Consistently mapping Differential Privacy paradigms between relational databases and RDF

Sara Taki[1], Adrien Boiret[1], Cédric Eichler[1], and Benjamin Nguyen[1]

Laboratoire d'Informatique Fondamentale d'Orléans, INSA Centre Val de Loire, Université d'Orléans, PETSCRAFT project-team, Inria, Bourges, France
{sara.taki,adrien.boiret,cedric.eichler,benjamin.nguyen}@insa-cvl.fr

Abstract. The Semantic Web represents an extension of the current web offering a metadata-rich environment based on the Resource Description Format (RDF) which supports advanced querying and inference. However, relational database (RDB) management systems remain the most widespread systems for (Web) data storage. Consequently, the key to populating the Semantic Web is the mapping of RDB to RDF, supported by standardized mechanisms. Confidentiality and privacy represent significant barriers for data owners when considering the translation and subsequent utilization of their data. In order to facilitate acceptance, it is essential to build privacy models that are equivalent, explainable, and usable within both data formats.
Differential Privacy (DP) has emerged to be the flagship of data privacy when sharing or exploiting data. Recent works have proposed DP-models tailored for either multi-relational databases or RDF. This paper leverages this field of work to study how privacy guarantees on RDB with foreign key constraints can be transposed to RDF databases and vice versa.
We use classical RDB and RDF formalisms as well as an established translation tool between the two (RDB2RDF) to compare DP models between the two data representations. We consider a promising DP model for RDB related to cascade deletion and demonstrate that it is sometimes similar to an existing DP graph privacy model, but inconsistently so. Consequently, we tweak this model in the relational world and propose a new model called *restrict deletion*. We show that it is equivalent to an existing DP graph privacy model, facilitating the comprehension, design and implementation of DP mechanisms in the context of the mapping of RDB to RDF. Conversely, we consider a useful DP model with label differentiation capabilities on graphs (QL-outedge), and propose its transposition into an original RDB distance.

Keywords: Differential Privacy · Relational Databases · Knowledge Graph · RDB2RDF mapping.

1 Introduction

The Semantic Web represents an extension of the current web standardized by the World Wide Web Consortium. It relies on the Resource Description Format (RDF) and provides metadata-rich, reusable, and shareable data. RDF is

a form of knowledge graph that can be coupled with ontologies such as OWL, thereby enhancing the semantic value of the data and inference capabilities. A key advantage of the Semantic Web is its ability to enrich data with well-defined semantics and to interconnect datasets through the RDF ontology language. A major motivation for this interlinking lies in the identification and integration of heterogeneous related data sources, significantly enhancing the value of using open datasets. As pointed out by reports such as Field *et al.* [1] and Michel *et al.* [2], many domains, such as neuroscience, biology, or social sciences, require combining and analyzing datasets that span multiple scales and representations. Achieving this integration demands explicit semantics that allow diverse databases to be interpreted in a common framework.

Currently, vast volumes of data still reside in relational databases (RDB), and relational database management systems (RDBMS), such as Oracle and PostgreSQL, remain among the most popular systems to manage data [1]. Mapping data from relational databases to RDF is a key to populate the Semantic Web. Transforming this massive amount of data into a machine-readable format is likely to facilitate the integration of various data sources and the emergence of new applications and innovative technological solutions. Mapping has been an active field of research during the last two decades [2,3,4] initiated by the RDB2RDF (Relational Database to Resource Description Format) incubator group[2].

However, the benefits of data integration and sharing also raise new concerns. Data collected (whether stored in RDB or RDF) can contain sensitive information. With the increasing attention on data privacy and the development of privacy regulations (e.g., the General Data Protection Regulation in the European Union), it is becoming increasingly important to protect sensitive information when sharing or allowing the utilization of data. Such concerns are a significant obstacle for RDB holders in accepting the translation and subsequent utilization of their data, as protection models vastly differ between RDF and RDB formats. *It is crucial to construct models that are equivalent and explainable within both formats, easing the comprehension of the guarantees provided in RDF within the familiar context of RDB.*

Differential Privacy (DP) [5] is a classical yardstick to measure privacy protection. Publication mechanisms that satisfy DP provide a form of indistinguishability. That is to say, it is difficult for an entity observing the output of a DP mechanism to determine which of several *adjacent* or *neighboring* databases was used as input. If two neighboring databases differ by the contribution of an individual, an external observer may therefore not know with high confidence whether the data pertaining to a particular individual has been used. Hence, it may not infer anything significant on such data. The concept of adjacency is thus a cornerstone of DP, defining what is protected. In the most simple context, a database is a single, monolithic table of records (or tuples) that holds private data. In this context, neighboring databases are those that differ by one

[1] See : https://db-engines.com/en/ranking
[2] `http://www.w3.org/2001/sw/rdb2rdf/`

record, meaning that DP protects (or "hides") the presence or absence of any single record in the database [5]. The intuition here is that each individual participates in, at most, one database record and therefore DP indeed protects the contribution of each individual.

Defining neighborhoods for multi-relational databases, i.e., databases composed of many tables, is challenging for many reasons (see, for instance, [6]). The introduction of several relations usually comes with *constraints*, each constraint arising from the semantics of the database. It is thus no longer possible to define an adjacent database simply by adding or removing a tuple in a table since this may violate the database constraints. In this paper, we consider an important type of constraint, foreign key (FK) constraints, sometimes associated to cardinality constraints.

In graphs, however, there exists several notions of distances where neighbors differ only locally, the most classic being the node distance (where two neighboring graphs differ only by the deletion of a node and its incident edges) and the edge distance (where two neighboring graphs differ only by the deletion of a single edge). We note that those different distances create different neighborhoods, and it turn, lead to DP models that provide different guarantees.

This paper focuses on distances and neighborhoods in the RDB and RDF worlds that are equivalent through standard translation mechanisms to build DP-models that are equivalent in both. More specifically, we focus on the following objectives:

O1 Establish a framework to match or compare DP models between the RDB and RDF formalisms. DP models can be characterized by their distance, which means we want mappings of databases from one formalism to the other with some distance-preserving properties.
O2 Apply this framework to the DP model on RDB centered around cascade deletion [7]. We aim to find a matching RDF DP model, and study potential adjustments.
O3 Apply this framework to the DP model on RDF centered around outedges of specific labels [8]. We aim to find a matching RDP DP model, and characterize mapping choices for the comparison to be meaningful.

The fulfillment of these objectives would offer the benefit of a common ground that actors considering DP in RDB and RDF can share, and leverage it to offer a variety of DP models, actable in both formalisms, that can provide a wide variety of privacy guarantees, depending on a specific application's requirements.

Contributions

To meet these objectives, this paper:

– formalizes the notion of encoding (or mapping) from RDB to RDF, consistent with R2RML mapping, that subsumes standard-compliant W3C recommendations.

- formalizes a generalized notion of cascade deletion in RDB covering both transitive deletion and our proposal.
- studies the translation in RDF of the DP model relying on transitive deletion.
- introduces a meaningful relaxation of this model and demonstrates that it is equivalent to an existing graph privacy model through mapping.
- proposes a RDB model and R2RML encoding choices that make it equivalent to QL-Outedge privacy [8], which provides semantics for the neighborhoods.

The remainder of this paper is structured as follows. The next section details relevant related works on mapping and DP. Section 3 introduces an illustrative example as well as the formalization of the considered databases and mappings. Section 4 proposes an analysis of the translation of transitive deletion in RDF and details our proposed relaxation, demonstrating its equivalence to a well-established DP model in graphs through mapping. Section 5 proposes a novel distance in relational databases which, under adequate encoding, is equivalent to the concept of QL-Outedge privacy. Adequacy of these encodings and their impact on privacy is further illustrated in Section 6. Finally, Section 7 concludes this paper and discusses future work.

This paper is an extended version of our previous works [9,10]. In [9], we first introduced the problem addressed here, but only in an informal manner and without formal proofs of model equivalency through mappings. A first formalization, focusing on cascade deletion, was later proposed in [10]. Building on these foundations, the present paper extends the formalization of RDB and RDF to the encodings that connect them, and broadens the comparative study of distances to include QL-outedge. This requires particular choices in R2RML mappings in order to admit an equivalent in RDB.

2 Related Work

This section introduces background on RDB to RDF mapping, before introducing DP and its adaptations to RDB and RDF databases. *To the best of our knowledge, this paper is the first at the intersection of these two fields*, focusing on the impact of the RDB to RDF translation on DP-models and the definition of equivalent DP-models in both worlds.

2.1 Mapping RDB to RDF

In September 2012, the RDB2RDF Working Group published two Recommendations: Direct Mapping (DM) [11] and customized mapping (CM) R2RML [12].

The W3C DM recommendation defines simple mapping rules to map relational data to RDF [13]. The RDF generated straightforwardly is based on the structure of the database schema. URIs are automatically generated [14]. Many-to-many relations in relational databases are generally represented as a join table where all its columns are foreign keys (FKs) to other tables (n-ary relations).

