Publicación de Datos Abiertos siguiendo los Principios de Datos Enlazados

José A. Gilliard[1], Omar R. Perin[1], Mariela G. Rico[1], María Laura Caliusco[1,2]

[1]Centro de Investigación y Desarrollo de Ingeniería en Sistemas de Información (CIDISI) - Universidad Tecnológica Nacional - Facultad Regional Santa Fe
Lavaise 610 - S3004EWB - Santa Fe - SF - Argentina

[2]Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)

jgilliard@frsf.utn.edu.ar, omar.perin@gmail.com,
mrico@frsf.utn.edu.ar, mcaliusc@frsf.utn.edu.ar

Abstract. *This paper presents an architecture based on a semantic model for publishing open data and a practical method for the instantiation and persistence of this model using D2RQ language and Jena TDB Triple Store. Currently, more and more governments publish their data following the doctrine of Open Government. While there are several proposals of how to perform this publication, none of them are complete and, therefore, they can not be easily replicated. The aim of this paper is to be a practical guide for data publication based on a semantic model using the tools afore mentioned.*

Resumen. *Este trabajo presenta una arquitectura basada en un modelo semántico para la publicación de datos abiertos y un método práctico para el poblado y persistencia de dicho modelo utilizando el lenguaje D2RQ y el Triple Store Jena TDB. Actualmente, cada vez son más los gobiernos que publican sus datos siguiendo la doctrina de Gobierno Abierto. Si bien existen varias propuestas de cómo realizar dicha publicación, ninguna de ellas es completa y, por lo tanto, no pueden ser facilmente replicadas. El objetivo de este trabajo es ser una guía práctica para la publicación de datos basados en un modelo semántico utilizando las herramientas previamente mencionadas.*

1. Introducción

Como nos encontramos en plena transición hacia la economía del conocimiento, todo el mundo asume como natural que ciertos datos procedentes de los gobiernos deben ser compartidos, fáciles de localizar, accesibles y manipulables por todo aquel que lo desee, estableciendo una relación más abierta y transparente con la sociedad toda, dando lugar a un nuevo modelo de gobierno llamado Gobierno Abierto.

La doctrina de Gobierno Abierto se basa en la premisa de que todos los datos que produce el gobierno son públicos [Brys y Aldana Montes 2011]. Un Gobierno Abierto es aquel que entabla una constante conversación con los ciudadanos con el fin de oír lo que ellos dicen y solicitan, que toma decisiones basadas en sus necesidades y preferencias, que facilita la colaboración de los ciudadanos y funcionarios en el desarrollo de los servicios que presta y que comunica todo lo que decide y hace de

forma abierta y transparente. El Gobierno Abierto se basa en tres pilares fundamentales: transparencia, colaboración y participación [Calderón y Lorenzo 2010].

Existen varias iniciativas de Gobierno Abierto que difieren en la estrategia seguida; mientras unas iniciativas, como la de Estados Unidos, se centraron en la publicación de la mayor cantidad de información en el menor plazo posible, volcando la información en crudo tal y como está disponible, otras iniciativas, como el caso de Gran Bretaña, realizaron un tratamiento previo de los datos para reducir los esfuerzos necesarios en su posterior reutilización, empleando tecnologías de la Web Semántica para modelar la información y establecer relaciones explícitas entre los distintos conjuntos de datos, siguiendo los principios de datos enlazados [Heath y Bizer 2011].

En este último sentido existen diferentes arquitecturas y metodologías [Heath y Bizer 2011], [Villazón-Terrazas et al. 2011], [Póveda-Villalón 2012]. Sin embargo, dichas metodologías son muy generales y no explicitan en forma clara cada uno de los pasos a seguir, con lo cual se hace muy difícil replicarlas en otros casos.

El objetivo del presente trabajo es presentar una arquitectura basada en un modelo semántico de los datos a publicar para dar soporte a la doctrina de datos abiertos y un método para el poblado y persistencia de dicho modelo. El desarrollo del método se va mostrando a través de un caso práctico basado en la información a publicar del personal del Gobierno de la Provincia de Santa Fe (Argentina).

