Transforming RDF Data into Property Graphs

R. Angles, and R. García

***Abstract*—RDF databases and graph databases are two approaches of data management which are based on modeling, storing and querying data following a graph structure. RDF databases are based on a single graph data model which allows to describe Web resources in terms of their relations and attributes. On the other hand, most graph databases are based on the property graph data model, a type of graph where nodes and edges can contain properties represented as key-value pairs. This paper presents two methods for transforming RDF data into property graphs. The first method defines schema and data transformations as it assumes the existence of an RDF schema. The second method is schema-independent, so it allows to transform any kind of RDF dataset. Both methods are useful to store RDF data into a Graph Data Management System.**

***Index Terms*—RDF, RDF Schema, Property Graph.**

I. Introducción

LAS bases de datos RDF y las bases de datos para grafos son dos enfoques de gestión de datos que están basados en modelar, almacenar y consultar datos con estructura de grafo. Los sistemas que cuentan con esta característica están ganando bastante relevancia en la industria debido a su uso en proyectos de *Big Data*, en particular para análisis de redes complejas [1].

Los sistemas de gestión de bases de datos (SGBD) para RDF y aquellos diseñados para grafos se basan en modelos de datos orientados a grafos [2]. Por una parte, los SGBD RDF (también llamados *RDF Triple Stores*) se basan en un modelo de datos de grafo donde los nodos representan recursos Web o valores simples (denominados Literales), mientras que las aristas representan los atributos y relaciones de dichos recursos Web.

Por otro lado, gran parte de los SGBD para grafos actuales se enfocan en almacenar grafos con propiedades. En comparación con un grafo tradicional, los nodos y las aristas de un grafo con propiedades pueden contener pares llave-valor (*key-value*), donde el primer elemento define el nombre de una propiedad y el segundo elemento corresponde al valor de dicha propiedad.

Desde el punto de vista de su uso, los SGBD RDF se emplean para gestionar datos del tipo *Linked Data* [3], y son diseñados para soportar diversos estándares de la Web Semántica como SPARQL [4] (el lenguaje de consulta estándar para RDF), RDF Schema [5] (un lenguaje para describir vocabularios RDF), y OWL [6] (un lenguaje para describir ontologías [7]). Por su parte, los SGBD para grafos son diseñados para almacenar datos complejos no estructurados, e incluyen lenguajes de consulta que permiten expresar consultas esenciales en grafos como patrones y caminos [8], [9].

Renzo Angles, Universidad de Talca, Instituto Milenio de Investigación sobre los Fundamentos de los Datos, Chile, rangles@utalca.cl.

Roberto García, Instituto Milenio de Investigación sobre los Fundamentos de los Datos, Chile, ramhgarcias@gmail.com.

1) Motivación y Problema.: Considerando las diferencias entre los SGBD para RDF y aquellos orientados a grafos, resulta necesario estudiar su interoperabilidad, es decir, la definición de métodos que permitan intercambiar información de manera correcta y significativa entre dichos sistemas [10]. La interoperabilidad entre sistemas de base de datos es muy importante por diversas razones, siendo algunas las siguientes:

- Promueve el intercambio de datos (*Data Exchange*), es decir, el estudio de métodos que permitan traducir o migrar esquemas y datos desde un sistema a otro [11];
- Facilita la reutilización de sistemas o programas existentes para construir nuevas aplicaciones [12];
- Permite explorar las mejores características de diversos enfoques o sistemas [13];
- Permite comparar los sistemas de manera justa a través del uso de *database benchmarks* [14];
- Soporta el éxito de los sistemas y tecnologías emergentes [12].

Adicionalmente, la interoperabilidad ayudaría a estrechar los lazos entre la comunidad de la Web Semántica y la comunidad de bases de datos orientadas a grafos.

2) Objetivos & Contribuciones.: El objetivo principal de este artículo es avanzar en el estudio formal de métodos que permitan interoperabilidad entre bases de datos RDF y bases de datos de grafos con propiedades. Para esto, primero definimos formalmente las nociones de base de datos RDF (Sección II) y base de datos de grafo (Sección III). Luego presentamos una revisión de los métodos de transformación de datos propuestos hasta el momento, describiremos sus características, y listaremos algunos problemas fundamentales (Sección IV).

Posteriormente presentamos la definición formal de dos métodos de transformación (o *mappings*) que permiten generar grafos con propiedades desde datos RDF (Sección V). El primer método asume que los datos RDF de entrada vienen acompañados de su esquema de datos RDF (es decir, se conoce la estructura de los datos), por lo tanto definimos transformaciones para esquema y datos. El segundo método funciona independientemente de que exista un esquema de datos RDF, por lo tanto puede ser empleado para transformar cualquier tipo de datos RDF. El articulo concluye con algunas conclusiones sobre los métodos propuestos (Sección VI).

II. Base de datos RDF

RDF [15] (*Resource Description Framework*) es un modelo de datos estándar para describir recursos Web. La noción de "describir" se refiere a la declaración explícita de los atributos de un recurso así como de sus relaciones con otros recursos.

Informalmente, el modelo de datos RDF puede definirse en base a cuatro conceptos:

- **Recursos.** Un recurso (*resource*) puede ser definido como un objeto o cosa que se desea describir. Los recursos pueden ser personas, libros, páginas Web, o cualquier otra cosa, real o abstracta. Cada recurso en RDF es identificado de manera única a través de una IRI[1]. Una URL es un tipo especial de IRI que se usa frecuentemente en las aplicaciones de la Web Semántica.
- **Propiedades.** Una propiedad (property) se refiere a un atributo o una relación de un recurso. Un atributo consiste en una característica propia de un recurso, la cual tiene un valor específico (ej. "nombre"). Una relación representa un vínculo entre dos recursos (ej. "autor"). En RDF, las propiedades son consideradas tipos especiales de recursos, por lo tanto también se identifican usando IRIs.
- **Declaración.** Una declaración (*statement*) establece una afirmación precisa sobre alguna propiedad de un recurso. Por ejemplo, la expresión "El autor de la Mona Lisa es Leonardo da Vinci" es una declaración respecto de la relación de "autor" entre los recursos "Mona Lisa" y "Leonardo da Vinci". Una declaración se representa usando una estructura especial denominada triple RDF.
- **Descripción.** El conjunto de declaraciones (o triples RDF) asociadas a un recurso conforman la descripción del mismo. La descripción de un dominio de aplicación esta conformada por la unión de las descripciones de todos los recursos del dominio.