One missing part from the DM is to represent many-to-many relations as simple triples [15]. When DM is applied, the join table will be translated into a distinct class, which conflicts with the canonical representation of many-to-many relationship in RDF.

CM R2RML [12] is a RDB to RDF mapping language that allows to manually customize the mapping. The expert user has to know the RDB and the domain ontology to express the schema utilizing an existing target ontology. The W3C RDB2RDF Working Group proposed a set of core requirements for R2RML [16], including the exposition of many-to-many join tables as simple triples [15].

Due to the representation of many-to-many join tables as simple RDF triples, we consider mapping mechanisms conform to the R2M2RL specifications.

Extended mapping models. RML [17][18] extends the R2RML mapping language to support the mapping of data sources with diverse formats, including data formats like XML, CSV/TSV and JSON. However, it does not tackle the constraints associated with handling various kinds of databases and query languages. xR2RML [19], a mapping language developed as an extension of R2RML and RML. Beyond relational databases, xR2RML also supports the mapping of many non-relational databases to RDF. It is intended to flexibly adjust to diverse data models and query languages. In addition, it can handle data under heterogeneous formats. In the rest of the article, we focus on R2RML as it is the original standard, but adapting our work to cover extensions such as RML and xR2RML could be an interesting perspective.

2.2 Differential privacy

Differential Privacy (DP) proposes a robust mathematical framework for privacy protection [20]. An algorithm respects DP if observing its output does not permit to determine with strong confidence which of several neighboring dataset was used as input.

Definition 1 (ϵ-differential privacy). *Given $\epsilon > 0$, a function $f : \mathcal{X} \to \mathcal{S}$ and a distance d over $\mathcal{X}$, is ϵ-differentially private if, for any couple of datasets $(D, D') \in \mathcal{D}^2$ such that $d(D, D') = 1$, and for any $S \subseteq \mathcal{S}$:*

$$Pr[f(D) = S] \leq e^{\epsilon} \times Pr[f(D') = S]$$

where probability Pr is over the randomness of f.

Parameter ϵ is also known as the privacy budget, a smaller value indicating stricter privacy requirements. Two datasets at a distance one are said to be neighbors. One classical way of achieving DP for a function (e.g. a query) f is to add an appropriate amount of noise to its results, calibrated by the *global sensitivity* (GS) of f. GS measures the maximal variation of the query result when evaluated upon any two neighboring databases.

Definition 2 (Global Sensitivity (GS)[5]). *For a function $f : \mathcal{D} \to \mathcal{S}$ and a distance d over $\mathcal{X}$ for all datasets $(D, D') \in \mathcal{D}^2$:*

$$df = GS_f = \max_{D,D':d(D,D')=1} \parallel f(D) - f(D') \parallel_1$$

where $\|\|\|_1$ *denotes the L1 norm.*

GS depends only on f, the considered space of databases $\mathcal{X}$, and the distance d it is associated with (i.e., that identify neighboring databases). It is independent of the database itself. For queries with low GS, only a small magnitude of noise needs to be added to respect DP. On the other hand, when the GS is high, a substantial amount of noise must be injected to achieve DP, which will impair data utility.

DP for RDF DP is immediately applicable to any space $\mathcal{X}$ given a proper distance d or notion of neighborhood over $\mathcal{X}$. When considering graphs, two models prevail : *node-DP* and *k-edge-DP*. We consider a third, more recent, proposal integrating semantics, *QL-edge-labeled-DP*.

In node-DP, neighboring graphs differ by a single node and all its incident edges, thereby protecting each node along with its incident edges. While node-DP is the strongest of these models, it poses a particular challenge: two neighboring graphs can differ by an arbitrarily large number of edges, which may lead to high variations in outputs across the neighborhood and result in low-utility DP mechanisms.

k-edge-DP [21] is a looser model in which two graphs are adjacent if they differ by up to k edges. Compared to node privacy, edge privacy is limited to protecting k relationships. 1-edge-DP is simply called edge-DP and the most commonly employed in the literature.

We note that these two models do not discriminate nodes or edges based on attributes or labels. Such considerations have value in privacy, as information is not uniformly sensitive: for instance, the timestamp of a tweet is usually considered less sensitive than a personal address. Reuben [7] studied the adaptation of DP to edge-labeled directed graphs by defining sets of sensitive labels to which the protection is restricted. Reuben introduced the notion of QL-edge-label neighboring graphs: graphs that differ by a set of outedges of a node with specific labels. The underlying idea behind this definition is that, for example, in RDF graphs, some relations of an entity may be innocuous and some should be considered sensitive, shown by particular labels. Given this neighboring definition, the author presented **QL-edge-labeled-DP**, that considers edges' semantics by only protecting edges of a given set QL (i.e., *sensitive* labels). As such, it only protects a predetermined subset of edges. This notion can be transposed to most models. For example, k-typed-edge DP could be defined as the model where two adjacent graphs differ by up to k *sensitive* edges.

Reuben considers the outedges of a node since it denotes the contributions made by that node within the graph. This semantically grabs the idea of the presence of an individual in a graph while not being present in another graph, analogous to how private data is modeled as a tuple in the relational model. Throughout this article, we will use **QL-Outedge DP** as a synonym for **QL-edge-labeled DP** to better reflect the considered model. The related distance for QL-Outedge privacy is presented in [8].

DP for multi-relational databases In the DP literature [5], a database is commonly a single, monolithic table of records (or tuples) that holds private data. Multi-relational databases, i.e., databases composed of many tables, are less popular. However, DP has also been investigated in this setting [22,23,24].

PINQ [22] and FLEX [23] consider a simple definition of neighboring databases, which does not consider foreign key (FK) constraints. According to their definition, neighboring databases possess the same set of relations and attributes and differ by exactly one tuple in one relation.

The PrivateSQL system [24] introduces a richer notion of neighboring databases that considers constraints in the schema, in particular primary and FK constraints. Under this model, upon deletion of one tuple from one relation, many tuples in other relations have to be deleted because of the existence of FK constraints. PrivateSQL enables privacy to be designated at multiple resolutions. The data owner can flexibly designate which entities in the schema need privacy. The key idea is that one relation is specified to be the primary private relation. Privacy protection extends to additional private relations linked to the primary one via FKs, which are called secondary private relations.

Under this DP policy, two database instances are considered neighbors when one can be obtained from the other by deleting a record x from the primary private relation and cascade deleting other records that refer to x through FKs. One requirement in this approach is that the schema needs to be acyclic. Based on this proposal, researchers began to consider FK constraints when defining neighboring databases [25,26].

Due to the consideration of FK constraints, the familiarity of cascade deletion on which its neighborhood concept relies, and the general adoption of the model presented in [24], we adopt this model as the starting point for O2 and aim at providing an equivalent model in RDF. The related formal distance definition will be restated in our model in Definition 9.

3 Setting: formalizing the concepts

This section introduces an illustrative example based on a Twitter dataset that will be used in the remainder of the paper. It then proposes the formalization of RDB, graphs representing RDF databases, and the mapping from RDB to RDF.

3.1 Illustrative example

Illustrative dataset In this paper, we use as an illustrative example a simple Twitter database, inspired by the Sentiment140 dataset composed of 1.6 million tweets [3]. Its ER diagram is presented in Fig. 1 and an instance of the database is illustrated in Fig. 2.

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

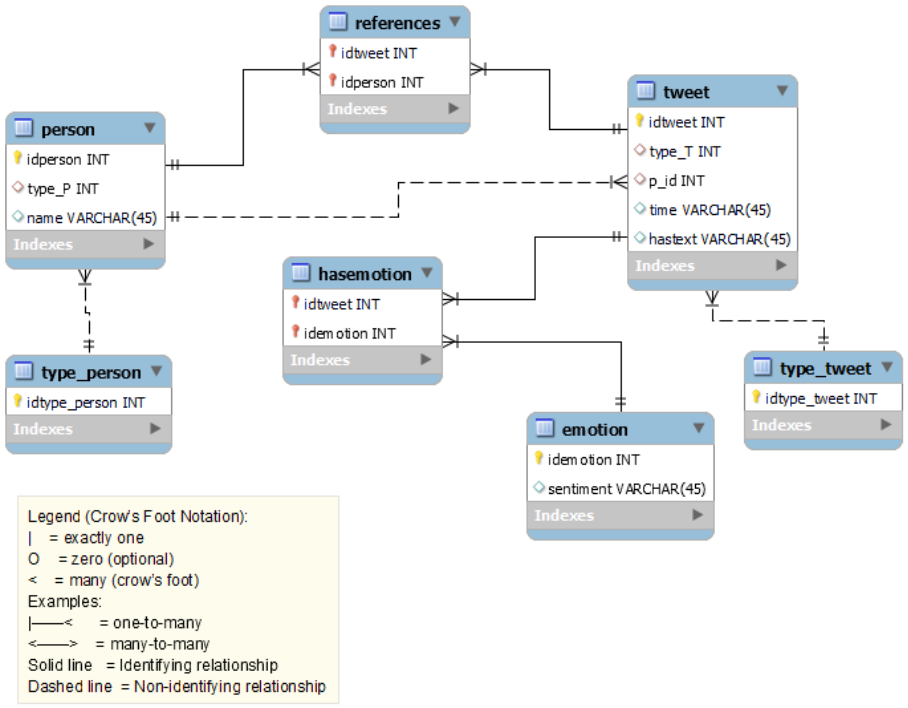


Fig. 1: Entity-Relation Schema of the Sentiment140 dataset

Person

idperson	type_P (fk)	name
1	100	Alice
2	100	Bob
3	100	Clara

Tweet

idtweet	type_T (fk)	p_id (fk)	time	hastext
30	200	1	Jan	This is a tweet
31	200	1	Feb	HelloWorld
32	200	2	March	What

