\mathcal{T}$. A triple pattern is defined as a triple in $(\mathcal{I} \cup \mathcal{B} \cup \mathcal{V}) \times (\mathcal{I} \cup \mathcal{V}) \times (\mathcal{T} \cup \mathcal{V})$. A *basic graph pattern (BGP)* is a finite set of triple patterns. The schema $sch(\psi)$ of a triple pattern ψ is the *RDF property and class* predicates [17] from the ψ.

Mappings. Formally, we adopt R2R mapping [6,22] that generate RDF triples from the active domain of a database $\Gamma_{\mathfrak{D}}$. Assume countably infinite and disjoint sets $\mathbb{F}$ and $\mathbb{T}$ of iri-template and typing *functions* respectively, with each function $\alpha \in \mathbb{F} \cup \mathbb{T}$ has an associated arity $n > 0$. W.l.o.g., we assume that functions $\mathbb{F} \cup \mathbb{T}$ are injective, and map only `null` to `null`.

Definition 3. *We specify R2R-mapping $\mathcal{M}$, from relational database-to-RDF, partitioned into three disjoint sets: $\mathcal{M}_{\mathcal{C}}$, $\mathcal{M}_{\mathcal{P}}$ and $\mathcal{M}_{\mathcal{U}}$ such that*

i. *$\mathcal{M}_{\mathcal{C}}$ is a set of data-to-RDF concept mappings, each one of the form*

$$Q_X \longrightarrow \langle \mathbf{f}(X), \texttt{rdf:type}, C\rangle,$$

where
- *a.* *Q_X is a source query Q over $\mathcal{R}$ with $X \lhd Q$,*
- *b.* *$\mathbf{f} \in \mathbb{F}$ and C is an RDF concept.*

ii. *$\mathcal{M}_{\mathcal{P}}$ is a set of data-to-RDF object property mappings, each one of the form*

$$Q_{X,Y} \longrightarrow \langle \mathbf{f}(X), P, \mathbf{f}'(Y)\rangle,$$

where
- *a.* *$Q_{X,Y}$ is a source query Q over $\mathcal{R}$ with $X, Y \lhd Q$,*
- *b.* *$\mathbf{f}, \mathbf{f}' \in \mathbb{F}$ and P is an RDF object property.*

iii. *$\mathcal{M}_{\mathcal{U}}$ is a set of data-to-RDF datatype property mappings, each one of the form*

$$Q_{X,Y} \longrightarrow \langle \mathbf{f}(X), U, \mathbf{t}(Y)\rangle,$$

where
- *a.* *$Q_{X,Y}$ is a source query Q over $\mathcal{R}$ with $X, Y \lhd Q$,*
- *b.* *$\mathbf{f} \in \mathbb{F}$, $\mathbf{t} \in \mathbb{T}$ and U is an RDF datatype property.*

Let m be a mapping $Q \longrightarrow \psi$ of a triple pattern ψ, as in Definition 3. The source query Q is the $body(m)$ of m, whereas the triple pattern ψ is the $head(m)$. The schema $sch(\mathcal{M})$ of a mapping set $\mathcal{M}$ is the union of $sch(head(m))$ of each $m \in \mathcal{M}$. For any two mapping sets $\mathcal{M}$ and $\mathcal{M}'$ defined over a relational schema $\mathcal{R}$ with source constraint Σ, we write $\mathcal{M}' \subseteq \mathcal{M}$, if for every mapping definition m, if $m \in \mathcal{M}'$ then $m \in \mathcal{M}$.

Definition 4. *Let Q_C, Q_P and Q_U be the source queries of mappings of an RDF concept C, object property P and datatype property U, respectively. Then, we say that a mapping set $\mathcal{M}$ (according to Definition 3) is a* simple mapping *if: (a) $\mathcal{M}$ contains exactly one mapping definition per concept C, object property P and datatype property U predicates in $sch(\mathcal{M})$; (b) each Q_P is a valid* `SPJ` *query with one join operation, (c) each Q_U is an* `SP` *query, d) if C and C' are the concepts whose instances are subject and object of an object property P, then the Q_C and $Q_{C'}$ are either equal to Q_P or* `SP` *queries with a projected set of attributes whose (tuple) values are mapped to instances of C and C', and (e) if C is the concept whose instances are the subject of a datatype property U, then Q_C is either equal to Q_U or an* `SP` *query with a projected set of attributes whose (tuple) values are mapped into the instances of C.*

Example 2. *Consider the mapping of object property '*`EnrolledFor`*' in Example 1. Instances of concepts '*`Student`*' and '*`Course`*' are mapped to subject and object of the property '*`EnrolledFor`*', respectively. Then, according to simple mapping in Definition 4, the source queries used in the mappings of those '*`Student`*' and '*`Course`*' concepts must be either the exact same source query used in the mapping of the property '*`EnrolledFor`*' or the* `SP` *source queries as in Example 1. Thus, a distinct simple mapping could be defined for the same purpose that maps RDF concepts '*`Student`*' and '*`Course`*' using the same* `SPJ` *source query Q_P,*

$$Q_P ::= \textit{Select}\ \texttt{S\_id, C\_id}\ \textit{from}\ \texttt{student}, \texttt{course}$$
$$\textit{where}\ \texttt{student.Code} = \texttt{course.C\_id}$$

as used in the mapping of object property '`EnrolledFor`*' as follows:*

$$Q_P \longrightarrow \langle \mathtt{iri}_1(\mathtt{S\_id}), \mathtt{rdf\!:\!type}, \mathtt{Student}\rangle.$$
$$Q_P \longrightarrow \langle \mathtt{iri}_2(\mathtt{C\_id}), \mathtt{rdf\!:\!type}, \mathtt{Course}\rangle.$$
$$Q_P \longrightarrow \langle \mathtt{iri}_1(\mathtt{S\_id}), \mathtt{enrolledFor}, \mathtt{iri}_2(\mathtt{C\_id})\rangle.$$

Let $t \in Q^{\mathcal{D}}$ be a tuple of constants, and let $\mathbf{f}(\mathrm{X})$ be a term such that $\mathbf{f} \in \mathbb{F}$ and $X \triangleleft Q$. Then, $\mathbf{f}(t(X))$ is a ground term of $\mathbf{f}(\mathrm{X})$ obtained by substituting occurrence of every $x \in X$ with $t(x)$.

Definition 5. *Let $\mathcal{M}_C \cup \mathcal{M}_{\mathcal{P}} \cup \mathcal{M}_{\mathcal{U}}$ be an R2R mapping set $\mathcal{M}$ defined over a relational schema $\mathcal{R}$, and $\mathcal{D}$ an instance of $\mathcal{R}$. Then, we call the set of well-defined RDF triple assertions $\mathcal{M}(\mathcal{D})$, i.e.,*

$$\begin{aligned}\mathcal{M}(\mathcal{D}) = \{&\langle \mathbf{f}(t(X)), \mathtt{rdf\!:\!type}, C\rangle \mid \{Q \longrightarrow \langle \mathbf{f}(X), \mathtt{rdf\!:\!type}, C\rangle\} \in \mathcal{M}_C, X \triangleleft Q\ \textit{and}\ t \in Q^{\mathcal{D}}\}\\ &\cup\ \{\langle \mathbf{f}(t(X)), P, \mathbf{f}'(t(Y))\rangle \mid \{Q \longrightarrow \langle \mathbf{f}(X), P, \mathbf{f}'(Y)\rangle\} \in \mathcal{M}_{\mathcal{P}}, X, Y \triangleleft Q\ \textit{and}\ t \in Q^{\mathcal{D}}\}\\ &\cup\ \{\langle \mathbf{f}(t(X)), U, \mathbf{t}(t(Y))\rangle \mid \{Q \longrightarrow \langle \mathbf{f}(X), U, \mathbf{t}(Y)\rangle\} \in \mathcal{M}_{\mathcal{U}}, X, Y \triangleleft Q\ \textit{and}\ t \in Q^{\mathcal{D}}\},\end{aligned}$$

the RDF graph projected by the mapping set $\mathcal{M}$ and the instance $\mathcal{D}$.

