Specification of data and databases

Cédric Eichler

PROJET SENDUP
ANR-18-CE23-0010

Specification of data and databases

DELIVERABLE D2

Related task	Task 2
Partner	LIFO
Redactor	Cédric Eichler
Contributors	Adrien Boiret, Jacques Chabin, Cédric Eichler, Mirian Halfeld-Ferrari, Nicolas Hiot, Benjamin Nguyen, Sara Taki
Versioning	30/10/2023, V1: initial report

Presentation of this deliverable

The goal of the SENDUP project is to propose anonymisation mechanisms for data organized as graphs with an underlying semantic. Such mechanisms triggers updates on the database. This deliverable presents the data and databases used in SENDUP.

Contents

Chapter 1

Databases, representations and examples

1.1 RDF/S databases

1.1.1 Representation

A collection of RDF statements intrinsically represents a typed attributed directed multi-graph, making the RDF model suited to certain kinds of knowledge representation [1]. A schema on RDF facts can be expressed in RDFS (Resource Description Framework Schema). RDF/S databases are formalized in two ways in SENDUP: as classical triple-based RDF statements and as a typed graph.

Logical representation of RDF/S databases In [3] we find a set of logical rules expressing the semantics of RDF/S (rules concerning RDF or RDFS) models. Let $\mathbf{A}_C$ and $\mathbf{A}_V$ be disjoint countably infinite sets of constants and variables, respectively. A *term* is a constant or a variable. Predicates are classified into two sets: (i) SchPred = $\{CL,\ Pr,\ CSub,\ Psub,Dom,\ Rng\}$, used to define the database schema, standing respectively for classes, properties, sub-classes, sub-properties, property domain and range, and (ii) InstPred = $\{CI, PI, Ind\}$, used to define the database instance, standing respectively for class and property instances and individuals. An *atom* has the form $P(u)$, where P is a predicate, and u is a list of terms. When all the terms of an atom are in $\mathbf{A}_C$, we have a fact.

RDF/S databases as a typed graph RDF/S type graphs comprise 4 node types (Class, Individual, Literal, and Prop) and 6 edge types (CI, PI, domain, range, subclass, and subproperty). Each nodes have one attribute representing an URI, an URI, a value, and a name, respectively. PI-typed edges are the only ones with an attribute which represent the name of the property the edge is an instance of.

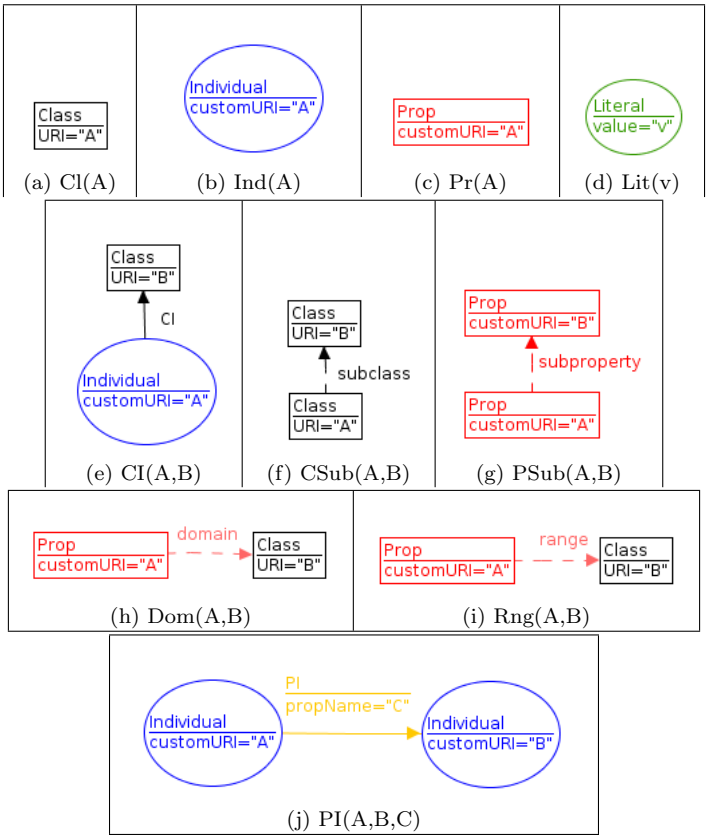

Figure 1.1: RDF triples in the type graph model

Fig 1.1 describes how each RDF triples are formalized and represented in the typed graph model.