Emotion

idemotion	sentiment
0	negative
4	positive

Type_tweet

idtype_tweet
200

Type_person

idtype_person
100

References

idtweet (fk)	idperson (fk)
30	2
31	3
32	1

hasemotion

idtweet (fk)	idemotion (fk)
30	0
31	0
32	4

Fig. 2: A database instance of the schema in Fig. 1

This example presents two many-to-many relationships: 1) between Person and Tweet, captured by the References table with FK referencing the id of a

tweet and the id of the persons it references; 2) between Tweet and Emotion, captured by the HasEmotion table. The Tweet table possesses a FK "p_id" from table Person referencing the person that authored the tweet.

Our mapping process As we aim to compare mechanisms and DP models in RDB and RDF, we require a framework in which we can specify encodings, or mappings, from one to the other. We select R2RML-F [27], an R2RML implementation available on Github[4]. The R2RML mapping process is done with R2RML-F whose engine takes as input the RDB, the R2RML mapping file that dictates the direction of relations, and the format of the output file to generate RDF data.

R2RML represents one-to-many and many-to-many relations as simple RDF triplets, which is to say edges in our graphs. Since RDF is an oriented formalism, such mappings require user input to specify in which directions such triplets will end up pointing. We will manually write R2RML mapping files exploring the consequences of those decisions. For instance, in Fig. 3a, the mapping of many-to-many relations, such as *References* and *HasEmotion*, can be made in either direction. We choose that Tweet references go from tweet to person, and Emotion labelling goes from tweet to emotion, but this choice is left up to the user by the W3C recommendations [5]. For the one-to-many relation that links a tweet to its author (modeled by the foreign key *p_id* in the *Tweet* table). R2RML also allows for this relation to be oriented either way. A default mapping would follow W3C recommendations to model it by an edge going from the referencing table (here, *Tweet*) to the referenced table (here, *Person*) as depicted in Fig. 3a. However, for our considerations, we will often consider non-default mappings that do not follow this rule, for instance in Fig. 3b, which depicts a fragment of another encoding of this database, where the edge representing the relation between *Tweet* and *Person* is "backwards", which we will find to sometimes be more appropriate (see Section 6). As such, the fact that "person 2 has tweeted tweet 32" is translated as an edge going from r:person_2 to r:tweet_32.

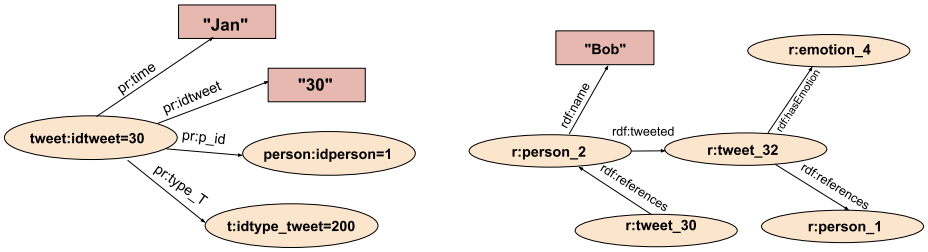
The R2RML-F engine takes the Twitter relational database and the R2RML mapping file as input and generates the output file in turtle format (.ttl) available online [6]. We use R2RML-F for the R2RML mapping code. However, the code (from 2012), needed some updates. Our 2025 runnable code for R2RML-F is available at :`https://github.com/sarataki/mapping/tree/main/code`.

Throughout this paper, IRIs are simplified using prefixes. In lisiting 1.1 we present all the prefixes used. For instance "rdf:" is a shorthand for the full prefix "http://www.w3. org/1999/02/22-rdf-syntax-ns#".

Listing 1.1: Prefixes

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX tweet: <http://foo.example/DB/tweet/>
```

[4] `https://github.com/chrdebru/r2rml`
[5] `https://github.com/sarataki/mapping/tree/main/defaultR2RML/r2rml.ttl`
[6] `https://github.com/sarataki/mapping/tree/main/defaultR2RML/output.ttl`



(a) ...with a default mapping (b) ...with another, arbitrary, mapping

Fig. 3: Extracts of the database mapped from Fig. 2...

