

A Source-to-Target Constraints Rewriting for Direct Mapping

Ratan B. Thapa and Martin Giese

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

Abstract. Most of the existing structured digital information today is still stored in relational databases. That’s why it is important for the Semantic Web effort to expose the information in relational databases as RDF, or allow to query it using SPARQL. ‘Direct mapping’ is a fully automated approach for converting well-structured relational data to RDF that does not require formulating explicit mapping rules [2,11]. Along with the mapped RDF data, it is desirable to have a description of that data. Previous work [11,3] has attempted to describe the RDF in terms of OWL axioms, which is problematic, partly due to the open world semantics of OWL. In the present paper, we present a source-to-target semantics preserving rewriting of constraints in an SQL database schema to equivalent SHACL [9] constraints. This means that we can provide a SHACL description of the RDF generated by direct mappings without the need to perform a costly validation of those constraints on the generated data. We define the rewriting from SQL constraints to SHACL by a set of Datalog rules, and prove central properties of the transformation.

1 Introduction

Relational *constraints*, a.k.a. *data dependencies* or *integrity constraints* in relational database theory [1], have traditionally been used to restrict the data in the database to those considered meaningful to the application at hand. Constraints are stated when a relational schema is defined and checked when the stored data are modified. When relational data is mapped into RDF [5], using Direct Mapping [2] or R2RML [6], the constraints on the original relational data imply certain constraints on the RDF, but existing tools do not make these constraints explicit, and the theory behind such constraints on the output of the mapping is not well explored. However, the integrity of the data that is being stored or represented in the RDF graph is a critical piece of information in practice, both to detect problems in the RDF dataset and provide data quality guarantees for the purpose of RDF data exchange and interoperability.

Previous work has attempted to capture the properties of the RDF graph resulting from direct mapping in *OWL axioms* [11] or as *DL-Lite_{RDFS}* axioms¹ with *identification constraints* [3]. However, as Sequeda et al. [11, Theorem 3] established, OWL axioms alone cannot provide a mapping that has both of the desirable properties of being monotone and semantics preserving. This is due to (1) DL semantics following the Open

¹ *DL-Lite_{RDFS}* is a variant of *DL-Lite_A* [10]

World Assumption (OWA), and (2) OWL not adopting the Unique Name Assumption (UNA). In our work, we transform integrity constraints on the source data into integrity constraints on the RDF, expressed in SHACL, the Shapes Constraint Language [9]. SHACL documents are collections of shapes which define a set of constraints and specify which nodes in a graph should be validated against these constraints. SHACL has a closed world semantics and is based on the unique name assumption, which makes it more suitable for expressing integrity constraints on RDF than OWL. Our mapping from SQL instance data to RDF, and from a relational schema to RDFS axioms is the same as the *direct mapping* described by Sequeda et al [11] in terms of Datalog rules. Our generation of SHACL constraints is also expressed in Datalog, building on the same vocabulary. Our transformation takes all explicit and implicit SQL domain constraints into account, which constrain the value and scope of attributes in a relation schema definition such as data types, nullability and uniqueness constraints.

Under certain reasonable assumptions (all relations have a primary key, and database instances satisfy their primary and foreign key constraints), our proposed SHACL constraints rewriting for direct mapping is constraints preserving and semantic preserving. Under these assumptions, our rewriting therefore preserves the good properties of the previous direct mapping [11].

In Sect. 2, we review central notions of relational databases, RDF, SHACL and Direct mapping. Sect. 3 introduces the central notion of SHACL constraints rewriting and the properties: constraints preservation and semantic preservation. In Sect. 4, we give the datalog rules for the proposed rewriting. Sect. 5 states the properties of the proposed rewriting.² Sect. 6 discusses shortcoming of the rewriting, and Sect. 7 concludes the paper.

2 Preliminaries

In this section, we fix notions and notations fundamental to the definition of direct mappings [11], and SHACL constraints [9].

Databases. Let Δ_c be a countably infinite set of constants, not including null. The value null is used to express ‘no information’. A *relational schema* $\mathcal{R}$ is a finite set of *relation schemas*. We regard each relation schema R as a fixed *set of named attributes*, and $\text{att}(R)$ denotes a nonempty finite set of attribute names associated to relation schema R . An *instance* $\mathcal{D}$ of $\mathcal{R}$ assigns each relation schema R in $\mathcal{R}$ a finite set of tuples $R^I = \{t_1 \dots t_n\}$. Each *tuple* $t \in R^I$ of R is a function that assigns to each attribute in $\text{att}(R)$ a value from $\Delta_c \cup \{\text{null}\}$. We write $t(X)$ to denote the value of tuple t restricted to an attribute $X \in \text{att}(R)$, which can be extended to a set $\emptyset \subsetneq \{X_1 \dots X_n\} \subseteq \text{att}(R)$ of attributes. Finally, we regard a *relational database* as a pair $(\mathcal{R}, \mathcal{D})$, where $\mathcal{R}$ is a relational schema and $\mathcal{D}$ is an instance of $\mathcal{R}$.

² Proofs are detailed in the appendix of the extended version at <https://www.mn.uio.no/ifi/english/people/aca/martingi/dm-shacl-long.pdf>, which will be published as a tech report, if this is accepted for ESWC.

SQL constraints. Assume $\mathcal{R}$ is a relational schema as in the SQL Data Definition Language (DDL), and Σ is a set of *SQL constraints* on $\mathcal{R}$. We consider two types of SQL constraints on relational schemas for the source constraints: *SQL domain constraints*, denoted with δ , and *SQL key constraints*, denoted with σ . The SQL domain constraints are *not null* (NN) and *unique* (UNQ) entity integrity constraints including *SQL data types*, and SQL Key constraints are *primary key* (PK) and *foreign key* (FK) constraints. Each attribute in a relation schema is required to have an SQL data type. SQL data types are the domain descriptions of attributes in a relation schema that define what type of value can be expected for the attributes in an instance of the relation schema. The NN, UNQ, and PK constraints over $\mathcal{R}$ are expressions of the form $\text{NN}(X_1 \dots X_n, R)$, $\text{UNQ}(X_1 \dots X_n, R)$ and $\text{PK}(X_1 \dots X_n, R)$, respectively, for any $\emptyset \subsetneq \{X_1 \dots X_n\} \subseteq \text{att}(R)$ such that $R \in \mathcal{R}$. An instance $\mathcal{D}$ of $\mathcal{R}$ satisfies $\text{NN}(X_1 \dots X_n, R)$, denoted by $\mathcal{D} \models \text{NN}(X_1 \dots X_n, R)$, if for every $t \in R^I$ and $k \in \{1, \dots, n\}$, it holds that $t(X_k) \neq \text{null}$. An instance $\mathcal{D}$ of $\mathcal{R}$ satisfies $\text{UNQ}(X_1 \dots X_n, R)$, denoted by $\mathcal{D} \models \text{UNQ}(X_1 \dots X_n, R)$, if for every $t, t' \in R^I$ and $k \in \{1, \dots, n\}$, if $t(X_k) = t'(X_k) \neq \text{null}$ then $t = t'$. An instance $\mathcal{D}$ of $\mathcal{R}$ satisfies $\text{PK}(X_1 \dots X_n, R)$, denoted by $\mathcal{D} \models \text{PK}(X_1 \dots X_n, R)$, if: (1) for every $t \in R^I$ and $k \in \{1, \dots, n\}$, it holds that $t(X_k) \neq \text{null}$, and (2) for every $t, t' \in R^I$ and $k \in \{1, \dots, n\}$, if $t(X_k) = t'(X_k)$ then $t = t'$. An FK key constraint over $\mathcal{R}$ is an expression of the form $\text{FK}(X_1 \dots X_n, R, Y_1 \dots Y_n, T)$ for any $\emptyset \subsetneq \{X_1 \dots X_n\} \subseteq \text{att}(R)$ and $\emptyset \subsetneq \{Y_1 \dots Y_n\} \subseteq \text{att}(T)$ such that $R, T \in \mathcal{R}$. An instance $\mathcal{D}$ of $\mathcal{R}$ satisfies $\text{FK}(X_1 \dots X_n, R, Y_1 \dots Y_n, T)$, denoted by $\mathcal{D} \models \text{FK}(X_1 \dots X_n, R, Y_1 \dots Y_n, T)$, if for every $t \in R^I$: either (1) there exists $k \in \{1, \dots, n\}$ such that $t(X_k) = \text{null}$, or (2) there exists a tuple $t' \in T^I$ such that $t(X_k) = t'(Y_k)$ for every $k \in \{1, \dots, n\}$.

Given a relational schema $\mathcal{R}$, we assume that: (1) every relation schema R in $\mathcal{R}$ must have exactly one PK key, *i.e.*, if $\text{PK}(X_1 \dots X_n, R)$ then there exists no $\text{PK}(Y_1 \dots Y_n, R)$ such that $\emptyset \subsetneq \{X_1 \dots X_n\}, \{Y_1 \dots Y_n\} \subseteq \text{att}(R)$, as is usual in all SQL implementations, (2) there exists no SQL key to be both primary and foreign key, *i.e.* if $\text{PK}(X_1 \dots X_n, R)$, then there exists no $\text{FK}(X_1 \dots X_n, R, Y_1 \dots Y_n, T)$ for any $\emptyset \subsetneq X_1 \dots X_n \subseteq \text{att}(R)$ and $\emptyset \subsetneq Y_1 \dots Y_n \subseteq \text{att}(T)$ such that $T \in \mathcal{R}$, and vice-versa. Further, we point out a couple of implications between SQL constraints. If we write $C \sqsubseteq C'$ for any $C, C' \in \Sigma$ to denote that for any instance $\mathcal{D}$ of $\mathcal{R}$, if $\mathcal{D} \models C(X_1 \dots X_n, R)$ then $\mathcal{D} \models C'(X_1 \dots X_n, R)$ for any non-empty set of attributes $\emptyset \subsetneq \{X_1 \dots X_n\} \subseteq \text{att}(R)$, then we have the following:

$$\text{PK} \sqsubseteq \text{NN}, \quad \text{and} \quad \text{PK} \sqsubseteq \text{UNQ}.$$

Finally, we say that $\mathcal{D}$ is a legal instance of $\mathcal{R}$, denoted by $\mathcal{D} \models \Sigma$, if $\mathcal{D}$ satisfies all SQL constraints Σ on $\mathcal{R}$.

Example 1 *Throughout the paper, we use the following running example. Consider a relational database consisting of relation schemas for the people employed, the beacon projects, and the research programs run by the SIRIUS centre for scalable data access in the oil and gas industry:*

```

CREATE TABLE EMP { E_ID INTEGER PRIMARY KEY, NAME VARCHAR NOT NULL, Post VARCHAR },
CREATE TABLE PROJ { B_ID INTEGER PRIMARY KEY, NAME VARCHAR UNIQUE },
CREATE TABLE PROG{ P_ID INTEGER PRIMARY KEY, NAME VARCHAR NOT NULL, ToPROJ INTEGER
NOT NULL UNIQUE FOREIGN KEY REFERENCES PROJ(B_ID)}, and
CREATE TABLE ENV { ToEMP INTEGER FOREIGN KEY REFERENCES EMP(E_ID), ToPROG INTEGER
FOREIGN KEY REFERENCES PROG(P_ID), CONSTRAINT (ToEMP,ToPROG) PRIMARY KEY }.

```

RDF Graph. 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 IRIs $\mathcal{I}$ are *global identifiers* that support easy integration of RDF triples, whereas blank nodes $\mathcal{B}$ are *local identifiers* of nodes in an RDF graph. The literals $\mathcal{L}$ are constants for RDF data type values. An *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 $o \in \mathcal{I} \cup \mathcal{B} \cup \mathcal{L}$ is called the object. An RDF graph $G \subset (\mathcal{I} \cup \mathcal{B}) \times \mathcal{I} \times (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L})$ is a finite set of RDF triples.

SHACL Constraints. SHACL is a language for validating RDF graphs against a set of conditions with the UNA, *i.e.*, two blank nodes in an RDF graph that is supposed to be validated against the SHACL constraints cannot denote the same individual. For this paper, we use the *core constraints component* of SHACL [9] recommended by the W3C standard. A SHACL document is a *set of shapes*, called *shape graph*, and the RDF graphs that are validated against a shape graph are called *data graph*. Each *shape* in a shape graph, known as *node shape*, consists of declarations of *shape name*, *target*, and the *constraint* component: a set of so-called *property shapes* which are used in conjunction. The target declaration is a SPARQL query [7] with one output variable whose propose is to retrieve target nodes of the shape from the data graph, *i.e.*, entities occurring in the data graph for which the constraints component of shape should be verified. W.l.o.g, we assume that there exists no shape in our shape graph that references another shape in its constraints declaration. Given a shape s in a shape graph $\mathcal{S}$, and a node v in a data graph G satisfying the target definition of s , v validates against s iff v satisfies all the constraints, *i.e.*, property shapes, of s . A data graph G validates against a shape graph $\mathcal{S}$, we formally write $G \models \mathcal{S}$, iff G is valid against all the shapes in $\mathcal{S}$. In addition, note that we use a specific constraint³ from the DASH extension to the SHACL core constraints.

Example 2 A SHACL node shape (left) :Employee with implicit class target, meaning that all members of the :Employee class are target nodes of the node shape, defines the constraints that all users must have a unique ID whose datatype must be xsd:integer. An instance of :Employee satisfies the node shape if it conforms to the property shape, *i.e.*, 'sh:property' A graph that validates the declared node shape is provided in the right, which can be made invalid by changing ID of 'Julie' to "002" ^xsd:integer.

³ dash:uniqueValueForClass

```

:Employee a sh:NodeShape, rdfs:Class;
sh:property [ sh:path :hasID;
               sh:nodeKind sh:Literal;
               sh:maxCount 1; sh:minCount 1;
               sh:datatype xsd:integer;
               dash:uniqueValueForClass :Employee ];
:Julie a :Employee;
:hasID "001"^^xsd:integer .
:Magnus a :Employee;
:hasID "002"^^xsd:integer .