We recall that R2R mappings in Definition 3 generate RDF triples from the active domain of a database $\Gamma_{\mathcal{D}}$, i.e., `null` cannot appear in the output RDF triples. Therefore, in this paper, we explicitly consider that (a) mappings $\mathcal{M}$ is simple, and (b) w.l.o.g., source query Q of each mapping in $\mathcal{M}$ contains $\sigma_{\neg\mathtt{isNull}(X)}$ and $\sigma_{\neg\mathtt{isNull}(Y)}$ filters over every projected set of $X, Y \triangleleft \mathsf{att}(Q)$.

SHACL. Our formal treatment of the *core constraints* of SHACL [15] is based on the approach of Corman et al. [12]. Each SHACL constraint is a set of conditions, usually referred to as shape, defined as a triple $\langle s, \tau_s, \phi_s \rangle$ consisting of a shape IRI s, a *target definition* τ_s, and a *constraint definition* ϕ_s. The τ_s and ϕ_s are expressions that determine for every RDF graph G and node n of G, whether n is a target of the shape, $G \models \tau_s(n)$, respectively, whether n satisfies the constraint, $G \models \phi_s(n)$. All shapes generated by our transformation have an 'implicit target class,' which means that s is also the IRI of a class and $G \models \tau(n)$ iff n is a SHACL instance of class s.[2] For the purpose of our work, the constraint ϕ_s is an expression defined according to the following grammar:

$$\begin{aligned} \phi &::= \phi \wedge \phi \ \mid \geq_n P^{\pm}.\alpha \ \mid \leq_n P^{\pm}.\alpha \ \mid \rhd_C P^{\pm} \\ \alpha &::= \top \ \mid \ \ell \ \mid \ \neg\ell \ \mid \ C \ \mid \ \neg C \end{aligned} \tag{1}$$

where $\top$ stands for truth, ℓ is an XML schema datatype, C and P are an RDF concept and property names respectively, the superscript $\pm$ stands for a property or its inverse, $n \in \mathbb{N}$, $\neg$ for negation, $(\geq_n P^{\pm}.\alpha)$ means 'must have at least n $P^{\pm}$-successor verifying α' for any $n \in \mathbb{N}$ and $(\rhd_C P^{\pm})$ means 'all values of $P^{\pm}$-successor must be unique[3] among instances of concept C'. As syntactic sugar, we use $(=_n P^{\pm}.\alpha)$ for $(\geq_n P^{\pm}.\alpha) \wedge (\leq_n P^{\pm}.\alpha)$, $(\rhd_C P^{\pm}.\alpha)$ for $(\leq_1 P^{\pm}.\alpha) \wedge (\rhd_C P^{\pm})$ and $(\unrhd_C P^{\pm}.\alpha)$ for $(=_1 P^{\pm}.\alpha) \wedge (\rhd_C P^{\pm})$.

A SHACL document is a set of SHACL shapes. An RDF graph G validates against a shape $\langle s, \tau_s, \phi_s \rangle$ if for every nodes n of G, if $G \models \tau_s(n)$ then $G \models \phi_s(n)$. An RDF graph G validates against a SHACL document S, written $G \models S$, iff G validates against all shapes in S. The schema $sch(s)$ of a SHACL shape s is the set of RDF concept and property predicates [17] used in the target τ_s and constraint ϕ_s definition. The schema $sch(S)$ of a SHACL document S is the union of $sch(s)$ of every shape $s \in S$.

3 Constraint Rewriting: Definition and Properties

Our goal is to generate a set of SHACL constraints that is as strong as possible while being guaranteed to hold for all RDF graphs resulting from valid database instances. Let $\mathcal{M}$ be a mapping set defined over a relational schema $\mathcal{R}$ with source constraints Σ.

Definition 6. *A SHACL document S is an Σ-implied set of shapes with respect to $\mathcal{M}$, written as $\Sigma \models_{\mathcal{M}} S$, if for every instance $\mathcal{D}$ of $\mathcal{R}$:*

$$\mathcal{D} \models \Sigma \rightarrow \mathcal{M}(\mathcal{D}) \models S.$$

Definition 7. *Let $\Sigma \models_{\mathcal{M}} S$. Then, we say that S is a maximally Σ-implied set of shapes with respect to $\mathcal{M}$, written as $\Sigma \models^{*}_{\mathcal{M}} S$, if for every $\Sigma \models_{\mathcal{M}} S'$ s.t. $sch(S') \subseteq sch(\mathcal{M})$ and every RDF graph $\mathcal{G}$:*

$$\mathcal{G} \models S \rightarrow \mathcal{G} \models S'.$$

We now formalise a constraint rewriting and some desirable properties. Let $\mathbb{S}$ be the set of all SHACL shapes and $\mathbb{Q}$ be the set of all pairs $(\mathcal{M}, \Sigma)$ such that $\mathcal{M}$ is a mapping set defined over a relational schema $\mathcal{R}$ with source constraints Σ.

[2] https://www.w3.org/TR/shacl/#implicit-targetClass.

[3] `dash:uniqueValueForClassConstraintComponent` from http://datashapes.org.

Definition 8 (Constraint rewriting). *A* constraint rewriting *is a function* $\mathcal{T} : \mathbb{Q} \rightarrow \mathcal{P}(\mathbb{S})$.

We next introduce central properties of a constraint rewriting $\mathcal{T}$.

Definition 9 (Semantics preservation). *A constraint rewriting* $\mathcal{T}$ *is* semantics preserving *if for every mapping set* $\mathcal{M}$ *and every source constraints* Σ*:*

$$\Sigma \models_{\mathcal{M}} \mathcal{T}(\mathcal{M}, \Sigma).$$

Definition 10 (Maximal semantics preservation). *A constraint rewriting* $\mathcal{T}$ *is* maximal semantics preserving *if for every mapping set* $\mathcal{M}$ *and every source constraints* Σ*:*

$$\Sigma \models^{*}_{\mathcal{M}} \mathcal{T}(\mathcal{M}, \Sigma).$$

Definition 11 (Monotonicity). *A constraint rewriting* $\mathcal{T}$ *is monotone if for any mapping sets* $\mathcal{M}' \subseteq \mathcal{M}$ *defined over a relational schema* $\mathcal{R}$ *with source constraint* Σ *and every RDF graph* $\mathcal{G}$*:*

$$\mathcal{G} \models \mathcal{T}(\mathcal{M}, \Sigma) \rightarrow \mathcal{G} \models \mathcal{T}(\mathcal{M}', \Sigma).$$

4 View Constraint: Definitions

As introduced in Sect. 2, R2R mapping relies on database views based on a source query to compute RDF terms from the database values. As a first step of our constraint transformation, we have to analyse the propagation of database constraints to these views.

Let $\mathcal{R}$ be a relational schema with source constraints Σ, and $R \in \mathcal{R}$. The constraint Σ restricted to the set of $\mathsf{att}(R)$, denoted by $\Sigma|_R$, is the set of constraints such that for every constraint $\sigma \in \Sigma$ on any $X \triangleleft R$, there is $\sigma \in \Sigma|_R$. For example, if $\mathtt{FK}(X, R, Y, S) \in \Sigma$ (resp., $\mathtt{FK}(Y, S, X, R) \in \Sigma$) on any $X \triangleleft R$, then there is $\mathtt{FK}(X, R, Y, S) \in \Sigma|_R$ (resp., $\mathtt{FK}(Y, S, X, R) \in \Sigma|_R$).