```
PREFIX person: <http://foo.example/DB/person/>
PREFIX references: <http://foo.example/DB/references/>
PREFIX reference: <http://foo.example/DB/references#>
PREFIX pr: <http://foo.example/DB/tweet#>
PREFIX t: <http://foo.example/DB/type_tweet/>
PREFIX r: <http://example.com/resource/>
```

3.2 Relational Database

We use a conventional notion of schema for relational databases.

Definition 3 (Database Schema).

- *A **table schema** T is a set of attribute names.*
- *A **primary key constraint** PK_T on T is a subset of the attributes of T.*
- *A **foreign key constraint** ϕ_{l_0,l_1} from T_0 to T_1 is a pair of equal-length lists (l_0, l_1) of attributes of T_0 and T_1 respectively.*
- *A **database schema** $\mathcal{D}$ is a finite set of tables $\mathcal{T}$ and of constraints $\mathcal{C}$ such that each table T has exactly one primary key constraint PK_T in $\mathcal{C}$, and for all foreign key constraint ϕ_{l_0,l_1} from T_0 to T_1, $PK_{T_0} = l_0$.*

We assume tables and attribute sets are disjoint, and that attributes do not have types or domains for this formalism.

Definition 4 (Database). *A **database** following a certain database schema $\mathcal{D} = (\mathcal{T}, \mathcal{C})$ is a set D of elements x (called records) such that:*

- *x belongs to exactly one table $T \in \mathcal{T}$, we note $x \in T$*
- *For each s an attribute of T, we note $x.s$ the value of attribute s for x. If it is undefined, we say $x.s = null$. By extension, if l is a n-uplet of attributes, $x.l$ is the n-uplet of their values in x.*

The constraints $C \in \mathcal{C}$ are interpreted as such:

- ***Primary key*** *For all PK constraint PK_T, for all $x \in T$, for each $s \in PK_T$, $x.s$ is defined, and if $x \neq x'$ then there exists $s \in PK_T$ such that $x.s \neq x'.s$.*
- *For all FK constraint ϕ_{l_0,l_1} from T_0 to T_1, for all $x' \in T_1$, there exists a unique element $x \in T_0$ called its* antecedent *such that $x.l_0 = x'.l_1$.*

In this setting, we identify a common type of tables in relational databases. A **relation table** is a table whose primary key contains all its attributes, and is composed of the disjoint union of the domain of two foreign keys. Other tables are called **entities**.

In the classical entity-relational model, many-to-many relations are stored in relation tables, whereas one-to-many relations are directly embedded in entities through foreign keys. In our example database, the relation "tweeted" is a one-to-many, as each tweet has only one author, and is thus stored as a foreign key in the Tweet table. However, the "references" relation is many-to-many, as a tweet can reference several people, and a person can be mentionned in several tweet. The table References stores this relation as pairs of foreign keys from Person and Tweet.

We say that a database schema $\mathcal{D} = (\mathcal{T}, \mathcal{C})$ is **ER-compliant** if for all FK constraints ϕ_{l_0,l_1} from some T_0 to some T_1:

- T_0 is an entity
- either T_1 is a relation and all attributes of l_1 are in its primary key PK_{T_1}, or T_1 is an entity and no attribute of of l_1 is in its primary key PK_{T_1}.

3.3 Graph Database and distances

We present a brief definition of graph databases. It is classical, but distinguishes attributes in a way that will facilitate encodings between RDB and RDF. A RDF dataset is represented as a graph:

Definition 5 (Graph Database). *A graph database is a tuple (A, L, V, E) such that:*

- *A is a potentially infinite set of attribute values*
- *L is a potentially infinite set of edge labels*
- *V is a finite set of vertices*
- *$E \subseteq V \times L \times (V \cup A)$ is a set of edges. In an edge v, l, v' we call v the subject, l the predicate, v' the object.*

In this definition, RDF triples are modeled as edges, either between two nodes or from a node to one of its attributes. Attributes here correspond to literals in RDF: they cannot be the subject of a relation and cannot appear isolated. We note that those attributes are not nodes themselves, and will not be counted as such in future distances. L denotes all possible predicates and A denotes the domain of definition of literals that may be object of a predicate.

In the figures, by convention, we represent nodes as yellow ovals and attributes as red rectangle. For example, in Fig. 3b, the node "r:person_2" has

an out-edge labeled "rdf:name" whose destination is an attribute "Bob". This represent an RDF triple whose object has URI "r:person_2", with predicate "rdf:name" and object the literal string "Bob". From the remainder of the graph, we see that the individual named Bob is the author of "r:tweet_32" which references "r:person_1", etc.

3.4 Encoding formalizing a mapping

The default mapping of the RDB presented in Fig. 2 is illustrated in Fig. 2. To produces another mapping, one may manually write R2RML mappings, which are tailored to their database schema. The choice of an encoding, or mapping, from RDB to graphs modeling a RDF dataset, can be characterized as picking a direction for all those triples. In our formalism, it is defined as follows:

Definition 6 (Orientation). *Let $\mathcal{D} = (\mathcal{T}, \mathcal{C})$ be an ER-compliant database schema. An orientation σ is a function that associates to each relation table and foreign key between entities a direction.*

- *For all T relation table, there exists two FK constraints $\phi_{l_0,l}$ from T_0 to T and $\phi_{l_1,l'}$ from T_1 to T such that l, l' are a partition of the attributes of t. An orientation for T is then (l, l') (from l to l') or (l', l) (from l' to l). We can also note these orientations (l_0, l_1) and (l_1, l_0), or (T_0, T_1) and (T_0, T_1) if $T_0 \neq T_1$.*
- *For all FK constraint ϕ_{l_0,l_1} from entity T_0 to entity T_1, an orientation is (l_0, l_1) (from l_0 to l_1) or (l_1, l_0) (from l_1 to l_0). We can also note these orientations (T_0, T_1) and (T_0, T_1) if $T_0 \neq T_1$.*

Definition 7 (ER encoding). *Let $\mathcal{D} = (\mathcal{T}, \mathcal{C})$ be an ER-database schema, and σ an orientation. An encoding of relational databases following $\mathcal{D}$ into graphs is an injective function f from the set of all relational databases following $\mathcal{D}$ into the set of graphs, that matches all ER-database D following $\mathcal{D}$ a graph $f(D) = (A, L, V, E)$ such that:*

- ***Nodes:*** *For each entity T, for each $x \in T$ in D, there is a node $x \in V$*
- ***Labels:*** *The labels L are the union of*
  - *The attributes s of all the tables of $\mathcal{T}$*
  - *The relation tables*
  - *The entity-to-entity foreign key constraints*
- ***Attributes:*** *For each entity T, for each attribute s such that for no FK constraint ϕ_{l_0,l_1}, $s \in l_1$, for each $x \in T$ with a defined value for s $x.s = a$ in D, there exists a value $a \in A$ and an edge $(x, s, a) \in E$*
- ***Many-to-many relations:*** *For each relation T, its two FK constraints $\phi_{l_0,l}$ from T_0 to T and $\phi_{l_1,l'}$ from T_1 to T such that l, l' are a partition of the attributes of t, of orientation $\sigma(T) = (l, l')$, for all $x \in T$, $y \in T_0$ the antecedent of $x.l$, and $z \in T_1$ the antecedent of $x.l'$, there is an edge $(y, T, z) \in E$*

- ***One-to-many relations:*** *For all FK constraint ϕ_{l_0,l_1} from entity T_0 to entity T_1, for all $x \in T_1$, $y \in T_0$ the antecedent of $x.l_0$, if $\sigma(\phi_{l_0,l_1}) = (l_0, l_1)$ there is an edge $(y, \phi_{l_0,l_1}, x) \in E$, if $\sigma(\phi_{l_1,l_0}) = (l_1, l_0)$ there is an edge $(x, \phi_{l_0,l_1}, y) \in E$*
- *There is no other node, attribute, or edge in $f(D)$*

In the rest of this paper, we will consider distances and how they are preserved through encodings. An ER encoding is an **isometry** w.r.t. a distance d on relational databases and a distance d' on graph databases iff for all D, D' relational databases following $\mathcal{D}$, if $d(D, D')$ is defined then $d'(f(D), f(D'))$ is defined and equal to $d(D, D')$.

For our recurring example, one possible encoding of some entries of Fig. 2 is the graph of Fig. 3b. The R2RML mapping file [7] and the output file are available online [8]. The entities (e.g. person 2, tweet 32, emotion 4) are translated into nodes. However, relations such as References and foreign key relations such as the *p_id* foreign key in the *Tweet* table become labellings in L and are represented as edges, e.g. (r:tweet_32,rdf:references,r:person_1).

4 Distances and isometries for DP in RDB and RDF

In this section, we formalize two classic distances, namely cascade deletion distance in RDB (4.1) and node distance in RDF (4.2). Because encodings transform entities into nodes, the cascade distance allows the deletion of several entities at once, but the node distance does not allow the deletion of several nodes at once, no encoding is an isometry between both distances in the general case. We present on the one had a RDF distance that matches cascade deletion (4.3), and on the other hand a RDB distance that matches the graphs' node distance (4.4).

4.1 Cascade Deletion in RDB

We first present a generalized notion of cascade deletion. Then we show the special case that corresponds to the transitive deletions introduced by Kotsogiannis et al. [24], and we analyze this notion on the RDB example, Twitter.

In general, a cascade deletion is the repercussion of the deletion of an element in a table to all other elements that depended on it in others. The characterization of such dependencies usually revolves around foreign keys, but may vary from a formalization to another. For this reason, we define here the cascade deletion as parameterized by its dependencies.