```

Direct Mapping. We now briefly comment on the direct mapping [11], defined in terms of a set $\mathcal{M}$ of datalog rules. The input of $\mathcal{M}$ is encoded as a set $\langle \mathcal{R}, \sigma, \mathcal{D} \rangle$ of relations such that $\mathcal{R}$ is a relational schema, σ is a set of PKs and FKs on $\mathcal{R}$ and $\mathcal{D}$ is an instance of $\mathcal{R}$, the output is an RDF graph with OWL axioms. Since we do not consider OWL axioms in our work, we simply ignore the rules [11, Section 4.3.2] used to generate OWL axioms from original work [11]. The remaining parts of the mapping rules $\mathcal{M}$ are further partitioned into two groups: the set of rules $\mathcal{M}_{\mathcal{R}}$ [11, Section 4.2 and 4.3.1,] that transforms the relational schema $\mathcal{R}$ into an RDF schema $\mathcal{G}_{\mathcal{R}}$, and the set $\mathcal{M}_{\mathcal{D}}$ [11, Section 4.4] that transforms an instance $\mathcal{D}$ of $\mathcal{R}$ into an RDF graph $\mathcal{G}$. Hence, w.l.o.g, given an input tuple $\langle \mathcal{R}, \sigma, \mathcal{D} \rangle$ of mapping $\mathcal{M}_{\mathcal{R}} \cup \mathcal{M}_{\mathcal{D}}$ the output is a directly mapped RDF graph $\mathcal{G} = \mathcal{M}_{\mathcal{D}}(\mathcal{G}_{\mathcal{R}}, \sigma, \mathcal{D})$ of RDF schema $\mathcal{G}_{\mathcal{R}}$, σ and $\mathcal{D}$, where $\mathcal{G}_{\mathcal{R}}$ is the directly mapped RDF schema $\mathcal{M}_{\mathcal{R}}(\mathcal{R}, \sigma)$ of $\mathcal{R}$ and σ .

Finally, we regard the set $\mathcal{M}_{\mathcal{R}} \cup \mathcal{M}_{\mathcal{D}}$ of mapping rules as direct mapping, denoted with $\mathbb{M}$, for the extension of direct mapping with SHACL constraints. We note that some of the $\mathcal{M}_{\mathcal{D}}$ rules specified in [11, Section 4.4.1] generate *blank nodes* for the tuples of relation schema not having a primary key. However, the mappings rule for generating such blank nodes is not applicable in our setting since we have assumed that all the input relation schemas of $\mathbb{M}$ must have a PK key.

3 SHACL Constraints Rewriting for Direct Mapping: Definitions

A source-to-target SHACL constraints rewriting for direct mapping translates SQL constraints of a relational schema into SHACL constraints in such a way that they are satisfied by the RDF data generated by the direct mapping. The input of a SHACL constraints rewriting for direct mapping is a directly mapped RDF schema $\mathcal{G}_{\mathcal{R}}$ of relational schema $\mathcal{R}$ and a set δ of SQL domain constraints on $\mathcal{R}$, and the output is a set of SHACL constraints, known as shapes.

Assume that $\mathbb{S}$ is the set of all SHACL shapes and $\mathbb{G}$ is the set of all pairs of the form $(\mathcal{G}_{\mathcal{R}}, \delta)$ such that $\mathcal{G}_{\mathcal{R}}$ is the directly mapped RDF schema of relational schema $\mathcal{R}$ and δ is a set of SQL domain constraints on $\mathcal{R}$. Note that $\mathbb{M}$ is the set of direct mapping rules $\mathcal{M}_{\mathcal{R}} \cup \mathcal{M}_{\mathcal{D}}$ from [11] that do not generate blank nodes in our setting.

Definition 1 (Directly mapped RDF graph) *Let σ be a set of PKs and FKs on relational schema $\mathcal{R}$ and $\mathcal{D}$ an instance of relational schema $\mathcal{R}$, and let $\mathbb{M}$ be the direct mapping. Then, $\mathcal{G}$ is the directly mapped RDF graph $\mathcal{M}_{\mathcal{D}}(\mathcal{G}_{\mathcal{R}}, \sigma, \mathcal{D})$ of a relational database $(\mathcal{R}, \mathcal{D})$ with σ , where $\mathcal{G}_{\mathcal{R}}$ is the RDF schema $\mathcal{M}_{\mathcal{R}}(\mathcal{R}, \sigma)$.*

Definition 2 (SHACL constraints rewriting) *A SHACL constraints rewriting $\mathcal{T}$ for direct mapping $\mathbb{M}$ is a function $\mathcal{T} : \mathbb{G} \rightarrow \mathcal{P}(\mathbb{S})$ that maps any directly mapped RDF*

schema $\mathcal{G}_R = M_R(\mathcal{R}, \sigma)$ of any relational schema $\mathcal{R}$ and set δ of SQL domain constraints on $\mathcal{R}$ to a set of SHACL shapes $\mathcal{S} = \mathcal{T}(\mathcal{G}_R, \delta) \subseteq \mathbb{S}$.

We note that all the good properties of the original work [11] still hold for the direct mapping $\mathbb{M}$, except the semantic preservation property [11, Definition 5] since we ignored the rules to generate OWL axioms. The *monotonicity* [11, Definition 4] property of direct mapping $\mathbb{M}$ assures that the re-computation of entire database-to-RDF graph is not required when the database is updated after the mapping. Here, it is straightforward to see that a SHACL constraint rewriting for direct mapping $\mathbb{M}$ in Definition 2 is independent of the database instance, hence, any updates in the database instance do not influence the rewriting of SHACL constraints for the $\mathbb{M}$ mapping. Therefore, when we state the additional properties of a SHACL constraints rewriting $\mathcal{T}$ for $\mathbb{M}$ that generates a desired one-to-one correspondence between relational databases and the RDF graphs with the SHACL constraints, we keep all the good properties of the $\mathbb{M}$ intact and undisturbed. Also note that two fundamental properties: information preservation [11, Theorem 1] and query preservation [11, Theorem 2], of direct mapping $\mathbb{M}$ are incomparable in a sense that the information preservation does not guarantee the query answering since the translated RDF data might require more complex SPARQL queries as compared to the SQL query over the relational database, and the query preservation does not guarantee the information preservation if relational data being translated is not properly stored.

With the definitions of directly mapped RDF graph and SHACL constraints rewriting for direct mapping $\mathbb{M}$ in place, we are now ready to introduce two fundamental properties of SHACL constraints rewriting $\mathcal{T}$ for direct mapping $\mathbb{M}$: *constraints preservation* and *semantic preservation*. A SHACL constraint rewriting for direct mapping is constraints preserving if it does not lose any SQL δ domain constraints on relational schema that is being translated into the SHACL constraints. However, a SHACL constraint rewriting for direct mapping is semantic preserving if (and only if) SHACL constraints rewriting preserves the semantics of SQL domains δ , including the semantics of PKs and FKs keys σ , constraints on input database to the translated RDF graph. The main force behind this requirement for the semantic preservation is that δ is the set of all explicit and implicit SQL domain constraints of the relational schema, *i.e.*, including all the ‘NN’ and ‘UNQ’ domain constraints implied by the PKs, and the domain constraints on the FKs (*i.e.*, on the attributes participating in FKs) of relation schemas.

Definition 3 (Constraints preservation) A SHACL constraints rewriting $\mathcal{T}$ for direct mapping $\mathbb{M}$ is constraints preserving if there is a computable mapping $N : \mathcal{P}(\mathbb{S}) \rightarrow \mathbb{G}$ such that for the directly mapped RDF schema $\mathcal{G}_R = M_R(\mathcal{R}, \sigma)$ of any relational schema $\mathcal{R}$ and set δ of SQL domain constraints on $\mathcal{R}$, where σ is the set of PKs and FKs on $\mathcal{R}$: $N(\mathcal{T}(\mathcal{G}_R, \delta)) = (\mathcal{G}_R, \delta)$.

Definition 4 (Semantic preservation) A SHACL constraints rewriting $\mathcal{T}$ for direct mapping $\mathbb{M}$ is semantic preserving if for any relational schema $\mathcal{R}$, set Σ of SQL domain δ and key σ constraints on $\mathcal{R}$, and an arbitrary instance $\mathcal{D}$ of $\mathcal{R}$:

$$\mathcal{D} \models \Sigma \iff \mathcal{G} \models S,$$

where $\mathcal{G}$ is the directly mapped RDF graph, and S is the set of SHACL constraints $\mathcal{T}(\mathcal{G}_R, \delta)$.

The direct mapping $\mathfrak{M}$ relies on primary keys to generate IRIs, and on foreign key references for object properties. The data mapping does not work if these constraints are violated. Sequeda et al. circumvent this problem in [11, Sect. 6.1] by taking the database instance as an extra argument of the rewriting, and generating an unsatisfiable axiom if constraints are violated. We prefer restricting the notion of semantic preservation to database instances that satisfy the key constraints:

Definition 5 (Weak semantic preservation) *A SHACL constraints rewriting $\mathcal{T}$ for direct mapping $\mathfrak{M}$ is weakly semantic preserving if for any relational schema $\mathcal{R}$, set Σ of SQL domain constraints δ and key constraints σ on $\mathcal{R}$, and an arbitrary instance $\mathcal{D}_\sigma$ of $\mathcal{R}$ that satisfies the set σ of primary and foreign keys in Σ :*

$$\mathcal{D}_\sigma \models \Sigma \iff \mathcal{G} \models \mathcal{S},$$

where $\mathcal{G}$ is the directly mapped RDF graph $\mathcal{M}_\mathcal{D}(\mathcal{G}_\mathcal{R}, \sigma, \mathcal{D}_\sigma)$ from Definition 1, and $\mathcal{S}$ is the set of SHACL constraints $\mathcal{T}(\mathcal{G}_\mathcal{R}, \delta)$.

There is a partial correspondence between the notions of constraints preservation and semantic preservation. The main force behind this is the semantic similarity between the SQL domain constraints in the relational database and the SHACL constraints over the RDF graph: *Closed World Assumptions* (CWA) and UNA *i.e.*, what causes an inconsistency in a relational database w.r.t. the SQL domain constraints, can also cause (partially) inconsistency in an RDF graph w.r.t. the SHACL constraints. For instance, if a SHACL constraints rewriting for direct mapping is semantic preserving, then it must be constraints preserving from the set δ of SQL constraints on a relational database to the translated the SHACL constraints on RDF graph. It is straightforward to see that the loss of any SQL domain constraints of the relational schema that is being translated into the SHACL constraints might always generate a consistent RDF graph even if the database under direct mapping is inconsistent, hence, a SHACL constraints rewriting cannot be semantics preserving without being constraints preserving. However, the opposite is not true, the constraints preservation does not guarantee the semantic preservation property of SHACL constraints rewriting for direct mapping in our SHACL constraints rewriting setting.

4 The SHACL Constraints Rewriting for Direct Mapping

We now introduce a SHACL constraints rewriting Γ for direct mapping $\mathfrak{M}$, defined in terms of a set of datalog rules [11], that extends the functionalities of $\mathfrak{M}$ with SHACL shapes. From Definition 2, the SHACL constraints rewriting Γ for $\mathfrak{M}$ translates the set δ of SQL domain constraints for the relational schema $\mathcal{R}$ into SHACL shapes that fit the RDF schema $\mathcal{G}_\mathcal{R}$ generated by the set of mapping $\mathcal{M}_\mathcal{R} \subset \mathfrak{M}$ rules. The set $\mathcal{M}_\mathcal{R}$ of rules generate $\mathcal{G}_\mathcal{R}$ in terms of the IRI identifiers in the form of datalog facts, we refer to Appendix A.1 for details, whereas for the SHACL constraints rewriting Γ we introduce a set of Datalog rules in Section 4.2. The rules for Γ start from the datalog facts $\mathcal{G}_\mathcal{R}$ produced by the set $\mathcal{M}_\mathcal{R} \subset \mathfrak{M}$ rules.

4.1 Datalog predicates

Predicates for relational schema. We use the following set of predicates to represent relational schema $\mathcal{R}$ and set Σ of SQL domain and key constraints on $\mathcal{R}$, where $\sigma \subseteq \Sigma$ is the set of PKs and FKs on $\mathcal{R}$. Hence, by restricting Σ to the σ on $\mathcal{R}$, the set $(\mathcal{R}, \sigma)$ of relations can be retrieved – an input to the direct mapping $\mathfrak{M}$. We reuse the representation from [11, section 4.1] with minor changes, but additionally introduce encoding for non-null and uniqueness SQL constraints.

1. $\text{REL}(R)$: Indicates R is a *relation schema* in $\mathcal{R}$.
2. $\text{ATTR}_n(X_1 \dots X_n, R)$: Indicates $\{X_1 \dots X_n\}$ are $\emptyset \subsetneq \{X_1 \dots X_n\} \subseteq \text{att}(R)$.
3. $\text{NN}_n(X_1 \dots X_n, R)$: Indicates SQL *non-null* constraints on $\emptyset \subsetneq \{X_1 \dots X_n\} \subseteq \text{att}(R)$.
4. $\text{UNQ}_n(X_1 \dots X_n, R)$: Indicates SQL *unique* constraints on $\emptyset \subsetneq \{X_1 \dots X_n\} \subseteq \text{att}(R)$.
5. $\text{PK}_n(X_1 \dots X_n, R)$: Indicates that $\emptyset \subsetneq \{X_1 \dots X_n\} \subseteq \text{att}(R)$ is the *primary key* of R .
6. $\text{FK}_n(X_1 \dots X_n, R, Y_1 \dots Y_n, T)$: Indicates that $\emptyset \subsetneq \{X_1 \dots X_n\} \subseteq \text{att}(R)$ is a *foreign key* in R that references parent key $\emptyset \subsetneq \{Y_1 \dots Y_n\} \subseteq \text{att}(T)$ in T .

In addition to the relational predicates specified above, we use a predicate $\text{TYPE}(X, R, T)$ to state that attribute $X \in \text{att}(R)$, *i.e.*, $\text{ATTR}_1(X, R)$, of relation schema R has an XML schema T , *e.g.*, xsd:string , xsd:integer , xsd:date *etc.*

Predicates for SHACL Syntax. We introduce a number of predicates to express SHACL shapes. A general vocabulary to encode SHACL shapes would have to take the recursive structure of SHACL into account, but this is not needed here:

- We generate exactly one shape for each class in $\mathcal{G}_{\mathcal{R}}$ mapped by the $\mathcal{M}_{\mathcal{R}} \subset \mathfrak{M}$. In fact, we use the class URI to identify the shape, *i.e.*, sh:NodeShape , as is done with implicit target classes in SHACL.⁴ This means that the sh:targetClass declaration is implicit, *i.e.*, also the class IRI.
- For each class, we generate a number of simple property shapes, *i.e.*, sh:property , based on the various datalog facts in $\mathcal{G}_{\mathcal{R}}$ corresponding to the class and the SQL constraints in the relational schema. There is no need for nested shapes.