Definition 12. *Let* Q *be a relational expression over a relational schema* $\mathcal{R}$ *with source constraints* Σ*. Then, the set* Σ *propagated to the set of* $\mathsf{att}(Q)$*, denoted by* $\Sigma|_Q$*, is recursively defined as follows,*

a. If $Q = R$ *such that* $R \in \mathcal{R}$*, then* $\Sigma|_Q = \Sigma|_R$*.*
b. $Q = \sigma_{\neg \mathtt{isNull}(X)}(Q')$ *where* $X \triangleleft Q'$*, then* $\Sigma|_Q = \Sigma|_{Q'}$*.*
c. If $Q = \pi_X(Q')$ *where* $X \triangleleft Q'$ *then* $\Sigma|_Q = \{\mathit{PK}(Y, R), \mathit{UNQ}(Y, R), \mathit{NN}(Y, R), \mathit{FK}(Y, R, Z, S), \mathit{FK}(Z, S, Y, R) \in \Sigma|_{Q'} \mid Y \subseteq X \text{ and } R, S \in \mathcal{R}\}$*.*
d. If $Q = Q_1 \,\mathit{OP}_{X=Y}\, Q_2$ *where* $X \triangleleft Q_1$ *and* $Y \triangleleft Q_2$ *have compatible data types, and* $\mathit{OP} \in \{\bowtie, ⟕, ⟖, ⟗\}$*, then* $\Sigma|_Q = \Sigma|_{Q_1} \cup \Sigma|_{Q_2}$*.*

SQL constraints are not well suited to direct translation to SHACL, so we introduce an intermediate representation similar to functional dependencies. Let R be a relation name with $X, Y \triangleleft R$. Then, we write a functional dependency as an expression of the form $\mathtt{FD}_{X \rightarrow Y}$, i.e., meaning $X \triangleleft R$ functionally determines $Y \triangleleft R$. Relational data dependencies, such as functional, multi-value and others, are originally defined on databases without `null` [3,5]. However, we need notions of data dependencies that also apply to databases with `null`, such as in [4], which we define as follows:

Definition 13. *Let Q be a source query over a relational schema $\mathcal{R}$ with source constraints Σ, $R \in \mathcal{R}$ a relation name and $\mathcal{D}$ an arbitrary instance of $\mathcal{R}$. Let V be the pair $(Q^{\mathcal{D}}, \Sigma|_Q)$ of projected view $Q^{\mathcal{D}}$ and propagated constraints $\Sigma|_Q$. Then, for any $X, Y \lhd Q$,*

a. $V \models \mathtt{FP}_{X\to Y}$ *if for every $t, t' \in Q^{\mathcal{D}}$, if $t(X) = t'(X)$ then $t(Y) = t'(Y)$.*
b. $V \models \mathtt{UF}_{X\to Y}$ *if $Q^{\mathcal{D}} \models \mathtt{FP}_{X\to Y}$ and $Q^{\mathcal{D}} \models \mathtt{FP}_{Y\to X}$.*
c. $V \models \mathtt{FD}_{X\to Y}$ *if $Q^{\mathcal{D}} \models \mathtt{FP}_{X\to Y}$ and $\mathtt{NN}(X,R), \mathtt{NN}(Y,R) \in \Sigma|_Q$.*
d. $V \models \mathtt{UFD}_{X\to Y}$ *if $Q^{\mathcal{D}} \models \mathtt{FD}_{X\to Y}$ and $Q^{\mathcal{D}} \models \mathtt{FD}_{Y\to X}$.*

Henceforth, we will keep the SQL notations intuitively simple in examples, i.e., we write $\mathtt{NN}(X) \in \Sigma|_{X\lhd R}$ instead of $\mathtt{NN}(X,R) \in \Sigma|_{X\lhd R}$ for the propagated $\mathtt{NN}(X,R) \in \Sigma$ to $\Sigma|_{X\lhd R}$.

Example 3. *Following Example 1, assume a mapping set $\mathcal{M}$ with $\mathbf{f}_S$ and $\mathbf{f}_C$ iri-templates and a typing function $\mathbf{t}_v$[4] as follows:*

a. $\pi_{\mathtt{S\_id,Name}}\sigma_{\neg\mathtt{isNull}(\mathtt{S\_id})\wedge\neg\mathtt{isNull}(\mathtt{Name})}(\mathtt{student}) \longrightarrow \langle \mathbf{f}_S(\mathtt{S\_id}), \mathtt{hasName}, \mathbf{t}_v(\mathtt{Name})\rangle$.
b. $\pi_{\mathtt{C\_id,Title}}\sigma_{\neg\mathtt{isNull}(\mathtt{C\_id})\wedge\neg\mathtt{isNull}(\mathtt{Title})}(\mathtt{course}) \longrightarrow \langle \mathbf{f}_C(\mathtt{C\_id}), \mathtt{hasTitle}, \mathbf{t}_v(\mathtt{Title})\rangle$.

Let $Q_1 = \pi_{\mathtt{S\_id,Name}}\sigma_{\neg\mathtt{isNull}(\mathtt{S\_id})\wedge\neg\mathtt{isNull}(\mathtt{Name})}(\mathtt{student})$, and $V_1 = ({Q_1}^{\mathcal{D}}, \Sigma|_{Q_1})$. Then,

- $\mathsf{att}(Q_1) = \{\mathtt{S\_id,Name}\}$ *and* $\Sigma|_{\mathsf{att}(Q_1)} = \{\mathtt{PK}(\mathtt{S\_id}), \mathtt{UNQ}(\mathtt{S\_id}), \mathtt{NN}(\mathtt{S\_id}), \mathtt{Type}(\mathtt{S\_id}, v), \mathtt{Type}(\mathtt{Name}, v)\}$, *i.e., from assumption in Sect. 2, if $\mathtt{PK}(\mathtt{S\_id})$ then $\mathtt{UNQ}(\mathtt{S\_id})$ and $\mathtt{NN}(\mathtt{S\_id})$.*
- $V_1 \models \mathtt{FP}_{\mathtt{S\_id}\to Name}$ *since for every $t, t' \in {Q_1}^{\mathcal{D}}$, if $t(\mathtt{S\_id}) = t'(\mathtt{S\_id})$ then $t(\mathtt{Name}) = t'(\mathtt{Name})$.*

Filter $\sigma_{\neg\mathtt{isNull}(\mathtt{Name})}$ excludes tuples from ${Q_1}^{\mathcal{D}}$ that contains null for the $\mathtt{Name} \in \mathsf{att}(Q_1)$.
Similarly, let $Q_2 = \pi_{\mathtt{C\_id,\ Title}}\sigma_{\neg\mathtt{isNull}(\mathtt{C\_id})\wedge\neg\mathtt{isNull}(\mathtt{Title})}(\mathtt{course})$, and $V_2 = ({Q_2}^{\mathcal{D}}, \Sigma|_{Q_2})$. Then,