Definition 8 ($\mathcal{C}'$ Cascade Deletion). *Let D be a database on a schema $\mathcal{D} = (\mathcal{T}, \mathcal{C})$, and $\mathcal{C}' \subseteq \mathcal{C}$ a set of FK. Let x be an element of D. The cascade deletion of x alongside $\mathcal{C}'$ defines a set of **deleted elements** $L_{rm(x)}$ as the smallest set of lines such that:*

[7] https://github.com/sarataki/mapping/tree/main/propR2RML/r2rml.ttl
[8] https://github.com/sarataki/mapping/tree/main/propR2RML/output.ttl

- $x \in L_{rm(x)}$
- *If z is an element in T_1, such that there exists a foreign key $\phi_{l_0,l_1} \in \mathcal{C}'$ from T_0 to T_1, and the antecedent of z by ϕ_{l_0,l_1} is $y \in L_{rm(x)}$, then $z \in L_{rm(x)}$.*

*This is turn defines a set $A_{rm(x)}$ of **deleted attributes**, pairs (y, s) such that:*

- *$y \in T_1$ is not in $L_{rm(x)}$*
- *there exists a foreign key constraint $\phi_{l_0,l_1} \notin \mathcal{C}'$ from T_0 to T_1*
- *the antecedent of y by ϕ_{l_0,l_1} is in $L_{rm(x)}$*

The result of the cascade deletion of x in D is a database D' of schema $\mathcal{D} = (\mathcal{T}, \mathcal{C})$ whose elements are all the elements of D that are not in $L_{rm(x)}$ where for every $(y, s) \in A_{rm(x)}$, $y.s$ is set to null.

We use this deletion formalism to define a distance:

Definition 9 ($\mathcal{C}'$ Cascade Distance). *Let D, D' be two databases of same schema, and $\mathcal{C}'$ a set of entity to entity FK. We say that D and D' are $\mathcal{C}'$ Cascade neighbors if D' is the result of the cascade deletion of an element x in D alongside $\mathcal{C}'$, or D is the result of the cascade deletion of an element x in D' alongside $\mathcal{C}'$.*
The $\mathcal{C}'$ cascade distance is defined over databases of same schema as the length of a shortest path connecting two databases neighbor by neighbor, if it exists.

We note that this definition is "eager" in its deletion, which is to say elements are deleted as soon as one of their relevant antecedents is deleted. There exists another, "cautious" (or lazy) cascade deletion, where elements get deleted only if all their relevant antecedents are deleted. The definition of transitive deletions [24] uses this eager deletion strategy. Our own proposed distance (4.4) will circumvent the problem by limiting deletions in a way this distinction no longer matters.

We also note that if C' does not contain all FK of relation tables, it is possible to have "dangling" relation records that potentially break primary key unicity. For simplicity's sake, we will focus on the cases where it does not happen.

4.2 Classical RDF node distance

In graphs, records in entities are encoded as nodes. As such, the classical measure most closely related to cascade deletion of a record is the node distance.

Definition 10 (Node Deletion). *Let $G = (A, L, V, E)$. We call E_{rmv} the set of edges incident to v: $(v_0, l, v_1) \in E_{rmv}$ iff $v_0 = v$ or $v_1 = v$. The result of the node deletion of v in G is a graph $G' = (A, L, V \backslash \{v\}, E \backslash E_{rmv})$.*

Accordingly, the transposition of the node distance in this formalism is:

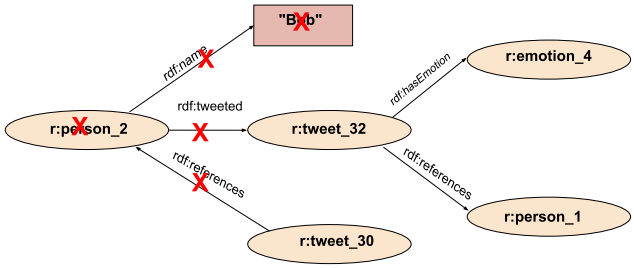


Fig. 4: Node deletion

Definition 11 (Node Distance). *Let $G = (A, L, V, E)$ and $G = (A, L, V', E')$ be two graph databases. We say that G and G' are node-distance neighbors if G' is the result of the deletion of a node $v \in V$ in G, or G is the result of the deletion of a node $v \in V$ in G'.*
The node distance is defined over graphs of same labels as the length of a shortest path connecting two databases neighbor by neighbor, if it exists.

Figure 4 illustrates the deletion of node "r:person_2" from the graph pictured in Fig. 3b. All the edges incident to it are deleted (those labeled "rdf:name", "rdf:tweeted", and "rdf:references"). While still formally in A, the attribute "Bob" does not appear in the graph anymore since the triple it was the object of has been suppressed. We recall that literals cannot appear while isolated but do not count this as the suppression of a node toward the distance. The resulting graph, containing 4 nodes and two edges, is a node-neighbor of the original graph.

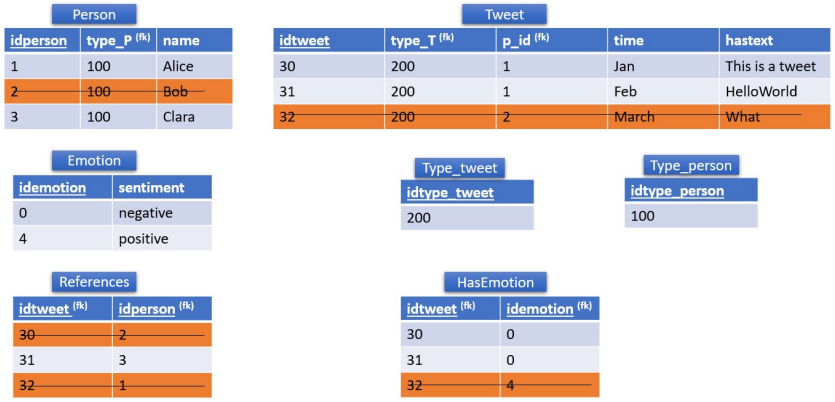
4.3 Cascade Deletion Distance in Graphs

It is immediate that in the general case, encodings cannot hope to be isometric from the RDB cascade distance to the RDF node distance. Indeed, some RDB cascade deletions lead to the deletion of several records in entity tables. In an encoding, this would translate as the deletion of several nodes. However, the RDF node distance only allows the deletion of one node at a time.

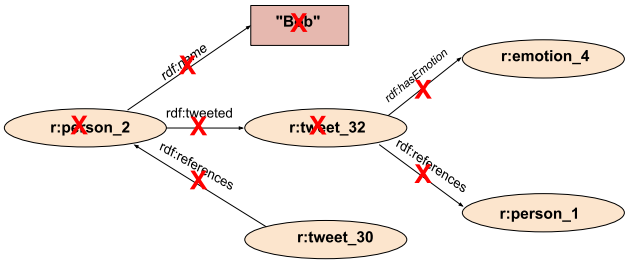
We present herein the graph distances and encodings for which this isometry holds. In such cases, comparisons of privacy guaranteeing mechanisms across formalisms would be possible. The definition of a cascade deletion in graphs would follow certain labels as follow:

Definition 12. *Let $G = (A, L, V, E)$ be a graph database and $L' \subseteq L$ a set of labels. Let x be an element of V. The cascade deletion of V alongside $\mathcal{L}'$ defines a set of **deleted nodes** $V_{rm(x)}$ as the smallest set of nodes such that:*

- $x \in V_{rm(x)}$
- *If $l \in L'$, $y, z \in V$ such that $(y, l, z) \in E$, and $y \in V_{rm(x)}$, then $z \in V_{rm(x)}$.*



(a) in RDB



(b) in RDF

Fig. 5: Transitive cascade deletion with Person as primary entity

This is turn defines a set $E_{rm(x)}$ *of* ***deleted edges****: for* $(y, l, z) \in E$, $(y, l, z) \in E_{rm(x)}$ *iff* $y \in V_{rm(x)}$ *or* $z \in V_{rm(x)}$. *This also defines a set* $A_{rm(x)}$ *of* ***deleted attributes****,* $a \in A_{rm(x)}$ *if for all* y, l *such that* (y, l, a), $y \in V_{rm(x)}$

The result of the cascade deletion of x *along* L' *in* G *is* $G' = (A \backslash A_{rm(x)}, L, V \backslash V_{rm(x)}, E \backslash E_{rm(x)},)$

Definition 13 (Cascade Deletion Graph Distance). *Let* $G = (A, L, V, E)$ *and* $G = (A', L, V', E')$ *be two graph databases. We say that* G *and* G' *are* L'*-cascade deletion-distance neighbors if* G' *is the result of the deletion of a node* $v \in V$ *along* L' *in* G, *or* G *is the result of the deletion of a node* $v \in V$ *along* L' *in* G'.

The L'*-cascade deletion-distance is defined over graphs of same labels as the length of a shortest path connecting two databases neighbor by neighbor, if it exists.*

This cascade deletion works in a way that propagates through some edges of label $l \in L'$. This means that this graph cascade deletion is dependant on the

choice of encoding, which is to say on our chosen R2RML mapping. However, for the (natural) choices of encodings and L', this new distance matches RDB cascade deletion, which is immediate by construction.

Lemma 1. *Let $\mathcal{D} = (\mathcal{T}, \mathcal{C})$ be a database schema, $\mathcal{C}'$ a set of FK (containing all relation table FK). We call $L' \subseteq \mathcal{C}'$ the entity to entity FK of $\mathcal{C}'$. Let σ be an orientation such that for all $\phi_{l_0,l_1} \in \mathcal{L}'$, $\sigma(\phi_{l_0,l_1}) = (l_0, l_1)$. Then the σ-encoding from RDB to RDF is an isometry from the C' cascade deletion distance in RDB to the L' cascade deletion distance in RDF.*

For instance, to model cascade deletions in a way that corresponds to [24], we consider that one can compute a join between tables starting from a primary entity T and following all entity to entity foreign key constraints. To study the impact a deletion in the primary table would have on the join, we can delete every line of every table that would no longer occur in it. This corresponds to a transitive deletion alongside those foreign keys.

As an example, the transitive deletion of Bob in the Person table, illustrated in Fig. 5, cascades to the Reference table as tweet 30 can no longer reference him, but also leads to the deletion of his tweet (tweet number 32) which in turns deletes two more lines in the database, one in References, one in HasEmotion.

Limitations. We now discuss the two limitations of cascade deletion as presented here as compared to Kotsogiannis *et al.* [24]. First, for the approach of [24], the choice of a primary table is restricting. While the join approach and transitive deletion as described in [24] necessitates picking a starting point, this has the undesirable side-effect of locking the privacy model towards certain protections and away from others. In RDF, it is possible to use a privacy model protecting any node (DP with node distance) or even nodes from one or several tables exclusively (DP with type-node distance). This is not always possible in databases once a primary table is picked. For instance, in the given database, if we pick Person as the primary table, Fig. 5a shows the only possible way to delete tweet 32. It is then impossible to delete a specific tweet without deleting its author and all its other tweets. In turn, choosing Tweet as a primary table would make it impossible to protect a Person. In a privacy setting, this a restriction that (typed) node DP would not exhibit.