A continuación presentamos la definición formal del modelo de datos RDF.

A. Modelo de Datos RDF

Asumir que I y L son dos conjuntos finitos disjuntos correspondientes a IRIs y Literales respectivamente[2]. Las IRIs actúan como identificadores de recursos, mientras que los literales son simples valores atómicos. Un término RDF es un elemento en el conjunto $T = I \cup L$.

Definición II.1. *Un grafo RDF es un conjunto* $G_R = \{t_1, \ldots, t_n\}$ *donde cada* t_i *es una tupla* $(v_1, v_2, v_3) \in I \times I \times T$. *Cada elemento* t_i *se denomina un Triple RDF.*

Dado un triple RDF (v_1, v_2, v_3), tendremos que v_1 se denomina el sujeto (*subject*), v_2 el predicado (*predicate*) y v_3 el objeto (*object*). Específicamente, el sujeto representa un recurso del cual se describe algo, el predicado representa una propiedad (atributo o relación) del sujeto, y el objeto representa el valor de la propiedad.

Un triple RDF se representa gráficamente como una estructura nodo-arista-nodo de un grafo. En consecuencia, un conjunto de triples RDF da lugar a una grafo RDF. Por ejemplo, la Figura 1 presenta un grafo RDF que describe la expresión "La Mona Lisa es una obra de arte (*artwork*) creada por un artista (*person*) de nombre Leonardo da Vinci".

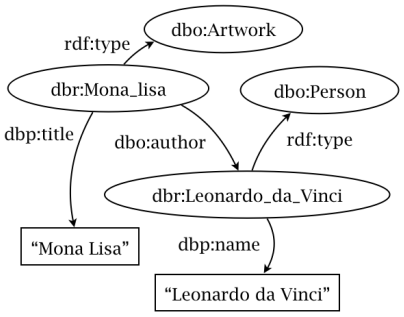

Fig. 1. Ejemplo de grafo RDF. Cada nodo elíptico esta etiquetado con una IRI y representa un recurso. Cada nodo rectangular esta etiquetado con un Literal que representa el valor de un atributo. Cada arista esta etiquetada con una IRI y representa una propiedad de un recurso.

Cabe resaltar que las IRIs empleadas en el grafo RDF de ejemplo se encuentran en su versión simplificada. Por ejemplo, la versión extendida de la IRI `dbr:Mona_lisa` es la URL `http://dbpedia.org/resource/Mona_Lisa`, donde `dbr` es un prefijo que actúa como abreviatura del espacio de nombres `http://dbpedia.org/resource/`. Con el fin de facilitar la legibilidad de los ejemplos, nos restringiremos a emplear únicamente IRIs abreviadas. Nótese además, que las IRIs empleadas en el ejemplo corresponden a recursos existentes en DBpedia[3], la fuente de datos RDF más popular de la Web Semántica.

B. Esquema de Datos RDF

RDF Schema (RDFS) [5] define un vocabulario estándar (es decir, un conjunto de términos con significado bien definido) que permite describir clases de recursos y propiedades para un dominio de datos particular. Desde el punto de vista de bases de datos, RDF schema permite definir la estructura de los datos en un grafo RDF, es decir un esquema de datos RDF.

En términos generales, un esquema RDF contiene definiciones de clases de recursos y clases de propiedades. Para esto, RDF schema define un vocabulario con los siguientes términos:

- `rdfs:class`, `rdf:Property` y `rdfs:Literal` representan las clases de los recursos, las propiedades y los literales respectivamente;
- `rdf:type` se emplea para definir que un recurso es una clase o una propiedad;
- `rdfs:domain` y `rdfs:range` son términos empleados para definir el dominio y rango de una propiedad respectivamente;
- el término `rdfs:label` permite describir el nombre concreto de una clase o propiedad.

Cabe destacar que las IRIs con prefijo `rdf` o `rdfs` pertenecen al vocabulario de RDF Schema.

Un esquema RDF se describe empleando triples RDF, por lo tanto también tiene una representación con forma de

[1]IRI, acrónimo de Internationalized Resource Identifier, es un estándar para crear identificadores de recursos web a través de cadenas compactas de caracteres. El uso de IRIs proporciona un sistema infinito de identificadores que pueden ser localizados y procesados de manera automática.

[2]Aparte de IRIs y Literales, el modelo de datos RDF considera un dominio de recursos anónimos denominados nodos blancos (*blank nodes*). Basándonos en un estudio previo [16], evitaremos el uso de nodos blancos asumiendo que su ausencia no afecta los resultados presentados en este artículo.

[3]DBpedia (http://wiki.dbpedia.org/) contiene información de Wikipedia pero descrita siguiendo el modelo de datos RDF.

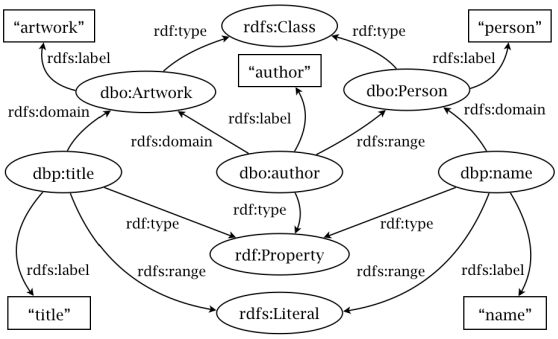

Fig. 2. Esquema de datos para el grafo RDF de la Figura 1.