- $\mathsf{att}(Q_2) = \{\mathtt{C\_id,\ Title}\}$ *and* $\Sigma|_{\mathsf{att}(Q_2)} = \{\mathtt{PK}(\mathtt{C\_id}), \mathtt{UNQ}(\mathtt{C\_id}), \mathtt{NN}(\mathtt{C\_id}), \mathtt{Type}(\mathtt{C\_id}, v), \mathtt{UNQ}(\mathtt{Title}), \mathtt{Type}(\mathtt{Title}, v), \mathtt{FK}(\mathtt{Code}, \mathtt{student}, \mathtt{C\_id}, \mathtt{course})\}$
- $V_2 \models \mathtt{FP}_{\mathtt{C\_id}\to\mathtt{Title}}$ *since for any $t, t' \in {Q_2}^{\mathcal{D}}$, if $t(\mathtt{C\_id}) = t'(\mathtt{C\_id})$ then $t(\mathtt{Title}) = t'(\mathtt{Title})$.*
- $V_2 \models \mathtt{FP}_{\mathtt{Title}\to\mathtt{C\_id}}$ *since for any $t, t' \in {Q_2}^{\mathcal{D}}$, if $t(\mathtt{Title}) = t'(\mathtt{Title})$ then $t(\mathtt{C\_id}) = t'(\mathtt{C\_id})$.*
- $V_2 \models \mathtt{UF}_{\mathtt{C\_id}\to\mathtt{Title}}$ *since ${Q_2}^{\mathcal{D}} \models \mathtt{FP}_{\mathtt{C\_id}\to\mathtt{Title}}$ and ${Q_2}^{\mathcal{D}} \models \mathtt{FP}_{\mathtt{Title}\to\mathtt{C\_id}}$.*

5 Source to View Constraint Implication

The next step is to determine which of the data dependencies from Definition 13 hold for the view defined by the source queries, i.e., they are implied by the propagated SQL constraints.

Let Q be a source query over a relational schema $\mathcal{R}$ with source constraints Σ. Then, we say that Σ implies a data dependency $\sigma_{X\to Y}$ s.t. $\sigma \in \{\mathtt{UFD,FD,UFP,FP}\}$ on $X, Y \lhd Q$, denoted by $\Sigma_Q \Vdash \sigma_{X\to Y}$, if $V \models \sigma_{X\to Y}$ for every legal instance $\mathcal{D}$ of $\mathcal{R}$, where $V = (Q^{\mathcal{D}}, \Sigma|_Q)$ is the pair of projected view $Q^{\mathcal{D}}$ and propagated constraints $\Sigma|_Q$. We now concentrate on SP source queries.

[4] $\mathbf{t}_v$ specify XML Schema datatype of RDF literal $\mathbf{t}_v(d)$ corresponding to the SQL data type v of the database constant $d \in \Delta_v$, e.g., $\mathbf{t}_v$ is an xsd:string IRI term if v is varchar SQL data type.

Lemma 1. *Let Q be a source query $\pi_{X,Y}\sigma_{\neg\texttt{isNull}(X)\wedge\neg\texttt{isNull}(Y)}(R)$ over a relational schema $\mathcal{R}$ with source constraints Σ, $R \in \mathcal{R}$ a relation name and $\Sigma|_Q$ the set Σ propagated to set of* $\mathsf{att}(Q)$. *Then, for any $X, Y \lhd Q$,*

a. $\Sigma_Q \Vdash \mathit{FP}_{X\to Y}$ *if* $\mathit{UNQ}(X,R) \in \Sigma|_Q$.
b. $\Sigma_Q \Vdash \mathit{UF}_{X\to Y}$ *if* $\mathit{UNQ}(X,R), \mathit{UNQ}(Y,R) \in \Sigma|_Q$.
c. $\Sigma_Q \Vdash \mathit{FD}_{X\to Y}$ *if* $\mathit{UNQ}(X,R) \in \Sigma|_Q$ *and* $\mathit{NN}(X,R), \mathit{NN}(Y,R) \in \Sigma|_Q$.
d. $\Sigma_Q \Vdash \mathit{UFD}_{X\to Y}$ *if* $\mathit{UNQ}(X,R), \mathit{UNQ}(Y,R) \in \Sigma|_Q$ *and* $\mathit{NN}(X,R), \mathit{NN}(Y,R) \in \Sigma|_Q$.

Corollary 1. *Let Q be a source query $\pi_{X,Y}\sigma_{\neg\texttt{isNull}(X)\wedge\neg\texttt{isNull}(Y)}(R)$ over a relational schema $\mathcal{R}$ with source constraints Σ, $R \in \mathcal{R}$ a relation name and $\Sigma|_Q$ the set Σ propagated to set of* $\mathsf{att}(Q)$. *Then, for any $X, Y \lhd Q$,*

a. $\Sigma_Q \Vdash \mathit{UFD}_{X\to Y} \to \Sigma_Q \Vdash \mathit{FD}_{X\to Y}$ *and* $\Sigma_Q \Vdash \mathit{FD}_{X\to Y} \to \Sigma_Q \Vdash \mathit{FP}_{X\to Y}$
b. $\Sigma_Q \Vdash \mathit{UFD}_{X\to Y} \to \Sigma_Q \Vdash \mathit{UF}_{X\to Y}$ *and* $\Sigma_Q \Vdash \mathit{UF}_{X\to Y} \to \Sigma_Q \Vdash \mathit{FP}_{X\to Y}$

We next concentrate on $\texttt{SPJ}$ source queries. An $\texttt{SPJ}$ source query Q over a relational schema $\mathcal{R}$ with source constraints Σ is a relational algebra expression of the form,

$$Q := \pi_{X,Y}\sigma_{\neg\texttt{isNull}(X)\wedge\neg\texttt{isNull}(Y)}(R_1\ \texttt{OP}_{U=V}\ R_2),$$

where $R_1, R_2 \in \mathcal{R}$ are relation names with $X, U \lhd R_1$ and $Y, V \lhd R_2$, $|U| = |V|$ and $\texttt{OP} \in \{\bowtie, ⟕, ⟖, ⟗\}$. Since mapping in Definition 3 generates RDF triples from the active domain $\Gamma_{\mathcal{D}} \subseteq \Delta \setminus \{\texttt{null}\}$ of the database, w.l.o.g., we equivalently express the stated $\texttt{SPJ}$ source query Q, that yields the same set of RDF triples as the original Q, as follows,

$$\pi_{X,Y}\sigma_{\neg\texttt{isNull}(X)\wedge\neg\texttt{isNull}(Y)}(\sigma_{\neg\texttt{isNull}(X)\wedge\neg\texttt{isNull}(U)}(R_1)\ \texttt{OP}_{U=V}\ \sigma_{\neg\texttt{isNull}(V)\wedge\neg\texttt{isNull}(Y)}(R_2)).$$

Note that the $\texttt{SPJ}$ query Q is valid if and only if $\mathtt{FK}(U, R_1, V, R_2) \in \Sigma|_Q$ or $\mathtt{FK}(V, R_2, U, R_1) \in \Sigma|_Q$, see Definition 1. Henceforth, we use symbol $\to^*$ to express dependency in the opposite direction of foreign key reference, i.e., we write $\mathrm{FD}_{X\to^* Y}$ to state functional dependency from $X \lhd Q$ to $Y \lhd Q$ if $\mathtt{FK}(Y, R_2, X, R_1) \in \Sigma|_Q$ or $\mathtt{FK}(V, R_2, U, R_1) \in \Sigma|_Q$ s.t. $X, U \lhd R_1$ and $Y, V \lhd R_2$.