We use the following predicates to represent SHACL shapes:

1. $\text{SHAPE}(R)$: Indicates that the IRI R designates a node shape with implicit class target, *i.e.*, meaning " R a sh:NodeShape , rdfs:Class ". Note that our transformation use the same IRI to identify node shape and the class target as is done with implicit target class, but this could be changed.
2. $\text{PROP}(R, P, S)$: Indicates that the node shape R has a property shape with default cardinality that requires the values of predicate path identified by the IRI P to be instances of rdfs:class identified by the IRI S , *i.e.*,

$\text{sh:property [sh:path P; sh:nodeKind sh:IRI; sh:class S]}.$

We have avoided the need for recursive shape by using the sh:class declaration.

⁴ <https://www.w3.org/TR/shacl/#implicit-targetClass>

3. $\text{DATA}(R, P, T)$: Indicates that the node shape R has a property shape with default cardinality that requires the values of predicate path identified by the IRI P to be the literals with XML schema identified by the fact T , *i.e.*,

`sh:property [ sh:path P; sh:nodeKind sh:Literal; sh:datatype T ].`

4. $\text{MINPROP}(R, P, S, N)$: Indicates that the node shape R has a property shape that requires the values of predicate path identified by the IRI P must have N or more instances of *rdfs:class* identified by the IRI S , *i.e.*,

`sh:property [ sh:path P; sh:nodeKind sh:IRI; sh:minCount N; sh:class S ].`

5. $\text{MAXPROP}(R, P, S, N)$: Indicates that the node shape R has a property shape that requires the values of predicate path identified by the IRI P must have no more than N instances of *rdfs:class* identified by the IRI S , *i.e.*,

`sh:property [ sh:path P; sh:nodeKind sh:IRI; sh:maxCount N; sh:class S ].`

6. $\text{CRDPROP}(R, P, S, N)$: Indicates that the node shape R has a property shape that requires that the predicate path identified by the IRI P has exactly N instances of *rdfs:class* identified by the IRI S , *i.e.*,

`sh:property [sh:path P; sh:nodeKind sh:IRI;
sh:minCount N; sh:maxCount N; sh:class S].`

7. $\text{MINDATA}(R, P, T, N)$, $\text{MAXDATA}(R, P, T, N)$, $\text{CRDDATA}(R, P, T, N)$: Indicate property shapes for predicate path identified by the IRI P of node shape R with min, max and exactly N cardinality restrictions, for datatype properties $\text{DATA}(R, P, T)$. E.g., for $\text{MINDATA}(R, P, T, N)$:

`sh:property [sh:path P; sh:nodeKind sh:Literal;
sh:minCount 1; sh:datatype T].`

8. $\text{UNQMAXDATA}(V, W, T, 1)$: Indicates that the node shape R has a property shape that requires values of the predicate path identified by the IRI P must have at most one literal with XML schema identified by the fact T , and the literal must be unique among all the members of *rdfs:class* identified by the IRI R , *i.e.*,

`sh:property [sh:path P; sh:nodeKind sh:Literal;
sh:maxCount 1; sh:datatype T; dash:uniqueValueForClass R].`

9. $\text{UNQCRDDATA}(V, W, T, 1)$: Indicates that the node shape R has a property shape that requires values of the predicate path identified by the IRI P must be exactly one literal with XML schema identified by the fact T , and the literal must be unique among all the members of *rdfs:class* identified by the IRI R , *i.e.*,

`sh:property [sh:path P; sh:nodeKind sh:Literal; sh:minCount 1;
sh:maxCount 1; sh:datatype T; dash:uniqueValueForClass R].`

In addition, we generate another disjoint set of predicates, named by concatenating a keyword ‘Inv’ in front of the each form of the SHACL predicates ‘Prop’ specified above. The ‘Inv’ predicates express the predicate path as the original predicates except in the *inverse* direction. For example, the predicate $\text{InvMaxProp}(R, P, S, N)$, which is the Inv of MaxProp , indicates that the node shape R has a property shape that requires values of *inverse* of predicate path identified by the IRI P must have no more than N instances of *rdfs:class* identified by the IRI S , *i.e.*,

$\text{sh:property} [\text{sh:inversePath } P; \text{ sh:nodeKind sh:IRI}; \text{ sh:maxCount } N; \text{ sh:class } S]$.

It is straightforward to transform a set of datalog facts using these SHACL predicates to a set of SHACL shapes: (1) one node shape, *i.e.*, $\text{SHAPE}(R)$, per CLASSIRI facts is needed, and (2) one SHACL property shape, *i.e.*, sh:property , (over one of the specified node shapes) for each of the other datalog facts is required.

4.2 SHACL Constraints Rewriting

We now define our SHACL constraints rewriting Γ . We assume that the input $(\mathcal{G}_R, \delta)$ is encoded as a set of relations, where $\mathcal{G}_R$ is the set of RDFS facts produced by the $\mathcal{M}_R \subset \mathbb{M}$, and δ is the datalog representation of the SQL domain constraints on $\mathcal{R}$. We define Γ by means of a set of datalog rules that act on the datalog facts corresponding to $\mathcal{G}_R$ and δ and produce a description of the generated SHACL shapes by means of the predicates defined in Sect. 4.1.

First, rule 1 is used to generate SHACL shapes for all the CLASSIRI facts produced by $\mathcal{M}_R$.

$$\text{SHAPE}(W) \leftarrow \text{CLASSIRI}(R, W) \quad (1)$$

For example, SHACL predicates $\text{SHAPE}(\text{:EMP})$, $\text{SHAPE}(\text{:PROG})$ and $\text{SHAPE}(\text{:PROJ})$ are derived in our Example 1, assuming that :EMP , etc., are the IRIs generated by the direct mappings $\mathcal{M}_R$ from the relation names, $\text{CLASSIRI}(\text{EMP}, \text{:EMP})$, etc.

Second, the rules 2 to 5 are used to generate SHACL property shapes for the case where the direct mapping identifies a database relation Q as binary, therefore mapping it to a single object property, as opposed to an RDF class and a property per attribute.⁵ We recall that in this case, $\mathcal{M}_R$ generates a fact $\text{OP_IRI}_1(Q, A, B, R, X, T, Y, W)$ where A and B are the two attributes of Q , A refers to attribute X of R , B to attribute Y of T , and W is the IRI of the RDF property. For instance, in our example, the fact $\text{OP_IRI}_1(\text{ENV}, \text{ToEMP}, \text{ToPROG}, \text{EMP}, \text{E_ID}, \text{PROG}, \text{P_ID}, \text{:ENV\#ToEMP,ToPROG,E_ID,P_ID})$ is generated from the ENV relation.

Rules 2 and 3 take care of the direction from A to B , with and without UNQ constraints on A , respectively. Rules 3 and 4 are for the direction from B to A , with and without UNQ constraints on B . Note that some of our rules use negation as failure: E.g., $\neg \text{UNQ}_n(X_1 \dots X_n, R)$ indicates that there is *no* unique constraint for attributes $\emptyset \subsetneq \{X_1 \dots X_n\} \subseteq \text{att}(R)$.

$$\begin{aligned} \text{MaxProp}(V, W, U, 1) \leftarrow & \text{UNQ}_1(A, Q), \text{OP_IRI}_1(Q, A, B, R, X, T, Y, W), \\ & \text{CLASSIRI}(R, V), \text{CLASSIRI}(T, U) \end{aligned} \quad (2)$$

⁵ As done by the direct mapping of Sequeda et al. [11], but not in the W3C recommendation [2]

$$\begin{aligned} \text{PROP}(V, W, U) \leftarrow & \neg \text{UNQ}_1(A, Q), \text{OP_IRI}_1(Q, A, B, R, X, T, Y, W), \\ & \text{CLASSIRI}(R, V), \text{CLASSIRI}(T, U) \end{aligned} \quad (3)$$

$$\begin{aligned} \text{INVMAXPROP}(U, W, V, 1) \leftarrow & \text{UNQ}_1(B, Q), \text{OP_IRI}_1(Q, A, B, R, X, T, Y, W), \\ & \text{CLASSIRI}(R, V), \text{CLASSIRI}(T, U) \end{aligned} \quad (4)$$

$$\begin{aligned} \text{INVPROP}(U, W, V) \leftarrow & \neg \text{UNQ}_1(B, Q), \text{OP_IRI}_1(Q, A, B, R, X, T, Y, W), \\ & \text{CLASSIRI}(R, V), \text{CLASSIRI}(T, U) \end{aligned} \quad (5)$$

Third, rules 6 to 9 are used to generate the SHACL property shapes for the object properties that stem from foreign key references in the non-binary relations. For these, $\mathcal{M}_R$ generates facts $\text{OP_IRI}_{2n}(X_1 \dots X_n, Y_1 \dots Y_n, R, T, W)$ where the attributes $\emptyset \subseteq \{X_1 \dots X_n\} \subseteq \text{att}(R)$ constitute a foreign key reference to the attributes $\emptyset \subseteq \{Y_1 \dots Y_n\} \subseteq \text{att}(T)$ (i.e., $\text{FK}_n(X_1 \dots X_n, R, Y_1 \dots Y_n, T)$), represented by an RDF property with IRI W . For instance, in Example 1, we get a fact $\text{OP_IRI}_2(\text{ToProj}, \text{B_ID}, \text{PROG}, \text{PROJ}, \text{:PROG}, \text{PROJ\#ToProj}, \text{B_ID})$.

Rules 6 and 7 treat the direction from R to T , with and without non-null constraints respectively. Note that since the direct mapping produces one resource per tuple in an instance of R , and a W triple only for non-null attribute values, the property W will have a cardinality of at most '1'. Rules 8 and 9 are for the inverse direction from T to R . The crucial observation here is that if there is a constraint $\text{UNQ}_n(X_1 \dots X_n, R)$, then the inverse property of W has a maximum cardinality of '1'. Otherwise, the typing is the only guarantee we have on the inverse.

$$\begin{aligned} \text{CRDPROP}(V, W, U, 1) \leftarrow & \text{NN}_n(X_1 \dots X_n, R), \\ & \text{OP_IRI}_{2n}(X_1 \dots X_n, Y_1 \dots Y_n, R, T, W), \\ & \text{CLASSIRI}(R, V), \text{CLASSIRI}(T, U) \end{aligned} \quad (6)$$

$$\begin{aligned} \text{MAXPROP}(V, W, U, 1) \leftarrow & \neg \text{NN}_n(X_1 \dots X_n, R), \\ & \text{OP_IRI}_{2n}(X_1 \dots X_n, Y_1 \dots Y_n, R, T, W), \\ & \text{CLASSIRI}(R, V), \text{CLASSIRI}(T, U) \end{aligned} \quad (7)$$

$$\begin{aligned} \text{INVMAXPROP}(U, W, V, 1) \leftarrow & \text{UNQ}_n(X_1 \dots X_n, R), \\ & \text{OP_IRI}_{2n}(X_1 \dots X_n, Y_1 \dots Y_n, R, T, W), \\ & \text{CLASSIRI}(R, V), \text{CLASSIRI}(T, U) \end{aligned} \quad (8)$$

$$\begin{aligned} \text{INVPROP}(U, W, V) \leftarrow & \neg \text{UNQ}_n(X_1 \dots X_n, R), \\ & \text{OP_IRI}_{2n}(X_1 \dots X_n, Y_1 \dots Y_n, R, T, W), \\ & \text{CLASSIRI}(R, V), \text{CLASSIRI}(T, U) \end{aligned} \quad (9)$$

Finally, rules 10 to 13 handle the datatype properties that are generated by $\mathcal{M}_R$ for every attributes of the non-binary relation schema. Recall that a fact $\text{DTP_IRI}(X, R, W)$ means that attribute X of relation R will be mapped to a datatype property with IRI W . E.g., in our Example, we will have facts like $\text{DTP_IRI}(\text{NAME}, \text{EMP}, \text{:EMP\#NAME})$, etc. The four rules treat all combinations of UNQ and NN constraints being present or absent for the attribute $X \in \text{att}(R)$. Note that like for rules 6 and 8, the maximum cardinality is 1 in

all cases. However, unlike object properties, UNQ constraints cannot be expressed by a cardinality constraints on the inverse property. We have to use UNQMAXDATA, resp, UNQCRDData, which again cannot be expressed in core SHACL, but require using the ‘uniqueValueForClass’ constraint from the DASH Data Shapes vocabulary.

$$\begin{aligned} \text{MAXDATA}(V, W, T, 1) \leftarrow & \neg \text{NN}_1(X, R), \neg \text{UNQ}_1(X, R), \text{DTP_IRI}(X, R, W), \\ & \text{TYPE}(X, R, T), \text{CLASSIRI}(R, V) \end{aligned} \quad (10)$$

$$\begin{aligned} \text{CRDData}(V, W, T, 1) \leftarrow & \text{NN}_1(X, R), \neg \text{UNQ}_1(X, R), \text{DTP_IRI}(X, R, W), \\ & \text{TYPE}(X, R, T), \text{CLASSIRI}(R, V) \end{aligned} \quad (11)$$

$$\begin{aligned} \text{UNQMAXDATA}(V, W, T, 1) \leftarrow & \neg \text{NN}_1(X, R), \text{UNQ}_1(X, R), \text{DTP_IRI}(X, R, W), \\ & \text{TYPE}(X, R, T), \text{CLASSIRI}(R, V) \end{aligned} \quad (12)$$

$$\begin{aligned} \text{UNQCRDData}(V, W, T, 1) \leftarrow & \text{NN}_1(X, R), \text{UNQ}_1(X, R), \text{DTP_IRI}(X, R, W), \\ & \text{TYPE}(X, R, T), \text{CLASSIRI}(R, V) \end{aligned} \quad (13)$$

This concludes the rules of our SHACL constraints rewriting Γ , and we shall see in the following section that they indeed cover all guarantees that can be given on the directly mapped RDF graph based on the SQL constraints.