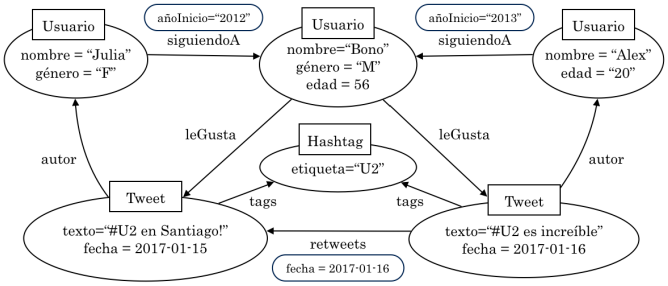

Fig. 3. Ejemplo de grafo con propiedades. Nótese que, en comparación con un grafo tradicional, tanto nodos como aristas pueden contener propiedades.

grafo. En la Figura 2 se muestra el esquema de datos para el grafo RDF de la Figura 1. Observe que una clase de recurso c se define básicamente a través de un triple RDF de la forma `(`c `rdf:type rdfs:Class)`, mientras que una clase de propiedad p requiere (idealmente) de tres triples de la forma `(`p `rdf:type rdf:Property)`, `(`p `rdfs:domain` d_p`)` y `(`p `rdf:range` r_p`)`. Nótese que d_p indica la clase de recursos que tendrá la propiedad p (es decir, su dominio), mientras que r_p indica la clase de recursos que determinará el valor de la propiedad p (es decir, su rango). Si una propiedad representa una relación, el rango será una clase de recursos definida en el esquema RDF, por ejemplo `(dbo:author rdfs:range dbo:person)`. Si una propiedad representa un atributo, entonces el rango será la clase de recursos `rdfs:Literal` o una clase de recursos que defina un tipo de datos específico. Por ejemplo, `xsd:string`, `xsd:integer` y `xsd:dateTime` hacen referencia a tipos de datos definidos por XML Schema[4].

III. Base de datos de grafo

Un grafo con propiedades (*Property Graph*), es un multi-grafo dirigido y etiquetado que se caracteriza porque cada nodo o arista mantiene un conjunto (posiblemente vacío) de propiedades representadas como pares llave-valor [17]. Desde el punto de vista de modelado de datos, un nodo representa una entidad, una arista representa un relación entre dos entidades, y una propiedad representa una característica específica de una entidad o una relación. A continuación presentamos una definición formal del modelo de datos de grafos con propiedades.[5]

A. Modelo de Datos de Grafos con Propiedades

Asumir que **L** es un conjunto infinito de etiquetas (para nodos y aristas), **P** es un conjunto infinito de nombres de propiedades, **V** es un conjunto infinito de valores atómicos, y **T** es un conjunto finito de tipos de datos (por ejemplo, *string*, *integer* y *date*). Dado un valor $v \in \mathbf{V}$, la función $\mathrm{type}(v)$ retorna el tipo de dato de v. Los valores en **V** se distinguirán como cadenas de caracteres entre comillas.

Definición III.1. *Un grafo con propiedades es una tupla* $G_P = (N, E, \rho, \lambda, \sigma)$ *donde:*

- N *es un conjunto finito de nodos (también llamados vértices);*
- E *es un conjunto finito de aristas, cumpliendo que* E *no tiene elementos en común con* N*;*
- $\rho : E \to (N \times N)$ *es una función total que asocia cada arista de* E *con un par de nodos en* N *(es decir,* ρ *es la usual función de incidencia en teoría de grafos);*
- $\lambda : (N \cup E) \to \boldsymbol{L}$ *es una función parcial que asocia un nodo/arista con un único valor en* L *(es decir,* λ *es una función de etiquetado para nodos y aristas);*
- $\sigma : (N \cup E) \times \boldsymbol{P} \to \boldsymbol{V}$ *es una función parcial que asocia nodos y aristas con propiedades, y asigna un valor*[6] *a cada una de dichas propiedades.*

Dados dos nodos $n_1, n_2 \in N$ y una arista $e \in E$, cumpliendo que $\rho(e) = (n_1, n_2)$, emplearemos $e = (n_1, n_2)$ como una representación corta de la arista e, donde n_1 y n_2 se denominan el nodo "origen" y el nodo "destino" del arista e respectivamente. No es obligatorio que todo nodo/arista tenga una etiqueta, es decir, el valor de $\lambda(o)$ puede ser indefinido (vacío) para un nodo/arista o. Por otra parte, dado un par $(o, p) \in (N \cup E) \times \mathbf{P}$ y la asignación $\sigma((o, p)) = v$, usaremos $(o, p) = v$ como una representación corta de una propiedad p donde o es el "dueño" de la propiedad, p es el "nombre" de la propiedad y v es el "valor" de la propiedad.

La Figura 3 muestra un ejemplo de grafo con propiedades el cual contiene información asociada a Twitter. Nótese que todos los nodos contienen una etiqueta que identifica su tipo (*Usuario*, *Tweet* y *Hashtag*), además de una o más propiedades (*nombre*, *genero*, *edad*, *texto* y *fecha*). En el caso de las aristas, podemos observar que todas tienen etiquetas (*siguiendoA leGusta*, *autor*, *tags* y *retweets*) pero solo dos de ellas contienen propiedades (*añoInicio* y *fecha*).

B. Esquema de Datos para un Property Graph

El esquema de un grafo con propiedades (*Property graph schema*) define los tipos de nodos, aristas y propiedades permitidos para una base de datos de grafo. A continuación

[4]https://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

[5]Cabe mencionar que actualmente no existe una definición formal estándar del *Property Graph Data Model*. La definición presentada en este artículo es suficientemente general para el contexto de las bases de datos de grafo.

[6]En este artículo no se consideran propiedades multi-valuadas por dos razones: (i) la mayoría de los sistemas de bases de datos de grafo existentes no soportan propiedades multi-valuadas; y (ii) la noción de propiedad multi-valuada contradice la primera forma normal del modelo de datos relacional.

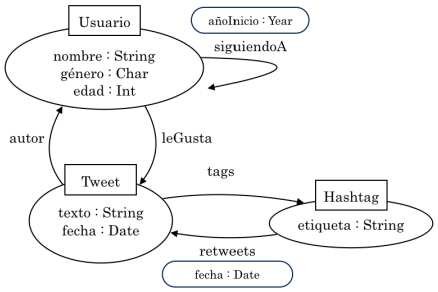

Fig. 4. Ejemplo de esquema de grafo con propiedades, el cual define la estructura del grafo con propiedades de la Figura 3.

definiremos formalmente la noción de esquema de grafo, además del significado de validar un grafo con propiedades.