Lemma 2. *Let $\mathcal{R}$ be a relational schema with source constraints Σ, and let Q be an* $\mathtt{SPJ}$ *source query over $\mathcal{R}$,*

$$Q := \pi_{X,Y}\sigma_{\neg\mathtt{isNull}(X)\wedge\neg\mathtt{isNull}(Y)}(Q_1\ \mathit{OP}_{U=V}\ Q_2)$$

s.t. Q_1 and Q_2 are SP expressions over $R_1 \in \mathcal{R}$ and $R_2 \in \mathcal{R}$ with $X, U \lhd Q_1$ and $Y, V \lhd Q_2$ respectively, $\mathit{OP} \in \{\bowtie, ⟕, ⟖, ⟗\}$ *and* $\mathit{FK}(U, R_1, V, R_2) \in \Sigma|_Q$. *Then, for any $X, Y \lhd Q$:*

a. $\Sigma_Q \Vdash \sigma_{X\to Y}$ *if* $\Sigma_{Q_1} \Vdash \sigma_{X\to U}$ *and* $\Sigma_{Q_2} \Vdash \sigma_{V\to Y}$ *s.t.* $\sigma \in \{\mathtt{UFD}, \mathtt{FD}, \mathtt{UF}\}$.
b. $\Sigma_Q \Vdash \sigma_{X\to Y}$ *if* $\Sigma_{Q_1} \Vdash \mathit{UFD}_{X\to U}$ *and* $\Sigma_{Q_2} \Vdash \sigma_{V\to Y}$ *s.t.* $\sigma \in \{\mathtt{FD}, \mathtt{UF}\}$.
c. $\Sigma_Q \Vdash \sigma_{X\to Y}$ *if* $\Sigma_{Q_1} \Vdash \sigma_{X\to U}$ *s.t.* $\sigma \in \{\mathtt{FD}, \mathtt{UF}\}$ *and* $\Sigma_{Q_2} \Vdash \mathit{UFD}_{V\to Y}$.
d. $\Sigma_Q \Vdash \mathit{FP}_{X\to Y}$ *if* $\Sigma_{Q_1} \Vdash \mathit{FD}_{X\to U}$ *and* $\Sigma_{Q_2} \Vdash \mathit{UF}_{V\to Y}$.
e. $\Sigma_Q \Vdash \mathit{FP}_{X\to Y}$ *if* $\Sigma_{Q_1} \Vdash \mathit{UF}_{X\to U}$ *and* $\Sigma_{Q_2} \Vdash \mathit{FD}_{V\to Y}$.
f. $\Sigma_Q \Vdash \mathit{FP}_{X\to Y}$ *if* $\Sigma_{Q_1} \Vdash \mathit{FP}_{X\to U}$.
g. $\Sigma_Q \Vdash \mathit{FP}_{X\to Y}$ *if* $\Sigma_{Q_1} \Vdash \sigma_{X\to U}$ *and* $\Sigma_{Q_2} \Vdash \mathit{FP}_{V\to Y}$ *s.t.* $\sigma \in \{\mathtt{UFD}, \mathtt{FD}, \mathtt{UF}\}$.

h. $\Sigma_Q \Vdash \sigma_{Y \to^* X}$ *if* $\Sigma_{Q_1} \Vdash \sigma_{U \to X}$ *and* $\Sigma_{Q_2} \Vdash \sigma_{Y \to V}$ *s.t.* $\sigma \in \{\mathtt{UFD}, \mathtt{UF}\}$.
i. $\Sigma_Q \Vdash \mathtt{FP}_{Y \to^* X}$ *if* $\Sigma_{Q_1} \Vdash \sigma_{U \to X}$ *s.t.* $\sigma \in \{\mathtt{UFD}, \mathtt{FD}, \mathtt{FP}\}$ *and* $\Sigma_{Q_2} \Vdash \mathtt{UF}_{Y \to V}$.
j. $\Sigma_Q \Vdash \sigma_{Y \to * X}$ *if* $\Sigma_{Q_1} \Vdash \sigma_{U \to X}$ *s.t.* $\sigma \in \{\mathtt{FD}, \mathtt{UF}, \mathtt{FP}\}$ *and* $\Sigma_{Q_2} \Vdash \mathtt{UFD}_{Y \to V}$.

On the correctness of Lemma 2, e.g., assume the case (f). Then, $\mathtt{UNQ}(V, R_2) \in \Sigma|_{Q_2}$ since $\mathtt{FK}(U, R_1, V, R_2) \in \Sigma|_Q$. Thus, $\Sigma_{Q_2} \Vdash \sigma_{V \to Y}$ s.t. $\sigma \in \{\mathtt{UFD}, \mathtt{FD}, \mathtt{UF}, \mathtt{FP}\}$ is the set of all possible constraints implication. Hence, the case (f) of Lemma 2 covers the following possible cases of constraints implication:

- $\Sigma_Q \Vdash \mathtt{FP}_{X \to Y}$ if $\Sigma_{Q_1} \Vdash \mathtt{FP}_{X \to U}$ and $\Sigma_{Q_2} \Vdash \sigma_{V \to Y}$ s.t. $\sigma \in \{\mathtt{UFD}, \mathtt{FD}, \mathtt{UF}, \mathtt{FP}\}$.

Further, by applying similar arguments and the implication rules stated in Corollary 1 to the rest of cases in Lemma 2, the correctness proof of the Lemma can be enumerated.

Example 4. *Following Examples 1 and 4, assume an R2R mapping:*

$$Q \longrightarrow \langle \mathbf{f}_S(\mathtt{S\_id}), \mathtt{enrolledFor}, \mathbf{f}_C(\mathtt{C\_id}) \rangle,$$

where Q is a source query $\pi_{\mathtt{S\_id},\mathtt{C\_id}} \sigma_{\neg \mathtt{isNull}(\mathtt{S\_id}) \wedge \neg \mathtt{isNull}(\mathtt{C\_id})}(Q_1 \bowtie_{\mathtt{Code}=\mathtt{C\_id}} Q_2)$ such that $Q_1 = \sigma_{\neg \mathtt{isNull}(\mathtt{S\_id}) \wedge \neg \mathtt{isNull}(\mathtt{Code})}(\mathtt{student})$ and $Q_2 = \sigma_{\neg \mathtt{isNull}(\mathtt{C\_id})}(\mathtt{course})$. Then,

a. for SP expression Q_1 :
- $\mathsf{att}(Q_1) = \{\mathtt{S\_id}, \mathtt{Code}\}$ *and* $\{\mathtt{UNQ}(\mathtt{S\_id}), \mathtt{NN}(\mathtt{S\_id}), \mathtt{NN}(\mathtt{Code})\} \subseteq \Sigma|_{Q_1}$ *from Definition 12.*
- $\Sigma_{Q_1} \Vdash \mathtt{FD}_{\mathtt{S\_id} \to \mathtt{Code}}$ *from the case (c) of Lemma 1*

b. for SP expression Q_2 :
- $\mathsf{att}(Q_2) = \{\mathtt{C\_id}\}$ *and* $\{\mathtt{UNQ}(\mathtt{C\_id}), \mathtt{NN}(\mathtt{C\_id})\} \subseteq \Sigma|_{Q_2}$ *from Definition 12.*
- $\Sigma_{Q_2} \Vdash \mathtt{UFD}_{\mathtt{C\_id} \to \mathtt{C\_id}}$ *from the case (d) of Lemma 1*

c. finally, for SPJ expression Q:
- $\mathsf{att}(Q) = \{\mathtt{S\_id}, \mathtt{C\_id}\}$
- $\mathtt{FK}(\mathtt{Code}, \mathtt{student}, \mathtt{C\_id}, \mathtt{course}) \in \Sigma|_{Q_1} \cap \Sigma|_{Q_2}$, *i.e., Q is a valid SPJ query.*
- $\Sigma_Q \Vdash \mathtt{FD}_{\mathtt{S\_id} \to \mathtt{C\_id}}$ *from case (c) of Lemma 2, since*
  - *i.* $\Sigma_{Q_1} \Vdash \mathtt{FD}_{\mathtt{S\_id} \to \mathtt{Code}}$, *and*
  - *ii.* $\Sigma_{Q_2} \Vdash \mathtt{FD}_{\mathtt{C\_id} \to \mathtt{C\_id}}$ *from* $\Sigma \Vdash \mathtt{UFD}_{\mathtt{C\_id} \to \mathtt{C\_id}} \to \Sigma \Vdash \mathtt{FD}_{\mathtt{C\_id} \to \mathtt{C\_id}}$ *following the case (a) of Corollary 1*