Example 3 *The following facts result from the application of rules 1–13 on the schema of Example 1. We use QNames like :EMP#E_ID to abbreviate the IRIs generated by the direct mapping rules $\mathcal{M}_R$. Here, $\text{SHAPE}(:\text{EMP})$ expresses that any resource of type :EMP has a unique ID, exactly one name, at most one post and they should be linked to zero or more resources of type :PROG. $\text{SHAPE}(:\text{PROG})$ says that any resource of type :EMP has a unique ID, exactly one name, and is linked to exactly one member of :PROJ and, inversely, to zero or more resource of type :EMP. Lastly, $\text{SHAPE}(:\text{PROJ})$ says that any resource of type :EMP has a unique ID, a unique name, and is inversely linked to at most one resource of type :PROG. The resulting SHACL is given in Fig. 1.*

$\text{SHAPE}(:\text{EMP})$	by Γ rule 1
$\text{UNQCRDData}(:\text{EMP}, : \text{EMP}\# \text{E_ID}, \text{xsd:integer}, 1);$	by Γ rule 13
$\text{CRDData}(:\text{EMP}, : \text{EMP}\# \text{NAME}, \text{xsd:string}, 1);$	by Γ rule 11
$\text{MAXData}(:\text{EMP}, : \text{EMP}\# \text{POST}, \text{xsd:string}, 1);$	by Γ rule 10
$\text{PROP}(:\text{EMP}, : \text{ENV}\# \text{ToEMP}, \text{ToPROG}, \text{E_ID}, \text{P_ID}, : \text{PROG}).$	by Γ rule 3
$\text{SHAPE}(:\text{PROG})$	by Γ rule 1
$\text{UNQCRDData}(:\text{PROG}, : \text{PROG}\# \text{P_ID}, \text{xsd:integer}, 1);$	by Γ rule 13
$\text{CRDData}(:\text{PROG}, : \text{PROG}\# \text{NAME}, \text{xsd:string}, 1);$	by Γ rule 11
$\text{UNQCRDData}(:\text{PROG}, : \text{PROG}\# \text{ToPROJ}, \text{xsd:integer}, 1);$	by Γ rule 13
$\text{CRDPROP}(:\text{PROG}, : \text{PROG}, \text{PROJ}\# \text{ToPROJ}, \text{B_ID}, : \text{PROJ}, 1);$	by Γ rule 6
$\text{INVPROP}(:\text{PROG}, : \text{ENV}\# \text{ToEMP}, \text{ToPROG}, \text{E_ID}, \text{P_ID}, : \text{EMP}).$	by Γ rule 5
$\text{SHAPE}(:\text{PROJ})$	by Γ rule 1
$\text{UNQCRDData}(:\text{PROJ}, : \text{PROJ}\# \text{B_ID}, \text{xsd:integer}, 1);$	by Γ rule 13
$\text{UNQMAXData}(:\text{PROJ}, : \text{PROJ}\# \text{NAME}, \text{xsd:string}, 1);$	by Γ rule 12
$\text{INVMAXPROP}(:\text{PROJ}, : \text{PROG}, \text{PROJ}\# \text{ToPROJ}, \text{B_ID}, : \text{PROG}, 1);$	by Γ rule 8

<pre> :EMP a sh:NodeShape, rdfs:Class; sh:property [sh:path :EMP#E_ID; sh:nodeKind sh:Literal; sh:maxCount 1; sh:minCount 1; sh:datatype xsd:integer; dash:uniqueValueForClass :EMP]; </pre>	<pre> sh:property [sh:path :EMP#NAME; sh:nodeKind sh:Literal; sh:maxCount 1; sh:minCount 1; sh:datatype xsd:string]; </pre>	<pre> sh:property [sh:path :EMP#Post; sh:nodeKind sh:Literal; sh:maxCount 1; sh:datatype xsd:string]; sh:property [sh:path :Env#ToEMP,ToPROG,E_ID,P_ID; sh:nodeKind sh:IRI; sh:class :PROG]; </pre>
<pre> :PROG a sh:NodeShape, rdfs:Class; sh:property [sh:path :PROG#P_ID; sh:nodeKind sh:Literal; sh:maxCount 1; sh:minCount 1; sh:datatype xsd:integer; dash:uniqueValueForClass :PROG]; </pre>	<pre> sh:property [sh:path :PROG#NAME; sh:nodeKind sh:Literal; sh:maxCount 1; sh:minCount 1; sh:datatype xsd:string]; sh:property [sh:path :PROG#ToPROJ; sh:nodeKind sh:Literal; sh:maxCount 1; sh:minCount 1; dash:uniqueValueForClass :PROG]; </pre>	<pre> sh:property [sh:inversePath :Env#ToEMP,ToPROG,E_ID,P_ID; sh:nodeKind sh:IRI; sh:class :EMP]; sh:property [sh:path :PROG,PROJ#ToPROJ,B_ID; sh:nodeKind sh:IRI; sh:maxCount 1; sh:minCount 1; sh:class :PROJ]; </pre>
<pre> :PROJ a sh:NodeShape, rdfs:Class; sh:property [sh:path :PROJ#B_ID; sh:nodeKind sh:Literal; sh:maxCount 1; sh:minCount 1; sh:datatype xsd:integer; dash:uniqueValueForClass :PROJ]; </pre>	<pre> sh:property [sh:path :PROG#NAME; sh:nodeKind sh:Literal; sh:maxCount 1; sh:datatype xsd:string]; dash:uniqueValueForClass :PROJ]; </pre>	<pre> sh:property [sh:inversePath :PROG,PROJ#ToPROJ,B_ID; sh:nodeKind sh:IRI; sh:maxCount 1; sh:class :PROG]; </pre>

Fig. 1. SHACL shapes for Example 1

5 Properties of direct mapping with SHACL constraints

We now study the properties of our SHACL constraints rewriting Γ for the direct mapping $\mathfrak{M}$: constraints preservation and semantics preservation, defined in Section 5.

We show that SHACL constraint rewriting Γ for direct mapping $\mathfrak{M}$ does not lose any SQL domain constraints of the relational database that is being translated into the RDF graph:

Theorem 1. *The constraints rewriting Γ for direct mapping $\mathfrak{M}$ is constraints preserving.*

The proof of this theorem is straightforward, and it involves providing a mapping $\mathcal{N} : \mathcal{P}(\mathbb{S}) \rightarrow \mathbb{G}$ that satisfies the condition stated in Definition 3. That is, a mapping $\mathcal{N}$ that can reconstruct the initial set $\mathcal{G}_R$ of directly mapped RDF schema of a relational schema $\mathcal{R}$ and the set δ of SQL domain constraints on $\mathcal{R}$, from the set $\mathcal{S}$ of SHACL shapes $\Gamma(\mathcal{G}_R, \delta)$. We refer reader to Appendix B.1 for the Proof of Theorem 1.

We now show that the SHACL constraints rewriting Γ for direct mapping $\mathfrak{M}$ is not semantics preserving. First, we recall that mapping $\mathfrak{M}$ does not generate: (1) an IRI from a null value, (2) distinct IRIs for the repeated tuples of relation schema. These facts can be used to construct a counterexample to show that mapping $\mathfrak{M}$ generates a consistent RDF graph w.r.t. the generated SHACL constraints when the primary keys of input database are violated. Observe that in Example 4, an obstacle to obtain a semantic-preserving SHACL constraints rewriting Γ for $\mathfrak{M}$ is the semantics of direct mappings $\mathfrak{M}$ w.r.t. the PKs of relation schema.

Example 4 *Consider a relation table: `CREATE TABLE USER{ID INTEGER PRIMARY KEY}`, with tuples $t_1.ID = 1$, $t_2.ID = 1$ and $t_3.ID = \text{null}$ violating PK of the relation schema USER. It is easy to see that the directly mapped RDF triples (on the right) validate against the generated SHACL shape (on the left), which leads to a contradiction w.r.t. the Definition 4.*

$\text{SHAPE}(:\text{USER})$ $: \text{USER}/\text{ID}=1 \quad \text{rdf:type} \quad : \text{USER}.$
 $\text{UNQCRDData}(:\text{USER}, : \text{USER}/\text{ID}, \text{xsd:integer}, 1).$ $: \text{USER}/\text{ID}=1 \quad : \text{EMP}/\text{ID} \quad 1.$

Proposition 1 *The constraints rewriting Γ for direct mapping $\mathfrak{M}$ is not semantics preserving.*

Example 5 *Consider, an instance of the SIRIUS database such as: $\text{EMP}(011, \text{Ida}, \text{PhD})$, $\text{PROG}(021, \text{Semantic Integration}, 031)$, $\text{PROJ}(034, \text{PeTWIN})$ and $\text{ENV}(011, 022)$, violating the ToPROJ and ToPROG FKs on relation schemas PROG and ENV, respectively, in the Example 1. It is easy to see that the directly mapped RDF triples of the database instance in Figure 2 violate the generated $\text{SHAPE}(:\text{PROG})$ in Figure 1.*

$: \text{EMP}/\text{E_ID}=011$	$\text{rdf:type} \quad : \text{EMP};$	$: \text{PROG}/\text{P_ID}=021$	$\text{rdf:type} \quad : \text{PROG};$	$: \text{PROJ}/\text{B_ID}=031$	$\text{rdf:type} \quad : \text{PROJ};$
$: \text{EMP}\#\text{E_ID} \quad 011;$		$: \text{PROG}\#\text{P_ID} \quad 021;$		$: \text{PROJ}\#\text{B_ID} \quad 034;$	
$: \text{EMP}\#\text{NAME} \quad \text{"Ida"};$		$: \text{PROG}\#\text{NAME} \quad \text{"Semantic Integration"};$		$: \text{PROJ}\#\text{NAME} \quad \text{"PeTWIN"}.$	
$: \text{EMP}\#\text{POST} \quad \text{"PhD"}.$		$: \text{PROG}\#\text{ToPROJ} \quad 031.$			

Fig. 2. RDF triples for Example 5

Observe that in Example 5, the SHACL shapes in Figure 1 fail to detect the violation of ToPROG foreign key on the ENV schema, essentially because the rewriting Γ for $\mathfrak{M}$ does not generate SHACL constraints for the binary schema. The main reason behind this flaw is that the direct mapping $\mathfrak{M}$ does not generate RDF class for the binary schema. Similarly, observe that the node $: \text{PROG}/\text{P_ID}=021$ in Example 5 could have validated the generated $\text{SHAPE}(:\text{PROG})$ if there would not have been 'not null' constraints on the ToPROG foreign key of the relation schema PROG. That means, only the 'not null' (strong) SQL domain constraints on FKs detects the violation of FKs on the relational schema.

We observe that (1) the direct mapping as defined by Sequeda et al., but also that from the W3C recommendation generate one resource per tuple of a database instance (for non-binary relations), and the IRI of this tuple is generated from the relation's primary key. This approach breaks down if the PK constraint is violated, which explains why semantics preservation does not hold as stated. (2) the binary relations 'BINREL' rule as defined by direct mapping of Sequeda et al., but not in the W3C recommendation, are not suitable for the SHACL constraints rewriting Γ since the mapping $\mathfrak{M}$ does not generate an RDF schema for the binary schema in the relational database. (3) not all domain constraints on FKs are strong enough to guarantee the semantics of FKs on relation schema in the rewriting Γ . However, if we restrict our attention to the database instances $\mathcal{D}_\sigma$ that satisfy their PKs and FKs constraints, then the semantics preservation is restored.

Theorem 2. *The constraints rewriting Γ for direct mapping $\mathfrak{M}$ is weakly semantic preserving.*

We refer reader to Appendix B.1 for the Proof of Theorem 2.

In summary, the SHACL constraints rewriting defined in Section 4 is both constraints preserving and weakly semantics preserving.

6 Discussion

We have extended the direct mapping $\mathfrak{M}$ from relational data to RDF, proposed in [11], with SHACL constraints by using the SQL domain constraints, including data-types which were missing from both previous extensions of direct mappings [11,3]. All of the good proprieties of the original extension of direct mapping [11] apply to our extension. Contrary to the previous works, our extension describes the mapped data using SHACL constraints instead of OWL axioms. This is what makes our mapping semantics preserving.

We note that our SHACL constraints rewriting Γ rules, specified in Section 4.2 are not semantic preserving if: (1) relation schemas without PKs are considered. This is essential because $\mathfrak{M}$ produces blank nodes for the tuples of relation schemas without PKs, which are problematic for SHACL which validates the RDF graph under the UNA, and (2) relational databases violating the PKs and FKs constraints are considered for the constraints rewriting Γ , because the mapping $\mathfrak{M}$ often produces a consistent graph w.r.t. the generated SHACL shapes even if the PKs and FKs are violated in the source database.

We observe that our SHACL constraints rewriting could be extended for relation schemas without PKs in combination with OWL axioms, in a similar manner as shown for the combination of DL-Lite_{RDFS} axioms and tree-based identification constraints in [3], where the relation schemas without PKs could be used to generate OWL axioms if there exists no foreign key referential integrity constraint between the schemas with and without PKs. However, the presence of referential integrity constraints between schemas with and without PKs might be an obstacle to generate a semantic-preserving constraints rewriting in this setting, therefore, we leave this transformation as an open question for now. As for the question of database instances that do not satisfy their primary or foreign key constraints, their interest lies purely in the formulation of ‘completeness’ direction (right to left) of semantics preservation. We believe that a more useful formulation of the completeness of constraint rewriting can be found.

7 Conclusion

In this paper, we have proposed an extension of direct mapping with SHACL constraints rewriting. It transfers the semantic information of SQL domain constraints and data types, from the relational database to the RDF graph while keeping intact all the good properties of the direct mapping [11], *i.e.*, information and query preservation. In contrast to previous work, we have studied the extension of direct mapping with SHACL constraints instead of the OWL axioms. Finally, we have shown that our SHACL constraints rewriting extends the original form of direct mapping of relational databases to an RDF graph while guaranteeing constraints and semantics preservation.

Application. A SHACL description of a directly mapped RDF graph could be useful for the semantic optimization of SPARQL queries in a similar manner as the database constraints. SHACL constraints on the RDF might also be helpful in an OBDA setting to optimise SPARQL queries before translating them to queries over the source. Further,

any ontology alignments that follow the W3C direct mapping directives to connect the ontological vocabulary to the relational database, such as BOOTOX [8], could be improved by extending bootstrapping with the SHACL description of source data that fits more closely with RDF/OWL representation.