Furthermore, cascade deletion can have a greater or lesser impact based on the chosen starting table. To illustrate this point, in Fig 6, we show a cascade deletion starting from a tweet. In the corresponding RDF graph, this leads to the deletion of a single node and all its adjacent edges, which is coherent with a node distance of 1. However, in Fig 5, we show a cascade deletion starting from a person. In the corresponding RDF graph, this leads to the deletion of two nodes and their adjacent edges, which is coherent with a node distance of 2. While it is possible to define an equivalent distance in graphs and propose DP mechanisms accordingly, they would be at risk of having low utility. Indeed, the number of nodes affected by a single deletion being unbounded is a problem in a DP setting, as it aims to guarantee a protection between neighbors. Providing

node-DP while maintaining acceptable utility can be challenging, and in the present case neighboring database would differ even more.

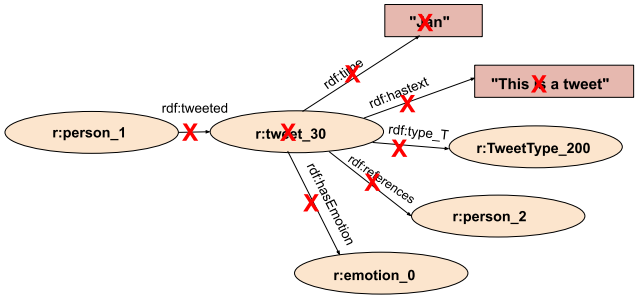


Fig. 6: Transitive cascade deletion of a tweet as shown on a graph

4.4 A new meaningful distance : Restrict Cascade Distance

The previous subsection resolves the mismatch between cascade deletion distance in RDB and node distance in RDF by creating an RDF distance that matches cascade deletion distance. Conversely, we can reverse the translation and search for an RDB distance that matches the node distance. We propose another instance of the cascade deletion: the **restrict cascade deletion**. The key idea is that the deletion of elements from an entity only propagates on the neighboring relations.

Definition 14 (Restrict Cascade Deletion). *Let D be an ER-compliant database on a schema $\mathcal{D} = (\mathcal{T}, \mathcal{C})$, $T \in \mathcal{T}$ a table, and $x \in T$ be an element of D. The restrict cascade deletion of x in D is its $\mathcal{C}'$ cascade deletion, where $\mathcal{C}'$ is the set of FK of relation tables.*

Definition 15 (Restrict Cascade Distance). *Let D, D' be two ER-compliant databases of same schema. We say that D and D' are Restrict Cascade neighbors if D' is the result of the cascade deletion of an element x in D, or D is the result of the cascade deletion of an element x in D'.*
The Restrict cascade distance is defined over ER-compliant databases of same schema as the length of a shortest path connecting two databases neighbor by neighbor, if it exists.

The restrict cascade deletion behaves similarly to the transitive distance except that in some cases it accepts a *null* as FK rather than deleting the concerned line. For example, the restrict cascade deletion is exactly equivalent to the transitive deletion in Fig. 6. Compared to the deletion performed in Fig. 5, restrict cascade deletion is gentler. It would replace the FK p_id with value 2 by a *null*,

but preserve the line, and the process would then stop, rather than leading to the suppression of lines in the tables References and HasEmotion.

Notably, this meaningful notion of distance in the relational world, is, no matter the starting table or the data we are trying to protect, always isometric to RDF node distance, as node deletion is exactly $\emptyset$-cascade deletion.

Theorem 1. *All ER encodings are isometric w.r.t the restrict cascade deletion distance and the node distance*

The proof of this theorem is made by establishing the following lemma:

Lemma 2. *In an ER database, restrict cascade deleting an element of an entity is exactly deleting adjacent relations and erasing adjacent foreign keys.*

Proof (Lemma). Restrict cascade deletion only propagates through foreign keys from an entity to a relation, hence the first part of the lemma: every element of a relation pointing towards the original element are deleted, and it does not propagate further. Erasure concerns foreign keys coming from deleted nodes. However, the only deleted nodes are the original and adjacent relations. Since in an ER database, foreign keys only come from entities, the foreign keys coming from deleted nodes all come from the original. □

The proof of the theorem is a direct consequent:

Proof (Theorem). Let D be a database, x one of it's elements, D' the database resulting from the cascade deletion of x in D, and f an ER encoding. D and D' are neighbors in the restrict cascade deletion distance, from Lemma 2. We will prove $f(D)$ and $f(D')$ are neighbors in the node deletion distance. The only difference between D and D' is:

- The deletion of x: this translates as the disappearance of the node x from V and the deletion of all its information, which translates as a deletion of all edges outgoing from x between $f(D)$ and $f(D')$.
- The deletion of every adjacent relation element: this translates as the disappearance of some edges between x and other elements of V that corresponds to the encoding of relations between $f(D)$ and $f(D')$.
- The erasure of every foreign key coming from x: this translates as the disappearance of all remaining edges between x and elements of V that corresponds to the encoding of entity-to-entity foreign keys between $f(D)$ and $f(D')$.

As a summary, between $f(D)$ and $f(D')$, we have removed x from V, and all edges incident to x from E. We note that this is the exact definition of the node deletion of x in $f(D)$, and conclude that $f(D)$ and $f(D')$ are neighbor in the node deletion distance.

Note that this argument goes both ways: any node deletion in $f(D)$ would result in a new graph which is the encoding of another relational database D'', which is identical to D save for one restrict cascade deletion.

Since both the cascade deletion distance and the node deletion distance are defined as the shortest distance from neighbor to neighbor between two points, this preservation of neighborhood is sufficient to prove that f is an isometry. □

5 QL-Outedge Privacy over Relational databases

In Section 4, we proposed a meaningful adaptation of a classical privacy model in relational databases, that when transformed to an RDF database, correspond to typed-node privacy. However, as discussed in Section 2, other privacy models on graphs are better adapted to RDF graphs, such as QL-Outedge privacy [8]. Thus in this section, we propose to study how we can translate the QL-Outedge privacy model to the relational database setting. The final goal is to propose a neighborhood definition for a relational database which would be the equivalent of the QL-Outedge neighborhood defined over RDF graphs.

Our approach is to map the relational database to RDF, apply QL-Outedge privacy in RDF, then return to relational world to define the corresponding neighborhood. When mapping a relational database to RDF, there is the edge direction to be tailored in order to provide the appropriate privacy guarantee under QL-Outedge privacy. Thereupon, we propose a model where one can choose the direction of edges according to what to protect.

Desired Privacy and Motivating Queries As a way to illustrate the importance of varied (and matching) distances between RDB and RDF formalisms, we consider a case where the desired protection of privacy is on the entire set of tweets of a single person. As such, any sufficiently private mechanism should be noisy enough for an attacker not to be able to significantly distinguish between the inclusion or exclusion of the set of tweets authored by a person from the mechanism's result. As a measuring stick, we study DP mechanisms that would answer the following queries while adding a noise calibrated by the global sensitivity of those queries under a particular distance.

- **Motivating Example 1:** Consider query Q1: Find the maximum number of tweets tweeted by a single person (maximum out-degree of *tweeted* outedges).
- **Motivating Example 2:** Consider query Q2: Count how many users "Alice" has referenced.

5.1 QL-outedge distance

We express the definition of QL-Outedge distance in our formalism.

Definition 16 (QL-Outedge pruning). *Let $G = (A, L, V, E)$. We call O_{QLv} the set of edges coming from v of label in QL: $(v, l, v') \in O_{QLv}$ iff $l \in QL$. The result of the QL pruning of v in G is a graph $G' = (A, L, V, E \backslash O_{QLv})$.*

Accordingly, the transposition of the node distance in this formalism is:

Definition 17 (QL-Outedge Distance). *Let $G = (A, L, V, E)$ and $G = (A, L, V, E')$ be two graph databases. We say that G and G' are QL-Outedge neighbors if G' is the result of the QL pruning of a node $v \in V$ in G, or G is the result of the QL pruning of a node $v \in V$ in G'.*
The QL-Outedge distance is defined over graphs of same labels as the length of a shortest path connecting two databases neighbor by neighbor, if it exists.

We note that only graphs of identical sets of nodes have a definable distance in QL-Outedge.

5.2 RDB Keywise deletion

Similarly to the previous section, we aim to find an RDB distance that matches the RDF QL-outedge distance. To do so, we will define keywise deletion as a direct analog of QL-outedge pruning.

Definition 18 (Relational Keywise Pruning). *Let D be a ER-compliant database on a schema $\mathcal{D} = (\mathcal{T}, \mathcal{C})$, and $\gamma \subseteq \sigma$ a restriction of an orientation σ to part of its domain. Let x be an element of T an entity table of $\mathcal{T}$. The relational keywise pruning of x alongside $\mathcal{C}', \sigma$ defines a set of **deleted elements** $L_{KW(x)}$ as the set of lines z such that:*

- *z is in a relation table T' of FK ϕ_{l,l'_0} from T to T' and ϕ_{l'',l'_1} from some T'' to T',*
- *$\gamma(T') = (l'_0, l'_1)$,*
- *x is the ϕ-antecedent of z for ϕ_{l,l'_0}.*

*The set $A_{KW(x)}$ of **deleted attributes** is the set of pairs (y, s) such that:*

- *$y \in T'$ where T' is an entity table*
- *there exists a FK constraint $\phi_{l,l'}$ from T to T'*
- *$\gamma(\phi_{l,l'}) = (l, l')$*
- *the antecedent of y by $\phi_{l,l'}$ is x*