Definition 1 (Database) *An RDF database $\mathscr{D}$ is a set of facts composed by two subsets: the database instance $\mathscr{D}_I$ (facts with predicates in* InstPred*) and the database schema $\mathscr{D}_S$ (facts with predicates in* SchPred*). We note $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ the typed graph that represents the same database. V are nodes with type in $\{CL, Pr, Ind, Lit\}$ and $\mathbb{E}$ are edges having type in $\{Dom, Rng, PSub, CSub, CI, PI\}$. The notation $\mathscr{D}/\mathbb{G}$ designates these two formats of a database.* □

1.1.2 Example

Fig. 1.2 shows an RDF instance and schema as a typed graph. The schema specifies that *Has Consequence* is a property having class *Drug* as its domain and the class *Effect* as its range. Property *Produces* is a sub-property of *Has Consequence* while *PosEffect* is a sub-class of *Effect*. Class *"rdfs:Resource"* symbolizes the root of an RDF class hierarchy. The instance is represented by individuals which are elements of a class (*e.g.* , *APAP* is an instance of

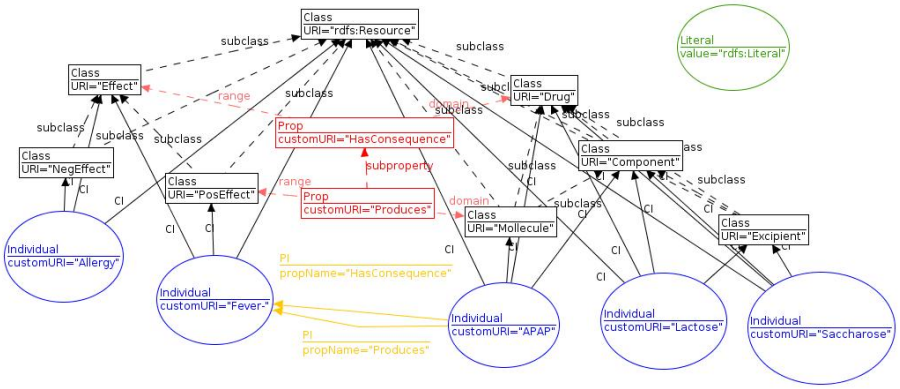

Figure 1.2: RDF schema and instance.

class *Mollecule*) and their relationships (*e.g.* , the property instance *Produces*, between $APAP$ and $Fever^-$).

The logical representation of this database is a set of facts. For instance facts such as *CL(Drug)* or *CSub(Drug, rdfs:Resource)* are for the schema description and *Ind(Saccharose)* or *CI(Saccharose, Excipient)* are for the instance description.

This dataset is mainly used as a motivating example for the management of consistent database updates.

1.2 Generic semantic graphs : the travels dataset example

1.2.1 Generic semantic graph representation

We consider databases to be modelled as attributed oriented multigraphs. In such models, it is customary for nodes and edges to have properties (among a finite set) and attributes (as words on a signature). In [2], RDF/S databases are modelled as a typed graph with 4 node types and 6 edge types. These types are inherent to RDF and thus the model can not be applied natively to arbitrary graph databases (*e.g.* neo4j).

We argue that considering a single node type and a single edge type having a single attribute (named *att* and *prop*, respectively) is in fact at least as expressive. Indeed, typing and additional properties can be encoded via special kinds of relation. We believe this model to be able to capture most –if not all- graph database representations.

1.2.2 Travel dataset

As an example of such datasets, we consider a graph database that contains information on travels, both professional and personal.

It has nodes for relevant entities, people and travels, whose attributes are an identifier. It also has nodes for every literal describing informations on those entities, e.g. last name, first name and address for people, date and destination for travels. We do not differentiate nodes representing entities or literals.

Its edges describe both relations between entities, e.g. "this person participated in this travel", represented by an edge of attribute `‘‘attends’’`, but also relations between entities and their information, e.g. "this person's name is in this literal", represented by an edge of attribute `‘‘name’’`. Typing falls within this second case e.g. "this node is a person" or "this literal is a city", represented by an edge of attribute `‘‘type’’`.