Future Work. In future work, we would like to concentrate on the more intuitive interpretation of SHACL constraints rewriting for the direct mapping specified in denotational semantics [2]. We also aim to extend our SHACL constraints rewriting from direct mapping to the interrelated and complementary W3C standard: R2RML[6].

Acknowledgements This work was supported by the Norwegian Research Council via the SIRIUS Centre for Research Based Innovation, Grant Nr. 237898.

References

1. Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of databases*, volume 8. Addison-Wesley Reading, 1995.
2. Marcelo Arenas, Alexandre Bertails, Eric Prud'hommeaux, and Juan Sequeda. A direct mapping of relational data to RDF. W3C recommendation, W3C, September 2012.
3. Diego Calvanese, Wolfgang Fischl, Reinhard Pichler, Emanuel Sallinger, and Mantas Simkus. Capturing relational schemas and functional dependencies in RDFS. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.
4. Julien Corman, Juan L. Reutter, and Ognjen Savković. Semantics and validation of recursive SHACL. In *International Semantic Web Conference*, pages 318–336. Springer, 2018.
5. Richard Cyganiak, David Wood, and Markus Lanthaler. RDF 1.1 concepts and abstract syntax. W3C recommendation, W3C, February 2014.
6. Souripriya Das, Seema Sundara, and Richard Cyganiak. R2RML: RDB to RDF mapping language, W3C recommendation, W3C, 2012.
7. Steven Harris and Andy Seaborne. SPARQL 1.1 query language. W3C recommendation, W3C, March 2013.
8. Ernesto Jiménez-Ruiz et al. BootOX: Bootstrapping OWL 2 ontologies and R2RML mappings from relational databases. In *ISWC (Posters & Demos)*, 2015.
9. Holger Knublauch and Dimitris Kontokostas. Shapes constraint language (SHACL). W3C recommendation, W3C, July 2017.
10. Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. In *Journal on data semantics X*, pages 133–173. Springer, 2008.
11. Juan F. Sequeda, Marcelo Arenas, and Daniel P. Miranker. On directly mapping relational databases to RDF and OWL. In *Proc. 21st Intl. Conf. on World Wide Web*, pages 649–658. ACM, 2012.

A APPENDIX

A.1 Rules $\mathcal{M}_R$ for translating relational schema into RDF schema

We briefly summarize the set of mapping $\mathcal{M}_R \subseteq \mathfrak{M}$ rules, defined in terms of Datalog syntax [11, Section 4.2 and 4.3.1.], used for transforming relational schema into an RDF schema. For details on each of these rules, we refer readers to the source [11].

- Predicate $\text{BINREL}(R, A, B, S, X, T, Y)$ is used to resolve any relation schema of the form $\text{REL}(R)$ as a many-to-many relations between the entities of relation schemas $\text{REL}(S)$ and $\text{REL}(T)$.

$$\begin{aligned} \text{BINREL}(R, A, B, S, X, T, Y) \leftarrow & \text{PK}_2(A, B, R), \neg \text{THREEATTR}(R), \\ & \text{FK}_1(A, R, X, S), R \neq S, \text{FK}_1(B, R, Y, T), R \neq T, \\ & \neg \text{TwoFK}(A, R), \neg \text{TwoFK}(B, R), \\ & \neg \text{ONEFK}(A, B, R), \neg \text{FKTO}(R). \end{aligned} \quad (14)$$

Where,

$$\begin{aligned} \text{THREEATTR}(R) \leftarrow & \text{ATTR}(X, R), \text{ATTR}(Y, R), \\ & \text{ATTR}(Z, R), X \neq Y, X \neq Z, Y \neq Z. \\ \text{TwoFK}(X, Y) \leftarrow & \text{FK}_1(X, Y, U_1, V_1), \text{FK}_1(X, Y, U_2, V_2), \\ & U_1 \neq U_2 \\ \text{TwoFK}(X, Y) \leftarrow & \text{FK}_1(X, Y, U_1, V_1), \text{FK}_1(X, Y, U_2, V_2), \\ & V_1 \neq V_2 \\ \text{ONEFK}(X, Y, Z) \leftarrow & \text{FK}_2(X, Y, Z, U, V, W) \\ \text{ONEFK}(X, Y, Z) \leftarrow & \text{FK}_2(Y, X, Z, U, V, W) \\ \text{FKTO}(X) \leftarrow & \text{FK}_1(U_1, Y, V, X) \\ \text{FKTO}(X) \leftarrow & \text{FK}_2(U_1, U_2, Y, V_1, V_2, X) \end{aligned}$$

The predicate $\neg \text{THREEATTR}(R)$ in combination with $\text{PK}_2(A, B, R)$ concludes that $\text{REL}(R)$ has exactly two attributes: $\text{ATTR}(A, R)$ and $\text{ATTR}(B, R)$. Predicates $\neg \text{TwoFK}(A, R)$ and $\neg \text{TwoFK}(B, R)$ define none of $\text{ATTR}(A, R)$ and $\text{ATTR}(B, R)$ are composite FK in R . Similarly, predicates $\neg \text{FKTO}(R)$ and $\neg \text{ONEFK}(A, B, R)$ respectively check that the $\text{REL}(R)$ does not have any attributes referenced by the FKs, and the attributes $\text{ATTR}(A, R)$ and $\text{ATTR}(B, R)$ in $\text{REL}(R)$ do not form a composite FK.

- Predicate $\text{CLASS}(R)$ is used to resolve any relation schema of the form $\text{REL}(R)$ that is not a 'BINREL', defined by $\neg \text{IsBINREL}(R)$, into an RDF schema *class*.

$$\begin{aligned} \text{CLASS}(R) \leftarrow & \text{REL}(R), \neg \text{IsBINREL}(R) \\ \text{IsBINREL}(R) \leftarrow & \text{BINREL}(R, A, B, S, X, T, Y) \end{aligned} \quad (15)$$

- Predicate $OP_{2n}(X_1 \dots X_n, Y_1 \dots Y_n, R, T)$ used to resolve every foreign keys, that is encoded as $FK_n(X_1 \dots X_n, R, Y_1 \dots Y_n, T)$, of any (non-binary) relation schema of the form $REL(R)$ into an object property predicate 'OP' between the entities of relation schemas $REL(R)$ and $REL(S)$.

$$OP_{2n}(X_1 \dots X_n, Y_1 \dots Y_n, R, T) \leftarrow FK(X_1 \dots X_n, R, Y_1 \dots Y_n, T), \quad (16)$$

$$\neg IsBINREL(R)$$

- Predicate $DATA(X, R, i)$ s used to resolve every $ATTR(X, R)$ of any non-binary relation schema of the form $REL(R)$ into a datatype property predicate 'DTP'.

$$DTP(X, R) \leftarrow ATTR(X, R), \neg IsBINREL(R) \quad (17)$$

Further, the following set of IRI predicates are used to construct IRIs identifiers for each of the RDFS facts produced by the datalog rule specified above. Consider a fixed IRI base,⁶ and a family of build-in predicates $CONCAT_n(n \geq 2)$ that has $n + 1$ arguments such that $CONCAT_n(X_1 \dots X_n, Y)$ holds if Y is the result of concatenation of the strings $X_1 \dots X_n$.

- Predicate $CLASSIRI(R, W)$ is used to generate IRI W^7 for all the RDFS predicate of the form $CLASS(R)$.

$$CLASSIRI(R, W) \leftarrow CLASS(R), CONCAT_2(base, R, W)$$

- Predicate $DTP_IRI(X, R, W)$ is used generate IRI W^8 for all the RDFS predicate of the form $DATA(X, R, .)$

$$DTP_IRI(X, R, W) \leftarrow DTP(X, R), CONCAT_4(base, R, "\#", X, W)$$

- Predicate OP_IRI_1 is used to generate an IRI W^9 for all the RDFS predicate of the form 'BINREL'.

$$OP_IRI_1(R, A, B, S, X, T, Y, W) \leftarrow$$

$$BINREL(R, A, B, S, X, T, Y),$$

$$CONCAT_{10}(base, R, "\#", A, ",", B, ",", X, ",", Y, W)$$

- Predicate OP_IRI_{2n} is used to generate an IRI W^{10} for all the RDFS predicate of the form $OP_{2n}(X_1 \dots X_n, Y_1 \dots Y_n, R, T)$.

$$OP_IRI_{2n}(X_1 \dots X_n, Y_1 \dots Y_n, R, T, W) \leftarrow$$

$$OP_{2n}(X_1 \dots X_n, Y_1 \dots Y_n, R, T),$$

$$CONCAT_{4n+4}(base, R, ",", T, "\#", X_1, ",", \dots, X_n, ",", Y_1, ",", \dots, Y_n, ",", W)$$

⁶ "http://resource.edu/example/db/"

⁷ "http://resource.edu/example/db/R"

⁸ http://resource.edu/example/db/R#X

⁹ "http://resource.edu/example/db/R#A,B,X,Y"

¹⁰ "http://resource.edu/example/db/R,T#X,Y"

A.2 SHACL constraints rewriting for Direct mapping

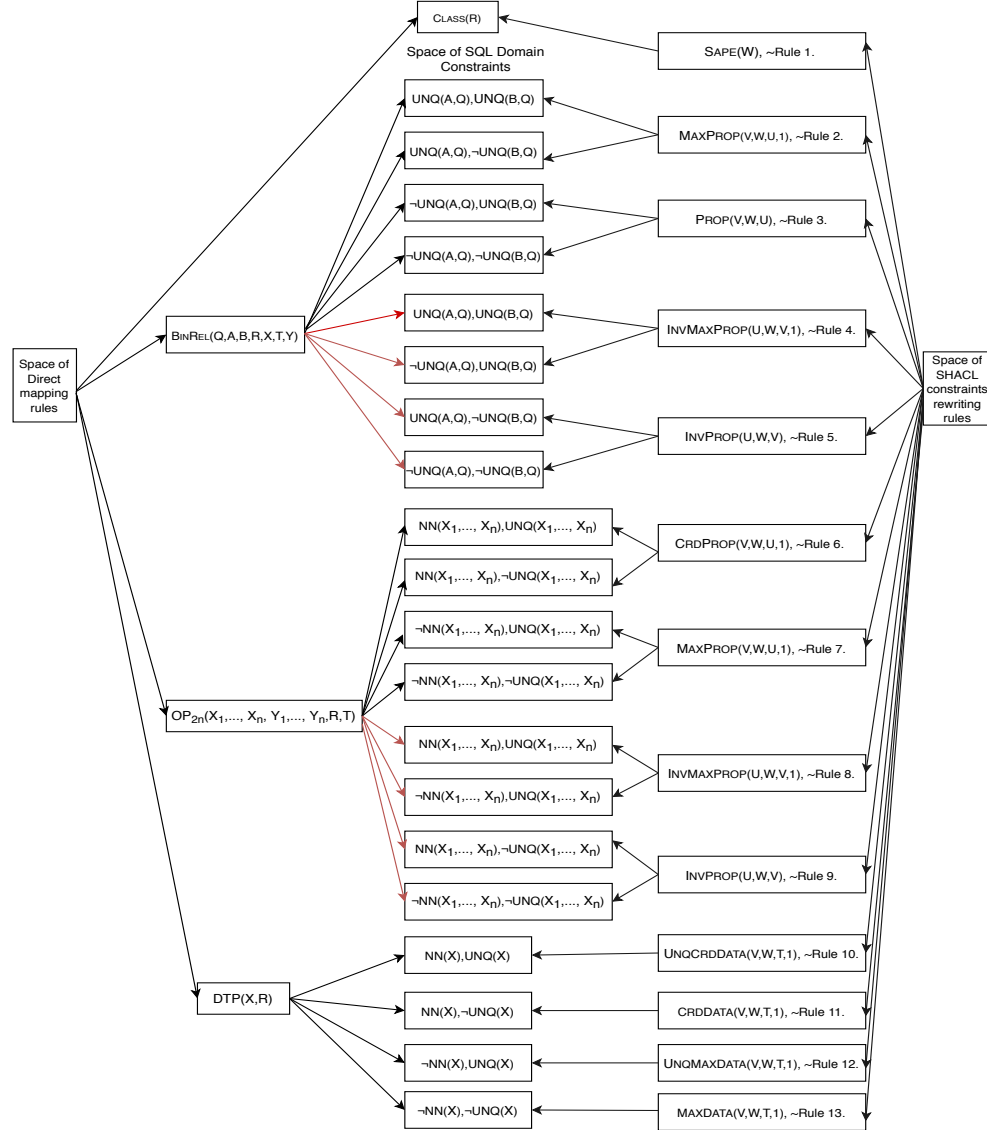


Fig. 3. The SHACL constraints rewriting rule Γ for the $\mathfrak{M}$. The red arrows point SQL δ constraints for the inverse object properties. The V, W and U in each rewriting Γ rule are the corresponding IRIs produced by the mapping $\mathcal{M}_R \subseteq \mathfrak{M}$. Note that nullability SQL constraint checks on $\text{ATTR}(A, Q)$ and $\text{ATTR}(B, Q)$ are not required for the $\text{BINREL}(R, A, B, S, X, T, Y)$. They are not-null by the definition of $\text{BINREL}(R, A, B, S, X, T, Y)$.