- *$s \in l'$*

or (x, s) such that:

- *there exists a FK constraint $\phi_{l',l}$ from T' to T*
- *$\gamma(\phi_{l',l}) = (l, l')$*
- *$s \in l$*

The result of the relational keywise deletion of x alongside γ is a database D' of schema $\mathcal{D} = (\mathcal{T}, \mathcal{C})$ whose elements are all the elements of D that are not in $L_{KW(x)}$ where for every $(y, s) \in A_{QL(x)}$, $y.s$ is set to null.

γ represents both a choice of relations, many-to-many or one-to-many (the restricted domain) and the only direction from which such relation must be pruned. We note that in one-to-many relationships, the resulting deletion of attributes can happen in a record away from x (if the orientation goes from the antecedent outward) or in x itself (if the orientation goes from the postcedent to the antecedent). We also note that both orientations cannot be picked for a single relation, mirroring a known limitation of QL-outedge.

Definition 19 (Relational Keywise Distance). *Let D, D' be two database of same schema, and γ a restriction of an orientation. We say that D and D' are relational keywise neighbors if D' is the result of the relational keywise deletion of an element x in D alongside γ, or D is the result of the relational keywise deletion of an element x in D' alongside γ.*
The relational keywise deletion distance is defined over databases of same schema as the length of a shortest path connecting two databases neighbor by neighbor, if it exists.

We note that only databases of identical entities have a definable distance in Keywise Distance, up to attribute deletion, as entity lines are never part of $L_{KW(x)}$.

5.3 Isometric Encoding RDB to RDF for QL-Outedge

We now characterize under which conditions the keywise distance and QL-outedge distance are linked by an isometric encoding. Since all entities/nodes remain untouched, and edge orientation is at the heart of QL-Outedge, this property will focus on finding the appropriate orientation.

Lemma 3 (QL-outedge isometry). *Let $\mathcal{D} = (\mathcal{T}, \mathcal{C})$ be a database schema, γ an orientation restriction, σ an orientation, QL a set of labels.*

The σ-encoding of databases of schema $\mathcal{D}$ is an isometry between γ-keywise distance and QL-outedge distance if γ is the restriction of σ on QL.

Proof. Let D be a database, and $f(D)$ its σ-encoding. Let x be an element of an entity T of D. It is a node in $f(D)$. This node's outgoing edges come from three sources:

- **Many-to-many relations:** the lines of D encoded as edges to or from x are lines of relations table that have x for antecedent in one of their FK. Hence the outgoing edges of x come from the set of lines z such that:
  - z is in a relation table T' of FK ϕ_{l,l'_0} from T to T' and ϕ_{l'',l'_1} from some T'' to T',
  - $\sigma(T') = (l'_0, l'_1)$,
  - x is the ϕ-antecedent of z for ϕ_{l,l'_0}.
- **One-to-many relations:** the entity to entity FK of D encoded as edges to or from x are those where x is the antecedent or postcedent. Depending on the role x takes and the chosen orientation, those will be incoming or outgoing. The outgoing ones specifically are the set of pairs (y, l') such that:
  - $y \in T'$ where T' is an entity table
  - there exists a FK constraint $\phi_{l,l'}$ from T to T'
  - $\sigma(\phi_{l,l'}) = (l, l')$
  - the antecedent of y by $\phi_{l,l'}$ is x

  or (x, l) such that:
  - there exists a FK constraint $\phi_{l',l}$ from T' to T
  - $\sigma(\phi_{l',l}) = (l, l')$

- $s \in l$

If we were to prune x in $f(D)$ along QL, we would restrict all of those edges to those that fulfill those condition and are from a relation or FK in QL. That is to say, it will be the same specification but every time we check for the orientation σ of a relation or FK, it should also be in QL. In other words, it is the same as checking this equality on the restriction of σ on QL, which is γ. However, replacing σ by γ in the specification above gives us exactly the definitions of $L_{KW(x)}$ and $A_{KW(x)}$. This means that γ-pruning x in D or QL-pruning x in $f(D)$ is identical. This means that finding D' a γ-keywise neighbor of D or finding f(D') a QL-outedge-neighbor of $f(D)$ is identical. This means that the σ-encoding f is an isometry between those two distances.

6 Illustrating Encodings Impact on QL-outedge Privacy

To illustrate the concept introduced in the previous section and the impact of mapping methods, we refer to our running example dataset and introduce two simple example queries :

Q1 Find the maximum number of tweets tweeted by a single person (maximum out-degree of *tweeted* outedges).
Q2 Count how many users “Alice" has referenced.

We discuss hereafter how protection and global sensitivity differ as different R2RML mappings are proposed. As such, we will present the privacy protection offered by the default mapping. We show that this may lead to weak protection and, given a stronger target, show how to design an appropriate, manually-written, R2RML mapping tailored to the use-case to achieve desired privacy protection. These two different protections are illustrated through Q1 (which they directly relate to) and Q2 (on which they have no impact).

6.1 Default R2RML, protecting authorship of a single tweet

In this first case, we use a R2RML mapping according to W3C R2RML recommendation[9]. This standard lets the choice of orientation open for many-to-many relationships in our example, References and HasEmotion. We translate both relations as outedges of Tweet. However, for one-to-many relations (between tweets and their author), the standard imposes that we choose a direction from the referencing table (here, the one-side Tweet) to the referenced table (here, the many-side Person). This means that, as is shown in Fig. 3a, the edges of this relation count as outgoing for tweets rather than persons.

The implicit protection provided is that on a QL-outedge distance, a DP mechanism would protect the information of any one tweet. This impacts the neighbohrood one can build with QL-outedge, and the resulting global sensitivity of certain queries is affected by this choice:

[9] `https://github.com/sarataki/mapping/tree/main/defaultR2RML/r2rml.ttl`

Q1 The global sensitivity of this query under QL-Outedge privacy over the RDF graph obtained from default R2RML mapping is 1, assuming that `pr:p_id` $\in$ QL, which represents the *tweetedBy* relation. The neighboring graphs differ by the QL-outedges of an arbitrary node. The *Tweet* node has exactly one *tweetedBy* outedge (along with other outedges of other labels). So, one possible neighboring graph differs by the outedges of node *Tweet*. The model related to this encoding protects the *tweetedBy* outedge, *i.e.,* the author of one tweet.

Q2 The global sensitivity of this query under QL-Outedge privacy over the RDF graph obtained from this default R2RML mapping is infinite, assuming both *tweetedby* and *references* are in QL. The node *Person* has one *name* outedge with Literal value "Alice" (along with other outedges). Any neighboring graphs differ by the QL-outedges of an arbitrary node, possibly a *Tweet*. The *Tweet* node has exactly one *tweetedBy* outedge plus some *references* outedges. Since the number of references outedges could be unbounded, so is this query's global sensitivity.

Discussion: As shown above, the default R2RML mapping restricts our choice on privacy protection and, for example, imposes to protect only the author of one tweet, as exemplified by the sensitivity of Q1 being 1. However, one may want to protect *all* the tweets of a person, of an author, rather than the author of one tweet. Notably, no choice of QL under default mapping would give us the desired graph neighborhood. Under QL-outedge DP, protecting all the tweets of a person requires the edges representing authorship to be outedges rather than inedges of nodes Person. This can be achieved by writing a customized R2RML mapping. We discuss the definition of the related mapping hereafter, as well as the impact of the example queries.

6.2 Custom mapping, protecting someone's tweets

Our objective is to find a distance for which DP guarantees protection on all the tweets of a person, which is to say a distance where such deletion is possible between a graph and one of its neighbors. As noted above, this is impossible under default mapping. However, by writing another R2RML file where we define our own mappings, we can better control the privacy protection as we reestablish appropriate distance, and consequently, appropriate noise requirement for DP.

Design of a mapping satisfying the target protection For our example dataset, protecting all the tweets authored by an individual corresponds in RDB to a keywise distance on the foreign key *p_ id* that points from *Tweets* to *Person*. To delete all pointers to one person in one go, we must choose the orientation that is contrary to the W3C recommendations, that is to say from the referenced table to the referencing table. To find an isometry of this to a QL-outedge distance,

Lemma 3 tells us that the encoding we pick must also choose the direction from *Person* to *Tweet* to encode this foreign key. This encoding corresponds to a different R2RML mapping[10], which outputs graphs with properly oriented edges (see Fig 7). In this new encoding and with the new QL-outedge distance it permits, we reconsider query Q1 and Q2. We study again the global sensitivity with our proposed R2RML mapping.