6 The Constraint Rewriting

We now introduce a constraint rewriting Γ for a simple mapping $\mathcal{M}$ (Definition 4), and prove the properties defined in Sect. 3. The constraint rewriting Γ in Definition 15 transforms the view constraints implied by the relational source Σ (as introduced in Sects. 4 and 5) into sets of SHACL shapes. Since the semantic equivalence of generated SHACL constraints to the source constraints Σ also depends on the combination of source queries used in mappings of RDF triples, we first introduce the classification functions ι and κ to distinguish between the various cases that can occur.

Let $\mathbf{f}_C$ and $\mathbf{f}_{C'}$ be $\mathtt{iri}$ mapping templates for the respective RDF concepts C and C', and let $\mathbf{t}$ be an $\mathtt{iri}$ typing template. Let Q_C, Q_P and Q_U be the source queries of mapping Definition 3 of an RDF concept C, object property P and datatype property U, respectively.

Definition 14. *Let $\mathcal{M}$ be a simple mapping with RDF predicates $C, C', P, U \in sch(\mathcal{M})$. Let ι and κ be classification functions that take a triple pattern of the form $\langle \mathbf{f}_C(X), P, \mathbf{f}_{C'}(Y)\rangle$ and $\langle \mathbf{f}_C(X), U, \mathbf{t}(Y)\rangle$ respectively, and the mapping set $\mathcal{M}$ as input, and classifies the groups of the respective source queries $(Q_C, Q_P, Q_{C'})$ and (Q_C, Q_U) as follows,*

$$\iota(\langle \mathbf{f}_C(X), P, \mathbf{f}_{C'}(Y)\rangle, \mathcal{M}) = \begin{cases} A & \text{if } Q_C \neq Q_P \\ B & \text{otherwise.} \end{cases} \quad \text{and } \kappa(\langle \mathbf{f}_C(X), U, \mathbf{t}(Y)\rangle, \mathcal{M}) = \begin{cases} A & \text{if } Q_C \neq Q_U \\ B & \text{otherwise.} \end{cases}$$

Let Q be a source query over a relational schema $\mathcal{R}$ with source constraint Σ. Then, we write $\Sigma_Q \Vdash \sigma_{X \twoheadrightarrow Y}$ s.t. $\sigma \in \{\texttt{UFD}, \texttt{FD}, \texttt{UF}, \texttt{FP}\}$ to express the dependency that is either $\Sigma_Q \Vdash \sigma_{X \rightarrow Y}$ or $\Sigma_Q \Vdash \sigma_{X \rightarrow^* Y}$ on $X, Y \lhd Q$.

Definition 15 (Constraint rewriting Γ). *Let $\mathcal{M}$ be a simple mapping defined over a relational schema $\mathcal{R}$ with source constraint Σ, and let ι and κ be the classification functions. Then, the constraint rewriting $\Gamma(\mathcal{M}, \Sigma)$ of Σ w.r.t. $\mathcal{M}$ is a set of SHACL shapes that for each RDF concept C with mapping $Q_X \longrightarrow \langle \mathbf{f}_C(X), \texttt{rdf:type}, C\rangle$, contains $\langle C, \tau_C, \phi_C\rangle$ with an implicit targetClass τ_C and conjunctive set of constraints $\phi_C = \bigwedge_{1 \le i \le 3} \Phi_i$, where*

1. *for mapping m of each object property P such as $Q_{X,Y} \longrightarrow \langle \mathbf{f}_C(X), P, \mathbf{f}_{C'}(Y)\rangle$,*

$$\Phi_1 = \begin{cases} (\le_0\ P.\neg C') \wedge (\ge_0\ P.C') \wedge (\bigwedge_{\Sigma_Q \Vdash \sigma} \lambda_1(\sigma)) & \text{if } \iota(head(m), \mathcal{M}) = A \\ (\le_0\ P.\neg C') \wedge (\ge_1\ P.C') \wedge (\bigwedge_{\Sigma_Q \Vdash \sigma} \lambda_2(\sigma)) & \text{if } \iota(head(m), \mathcal{M}) = B \end{cases}$$

where

$$\lambda_1(\sigma) = \begin{cases} (\trianglerighteq_C\ P.C') & \text{if } \sigma = \mathtt{UFD}_{X \rightarrow Y} \\ (=_1\ P.C') & \text{if } \sigma = \mathtt{FD}_{X \rightarrow Y} \\ (\triangleright_C\ P.C') & \text{if } \sigma = \mathtt{UF}_{X \twoheadrightarrow Y} \\ (\le_1\ P.C') & \text{if } \sigma = \mathtt{FP}_{X \twoheadrightarrow Y} \end{cases} \quad \text{and } \lambda_2(\sigma) = \begin{cases} (\trianglerighteq_C\ P.C') & \text{if } \sigma = \mathtt{UF}_{X \twoheadrightarrow Y} \\ (=_1\ P.C') & \text{if } \sigma = \mathtt{FP}_{X \twoheadrightarrow Y} \end{cases}$$

2. *for mapping m of each object property P such as $Q_{X,Y} \longrightarrow \langle \mathbf{f}_{C'}(X), P, \mathbf{f}_C(Y)\rangle$,*

$$\Phi_2 = \begin{cases} (\le_0\ P^-.\neg C') \wedge (\ge_0\ P^-.C') \wedge (\bigwedge_{\Sigma_Q \Vdash \sigma} \delta_1(\sigma)) & \text{if } \iota(head(m), \mathcal{M}) = A \\ (\le_0\ P^-.\neg C') \wedge (\ge_1\ P^-.C') \wedge (\bigwedge_{\Sigma_Q \Vdash \sigma} \delta_2(\sigma)) & \text{if } \iota(head(m), \mathcal{M}) = B \end{cases}$$

where

$$\delta_1(\sigma) = \begin{cases} (\trianglerighteq_C\ P^-.C') & \text{if } \sigma = \mathtt{UFD}_{X \rightarrow Y} \\ (=_1\ P^-.C') & \text{if } \sigma = \mathtt{FD}_{X \rightarrow Y} \\ (\triangleright_C\ P^-.C') & \text{if } \sigma = \mathtt{UF}_{X \twoheadrightarrow Y} \\ (\le_1\ P^-.C') & \text{if } \sigma = \mathtt{FP}_{X \twoheadrightarrow Y} \end{cases} \quad \text{and } \delta_2(\sigma) = \begin{cases} (\trianglerighteq_C\ P^-.C') & \text{if } \sigma = \mathtt{UF}_{X \twoheadrightarrow Y} \\ (=_1\ P^-.C') & \text{if } \sigma = \mathtt{FP}_{X \twoheadrightarrow Y} \end{cases}$$