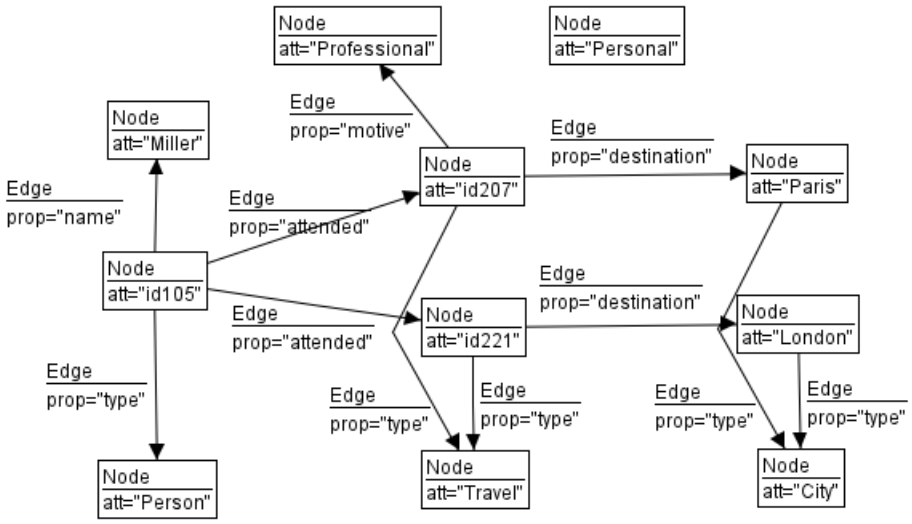

Figure 1.3: Running example: instance of a database

An example of such a database is provided in Fig. 1.3. In this instance, id105 (named Miller) attended travel id207 to Paris for professional reasons.

In this dataset, we consider the names to be direct indentifier, travels to be quasi identifiers, and the destination of personal travels to be sensitive.

Chapter 2

Real databases for experimentation

2.1 Dataset

For the experiments we rely on a real dataset: the Sentiment140 dataset with 1.6 million tweets[1]. This dataset has a simple schema, has exploitable semantics, and is big enough to conduct performance analysis. Its schema is shown in Fig. 2.1.

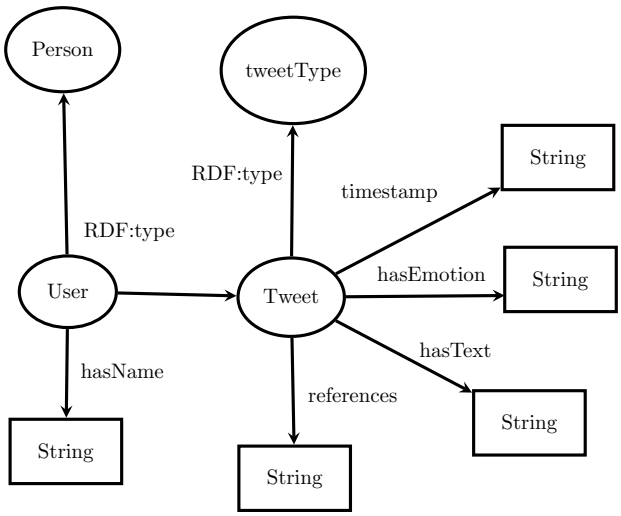

Figure 2.1: RDF schema for Sentiment140

[1] `https://www.kaggle.com/kazanova/sentiment140`

2.2 Parser

We provide code to parse this dataset, serialize it in XML and transform it into the two considered formats:

- In RDF/XML format[2].
- In the attributed multi-graph format supported by GrAnon[3].

In general, we consider the name of a user to be a direct identifier. Both the text of a tweet, the users it references and its timestamp may be considered as quasi-identifiers.

[2]https://github.com/sarataki/dp-projection-queries
[3]https://github.com/ceichler/granon/tree/master/src/executable

Bibliography

[1] Abiteboul, S., Manolescu, I., Rigaux, P., Rousset, M.C., Senellart, P.: Web data management. Cambridge University Press (2011)

[2] Chabin, J., Eichler, C., Ferrari, M.H., Hiot, N.: Graph rewriting rules for RDF database evolution: optimizing side-effect processing. Int. J. Web Inf. Syst. **17**(6), 622–644 (2021)

[3] Flouris, G., Konstantinidis, G., Antoniou, G., Christophides, V.: Formal foundations for RDF/S KB evolution. Knowl. Inf. Syst. **35**(1), 153–191 (2013)