Definición III.2. *Un esquema de grafo con propiedades es una tupla* $S_P = (T_N, T_E, \beta, \delta)$ *donde:*

- $T_N \subset \boldsymbol{L}$ *es un conjunto finito de tipos de nodos;*
- $T_E \subset \boldsymbol{L}$ *es un conjunto finito de tipos de aristas;*
- $\beta : (T_N \cup T_E) \times \boldsymbol{P} \rightarrow \boldsymbol{T}$ *es una función parcial que define las propiedades de nodos y aristas, y asigna un tipo de datos a dichas propiedades;*
- $\delta : (T_N \times T_N) \rightarrow Set(T_E)$ *es una función parcial que define los tipos de aristas que pueden existir entre cada par de tipos de nodos.*

La Figura 4 muestra un ejemplo de esquema de grafo con propiedades, el cual define la estructura del grafo presentado en la Figura 3. Observe que el esquema define los tipos de nodo "Usuario", "Tweet" y "Hashtag", y los tipos de arista "autor", "leGusta", "siguiendoA", "tags" y "retweets". Además, las propiedades de nodos y aristas incluyen un nombre y su tipo de dato.

Dada la simplicidad del ejemplo, es posible verificar visualmente que el grafo con propiedades de la Figura 3 satisface las restricciones estructurales y de tipos definidas por el esquema. A continuación, definimos formalmente como se realizaría dicha verificación.

Definición III.3. *Dado un grafo con propiedades* $G_P = (N, E, \rho, \lambda, \sigma)$ *y un esquema de grafo* $S_P = (T_N, T_E, \beta, \delta)$*, diremos que* G_P *es válido con respecto a* S_P *si y solo si:*

(i) *por cada nodo* $n \in N$ *se cumple que* $\lambda(n) \in T_N$*;*
(ii) *por cada arista* $e \in E$ *se cumple que* $\lambda(e) \in T_E$*;*
(iii) *por cada propiedad* $(o, p) = v$ *en* G_P *se cumple que* $\text{type}(v) = \beta(\lambda(o), p)$*;*
(iv) *por cada arista* $e = (n, n') \in E$ *se cumple que* $\lambda(e) \in \delta(\lambda(n), \lambda(n'))$.

De esta manera, las condiciones (i) y (ii) validan que tanto nodos como aristas están etiquetados con tipos definidos por el esquema, la condición (iii) verifica que cada nodo o arista contiene únicamente las propiedades definidas en el esquema, y la condición (iv) verifica que todo par de nodos conectados a través una arista cumplen lo definido por el esquema.

IV. Interoperabilidad Semántica entre Grafos RDF y Grafos con Propiedades

La interoperabilidad entre bases de datos puede dividirse en interoperabilidad semántica (esto es, intercambio de esquemas y datos) e interoperabilidad de consultas (esto es, transformaciones entre diferentes lenguajes de consulta o métodos de acceso). En este artículo nos concentraremos en interoperabilidad semántica, es decir, estudiaremos métodos que permiten transformar o mapear[7] un grafo RDF en un grafo con propiedades (y viceversa). Primero presentaremos una revisión de los métodos disponibles en la literatura, y luego describiremos problemas fundamentales asociados a la transformación de esquemas y datos.

A. Métodos de Transformación Existentes

En la literatura podemos encontrar algunos trabajos que estudian métodos de transformación entre RDF y grafos tradicionales. Hayes y Gutierrez [18] propusieron modelar grafos RDF como grafos bipartitos haciendo uso de reificación[8]. Esta propuesta permite dividir un grafo RDF en varias capas (o subgrafos no conectados), donde una capa corresponde a los datos y el resto al esquema. En [19], los autores presentan un método para almacenar datos RDF en una base de datos orientada a objetos. Para esto se definió un modelo basado en grafos donde los recursos, propiedades y literales del grafo RDF son modelados como objetos. En [20], los autores proponen un modelo de grafos para datos RDF denominado LDM-3N (*labeled directed multigraph-three nodes*). Este modelo emplea el método *Singleton Property* [21] para representar datos RDF de manera reificada. Aunque los métodos anteriores pueden ser empleados para generar grafos con propiedades, no se explotan las características de estos últimos (en particular, el uso de las propiedades).

Respecto de los métodos para transformar una base de datos RDF en una base de datos de grafos con propiedades, las propuestas existentes solo se concentran en transformar los triples RDF sin considerar la existencia de un esquema RDF. En [22], Hartig propone dos transformaciones basadas en un modelo de datos intermedio denominado RDF*. El primer método transforma cada triple RDF en un arista del grafo con propiedades, y cada nodo incluye el atributo "kind" para describir el tipo del recurso origen. La segunda transformación distingue entre atributos y relaciones de un recurso, los cuales son transformados en propiedades y aristas de un nodo en el grafo resultante. La limitación de la segunda transformación es que los triples que describen metadatos no pueden ser transformados adecuadamente. Una desventaja general de ambos métodos es que el modelo RDF* no es soportado por los SGBD RDF.

[7]Mapear es la traducción técnica de *mapping* en inglés. En matemática, un *mapping* se refiere a una operación que asocia cada elemento de un conjunto (el origen) con uno o más elementos de otro conjunto (el rango). En bases de datos, un *mapping* se refiere a un método para convertir datos de un modelo o sistema a otro.

[8]La reificación de un triple RDF (s,p,o) implica la generación de cuatro triples de la forma (_:b,rdf:type,rdf:Statement), (_:b,rdf:subject,s), (_:b,rdf:predicate,p), (_:b,rdf:object,o), donde _:b es un nodo blanco.

En [23], los autores presentan un *mapping* que es usado para la implementación de un procesador de consultas SPARQL sobre GraphX (un sistema de procesamiento de grafos en paralelo implementado sobre Apache Spark). El *mapping* emplea un atributo reservado *label* para almacenar un identificador en el caso de los nodos, y una etiqueta en el caso de las aristas. Básicamente, cada triple (s, p, o) del grafo RDF de entrada es representado (en el grafo de salida) a través de dos nodos n, n' y una arista $e = (n, n')$, de manera tal que n.`label` = s, e.`label` = p y n'.`label` = o. Cabe destacar que el método no hace diferencia entre atributos y relaciones, y tampoco considera la existencia de nodos blancos.