3. *for mapping m of each datatype property U such as $Q_{X,Y} \longrightarrow \langle \mathbf{f}_C(X), U, \mathbf{t}(Y)\rangle$,*

$$\Phi_3 = \begin{cases} (\le_0\ U.\neg\mathbf{t}) \wedge (\ge_0\ U.\mathbf{t}) \wedge (\bigwedge_{\Sigma_Q \Vdash \sigma} \mu_1(\sigma)) & \text{if } \iota(head(m), \mathcal{M}) = A \\ (\le_0\ U.\neg\mathbf{t}) \wedge (\ge_1\ U.\mathbf{t}) \wedge (\bigwedge_{\Sigma_Q \Vdash \sigma} \mu_2(\sigma)) & \text{if } \iota(head(m), \mathcal{M}) = B \end{cases}$$

where

$$\mu_1(\sigma) = \begin{cases} (\trianglerighteq_C\ U.\mathbf{t}) & \text{if } \sigma = \mathtt{UFD}_{X \rightarrow Y} \\ (=_1\ U.\mathbf{t}) & \text{if } \sigma = \mathtt{FD}_{X \rightarrow Y} \\ (\triangleright_C\ U.\mathbf{t}) & \text{if } \sigma = \mathtt{UF}_{X \twoheadrightarrow Y} \\ (\le_1\ U.\mathbf{t}) & \text{if } \sigma = \mathtt{FP}_{X \twoheadrightarrow Y} \end{cases} \quad \text{and } \mu_2(\sigma) = \begin{cases} (\trianglerighteq_C\ U.\mathbf{t}) & \text{if } \sigma = \mathtt{UF}_{X \twoheadrightarrow Y} \\ (=_1\ U.\mathbf{t}) & \text{if } \sigma = \mathtt{FP}_{X \twoheadrightarrow Y} \end{cases}$$

Observe that in Definition 15, the first constraint components, such as $(\leq_0\ P^{\pm}.\neg C')$ and $(\leq_0\ U.\neg \mathbf{t})$ in the definitions of Φ_i, are implied by the restriction on the mapping set $\mathcal{M}$, i.e., by the fact that $\mathcal{M}$ contains exactly one mapping defining per object and datatype property predicates. The second constraint components, such as $(\geq_0\ P^{\pm}.\neg C')$ or $(\geq_1\ P^{\pm}.\neg C')$ and $(\geq_0\ U.\neg \mathbf{t})$ or $(\geq_1\ U.\neg \mathbf{t})$, in the Φ_i are implied by the combination of ι- and κ-classifications. Finally, the third constraints components $\bigwedge_{\Sigma_Q \Vdash \sigma} f(\sigma)$ s.t. $f \in \lambda_i \cup \delta_i \cup \mu_i$ for $1 \leq i \leq 2$ are implied by the source constraint Σ w.r.t. $\mathcal{M}$.

The constraint definition $\phi_C := (\leq_0\ P^{\pm}.\neg C')$ requires all nodes n' in the graph that are reachable from a node n s.t. $\langle n, \texttt{rdf:type}, C\rangle$ via property path $P^{\pm}$ to have a typing triple s.t. $\langle n', \texttt{rdf:type}, C'\rangle$, which is exactly what we needed for the mapped object property paths $P^{\pm}$ in the RDF graph given the restriction that set $\mathcal{M}$ contains exactly one mapping definitions per object property predicates. Thus, to extend the constraint rewriting Γ Definition in 15 beyond the simple mapping $\mathcal{M}$, the rewriting Γ must: (i) not generate constraint components such as $(\leq_0\ P^{\pm}.\neg C')$ and $(\leq_0\ U.\neg \mathbf{t})$ when there exist more than one mapping definition per object P and datatype U properties, respectively, in the set $\mathcal{M}$, (ii) accommodate classification of all possible combinations of sources queries in the definitions of ι and κ, and (iii) revise the definitions of λ_i, δ_i and μ_i for additional consequences of Σ-implications w.r.t. the extended $\mathcal{M}$.

We now state the properties of the constraint Γ rewriting. Theorem 1 is a soundness statement that guarantees that all constraints produced by Γ will be validated by the RDF graph mapped from any valid database instance.

Theorem 1. *The constraint rewriting Γ is semantics preserving.*

Theorem 2 expresses the completeness of Γ, i.e., every SHACL constraint expressible with the schema $sch(\mathcal{M})$ of the mappings, and that is implied by Σ is implied by the generated shapes $\Gamma(\mathcal{M}, \Sigma)$. This does not hold in general for SHACL constraints on predicates not in $sch(\mathcal{M})$. Finally, Theorem 3 expresses that adding mappings will never invalidate generated constraints.

Theorem 2. *The constraint rewriting Γ is maximal semantics preserving.*

Theorem 3. *The constraint rewriting Γ is monotone.*

7 Discussion

We have presented a constraint rewriting Γ for simple R2R mapping that is useful in the context of relational to RDF data transformation [13,19,23] and data integration [22, 31]. Observe that simple R2R mappings can express a comprehensive catalog of useful mapping patterns studied in [8,24,25]. Simplifying simple R2R mapping further yields direct mapping [2] since that requires additional restrictions on Definition 4; therefore, the results for our constraint rewriting for simple mappings also seamlessly holds for direct mapping [2,23,27]. In future work, we believe that it would interesting to extend our constraint rewriting Γ in two different directions: (a) for arbitrary R2R mappings, e.g., admitting the full relational algebra or arbitrary `SPJ` expressions as the source query in mapping Definition 3, and (b) for a broader class of relational constraints such as (disjunctive) tuple and equality generating dependencies [1].

There are several approaches that map relational schemas and constraints to RDFS and OWL/Epistemic DL axioms since, with an appropriate closed world semantics, OWL can express integrity constraints. In particular, we first refer the reader to the implications of constraints in ontology-based data access platform under different names, such as protection and faithfulness in [10,11], which is equivalent to relational constraints-to-OWL, i.e., to check whether the mapped RDF of every source dataset satisfying the source constraints can be extended to a model of the mapped $\texttt{DL-Lite}_{\texttt{A}}$ axioms, and OWL-to-relational constraints, i.e., opposite of former, constraints implication in [21]. Even though these proposals for combining OWL/Epistemic DL axioms with integrity constraints have some promising results for target constraints specification in the OBDA setting, there has been no unanimity on the correct semantics.

The problem of direct mapping of source schemas and constraints into RDFS/OWL axioms has been studied in [7,23]. Sequeda et al. [23] attempted to capture the database constraints on the RDF graph resulting from direct mapping using OWL. However, the bootstrapped OWL axioms did not trigger the unsatisfiability of the directly mapped graph whenever keys are violated in the source database unless the database instance is explicitly encoded in the constraint rewriting. Further, Sequeda et al. [23, Theorem 3] established that the desirable monotonicity property of direct mapping is an obstacle to obtain a semantics preserving OWL axioms even if the database instance is explicitly encoded in the constraint rewriting. To accomplish the desired one-to-one semantics correspondence between legal relational data and RDF graph satisfying OWL axioms, Calvanese et al. [7] further extended the direct mapping of relational schemas into $\texttt{DL-Lite}_{\texttt{RDFS}}$ with disjointness - as constraints over mapped RDF graphs.

Finally, Thapa et al. [26] have studied the problem of translating database constraints into SHACL, instead of OWL/Epistemic DL, giving a direct transformation from SQL constraints to SHACL, preserving their semantics when source key constraints are satisfied [26, Theorem 2]. The present work improves on this by a) not being restricted to direct mappings, and b) lifting the requirement on satisfied key constraints.

8 Conclusion

In this paper, we study the problem of constraint rewriting for relational to RDF data transformation based on the central property of maximal semantics preservation. We translate standard SQL database constraints to shapes in the SHACL constraint language for RDF graphs. We show that our proposed rewriting Γ for the simple relational to RDF mappings satisfies the central properties of a constraint rewriting.