Q1 The sensitivity of Q1 in this new distance is the maximal number of edges *tweeted* that can separate a graph from its neighbors. This maximal number does not exist, as an arbitrarily great number of edges can be deleted from a single *Person* node at once. The global sensitivity of this query is therefore infinite. This means that we now correctly assess that no amount of noise can obfuscate the presence or absence of a single person's tweet output to the satisfaction of DP criteria.

Note that, if the number of tweets outedges of node person is bounded, then the global sensitivity of Q1 is bounded. This can also be achieved by using a projection method (e.g. [8]).

Q2 Two possible neighbouring graphs differ by the QL-outedges of some nodes. *Tweet* (resp. *Person*) nodes may have an arbitrary number of references (resp. tweeted) outedges. Again, assuming *tweeted* $\in$ QL or references $\in$ QL, we have global sensitivity of Q2 equal to infinite. Here, the change in mapping (and hence privacy protection) does not impact Q2, since the number of people referenced in a single tweet is unbounded, and protecting one or several tweet still lead to the protection of an arbitrarily high number of referenced individuals.

Discussion Hence, the mappings influence privacy protection and, in turn, may impact queries' sensitivity. We note that this choice of orientation is not objectively better, but merely adapted to the protection we use as an example. Every such decision represents some form of tradeoff on what privacy protection can or cannot be expressed. For instance, if we represent authorship information from *Person* to *Tweet*, then privacy protection about a single tweet becomes incongruous: even a complete deletion of all the information outgoing from a *Tweet* node would leave its author edge untouched, as it is now incoming. If translated into RDB, this gives an incomplete line pruning, as shown in Fig. 8.

7 Conclusion

In this paper, we analyze the transposition to RDF through mapping of a popular DP model for multi-tables relational databases with FK constraints related to transitive deletion [24]. We show it has an equivalent in RDF, an extension of

[10] `https://github.com/sarataki/mapping/tree/main/propR2RML/r2rml.ttl`

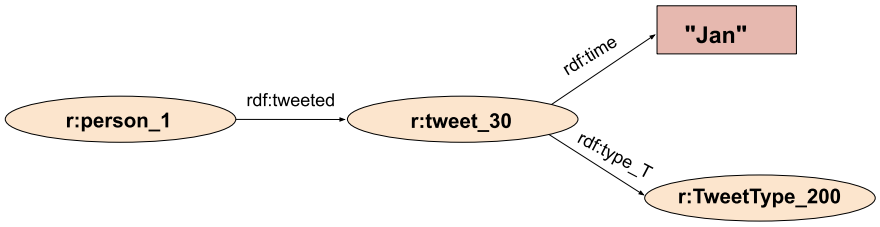


Fig. 7: Proposed R2RML: translation of one-to-many relationship

Person

idperson	type_P	name
1	100	Alice
2	~~100~~	~~Bob~~
3	100	Clara

Tweet

idtweet	type_T	p_id	time	hastext
30	200	1	Jan	This is a tweet
31	200	1	Feb	HelloWorld
32	200	~~2~~	March	What

Fig. 8: Tweet (incomplete) pruning under nonstandard edge orientation

node DP, under a choice of R2RML mapping made to correspond to a choice of primary relation. To ease the construction of RDF DP mechanisms while remaining explainable in the relational world, we tweak the original privacy model in a meaningful way so that its translation is always equivalent to classical node DP. Thus, we proposed the restrict deletion for relational databases, which captures privacy policies and FK constraints.

Furthermore, we formalize neighborhoods in relational databases to match the concept of QL-Outedge privacy, which was previously defined over RDF graphs. We believe this approach is particularly suitable to the context of RDF, offering meaningful semantics for the neighborhoods. When mapping relational databases to RDF, we propose a model where one can choose what they want to

protect. This is once again done by deciding the edge direction in the R2RML mapping. The decision on which direction to choose will affect what edges are incoming or outgoing, and, therefore, the QL-Outedge neighborhood definition in the relational database. Finally, we proposed an implementation based on R2RML to illustrate our approach.

For future work, we plan to strengthen and implement relational-to-graph and graph-to-relational database mapping methods, by matching known and useful distances of RDB or RDF as well as neighborhood definitions which would make more sense in this context into corresponding notions in the other formalism. Furthermore, another interesting research direction is establishing a benchmark to compare the efficiency of different privacy methods through mapping. This would lead to a wider choice of comparable options for information stored as RDB or RDF while preserving important privacy guaranteeing properties.

Acknowledgments. This work is supported by the french National Research Agency (ANR) under grants iPoP (ANR-22-PECY-0002), CyberINSA (ANR-23-CMAS-0019), and DiffPriPos (ANR-23-CE23-0032).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Laurence Field, Stephanie Suhr, John Ison, Wouter Los, Peter Wittenburg, Daan Broeder, Alex Hardisty, Susanna Repo, and Andy Jenkinson. Realising the full potential of research data: common challenges in data management, sharing and integration across scientific disciplines. Technical report, ZENODO, 2013.
2. Franck Michel, Johan Montagnat, and Catherine Faron-Zucker. A survey of rdb to rdf translation approaches and tools. Technical report, I3S, 2013.
3. Mohamed AG Hazber, Ruixuan Li, Bing Li, Yuqi Zhao, and Khaled MA Alalayah. A survey: Transformation for integrating relational database with semantic web. In *Proceedings of the 2019 3rd International Conference on Management Engineering, Software Engineering and Service Sciences*, pages 66–73, 2019.
4. Iovka Boneva, Slawomir Staworko, and Jose Lozano. Sherml: mapping relational data to rdf. 2019.
5. Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.
6. Joseph P Near, Xi He, et al. Differential privacy for databases. *Foundations and Trends® in Databases*, 11(2):109–225, 2021.
7. Jenni Reuben. Towards a differential privacy theory for edge-labeled directed graphs. *SICHERHEIT 2018*, 2018.
8. Sara Taki, Cédric Eichler, and Benjamin Nguyen. It's too noisy in here: Using projection to improve differential privacy on rdf graphs. In *European Conference on Advances in Databases and Information Systems*, pages 212–221. Springer, 2022.
9. Sara Taki. *Linked Data Sanitization with Differential Privacy.* Theses, INSA Centre Val de Loire, December 2023.

10. Sara Taki, Adrien Boiret, Cédric Eichler, and Benjamin Nguyen. Cohesive database neighborhoods for differential privacy: Mapping relational databases to RDF. In *WISE (5)*, volume 15440 of *Lecture Notes in Computer Science*, pages 231–242. Springer, 2024.
11. Marcelo Arenas, Alexandre Bertails, Eric Prud'hommeaux, Juan Sequeda, et al. A direct mapping of relational data to rdf. *W3C recommendation*, 27:1–11, 2012.
12. Das Souripriya, Sundara Seema, and Cyganiak Richard. R2rml: Rdb to rdf mapping language. *W3C Recommendation*, 27, 2012.
13. Luciano Frontino de Medeiros, Freddy Priyatna, and Oscar Corcho. Mirror: Automatic r2rml mapping generation from relational databases. In *Engineering the Web in the Big Data Era: 15th International Conference, ICWE 2015, Rotterdam, The Netherlands, June 23-26, 2015, Proceedings 15*, pages 326–343. Springer, 2015.
14. Tim Berners-Lee. Relational databases on the semantic web, 1998. *Via https://www. w3. org/DesignIssues/RDBRDF. html, last accessed January*, 2015.
15. Franck Michel, Johan Montagnat, and Catherine Faron Zucker. A survey of rdb to rdf translation approaches and tools. 2013.
16. Sören Auer, L Feigenbaum, D Miranker, A Fogarolli, and J Sequeda. Use cases and requirements for mapping relational databases to rdf. *W3c working draft*, 2010.
17. Anastasia Dimou, Miel Vander Sande, Jason Slepicka, Pedro Szekely, Erik Mannens, Craig Knoblock, and Rik Van de Walle. Mapping hierarchical sources into rdf using the rml mapping language. In *2014 IEEE International Conference on Semantic Computing*, pages 151–158. IEEE, 2014.
18. Anastasia Dimou, Miel Vander Sande, Pieter Colpaert, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. Rml: A generic language for integrated rdf mappings of heterogeneous data. *Ldow*, 1184, 2014.
19. Franck Michel, Loïc Djimenou, Catherine Faron Zucker, and Johan Montagnat. Translation of relational and non-relational databases into rdf with xr2rml. In *11th International Confenrence on Web Information Systems and Technologies (WEBIST'15)*, pages 443–454, 2015.
20. Cynthia Dwork. Differential privacy. In *Proceedings of the Automata, Languages and Programming, 33rd International Colloquium, ICALP*, 2006.
21. Michael Hay, Chao Li, Gerome Miklau, and David Jensen. Accurate estimation of the degree distribution of private networks. In *2009 Ninth IEEE International Conference on Data Mining*, pages 169–178. IEEE, 2009.
22. Frank D McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 19–30, 2009.
23. Noah Johnson, Joseph P Near, and Dawn Song. Towards practical differential privacy for sql queries. *Proceedings of the VLDB Endowment*, 11(5):526–539, 2018.
24. Ios Kotsogiannis, Yuchao Tao, Xi He, Maryam Fanaeepour, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. Privatesql: a differentially private sql query engine. *Proceedings of the VLDB Endowment*, 12(11):1371–1384, 2019.
25. Yuchao Tao, Xi He, Ashwin Machanavajjhala, and Sudeepa Roy. Computing local sensitivities of counting queries with joins. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 479–494, 2020.
26. Wei Dong, Juanru Fang, Ke Yi, Yuchao Tao, and Ashwin Machanavajjhala. R2t: Instance-optimal truncation for differentially private query evaluation with foreign keys. In *Proceedings of the International Conference on Management of Data*, 2022.
27. Christophe Debruyne and Declan O'Sullivan. R2rml-f: Towards sharing and executing domain logic in r2rml mappings. *LDOW@ WWW*, 1593, 2016.