En [24], Tomaszuk presenta un algoritmo para serializar datos RDF siguiendo el formato de un grafo con propiedades. Este método es básicamente una transformación entre formatos de codificación que no considera nodos blancos.

Respecto de los métodos para transformar grafos con propiedades en grafos RDF, la literatura es aún más reducida. Los dos métodos existentes [22], [25] están basados en reificación, es decir, por cada arista en un grafo por propiedades, el grafo RDF tendrá un nodo blanco representando el arista, y este incluirá al menos tres nodos (recursos o literales) y tres aristas (propiedades) para describir la información del arista original.

B. Problemas Fundamentales

Luego de analizar los métodos de transformación existentes, se pudieron identificar los siguientes problemas o desafíos:

- Es muy común que una base de datos RDF presente una mezcla de datos y esquema. En este caso, es necesario elegir entre extraer el esquema (y transformarlo de manera independiente), o procesar el esquema como parte de los datos.
- Otra característica intrínseca de una base de datos RDF es la ocurrencia de un esquema parcial. En este caso, será necesario definir si se empleará o no dicho esquema. Esto puede implicar el uso de un método de transformación independiente del esquema, o un método mixto que soporte datos con y sin esquema.
- El modelo RDF incluye componentes o características especiales (como nodos blancos, reificación, sub-clases, sub-propiedades, y *entailment*) las cuales no pueden ser modeladas directamente o de manera sencilla en un grafo con propiedades.
- La existencia de datos RDF reificados introduce desafíos sintácticos y semánticos. Un desafío sintáctico es la representación multinivel en un grafo con propiedades. Un problema semántico es la interpretación completa y correcta de un triple reificado, y su representación en un grafo con propiedades.
- Es posible que un esquema RDF defina herencia a través de relaciones de sub-clase y sub-propiedad. Esta característica no es soportada (actualmente) por los SGBD orientados a grafos.
- Una base de datos RDF puede contener información semántica que permite realizar inferencia de datos (es decir, deducir triples en base a los triples existentes). Las bases de datos orientadas a grafos no están pensadas para realizar inferencia, por lo que los sistemas asociados no soportan esta característica.

En la sección siguiente, definiremos dos métodos que pretenden solucionar algunos de los problemas mencionados anteriormente.

V. Métodos para Transformar Grafos RDF en Grafos con Propiedades

Si comparamos un grafo RDF con un grafo con propiedades, encontraremos que ambos comparten las características básicas de un grafo: están compuestos de nodos y aristas los cuales están etiquetados con algún valor; las aristas son dirigidas y etiquetadas; y es posible tener varias aristas entre dos nodos.

Por otro lado, también se pueden encontrar diferencias importantes entre ambos tipos de grafos:

- Un grafo RDF contiene nodos de tipo Recurso (IRI) y nodos de tipo Literal (valor). En contraste, un grafo con propiedades solo permite nodos de tipo entidad, mientras que los atributos y sus valores se encuentran contenidos en los nodos (como propiedades).
- Cada nodo/arista de un grafo RDF se etiqueta con un único valor (IRI o Literal), mientras que cada nodo/arista de un grafo con propiedades puede incluir, además de una etiqueta, propiedades clave-valor.
- Un grafo RDF soporta propiedades multi-valuadas. En un grafo con propiedades, la propiedades son definidas como pares key-value, es decir, son mono-valuadas.
- En un grafo RDF es posible tener aristas entre aristas, lo cual no es permitido en un grafo con propiedades (por definición).
- Un nodo de un grafo RDF puede estar asociado a cero o más clases de recursos. En un grafo con propiedades es usual que un nodo este asociado a un único tipo de nodo.

A continuación presentaremos dos métodos para transformar un grafo RDF en un grafo con propiedades. La diferencia entre ambos métodos radica en la existencia de un esquema de datos RDF.

A. Transformación con Esquema de Datos

Este método asume que la base de datos RDF incluye datos y un esquema RDF. La existencia del esquema RDF permite conocer la estructura de los datos, es decir, las clases de recursos y sus propiedades (ya sean relaciones o atributos).

El método de transformación considera dos etapas: en la primera etapa, se crea un esquema de grafo con propiedades a partir del esquema de datos RDF; en la segunda etapa, los datos RDF son transformados en un grafo con propiedades siguiendo las restricciones establecidas por su esquema de datos. Considerando que el esquema RDF describe las clases de recursos y propiedades, no es difícil transformar estos elementos en tipos de nodos, aristas y propiedades. A continuación describiremos en detalle ambas etapas.

Para la primera etapa de la transformación, consideremos la función de transformación $\mathcal{F}_S$ la cual recibe como entrada

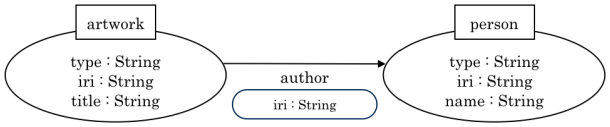

Fig. 5. Esquema de grafo con propiedades obtenido desde el esquema RDF presentado en la Figura 2.

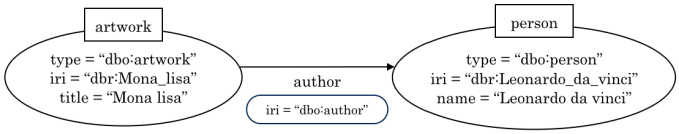

Fig. 6. Grafo con propiedades obtenido desde el grafo RDF presentado en la Figura 1.

un esquema de datos RDF S_R, y entrega como resultado un esquema de grafo con propiedades $S_P = (T_N, T_E, \beta, \delta)$. Formalmente, la función $\mathcal{F}_S(S_R) = S_P$ se define de la siguiente manera:

1) Por cada clase de recurso c definido en S_R a través de un par de triples de la forma `(`c `rdf:type rdf:class)` y `(`c `rdfs:label` l_c`)`, S_P tendrá un tipo de nodo $t_n \in T_N$ etiquetado con el valor de l_c.[9]
2) Por cada clase de propiedad p definida en S_R a través de un conjunto de triples de la forma `(`p `rdf:type rdf:Property)`, `(`p `rdfs:label` l_p`)`, `(`p `rdfs:domain` d_p`)` y `(`p `rdfs:range` r_p`)`, se tendrá que:
   a) Si r_p es la clase de recursos `rdf:Literal` o un tipo de datos específico (ej. `xsd:integer`), entonces para el tipo de nodo $t_n \in T_N$ asociado a d_p, se tendrá que $\beta(t_n, l_p) = D$ donde D es un tipo de datos compatible con r_p.
   b) Si r_p es una clase de recursos definida en S_R, entonces S_P definirá un tipo de arista $t_e \in T_E$ cumpliendo que $t_e \in \delta(t_n, t'_n)$ donde t_n es el tipo de nodo asociado a d_p y t'_n es el tipo de nodo asociado a r_p.
3) Todo tipo de nodo $t_n \in T_N$ definirá un atributo por defecto $\beta(t_n, \texttt{'type'}) = \texttt{String}$, el cual mantendrá la IRI de una clase RDF.
4) Todo tipo (de nodo o arista) $t \in T_N \cup T_E$ definirá un atributo por defecto $\beta(t, \texttt{'iri'}) = \texttt{String}$ el cual mantendrá la IRI de un recurso RDF.

Cabe mencionar que los atributos por defecto `type` e `iri`, definidos en los puntos 3 y 4, fueron incluidos con la finalidad de mantener información original de los datos RDF. Ambos atributos no son fundamentales para la transformación por lo que podrían ser eliminados.

Para la segunda etapa de la transformación, definiremos una función $\mathcal{F}_G$ la cual recibe como entrada un grafo RDF G_R junto a su esquema de datos S_R, y entrega como resultado un grafo con propiedades $G_P = (N, E, \rho, \lambda, \sigma)$, cumpliéndose que G_P es válido con respecto al esquema de grafo con propiedades $S_P = \mathcal{F}_S(S_R)$.

La función $\mathcal{F}_G(G_R, S_R) = G_P$ se define formalmente de la siguiente manera: Por cada recurso $r \in I$ descrito en G_R, existirá un nodo $n \in N$ en G_P cumpliendo que:

1) Si G_R contiene un triple `(`r `rdf:type` c`)` y S_R contiene el triple `(`c `rdfs:label` l_c`)`, entonces se tendrá que $\lambda(n) = l_c$, $\sigma(n, \texttt{'type'}) = c$ y $\sigma(n, \texttt{'iri'}) = r$;
2) Por cada propiedad p de r, descrita a través de los triples `(`r p v`)` en G_R y `(`p `rdfs:label` l_p`)` en S_R, donde v es un literal, se tendrá que n presenta un atributo $\sigma(n, l_p) = v$;
3) Por cada propiedad p de r, descrita a través de los triples `(`r p r'`)` en G_R y `(`p `rdfs:label` l_p`)` en S_R, donde $r' \in I$ es otro recurso en G_R, existirán un nodo $n' \in N$ (asociado a r') y una arista $e \in E$ cumpliendo que $\rho(n, n') = e$, $\lambda(e) = l_p$ y $\sigma(e, \texttt{'iri'}) = p$.

Asuma que G_R es el grafo RDF de la Figura 1. La Figura 5 muestra el esquema de grafo con propiedades S_P resultante de aplicar $\mathcal{F}_S(G_R)$, mientras que la Figura 6 muestra el grafo con propiedades obtenido luego de aplicar $\mathcal{F}_G(G_R, S_R)$.

B. Transformación sin Esquema de datos

Este segundo método asume que la fuente de datos RDF no contiene un esquema RDF explícito, es decir, solo disponemos de un conjunto de triples RDF que conforman un grafo RDF de entrada. Dada esta condición, no podemos crear un esquema de grafo con propiedades que refleje la estructura de los datos RDF. En este sentido, la estrategia de transformación consistirá en definir un esquema de grafo que sea lo suficientemente flexible para soportar cualquier grafo RDF. Lo primero que haremos es definir este esquema de grafo genérico.

Asuma un esquema de grafo $S_P = (T_N, T_E, \beta, \delta)$ como el que se muestra en la Figura 7. Formalmente. tendremos que:

$T_N = \{\texttt{Recurso}, \texttt{Literal}\}$,
$T_E = \{\texttt{Atributo}, \texttt{Relación}\}$,
$\beta(\texttt{Recurso}, \texttt{iri}) = \texttt{String}$,
$\beta(\texttt{Literal}, \texttt{valor}) = \texttt{String}$,
$\beta(\texttt{Atributo}, \texttt{iri}) = \texttt{String}$,
$\beta(\texttt{Relación}, \texttt{iri}) = \texttt{String}$,
$\delta(\texttt{Recurso}, \texttt{Literal}) = \{\texttt{Atributo}\}$,
$\delta(\texttt{Recurso}, \texttt{Recurso}) = \{\texttt{Relación}\}$.

Dado el esquema genérico S_P, definiremos la función $\mathcal{T}_G$ la cual recibe como entrada un grafo RDF G_R y entrega como resultado un grafo con propiedades $G_P = (N, E, \rho, \lambda, \sigma)$, satisfaciendo que G_P es válido con respecto a S_P. Específicamente, la función $\mathcal{T}_G(G_R) = G_P$ se define de la siguiente manera:

Por cada triple $t = (v_1, v_2, v_3)$ en G_R tendremos que:

1) Si $v_3 \in I$ (es decir, t define una relación de v_1) entonces G_P contendrá dos nodos $n, n' \in N$ y una arista $e \in E$ satisfaciendo que $\rho(n, n') = e$, $\lambda(n) = \texttt{'Recurso'}$, $\lambda(n') = \texttt{'Recurso'}$, $\lambda(e) =$

[9]Un inconveniente que puede surgir al momento de generar las etiquetas para tipos de nodos y aristas, es que el valor l de una propiedad *rdfs:label* sea un texto con múltiples palabras. En este caso, la solución simple es reemplazar los espacios en blanco en l por sub-guiones "_".

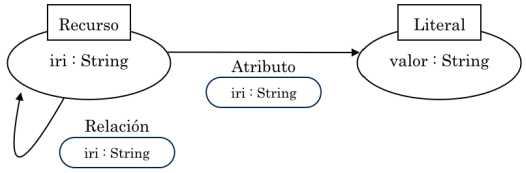

Fig. 7. Esquema de grafo con propiedades genérico.