2. Arquitectura basada en un Modelo Semántico para la Publicación de Datos Abiertos siguiendo los Principios de Datos Enlazados

El término Datos Enlazados (*Linked Data*) refiere a una forma de publicar y enlazar datos estructurados en la Web utilizando RDF (*Resource Description Framework*)[1], un lenguaje para representar información sobre recursos. El valor y la utilidad de los datos enlazados es mayor en tanto y en cuanto éstos estén interconectados con otros datos en la Web de datos. Los cuatro principios de diseño en los que se basa la Web de datos enlazados son [Heath y Bizer 2011]:

- Utilizar URIs (*Uniform Resource Identifier*) como nombres únicos para los recursos.
- Utilizar el protocolo HTTP (*HyperText Transfer Protocol*) para nombrar y resolver la ubicación de los datos identificados mediante esas URIs.
- Representar los datos en RDF y utilizar SPARQL[2], un lenguaje de consulta para RDF, como lenguaje de consulta de dichos datos.
- Incluir enlaces a otras URIs para permitir la localización de más datos enlazados.

La aplicación del concepto de datos enlazados tiene dos objetivos principales en el área de gobierno: el primero es proporcionar datos públicos más accesibles a la ciudadanía en un formato reutilizable; el segundo es proporcionar un punto de acceso único a la información gubernamental, en el que los datos están conectados y es posible utilizarlos de forma automatizada por sistemas software [Villazón-Terrazas et al. 2011].

---

[1] http://www.w3.org/standards/techs/rdf#w3c_all

[2] http://www.w3.org/TR/sparql11-query/

Una arquitectura de implementación de datos abiertos siguiendo los principios de datos enlazados depende del tipo de dato del conjunto de datos origen. Si el conjunto de datos está almacenado en una base de datos (BD) relacional, los mismos pueden ser fácilmente publicables usando un *wrapper* de BD a RDF, como lo es el servidor D2R[3]. Estas herramientas permiten publicar datos definiendo mapeos entre las estructuras de la BD relacional y los grafos RDF que se suben a la Web según los principios de datos enlazados. Esta propuesta tiene dos desventajas importantes: (1) los datos se publican en crudo con lo cual un usuario final podría no obtener una respuesta adecuada a su consulta, ya que la misma va a depender de la calidad de la BD; y (2) cada vez que se realiza una consulta, la misma se debe resolver en memoria.

Para resolver los problemas antes planteados, en este trabajo se propone la arquitectura que se muestra en la Figura 1. En dicha arquitectura puede observarse que a partir de la BD relacional, fuente de los datos de origen a publicar, se crea un modelo semántico para explicitar el significado de los datos allí almacenados y así mejorar la búsqueda. Luego, utilizando un *wrapper*, se generan tripletas de valores en función de los datos almacenados en la BD y la estructura semántica del modelo semántico. A continuación, dichas tripletas se guardan en un medio de almacenamiento adecuado como lo es una BD semántica o *Triple Store* (T-Store). Finalmente, el servicio de consulta en SPARQL y los servicios de datos enlazados se basarían en dicha T-Store que se publicaría en la Web y que puede almacenar gran cantidad de datos sobre los cuales también se pueden realizar inferencias.