A.3 SHACL Constraints

We briefly recall the convenient abstract syntax of SHACL [9], introduced in [4]¹¹, with necessary changes that suits purpose. A SHACL document is a set of shapes $\mathcal{S}$, called *shape graph*, and the RDF graphs that are validated against a shape graph are called *data graph*. Each shape in $\mathcal{S}$ is defined as a triple $\langle s, \tau_s, \phi_s \rangle$, where s is the *name* of shape, τ_s is the *target declarations* and ϕ_s is the *constraints declarations*. The target declaration τ_s is a SPARQL query [7] with one output variable whose propose is to retrieve target nodes of shape s from the data graph, *i.e.*, entities occurring in the data graph for which the constraint ϕ_s of the shape s should be verified. The constraint ϕ_s is an expression defined according to the following grammar:

$$\phi ::= \top \mid I \mid C \mid \phi_1 \wedge \phi_2 \mid \neg \phi \mid \geq_n r.\phi \mid EQ(r_1, r_2),$$

where $\top$ stands for the *Boolean true value*, I is an IRI,¹² C is an RDF concept, r is a *SHACL property path* built like *SPARQL property path* [7], but without the *NegatedPropertySet* operator, $\neg$ for negation, $\geq_n r.\phi$ means "must have at least n r -successors verifying ϕ " for any natural number n , and $EQ(r_1, r_2)$ stands for the r_1 and r_2 -successors of a node must coincide. As a syntactic sugar, it is common to use $(\phi_1 \vee \phi_2)$ for $\neg(\neg\phi_1 \wedge \neg\phi_2)$, $\leq_n r.\phi$ for $\neg(\geq_{n+1} r.\phi)$ and $=_n r.\phi$ for $(\geq_n r.\phi) \wedge (\leq_n r.\phi)$. In addition, we use $\boxed{\leq 1} r.\phi$ as syntactic construct to mean "must have at most one r -successors verifying ϕ and the successors must be 'unique' ¹³ among all the focus nodes". Similarly, we use $\boxed{= 1} r.\phi$ as syntactic construct to mean "must have exactly one r -successors verifying ϕ and the successors must be 'unique' among all the focus nodes".

The notion of property path r in SHACL syntax describes the regular constraint holding over a path in an RDF graph.

Data graph validation against a shape graph can be viewed as a two-step process: the first step consists of iterating over all shapes in the shape graph, and retrieve their respective target nodes from the data graph, known as *shape assignment*; the second step consists of iterating over each target node of each shape, and check whether the node conforms the constraints defined for the shape, known as *constraints validation*. A *shape assignment* σ for an RDF G and a shape graph $\mathcal{S}$ is a total function mapping nodes v in G to subsets of $(\mathcal{S} \cup \{\neg s \mid s \in \mathcal{S}\})$ such that s and $\neg s$ can not be both in $\sigma(v)$. The assignment is called total if either $s \in \sigma(v)$ or $\neg s \in \sigma(v)$ for each node v in G and shape s in $\mathcal{S}$.

The semantics of *constraint ϕ validation* is given in terms of a function $[\phi]^{G,v,\sigma}$, for an RDF graph G , a node $v \in G$ and an *assignment* σ . The function evaluates whether v satisfies ϕ based on the semantics of assignment σ . A *total assignment* σ is defined in terms of 2-value logic, therefore, always evaluates $[\phi]^{G,v,\sigma}$ to either true ('1') or false ('0'). Formally, the evaluation of constraints ϕ at node v in an RDF graph G given the

¹¹ Note that SHACL constraints do not require recursion in our setting.

¹² I is an abstraction, standing for any syntactic constraint over an *RDF term*: exact value, datatype, regex etc.

¹³ dash:uniqueValueForClass

total assignment σ is inductively defined as follows:

$$\begin{aligned}
[\top]^{G,v,\sigma} &= 1, \\
[\neg\phi]^{G,v,\sigma} &= 1 - [\phi]^{G,v,\sigma}, \\
[\phi_1 \wedge \phi_2]^{G,v,\sigma} &= \min \{[\phi_1]^{G,v,\sigma}, [\phi_2]^{G,v,\sigma}\}, \\
[EQ(r_1, r_2)]^{G,v,\sigma} &= 1 \text{ if and only if } \{v' \mid (v, v') \in \llbracket r_1 \rrbracket^G\} = \{v' \mid (v, v') \in \llbracket r_2 \rrbracket^G\}, \\
[I]^{G,v,\sigma} &= 1 \text{ if and only if } v \text{ is the IRI } I, \\
[C]^{G,v,\sigma} &= 1 \text{ if and only if "v rdf:type C", and} \\
[\geq_n r.\phi]^{G,v,\sigma} &= 1 \text{ if and only if } |\{v' \mid (v, v') \in \llbracket r \rrbracket^G \text{ and } [\phi]^{G,v,\sigma} = 1\}| \geq n,
\end{aligned}$$

where we use $(v, v') \in \llbracket r \rrbracket^G$ to say that the v and v' are connected via a SHACL property path r . We now ready to define graph validation. An RDF graph G is valid against a set $\mathcal{S}$ of shapes if there exists at least an assignment σ for G and $\mathcal{S}$ complying with both targets and constraints, known as *faithful assignment* between G and $\mathcal{S}$. An assignment σ for G and $\mathcal{S}$ is faithful if $S \in \sigma(v)$ for each shape $\langle S, \tau_S, \phi_S \rangle$ such that $v \in [\tau_S]^G$, and for each node $v \in G$:

$$\begin{aligned}
&\text{if } S \in \sigma(v), \text{ then } \llbracket \phi_S \rrbracket^{G,v,\sigma} = 1, \\
&\text{if } \neg S \in \sigma(v), \text{ then } \llbracket \phi_S \rrbracket^{G,v,\sigma} = 0,
\end{aligned}$$

where τ_S is a SPARQL query by definition, and we use $[\tau_S]^G$ to denote the evaluation of τ_S in G .

A.4 SHACL predicates to SHACL abstract Syntax translation

We translate SHACL predicates generated by the SHACL constraints rewriting for direct mappings into the set of SHACL shapes, defined by SHACL abstract syntax. Note that translation is conceptually simple, and sufficient for our purpose.

One SHACL shape triplet $\langle R, \tau_R, \phi_R \rangle$ per $\text{SHAPE}(R)$ facts is needed, where same URI fact R identifies both the node shape and target class:

$$\begin{aligned}
R &:= \text{a sh:NodeShape}, \\
\tau_R &:= \text{sh:targetClass } R, \\
\phi_R &:= \emptyset.
\end{aligned}$$

Then, constraints component ϕ_R of node shape s can be enumerated (one-by-one in conjunction) from the set of SHACL property shapes defined on the $\text{SHAPE}(R)$. A constraint, in the set ϕ_R , per SHACL property shape on $\text{SHAPE}(R)$ is needed.

- $\phi_R = (\leq_0 P. \neg C)$ per $\text{PROP}(R, P, C)$.
- $\phi_R = (\leq_N P. C)$ per $\text{MAXPROP}(R, P, C, N)$.
- $\phi_R = (\geq_N P. C)$ per $\text{MINPROP}(R, P, C, N)$.
- $\phi_R = (=_N P. C)$ per $\text{CRDPROP}(R, P, C, N)$.
- $\phi_R = (\leq_0 P^-. \neg C)$ per $\text{INVPROP}(R, P, C)$.

- $\phi_R = (\leq_N P^- . C)$ per $\text{INVMAXPROP}(R, P, C, N)$.
- $\phi_R = (\geq_N P^- . C)$ per $\text{INVMINPROP}(R, P, C, N)$.
- $\phi_R = (\leq_0 P . \neg T)$ per $\text{DATA}(R, P, T)$.
- $\phi_R = (\geq_N P . T)$ per $\text{MINDATA}(R, P, T, N)$.
- $\phi_R = (\leq_N P . T)$ per $\text{MAXDATA}(R, P, T, N)$.
- $\phi_R = (=_N P . T)$ per $\text{CRDData}(R, P, T, N)$.
- $\phi_R = (\boxed{=1} P . T)$ per $\text{UNQCRDData}(R, P, T, 1)$.
- $\phi_R = (\boxed{\leq 1} P . T)$ per $\text{UNQMAXDATA}(R, P, T, 1)$.

B APPENDIX

B.1 Proof of Section 5

Theorem 1. *The constraints rewriting Γ for direct mapping $\mathfrak{M}$ is constraints preserving.*

Proof. We now prove constraints rewriting Γ for direct mapping $\mathfrak{M}$ is constraints preserving by defining a mapping $\mathcal{N} : \mathcal{P}(\mathbb{S}) \rightarrow \mathbb{G}$. That is, given a relational schema $\mathcal{R}$ and set δ of all explicit and implicit SQL domain constraints on $\mathcal{R}$, next we show how $\mathcal{N}(\mathcal{S})$ is defined for $\mathcal{S} = \Gamma(\mathcal{G}_{\mathcal{R}}, \delta)$, where $\mathcal{S} \subseteq \mathbb{S}$. Note that in the following we use bold latter such as **I** to denote IRI identifiers corresponding to the symbol I .

- For each shape $s \in \mathcal{S}$ such that $s := \langle \mathbf{R}, \tau_{\mathbf{R}}, \phi_{\mathbf{R}} \rangle$, where

$\mathbf{R} := \text{a sh:NodeShape},$

$\tau_{\mathbf{R}} := \text{sh:targetClass } \mathbf{R}, \text{ and}$

$\phi_{\mathbf{R}}$ is a non-empty set of constraints.

From rewriting Γ rule 1, it is clear that $\mathbf{R}$ is an IRI identifier corresponding to an RDFS fact of the form $\text{CLASSIRI}(R, \mathbf{R})$. It is straightforward to construct fact $\text{CLASSIRI}(R, \mathbf{R})$ from $\mathbf{R}$ since, by the definition of $\mathfrak{M}$ rule, $\mathbf{R}$ contains essential information for the first argument of the fact $\text{CLASSIRI}(R, \mathbf{R})$. For instance, if $\mathbf{R}$ is an IRI fact 'baseIRI/student' then facts $\text{CLASSIRI}(\text{student}, \text{'baseIRI/student'})$ can be constructed by following rule 1. Thus, the fact of the form $\text{CLASSIRI}(R, \mathbf{R})$ that was constructed based on the identifier $\mathbf{R}$, we collect in a set, known as RDF schema $\mathcal{G}_{\mathcal{R}}$. In the rest part of proof enumeration, we consider the following case for the constraints component $\phi_{\mathbf{R}}$,

- First, we consider all possible SHACL property shapes corresponding to the datatype property, related to 'DTP_IRI':
 1. If $\phi_{\mathbf{R}}$ is a SHACL property shape $(\leq_1 \mathbf{P} . T)$ such that $\mathbf{P}$ is an IRI identifier of the forms $\text{'baseIRI/R\#X'}$ and T is an XML schema, then the facts of the form $\text{DTP_IRI}(X, R, \mathbf{R})$, $\neg \text{NN}_1(X, R)$, $\neg \text{UNQ}_1(X, R)$ and $\text{TYPE}(X, R, T)$ can be constructed by following the Γ rule 10. We collect constraints: $\neg \text{NN}_1(X, R)$, $\neg \text{UNQ}_1(X, R)$ and $\text{TYPE}(X, R, T)$, in a set δ , known as SQL constraints vocabulary, and rest of the facts: $\text{DTP_IRI}(X, R, \mathbf{R})$, in the set $\mathcal{G}_{\mathcal{R}}$, respectively.

2. If $\phi_{\mathbf{R}}$ is a SHACL property shape ($=_1 \mathbf{P} \cdot \mathbf{T}$) such that $\mathbf{P}$ is an IRI identifier of the forms $':\text{baseIRI}/\mathbf{R}\#X'$ and $\mathbf{T}$ is an XML schema, then the facts of the form $\text{DTP_IRI}(X, R, \mathbf{R})$, $\text{NN}_1(X, R)$, $\neg\text{UNQ}_1(X, R)$ and $\text{TYPE}(X, R, T)$ can be constructed by following the Γ rule 11. We collect constraints: $\text{NN}_1(X, R)$, $\neg\text{UNQ}_1(X, R)$ and $\text{TYPE}(X, R, T)$, in the set δ , and rest of the fact: $\text{DTP_IRI}(X, R, \mathbf{R})$, in the set $\mathcal{G}_{\mathcal{R}}$, respectively.
 3. If $\phi_{\mathbf{R}}$ is a SHACL property shape ($\leq_1 \mathbf{P} \cdot \mathbf{T}$) such that $\mathbf{P}$ is an IRI identifier of the forms $':\text{baseIRI}/\mathbf{R}\#X'$ and $\mathbf{T}$ is an XML schema, then the facts of the form $\text{DTP_IRI}(X, R, \mathbf{R})$, $\neg\text{NN}_1(X, R)$, $\text{UNQ}_1(X, R)$ and $\text{TYPE}(X, R, T)$ can be constructed by following the Γ rule 12. We collect constraints: $\neg\text{NN}_1(X, R)$, $\text{UNQ}_1(X, R)$ and $\text{TYPE}(X, R, T)$, in the set δ , and rest of the fact: $\text{DTP_IRI}(X, R, \mathbf{R})$, in the set $\mathcal{G}_{\mathcal{R}}$, respectively.
 4. If $\phi_{\mathbf{R}}$ is a SHACL property shape ($\equiv_1 \mathbf{P} \cdot \mathbf{T}$) such that $\mathbf{P}$ is an IRI identifier of the forms $':\text{baseIRI}/\mathbf{R}\#X'$ and $\mathbf{T}$ is an XML schema, then the facts of the form $\text{DTP_IRI}(X, R, \mathbf{R})$, $\text{NN}_1(X, R)$, $\text{UNQ}_1(X, R)$ and $\text{TYPE}(X, R, T)$ can be constructed by following the Γ rule 13. We collect constraints: $\text{NN}_1(X, R)$, $\text{UNQ}_1(X, R)$ and $\text{TYPE}(X, R, T)$, in the set δ , and rest of the fact: $\text{DTP_IRI}(X, R, \mathbf{R})$, in the set $\mathcal{G}_{\mathcal{R}}$, respectively.
- Second, we consider all possible SHACL property shapes corresponding to the object property, related to $':\text{OP_IRI}_{2n}':$
 1. If $\phi_{\mathbf{R}}$ is a SHACL property shape ($=_1 \mathbf{P} \cdot \mathbf{C}$) such that $\mathbf{P}$ and $\mathbf{C}$ are IRI-identifiers of the forms $':\text{baseIRI}/\mathbf{R}, \mathbf{C}\#X, Y'$ and $':\text{baseIRI}/\mathbf{C}'$, then the facts of the form $\text{OP_IRI}_{2n}(X_1 \dots X_n, Y_1 \dots Y_n, R, C, W)$ and $\text{NN}_n(X_1 \dots X_n, R)$ can be constructed by following the Γ rule 6. We collect $\text{NN}_n(X_1 \dots X_n, R)$ in the set δ , and rest of the fact: $\text{OP_IRI}_{2n}(X_1 \dots X_n, Y_1 \dots Y_n, R, C, W)$, in the set $\mathcal{G}_{\mathcal{R}}$, respectively.
 2. If $\phi_{\mathbf{R}}$ is a SHACL property shape ($\leq_1 \mathbf{P} \cdot \mathbf{C}$) such that $\mathbf{P}$ and $\mathbf{C}$ are IRI-identifiers of the forms $':\text{baseIRI}/\mathbf{R}, \mathbf{C}\#X, Y'$ and $':\text{baseIRI}/\mathbf{C}'$, then the facts of the form $\neg\text{NN}_n(X_1 \dots X_n, R)$ and $\text{OP_IRI}_{2n}(X_1 \dots X_n, Y_1 \dots Y_n, R, C, W)$ can be constructed by following the Γ rule 7. We collect $\neg\text{NN}_n(X_1 \dots X_n, R)$ in the set δ , and rest of the fact: $\text{OP_IRI}_{2n}(X_1 \dots X_n, Y_1 \dots Y_n, R, C, W)$, in the set $\mathcal{G}_{\mathcal{R}}$, respectively.
 3. If $\phi_{\mathbf{R}}$ is a SHACL property shape ($\leq_1 \mathbf{P}^- \cdot \mathbf{C}$) such that $\mathbf{P}$ and $\mathbf{C}$ are IRI-identifiers of the forms $':\text{baseIRI}/\mathbf{C}, \mathbf{R}\#X, Y'$ and $':\text{baseIRI}/\mathbf{C}'$, then the facts of the form $\text{UNQ}_n(X_1 \dots X_n, C)$ and $\text{OP_IRI}_{2n}(X_1 \dots X_n, Y_1 \dots Y_n, C, R, W)$ can be constructed by following the Γ rule 8. We collect constraint fact $\text{UNQ}_n(X_1 \dots X_n, C)$ in the set δ , but avoid RDFS fact for the redundancy reason. The RDFS fact $\text{OP_IRI}_{2n}(X_1 \dots X_n, Y_1 \dots Y_n, C, R, W)$ will eventually be collected in the set $\mathcal{G}_{\mathcal{R}}$ when we reason for the shape $\langle \mathbf{C}, \tau_{\mathbf{C}}, \phi_{\mathbf{C}} \rangle$ corresponding to the IRI fact $\mathbf{C}$.
 4. If $\phi_{\mathbf{R}}$ is a SHACL property shape ($\leq_0 \mathbf{P}^- \cdot \neg\mathbf{C}$) such that $\mathbf{P}$ and $\mathbf{C}$ are IRI-identifiers of the forms $':\text{baseIRI}/\mathbf{R}, \mathbf{C}\#X, Y'$ and $':\text{baseIRI}/\mathbf{C}'$, then the facts of the form $\neg\text{UNQ}_n(X_1 \dots X_n, C)$ and $\text{OP_IRI}_{2n}(X_1 \dots X_n, Y_1 \dots Y_n, C, R, W)$ can be constructed by following the Γ rule 9. We collect constraint fact $\text{UNQ}_n(X_1 \dots X_n, C)$ in the set δ , but avoid RDFS fact for the redundancy reason. The RDFS fact $\text{OP_IRI}_{2n}(X_1 \dots X_n, Y_1 \dots Y_n, C, R, W)$ will eventu-