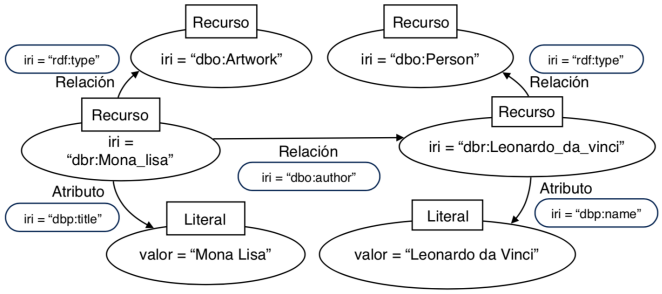

Fig. 8. Grafo con propiedades obtenido desde el grafo RDF de la Figura 1. Este grafo satisface la estructura definida por el esquema de la Figura 7.

$\texttt{'Relación'}$, $\sigma(n, \texttt{'iri'}) = v_1$, $\sigma(e, \texttt{'iri'}) = v_2$ y $\sigma(n', \texttt{'iri'}) = v_3$.

2) Si $v_3 \in L$ (es decir, t define un atributo de v_1) entonces P_G contendrá dos nodos $n, n' \in N$ y una arista $e \in E$ satisfaciendo que $\rho(n, n') = e$, $\lambda(n) = \texttt{'Recurso'}$, $\lambda(n') = \texttt{'Literal'}$, $\lambda(e) = \texttt{'Atributo'}$, $\sigma(n, \texttt{'iri'}) = v_1$, $\sigma(e, \texttt{'iri'}) = v_2$ y $\sigma(n', \texttt{'valor'}) = v_3$.

Asuma que G_R es el grafo RDF de la Figura 1 y S_P es el esquema genérico de la Figura 7. La Figura 8 muestra el grafo con propiedades obtenido luego de aplicar $\mathcal{T}_G(G_R)$.

Nótese que el grafo de la Figura 8 es menos natural que el grafo de la Figura 6, lo cual afecta en cierto grado su uso (en particular, el diseño de consultas). Sin embargo, el segundo método de transformación es mucho más flexible, por lo que permite transformar cualquier tipo de grafo RDF, sin importar si su esquema de datos se encuentra descrito total o parcialmente.

VI. Conclusiones

Este artículo presenta dos métodos para transformar un grafo RDF en un grafo con propiedades. El primer método asume la existencia de un esquema RDF, lo cual permite crear un esquema de grafo con propiedades que refleje la estructura de los datos originales. El segundo método no considera la existencia de un esquema RDF, por lo tanto define un esquema de grafo genérico el cual es lo suficientemente flexible para modelar cualquier tipo de grafo RDF.

Los dos métodos propuestos en este artículo comparten algunas características de las propuestas existentes (descritas en la Sección IV). La principal diferencia y novedad de nuestra propuesta es que se considera la existencia de un esquema RDF, por lo que se incluyó la correspondiente transformación hacia un esquema de grafo con propiedades.

Por otro lado, los métodos propuestos en este artículo presentan algunas restricciones.

- RDF Schema permite la definición de relaciones de herencia a través de los términos `rdfs:subClassOf` y `rdfs:subPropertyOf`. Considerando que las bases de datos de grafo no soportan herencia, esta noción no es soportada por ninguno de los métodos de transformación. Sin embargo, podría simularse duplicando atributos y relaciones en concordancia con noción de herencia definida por RDF Schema.
- En un grafo RDF es posible encontrar que un nodo está asociado a distintas clases de recursos. Para simular esta característica en una base de datos de grafo, sería necesario que un nodo pueda estar asociado a distintos tipos de nodos. Sin embargo, no todos los SGBD para grafos soportan esta funcionalidad.
- En un grafo RDF, un recurso puede aparecer como sujeto y propiedad al mismo tiempo, es decir, es factible tener aristas entre aristas. Los métodos de transformación propuestos no se ven afectados por descripciones RDF del tipo anterior. Sin embargo, en caso de que ocurra, la representación en el grafo con propiedades resultante será extraña ya que se tendrá una duplicación del recurso (un nodo representando al recurso en su rol de sujeto, y una arista representando al recurso en su rol de propiedad).
- Otra característica de RDF, no considerada en este artículo, es la noción de nodos blancos. En términos sencillos, un nodo blanco es un tipo de recurso anónimo cuyo identificador es un número en lugar de una IRI. Para transformar un grafo RDF conteniendo nodos blancos, una opción razonable y sencilla consiste en reemplazar consistentemente los identificadores de nodos blancos por IRIs.
- En un grafo RDF es posible que un recurso tenga múltiples valores para una misma propiedad. Esta característica no es soportada por el primer método de transformación ya que nuestra definición de grafo con propiedades indica que cada propiedad de un nodo tiene asignado un único valor (esto es consistente con el hecho que la mayoría de las bases de datos de grafo indican que los atributos son pares clave-valor). Una solución sencilla consiste en modificar nuestra definición de grafo con propiedades, permitiendo que una propiedad contenga múltiples valores. Una segunda solución, empleada por algunos sistemas para bases de datos de grafo, consiste en soportar el tipo de datos *Array*. De esta manera, un atributo podría contener múltiples valores codificados como un array.

Cabe mencionar que sería posible definir un método intermedio, pensado para un grafo RDF cuyo contenido incluye una descripción parcial de su esquema de datos. Bajo esta situación, es factible hacer un análisis de los triples RDF con el fin de "descubrir" el esquema de datos subyacente. Este proceso de análisis, denominado *schema discovery* [26], podría ser aplicado inicialmente para obtener el esquema de datos, y posteriormente aplicar el primer método de transformación.

Como trabajo futuro podemos mencionar la necesidad de implementar los métodos propuestos, lo cual permitirá vali-

dar su definición, además de realizar algunos experimentos para evaluar su aplicabilidad y eficiencia. Adicionalmente, se plantea diseñar otros métodos de transformación, además de estudiar el problema inverso, es decir, transformar grafos con propiedades en grafos RDF.

AGRADECIMIENTOS

Renzo Angles y Roberto García son financiados por el Instituto Milenio de Investigación sobre los Fundamentos de los Datos (Chile). Los autores desean agradecer a Harsh Thakkar (University of Bonn) por sus comentarios y recomendaciones respecto de los métodos presentados en este artículo.