We believe that the propose constraint rewriting constitutes a core component of R2R mapping tools for the crucial task of constructing and maintaining a quality-assured RDF graph with SHACL constraints. The SHACL description of the generated RDF graph provides a data quality guarantee for data exchange, interoperability and query optimization. Hence, an important direction for future work will be the implementation and practical evaluation of our rewriting for relational to RDF data transformation and query optimization [30] in an ontology-based data access platform [29,31].

Acknowledgements. This work is supported by the Norwegian Research Council via the SIRIUS SFI (237898). We thank Egor Kostylev for many constructive suggestions.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases, vol. 8. Addison-Wesley, Reading (1995)
2. Arenas, M., Bertails, A., Prud'hommeaux, E., Sequeda, J.: A direct mapping of relational data to RDF. W3C Recommendation **27**, 1–11 (2012)
3. Armstrong, W.W.: Dependency structures of data base relationships. In: IFIP Congress, vol. 74, pp. 580–583. Geneva, Switzerland (1974)
4. Badia, A., Lemire, D.: Functional dependencies with null markers. Comput. J. **58**(5), 1160–1168 (2015)
5. Beeri, C., Fagin, R., Howard, J.H.: A complete axiomatization for functional and multivalued dependencies in database relations. In: Proceedings of the 1977 ACM SIGMOD International Conference on Management of Data, pp. 47–61 (1977)
6. Calvanese, D.: Ontologies and databases: the *DL-Lite* approach. In: Tessaris, S., et al. (eds.) Reasoning Web 2009. LNCS, vol. 5689, pp. 255–356. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03754-2_7
7. Calvanese, D., Fischl, W., Pichler, R., Sallinger, E., Simkus, M.: Capturing relational schemas and functional dependencies in RDFS. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 28 (2014)
8. Calvanese, D., Gal, A., Lanti, D., Montali, M., Mosca, A., Shraga, R.: Mapping patterns for virtual knowledge graphs. arXiv preprint arXiv:2012.01917 (2020)
9. Civili, C., Mora, J., Rosati, R., Ruzzi, M., Santarelli, V.: Semantic analysis of R2RML mappings for ontology-based data access. In: Ortiz, M., Schlobach, S. (eds.) RR 2016. LNCS, vol. 9898, pp. 25–38. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45276-0_3
10. Console, M., Lenzerini, M.: Data quality in ontology-based data access: the case of consistency. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 28 (2014)
11. Console, M., Lenzerini, M.: Epistemic integrity constraints for ontology-based data management. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 2790–2797 (2020)
12. Corman, J., Reutter, J.L., Savković, O.: Semantics and validation of recursive SHACL. In: Vrandečić, D., Bontcheva, K., Suárez-Figueroa, M.C., Presutti, V., Celino, I., Sabou, M., Kaffee, L.-A., Simperl, E. (eds.) ISWC 2018. LNCS, vol. 11136, pp. 318–336. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00671-6_19
13. Das, S., Sundara, S., Cyganiak, R.: R2RML: RDB to RDF mapping language, September 2012. http://www.w3.org/TR/2012/REC-r2rml-20120927/
14. Donini, F.M., Nardi, D., Rosati, R.: Description logics of minimal knowledge and negation as failure. ACM Trans. Comput. Logic (ToCL) **3**(2), 177–225 (2002)
15. Knublauch, H., Kontokostas, D.: Shapes constraint language (SHACL). W3C recommendation, W3C, July 2017. http://www.w3.org/TR/2017/REC-shacl-20170720/
16. Lembo, D., Mora, J., Rosati, R., Savo, D.F., Thorstensen, E.: Mapping analysis in ontology-based data access: algorithms and complexity. In: Arenas, M., Corcho, O., Simperl, E., Strohmaier, M., d'Aquin, M., Srinivas, K., Groth, P., Dumontier, M., Heflin, J., Thirunarayan, K., Staab, S. (eds.) ISWC 2015. LNCS, vol. 9366, pp. 217–234. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25007-6_13
17. Manola, F., Miller, E., McBride, B., et al.: RDF primer. W3C Recommendation **10**(1–107), 6 (2004)
18. Mecca, G., Rull, G., Santoro, D., Teniente, E.: Ontology-based mappings. Data Knowl. Eng. **98**, 8–29 (2015)

19. De Medeiros, L.F., Priyatna, F., Corcho, O.: MIRROR: automatic R2RML mapping generation from relational databases. In: Cimiano, P., Frasincar, F., Houben, G.-J., Schwabe, D. (eds.) ICWE 2015. LNCS, vol. 9114, pp. 326–343. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19890-3_21
20. Motik, B., Horrocks, I., Sattler, U.: Bridging the gap between OWL and relational databases. J. Web Semant. **7**(2), 74–89 (2009)
21. Nikolaou, C., Grau, B.C., Kostylev, E.V., Kaminski, M., Horrocks, I.: Satisfaction and Implication of Integrity Constraints in Ontology-based Data Access. In: IJCAI, pp. 1829–1835 (2019)
22. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. In: Spaccapietra, S. (ed.) Journal on Data Semantics X. LNCS, vol. 4900, pp. 133–173. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-77688-8_5
23. Sequeda, J.F., Arenas, M., Miranker, D.P.: On directly mapping relational databases to RDF and OWL. In: Proceedings of the 21st International Conference on World Wide Web, pp. 649–658 (2012)
24. Sequeda, J.F., Miranker, D.P.: Ultrawrap mapper: a semi-automatic relational database to RDF (RDB2RDF) mapping tool. In: International semantic web conference (posters & demos) (2015)
25. Juan, F., Sequeda, F.P., Villazón-Terrazas, B.: Relational database to RDF mapping patterns. In: WOP (2012)
26. Thapa, R.B., Giese, M.: A source-to-target constraint rewriting for direct mapping. In: Hotho, A., Blomqvist, E., Dietze, S., Fokoue, A., Ding, Y., Barnaghi, P., Haller, A., Dragoni, M., Alani, H. (eds.) ISWC 2021. LNCS, vol. 12922, pp. 21–38. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-88361-4_2
27. Thapa, R.B., Giese, M.: A source-to-target constraint rewriting for direct mapping (extended version). Research Report 498, Dept. of Informatics, University of Oslo, September 2021. http://www.urn.nb.no/URN:NBN:no-90764
28. Thapa, R.B., Giese, M.: Mapping relational database constraints to SHACL (extended version). Research Report 503, Dept. of Informatics, University of Oslo, July 2022. http://www.urn.nb.no/URN:NBN:no-35645
29. Xiao, G., et al.: A survey. IJCAI Organization, Ontology-based data access (2018)
30. Xiao, G., Kontchakov, R., Cogrel, B., Calvanese, D., Botoeva, E.: Efficient handling of SPARQL OPTIONAL for OBDA. In: Vrandečić, D., Bontcheva, K., Suárez-Figueroa, M.C., Presutti, V., Celino, I., Sabou, M., Kaffee, L.-A., Simperl, E. (eds.) ISWC 2018. LNCS, vol. 11136, pp. 354–373. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00671-6_21
31. Xiao, G., Lanti, D., Kontchakov, R., Komla-Ebri, S., Güzel-Kalaycı, E., Ding, L., Corman, J., Cogrel, B., Calvanese, D., Botoeva, E.: The virtual knowledge graph system ontop. In: Pan, J.Z., Tamma, V., d'Amato, C., Janowicz, K., Fu, B., Polleres, A., Seneviratne, O., Kagal, L. (eds.) ISWC 2020. LNCS, vol. 12507, pp. 259–277. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-62466-8_17