- ally be collected in the set $\mathcal{G}_{\mathcal{R}}$ when we reason for the shape $\langle C, \tau_C, \phi_C \rangle$ corresponding to the IRI fact C .
- Third, we consider all possible SHACL property shapes corresponding to the 'BINREL' properties, related to the 'OP_IRI₁':
 1. If $\phi_{\mathbf{R}}$ is a SHACL property shape ($\leq_1 \mathbf{P} \cdot \mathbf{C}$) such that $\mathbf{P}$ and $\mathbf{C}$ are IRI-identifiers of the forms $':\text{baseIRI}/Q\#A,B,X,Y'$ and $':\text{baseIRI}/C'$, then the facts of the form $\text{UNQ}_1(A, Q)$ and $\text{OP_IRI}_1(Q, A, B, R, X, C, Y, W)$ can be constructed by following the Γ rule 2. We collect them in their respective set $\mathcal{G}_{\mathcal{R}}$ and δ of vocabulary.
 2. If $\phi_{\mathbf{R}}$ is a SHACL property shape ($\leq_0 \mathbf{P} \cdot \neg \mathbf{C}$) such that $\mathbf{P}$ and $\mathbf{C}$ are IRI identifiers of the forms $':\text{baseIRI}/Q\#A,B,X,Y'$ and $':\text{baseIRI}/C'$, respectively, then the facts of the form $\neg \text{UNQ}_1(A, Q)$ and $\text{OP_IRI}_1(Q, A, B, R, X, C, Y, W)$ can be constructed by following the Γ rule 3. We collect them in their respective set $\mathcal{G}_{\mathcal{R}}$ and δ of vocabulary.
 3. If $\phi_{\mathbf{R}}$ is a SHACL property shape ($\leq_1 \mathbf{P}^- \cdot \mathbf{C}$) such that $\mathbf{P}$ and $\mathbf{C}$ are IRI identifiers of the forms $':\text{baseIRI}/Q\#A,B,X,Y'$ and $':\text{baseIRI}/C'$, then the facts of the form $\text{UNQ}_1(B, Q)$ and $\text{OP_IRI}_1(Q, A, B, C, X, R, Y, W)$ can be constructed by following the Γ rule 4. We collect constraint fact $\text{UNQ}_1(B, Q)$ in the set δ , but avoid RDFS fact for the redundancy reason. The RDFS fact $\text{OP_IRI}_1(Q, A, B, C, X, R, Y, W)$ will eventually be collected in the set $\mathcal{G}_{\mathcal{R}}$ when we reason for the shape $\langle C, \tau_C, \phi_C \rangle$ corresponding to the IRI fact C .
 4. If $\phi_{\mathbf{R}}$ is a SHACL property shape ($\leq_0 \mathbf{P}^- \cdot \neg \mathbf{C}$) such that $\mathbf{P}$ and $\mathbf{C}$ are IRI identifiers of the forms $':\text{baseIRI}/Q\#A,B,X,Y'$ and $':\text{baseIRI}/C'$, then the facts of the form $\neg \text{UNQ}_1(B, Q)$ and $\text{OP_IRI}_1(Q, A, B, C, X, R, Y, W)$ can be constructed by following the Γ rule 5. We collect constraint fact $\text{UNQ}_1(B, Q)$ in the set δ , but avoid RDFS fact for the redundancy reason. The RDFS fact $\text{OP_IRI}_1(Q, A, B, C, X, R, Y, W)$ will eventually be collected in the set $\mathcal{G}_{\mathcal{R}}$ when we reason for the shape $\langle C, \tau_C, \phi_C \rangle$ corresponding to the IRI fact C .

This concludes the proof of the theorem.

Proposition 1 *The constraints rewriting Γ for direct mapping $\mathfrak{M}$ is not semantics preserving.*

Proof. Trivial by the Example 4 and 5.

Theorem 2. *The constraints rewriting Γ for direct mapping $\mathfrak{M}$ is weakly semantic preserving.*

Proof. We first proceed for the right direction $\mathcal{D}_{\sigma} \models \Sigma \implies \mathcal{G} \models S$. Let $\mathcal{D}_{\sigma}$ be an instance of relational schema $\mathcal{R}$ that never violates the set of Pks and FKs on $\mathcal{R}$, and $\mathcal{D}_{\sigma}$ is legal for the set Σ of SQL constraints on $\mathcal{R}$, i.e., $\mathcal{D}_{\sigma} \models \Sigma$. Now, there must be the case $\mathcal{G} \models S$ under SHACL semantics, where $\mathcal{G}$ is the directly mapped RDF graph $\mathcal{M}_{\mathcal{D}}(\mathcal{G}_{\mathcal{R}}, \sigma, \mathcal{D}_{\sigma})$ from Definition 1 and S is the set of SHACL shapes $\Gamma(\mathcal{G}_{\mathcal{R}}, \delta)$ from rewriting rules in Section 4.2.

Before, we proceed to the proof, we recall fact that the direct mapping $\mathcal{M}$ is information preserving [11, Theorem 1], therefore, there exists an inverse mapping $\mathcal{K} : \mathcal{G} \rightarrow \mathcal{D}$,

from the directly mapped RDF graph $\mathcal{G} = \mathcal{M}_{\mathcal{D}}(\mathcal{G}_{\mathcal{R}}, \sigma, \mathcal{D}_{\sigma})$ from Definition 1 to the database instance $\mathcal{D}_{\sigma}$. Similarly, from Theorem 1, the SHACL constraints rewriting Γ for direct mapping $\mathfrak{M}$ is constraints preserving, therefore, there exists an inverse mapping $N(S)$ for any $S = \Gamma(\mathcal{G}_{\mathcal{R}}, \delta)$.

With these facts in place, we now turn out attention to the problem $\mathcal{G} \models S$, which we establish by defining a faithful assignment between $\mathcal{G}$ and S , *i.e.*, meaning "for every nodes v in $\mathcal{G}$, there exists a node shape s in S complying both *shape assignment* and *constraints validation*". Note that in the following we use bold latter such as **I** to denote IRI identifiers corresponding to the symbol I .

- For each node shape s in S such that $s := \langle \mathbf{R}, \tau_{\mathbf{R}}, \phi_{\mathbf{R}} \rangle$, where

$$\tau_{\mathbf{R}} := \text{sh:targetClass} : \mathbf{R}.$$

From proof of Theorem 1, it is clear that **R** is an IRI identifier corresponding to an RDFS fact of the form $\text{CLASSIRI}(R, \mathbf{R})$ in $\mathcal{G}_{\mathcal{R}}$. Given that $\mathcal{G}$ is the directly mapped RDF graph $\mathcal{M}_{\mathcal{D}}(\mathcal{G}_{\mathcal{R}}, \sigma, \mathcal{D}_{\sigma})$, if there exists a mapping of a node v in $\mathcal{G}$ such that

$$v \text{ rdf:type } \mathbf{R}.$$

then, there exists a *shape assignment* σ such that $s \in \sigma(v)$ for the $v \in \mathcal{G}$ and $s \in S$. The node $v \in \mathcal{G}$ validates against the node shape $s \in S$ if $\llbracket \phi_{\mathbf{R}} \rrbracket^{\mathcal{G}, v, \sigma} = 1$ holds by following the semantics of SHACL constraints validation. Now, to establish $\llbracket \phi_{\mathbf{R}} \rrbracket^{\mathcal{G}, v, \sigma} = 1$, we consider all possible case for the $\phi_{\mathbf{R}}$.

- First, we consider possible property shapes corresponding to the datatype property.
 1. If $\phi_{\mathbf{R}}$ is a SHACL property shape ($\leq_1 \mathbf{P}. T$) such that **P** is an IRI identifier of the forms $\text{'baseIRI/R\#X'}$ and T is an XML schema, then there exist facts $\text{DTP_IRI}(X, R, \mathbf{R})$, $\neg \text{NN}_1(X, R)$, $\neg \text{UNQ}_1(X, R)$ and $\text{TYPE}(X, R, T)$ from Theorem 1. From the definition of $\mathcal{G} = \mathcal{M}_{\mathcal{D}}(\mathcal{G}_{\mathcal{R}}, \sigma, \mathcal{D}_{\sigma})$ together with SQL constraints $\neg \text{NN}_1(X, R)$, $\neg \text{UNQ}_1(X, R)$ and $\text{TYPE}(X, R, T)$ on $\text{ATTR}_1(X, R)$, for mapping of each node v in $\mathcal{G}$ such that

$$v \text{ rdf:type } \mathbf{R}.$$

there exists mapping of at most one value c of $\text{ATTR}_1(X, R)$ from database $\mathcal{D}$ to the $\mathcal{G}$ in the form of triple,

$$v \text{ } \mathbf{P} \text{ } c.$$

Thus, the node v conform both the cardinality restriction and the XML type requirement for the constraint $\phi_{\mathbf{R}}$, therefore, $\llbracket \phi_{\mathbf{R}} \rrbracket^{\mathcal{G}, v, \sigma} = 1$ holds.

2. If $\phi_{\mathbf{R}}$ is a SHACL property shape ($=_1 \mathbf{P}. T$) such that **P** is an IRI identifier of the forms $\text{'baseIRI/R\#X'}$ and T is an XML schema, then the facts of the form $\text{DTP_IRI}(X, R, \mathbf{R})$, $\text{NN}_1(X, R)$, $\neg \text{UNQ}_1(X, R)$ and $\text{TYPE}(X, R, T)$ exist from the Theorem 1. From the definition of $\mathcal{G} = \mathcal{M}_{\mathcal{D}}(\mathcal{G}_{\mathcal{R}}, \sigma, \mathcal{D}_{\sigma})$ together with SQL constraints $\text{NN}_1(X, R)$, $\neg \text{UNQ}_1(X, R)$ and $\text{TYPE}(X, R, T)$ on $\text{ATTR}_1(X, R)$, for mapping of each node v in $\mathcal{G}$ such that

$$v \text{ rdf:type } \mathbf{R},$$

there exists mapping of exactly one value c of $\text{ATTR}_1(X, R)$ from database $\mathcal{D}$ to the $\mathcal{G}$ in the form of triple,

$$v \quad \mathbf{P} \quad c.$$

Thus, the node v conform both the cardinality restriction and the XML type requirement for the constraint $\phi_{\mathbf{R}}$, therefore, $\llbracket \phi_{\mathbf{R}} \rrbracket^{\mathcal{G}, v, \sigma} = 1$ holds.