REFERENCIAS

[1] S. Sakr and E. Pardede, *Graph Data Management: Techniques and Applications*, 1st ed. Information Science Reference - IGI Publishing, 2011.
[2] R. Angles and C. Gutierrez, "Survey of graph database models," *ACM Computing Surveys (CSUR)*, vol. 40, no. 1, pp. 1–39, 2008.
[3] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data-the story so far," *International journal on Semantic Web and Information Systems*, 2009.
[4] S. Harris and A. Seaborne, "SPARQL 1.1 Query Language, W3C Recommendation," https://www.w3.org/TR/sparql11-query/, March 21 2013.
[5] D. Brickley and R. V. Guha, "RDF Schema 1.1, W3C Recommendation," https://www.w3.org/TR/rdf-schema/, 2014.
[6] W3C, "Web Ontology Language (OWL)," https://www.w3.org/OWL/.
[7] M. A. Corral, L. Antonelli, and L. E. Sánchez, "Ontologías de Salud y Sistemas de Información: Revisión Sistemática," *IEEE Latin American Transactions*, vol. 15, no. 1, pp. 103–120, Jan 2017.
[8] R. Angles and C. Gutierrez, "An Introduction to Graph Data Management," in *Graph Data Management*, ser. Data-Centric Systems and Applications. Springer Nature, 2018, ch. 1.
[9] J. L. C. Silva, L. Rocha, and B. C. H. Silva, " New Algorithm for Finding All Tours and Hamiltonian Circuits in Graphs," *IEEE Latin American Transactions*, vol. 14, no. 2, pp. 831–836, Feb 2016.
[10] C. Parent and S. Spaccapietra, "Database integration: the key to data interoperability," *Advances in Object-Oriented Data Modeling*, 2000.
[11] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa, "Data Exchange: Semantics and Query Answering," *Theoretical Computer Science*, vol. 336, no. 1, pp. 89–124, May 2005.
[12] S.Ceri, L. Tanca, and R. Zicari, "Supporting interoperability between new database languages," in *Proc. of the 5th Annual European Computer Conference (CompEuro)*, 1991.
[13] J. Park and S. Ram, "Information Systems Interoperability: What Lies Beneath?" *ACM Transactions on Information Systems*, vol. 22, no. 4, pp. 595–632, Oct. 2004.
[14] R. Angles, P. Boncz, J. Larriba-Pey, I. Fundulaki, T. Neumann, O. Erling, P. Neubauer, N. Martinez-Bazan, V. Kotsev, and I. Toma, "The Linked Data Benchmark Council: a Graph and RDF industry benchmarking effort," *Sigmod Record*, vol. 43, no. 1, March 2014.
[15] G. Klyne and J. Carroll, "Resource Description Framework (RDF) Concepts and Abstract Syntax," http://www.w3.org/TR/2004/REC-115-concepts-20040210/, February 2004.
[16] A. Hogan, M. Arenas, A. Mallea, and A. Polleres, "Everything you always wanted to know about blank nodes," *Journal of Web Semantics*, vol. 27, no. 1, 2014.
[17] R. Angles, "The Property Graph Database Model," in *Proc. Alberto Mendelzon International Workshop on Foundations of Data Management (AMW)*, 2018.
[18] J. Hayes and C. Gutierrez, "Bipartite Graphs as Intermediate Model for RDF," in *Proc. of the Int. Semantic Web Conference (ISWC)*, ser. LNCS. Springer-Verlag, 2004, pp. 47–61.
[19] Valerie Bönström and Annika Hinze and Heinz Schweppe, "Storing RDF as a graph," in *Proceedings of the First Latin American Web Congress*. IEEE Computer Society, 2003.
[20] V. Nguyen, J. Leeka *et al.*, "A formal graph model for RDF and its implementation," arXiv:1606.00480, Tech. Rep., June 2016.
[21] V. Nguyen, O. Bodenreider, and A. Sheth, "Don't Like RDF Reification?: Making Statements About Statements Using Singleton Property," in *Proc. of the International Conference on World Wide Web*. ACM, 2014, pp. 759–770.
[22] O. Hartig, "Reconciliation of RDF* and Property Graphs," *The Computing Research Repository (CoRR)*, vol. abs/1409.3288, 2014.
[23] A. Schätzle, M. Przyjaciel-Zablocki, T. Berberich, and G. Lausen, "S2X: Graph-Parallel Querying of RDF with GraphX," in *Biomedical Data Management and Graph Online Querying (VLDB Workshop)*, no. 155–168. Springer, 2016.
[24] D. Tomaszuk, "RDF Data in Property Graph Model," in *Proc. of the 10th International Conference on Metadata and Semantics Research (MTSR)*, vol. 672, 2016.
[25] S. Das, J. Srinivasan *et al.*, "A tale of two graphs: Property graphs as rdf in oracle." in *Proc. of the International Conference on Extending Database Technology (EDBT)*, 2014, pp. 762–773.
[26] M. Pham and P. A. Boncz, "Exploiting emergent schemas to make RDF systems more efficient," in *Proc. of the 15th International Semantic Web Conference (ISWC)*, ser. LNCS. Springer, 2016, pp. 463–479.

Renzo Angles recibió su grado de Doctor en Ciencias, mención Computación en la Universidad de Chile y de Bachiller en Ingeniería de Sistemas en la Universidad Católica de Santa María (Arequipa, Perú). Actualmente, es Profesor Asistente del Departamento de Ciencias de la Computación de la Universidad de Talca (Chile). Además es investigador adjunto en el Instituto Milenio de Investigación sobre los Fundamentos de los Datos (Chile). Sus áreas de interés son Bases de Datos para Grafos y Web Semántica. De manera especial investiga en la teoría y el diseño de lenguajes de consulta para grafos, benchmarking de bases de datos para grafos y RDF, y el análisis de datos con estructura de grafo.

Roberto García recibió su título profesional de Ingeniero Civil en Computación en la Universidad de Talca, Chile. Actualmente trabaja como ingeniero de software en el Instituto Milenio de Investigación sobre los Fundamentos de los Datos. Sus áreas de interés son bases de datos orientadas a grafos y lenguajes de consulta para grafos.