3. If $\phi_{\mathbf{R}}$ is a SHACL property shape $(\leq_1 \mathbf{P} \cdot \mathbf{T})$ such that $\mathbf{P}$ is an IRI identifier of the forms $\text{'baseIRI/R\#X'}$ and $\mathbf{T}$ is an XML schema, then the facts of the form $\text{DTP_IRI}(X, R, \mathbf{R})$, $\neg \text{NN}_1(X, R)$, $\text{UNQ}_1(X, R)$ and $\text{TYPE}(X, R, T)$ exist from the Theorem 1. From the definition of $\mathcal{G} = \mathcal{M}_{\mathcal{D}}(\mathcal{G}_{\mathcal{R}}, \sigma, \mathcal{D}_{\sigma})$ together with SQL constraints $\neg \text{NN}_1(X, R)$, $\text{UNQ}_1(X, R)$ and $\text{TYPE}(X, R, T)$ on $\text{ATTR}_1(X, R)$, for mapping of each node v in $\mathcal{G}$ such that

$$v \quad \text{rdf:type} \quad \mathbf{R},$$

there exists mapping of at most one unique value c of $\text{ATTR}_1(X, R)$ from database $\mathcal{D}$ to the $\mathcal{G}$ in the form of triple,

$$v \quad \mathbf{P} \quad c.$$

Thus, the node v conform both the cardinality restriction and the XML schema type requirement for the constraint $\phi_{\mathbf{R}}$, therefore, $\llbracket \phi_{\mathbf{R}} \rrbracket^{\mathcal{G}, v, \sigma} = 1$ holds.

4. If $\phi_{\mathbf{R}}$ is a SHACL property shape $(=1 \mathbf{P} \cdot \mathbf{T})$ such that $\mathbf{P}$ is an IRI identifier of the forms $\text{'baseIRI/R\#X'}$ and $\mathbf{T}$ is an XML schema, then the facts of the form $\text{DTP_IRI}(X, R, \mathbf{R})$, $\text{NN}_1(X, R)$, $\text{UNQ}_1(X, R)$ and $\text{TYPE}(X, R, T)$ exist from the Theorem 1. From the definition of $\mathcal{G} = \mathcal{M}_{\mathcal{D}}(\mathcal{G}_{\mathcal{R}}, \sigma, \mathcal{D}_{\sigma})$ together with SQL constraints $\text{NN}_1(X, R)$, $\text{UNQ}_1(X, R)$ and $\text{TYPE}(X, R, T)$ on $\text{ATTR}_1(X, R)$, for mapping of each node v in $\mathcal{G}$ such that

$$v \quad \text{rdf:type} \quad \mathbf{R},$$

there exists mapping of exactly one unique value c of $\text{ATTR}_1(X, R)$ from database $\mathcal{D}$ to the $\mathcal{G}$ in the form of triple,

$$v \quad \mathbf{P} \quad c.$$

Thus, the node v conform both the cardinality restriction and the XML schema type requirement for the constraint $\phi_{\mathbf{R}}$, therefore, $\llbracket \phi_{\mathbf{R}} \rrbracket^{\mathcal{G}, v, \sigma} = 1$ holds.

- Second, we consider possible property shapes corresponding to the object property.

1. If $\phi_{\mathbf{R}}$ is a SHACL property shape $(=1 \mathbf{P} \cdot \mathbf{C})$ such that $\mathbf{P}$ and $\mathbf{C}$ are IRI-identifiers of the forms $\text{'baseIRI/R,C\#X,Y'}$ and 'baseIRI/C' , then the facts of the form $\text{OP_IRI}_{2n}(X_1 \dots X_n, Y_1 \dots Y_n, R, C, W)$ and $\text{NN}_n(X_1 \dots X_n, R)$ exist from the Theorem 1. From the definition of $\mathcal{G} = \mathcal{M}_{\mathcal{D}}(\mathcal{G}_{\mathcal{R}}, \sigma, \mathcal{D}_{\sigma})$

together with SQL constraints $NN_n(X_1 \dots X_n, R)$ on $ATTR_n(X_1 \dots X_n, R)$, for mapping of each node v in $\mathcal{G}$ such that

$$v \text{ rdf:type } \mathbf{R}.$$

, there exists mapping of exactly one node v' in $\mathcal{G}$ such that

$$v \text{ } \mathbf{P} \text{ } v'.$$

,where " $v' \text{ rdf:type } \mathbf{C}$ ". Thus, the node v conforms both the cardinality restriction and the property value type requirement for the constraint $\phi_{\mathbf{R}}$, therefore, $\llbracket \phi_{\mathbf{R}} \rrbracket^{\mathcal{G}, v, \sigma} = 1$ holds.

2. If $\phi_{\mathbf{R}}$ is a SHACL property shape $(\leq_1 \mathbf{P} \cdot \mathbf{C})$ such that $\mathbf{P}$ and $\mathbf{C}$ are IRI-identifiers of the forms $':\text{baseIRI}/\mathbf{R}, \mathbf{C}\#X, Y'$ and $':\text{baseIRI}/\mathbf{C}'$, then the facts of the form $OP_IRI_{2n}(X_1 \dots X_n, Y_1 \dots Y_n, R, C, W)$ and $\neg NN_n(X_1 \dots X_n, R)$ exist from the Theorem 1. From the definition of $\mathcal{G} = \mathcal{M}_{\mathcal{D}}(\mathcal{G}_{\mathcal{R}}, \sigma, \mathcal{D}_{\sigma})$ together with SQL constraints $\neg NN_n(X_1 \dots X_n, R)$ on $ATTR_n(X_1 \dots X_n, R)$, for mapping of each node v in $\mathcal{G}$ such that

$$v \text{ rdf:type } \mathbf{R}.$$

, there exists mapping of at most one node v' in $\mathcal{G}$ such that

$$v \text{ } \mathbf{P} \text{ } v'.$$

,where " $v' \text{ rdf:type } \mathbf{C}$ ". Thus, the node v conforms both the cardinality restriction and the property value type requirement for the constraint $\phi_{\mathbf{R}}$, therefore, $\llbracket \phi_{\mathbf{R}} \rrbracket^{\mathcal{G}, v, \sigma} = 1$ holds.

3. If $\phi_{\mathbf{R}}$ is a SHACL property shape $(\leq_1 \mathbf{P}^- \cdot \mathbf{C})$ such that $\mathbf{P}$ and $\mathbf{C}$ are IRI-identifiers of the forms $':\text{baseIRI}/\mathbf{C}, \mathbf{R}\#X, Y'$ and $':\text{baseIRI}/\mathbf{C}'$, then the facts of the form $OP_IRI_{2n}(X_1 \dots X_n, Y_1 \dots Y_n, C, R, W)$ and $UNQ_n(X_1 \dots X_n, C)$ exist from the Theorem 1. From the definition of $\mathcal{G} = \mathcal{M}_{\mathcal{D}}(\mathcal{G}_{\mathcal{R}}, \sigma, \mathcal{D}_{\sigma})$ together with SQL constraints $UNQ_n(X_1 \dots X_n, C)$ on $ATTR_n(X_1 \dots X_n, C)$, for mapping of each node v in $\mathcal{G}$ such that

$$v \text{ rdf:type } \mathbf{R},$$

there exists mapping of at most node v' in $\mathcal{G}$ such that

$$v' \text{ } \mathbf{P} \text{ } v.$$

,where " $v' \text{ rdf:type } \mathbf{C}$ ". Thus, the node v conforms both the cardinality restriction and the property value type requirement for the constraint $\phi_{\mathbf{R}}$, therefore, $\llbracket \phi_{\mathbf{R}} \rrbracket^{\mathcal{G}, v, \sigma} = 1$ holds.

4. If $\phi_{\mathbf{R}}$ is a SHACL property shape $(\leq_0 \mathbf{P}^- \cdot \neg \mathbf{C})$ such that $\mathbf{P}$ and $\mathbf{C}$ are IRI-identifiers of the forms $':\text{baseIRI}/\mathbf{R}, \mathbf{C}\#X, Y'$ and $':\text{baseIRI}/\mathbf{C}'$, then the facts of the form $\neg UNQ_n(X_1 \dots X_n, C)$ and $OP_IRI_{2n}(X_1 \dots X_n, Y_1 \dots Y_n, C, R, W)$

exist from the Theorem 1. From the definition of $\mathcal{G} = \mathcal{M}_{\mathcal{D}}(\mathcal{G}_{\mathcal{R}}, \sigma, \mathcal{D}_{\sigma})$ together with SQL constraints $\text{UNQ}_n(X_1 \dots X_n, C)$ on $\text{ATTR}_n(X_1 \dots X_n, C)$, for mapping of each node v in $\mathcal{G}$ such that

$$v \text{ rdf:type } \mathbf{R},$$

there exists mapping of zero or more nodes v' in $\mathcal{G}$ such that

$$v' \mathbf{P} v.$$

, where " $v' \text{ rdf:type } \mathbf{C}$ ". Thus, the node v conforms both the cardinality restriction and the property value type requirement for the constraint $\phi_{\mathbf{R}}$, therefore, $\llbracket \phi_{\mathbf{R}} \rrbracket^{\mathcal{G}, v, \sigma} = 1$ holds.

- Third, we consider possible SHACL property shapes corresponding to the 'OP_IRI₁':

1. If $\phi_{\mathbf{R}}$ is a SHACL property shape ($\leq_1 \mathbf{P} \mathbf{C}$) such that $\mathbf{P}$ and $\mathbf{C}$ are IRI identifiers of the forms $\text{'baseIRI/Q\#A,B,X,Y'}$ and 'baseIRI/C' , then the facts of the form $\text{UNQ}_1(A, Q)$ and $\text{OP_IRI}_1(Q, A, B, R, X, C, Y, W)$ exist from the Theorem 1. From the definition of $\mathcal{G} = \mathcal{M}_{\mathcal{D}}(\mathcal{G}_{\mathcal{R}}, \sigma, \mathcal{D}_{\sigma})$ together with SQL constraints $\text{UNQ}_1(A, Q)$ on $\text{ATTR}_1(A, Q)$, for mapping of each node v in $\mathcal{G}$ such that

$$v \text{ rdf:type } \mathbf{R}.$$

there exists mapping of at most node v' in $\mathcal{G}$ such that

$$v \mathbf{P} v'.$$

, where " $v' \text{ rdf:type } \mathbf{C}$ ". Thus, the node v conforms both the cardinality restriction and the property value type requirement for the constraint $\phi_{\mathbf{R}}$, therefore, $\llbracket \phi_{\mathbf{R}} \rrbracket^{\mathcal{G}, v, \sigma} = 1$ holds.

2. If $\phi_{\mathbf{R}}$ is a SHACL property shape ($\leq_0 \mathbf{P} \neg \mathbf{C}$) such that $\mathbf{P}$ and $\mathbf{C}$ are IRI identifiers of the forms $\text{'baseIRI/Q\#A,B,X,Y'}$ and 'baseIRI/C' , then the facts of the form $\neg \text{UNQ}_1(A, Q)$ and $\text{OP_IRI}_1(Q, A, B, R, X, C, Y, W)$ exist from the Theorem 1. From the definition of $\mathcal{G} = \mathcal{M}_{\mathcal{D}}(\mathcal{G}_{\mathcal{R}}, \sigma, \mathcal{D}_{\sigma})$ together with SQL constraints $\neg \text{UNQ}_1(A, Q)$ on $\text{ATTR}_1(A, Q)$, for mapping of each node v in $\mathcal{G}$ such that

$$v \text{ rdf:type } \mathbf{R}.$$

there exists mapping of zero or more nodes v' in $\mathcal{G}$ such that

$$v \mathbf{P} v'.$$

, where " $v' \text{ rdf:type } \mathbf{C}$ ". Thus, the node v conforms both the cardinality restriction and the property value type requirement for the constraint $\phi_{\mathbf{R}}$, therefore, $\llbracket \phi_{\mathbf{R}} \rrbracket^{\mathcal{G}, v, \sigma} = 1$ holds.

3. If $\phi_{\mathbf{R}}$ is a SHACL property shape ($\leq_1 \mathbf{P}^- \mathbf{C}$) such that $\mathbf{P}$ and $\mathbf{C}$ are IRI identifiers of the forms $\text{'baseIRI/Q\#A,B,X,Y'}$ and 'baseIRI/C' , then the facts of the form $\text{UNQ}_1(B, Q)$ and $\text{OP_IRI}_1(Q, A, B, C, X, R, Y, W)$ exist

from the Theorem 1. From the definition of $\mathcal{G} = \mathcal{M}_{\mathcal{D}}(\mathcal{G}_{\mathcal{R}}, \sigma, \mathcal{D}_{\sigma})$ together with SQL constraints $\text{UNQ}_1(B, Q)$ on $\text{ATTR}_1(B, Q)$, for mapping of each node v in $\mathcal{G}$ such that

$$v \text{ rdf:type } \mathbf{R}.$$

there exists mapping of at most node v' in $\mathcal{G}$ such that

$$v' \text{ } \mathbf{P} \text{ } v.$$

, where " $v' \text{ rdf:type } \mathbf{C}$ ". Thus, the node v conforms both the cardinality restriction and the property value type requirement for the constraint $\phi_{\mathbf{R}}$, therefore, $\llbracket \phi_{\mathbf{R}} \rrbracket^{\mathcal{G}, v, \sigma} = 1$ holds.

4. If $\phi_{\mathbf{R}}$ is a SHACL property shape $(\leq_0 \text{ } \mathbf{P}^- . \neg \mathbf{C})$ such that $\mathbf{P}$ and $\mathbf{C}$ are IRI identifiers of the forms $\text{'baseIRI/Q\#A,B,X,Y'}$ and 'baseIRI/C' , then the facts of the form $\neg \text{UNQ}_1(B, Q)$ and $\text{OP_IRI}_1(Q, A, B, C, X, R, Y, W)$ exist from the Theorem 1. From the definition of $\mathcal{G} = \mathcal{M}_{\mathcal{D}}(\mathcal{G}_{\mathcal{R}}, \sigma, \mathcal{D}_{\sigma})$ together with SQL constraints $\neg \text{UNQ}_1(B, Q)$ on $\text{ATTR}_1(B, Q)$, for mapping of each node v in $\mathcal{G}$ such that

$$v \text{ rdf:type } \mathbf{R}.$$

there exists mapping of zero or more nodes v' in $\mathcal{G}$ such that

$$v' \text{ } \mathbf{P} \text{ } v.$$

, where " $v' \text{ rdf:type } \mathbf{C}$ ". Thus, the node v conforms both the cardinality restriction and the property value type requirement for the constraint $\phi_{\mathbf{R}}$, therefore, $\llbracket \phi_{\mathbf{R}} \rrbracket^{\mathcal{G}, v, \sigma} = 1$ holds.

For the opposite direct $\mathcal{D}_{\sigma} \models \Sigma \iff \mathcal{G} \models \mathcal{S}$, similar proof arguments exists.

This concludes the proof of the theorem.