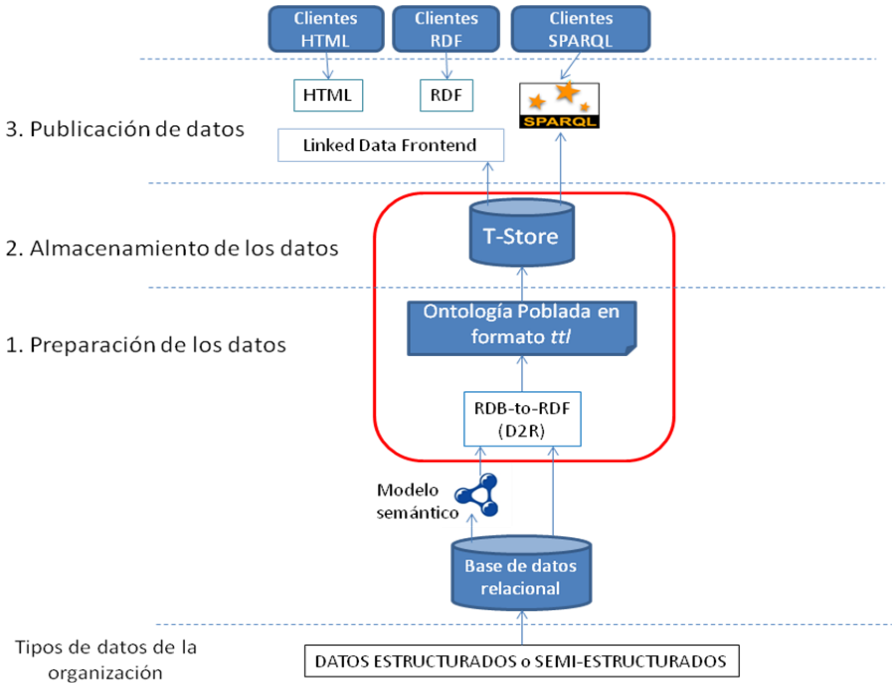

Figura 1. Arquitectura basada en un modelo semántico para la publicación de datos.

Para la implementación de dicha arquitectura se pueden utilizar diferentes *wrappers* y diferentes T-Stores. En este trabajo se presenta a continuación un método

[3] http://d2rq.org/

práctico para la implementación de dicha arquitectura utilizando D2R como *wrapper* y Jena TDB[4] como T-Store.

3. Método Práctico para la Población y Persistencia de un Modelo Semántico

El objetivo del método propuesto es obtener un almacén de datos físico donde se encuentre alojado de forma persistente un modelo semántico junto con sus instancias, generadas a partir de datos obtenidos de una BD relacional de gran porte, para poder consultarlo. El método establece una serie de actividades que permiten llevar a cabo el trabajo en forma práctica y ordenada para cualquier caso genérico.

El método consta de tres procesos, que se muestran en la Figura 2. Cada uno de los procesos tiene varias actividades y un resultado específico.

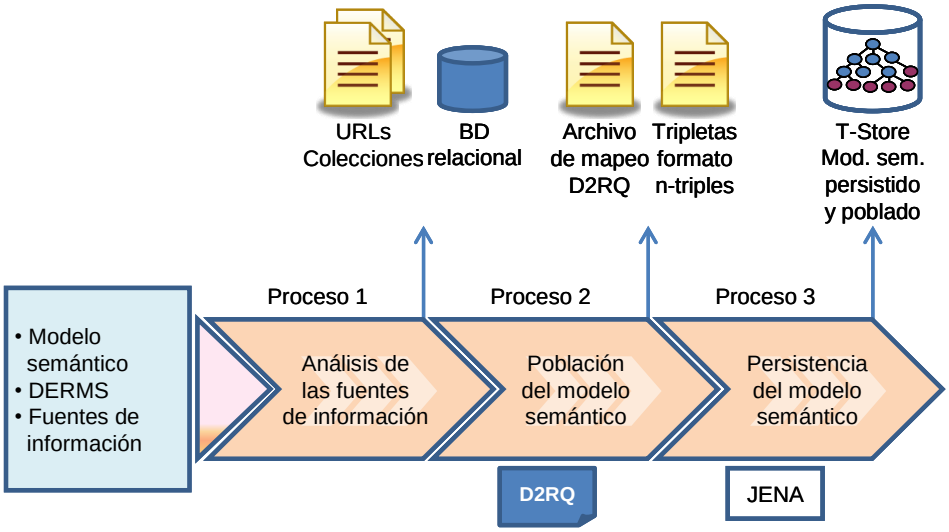

Figura 2. Método práctico para la población y persistencia de un modelo semántico.

Con el propósito de ejemplificar cada uno de los procesos y actividades, se considera el modelo semántico de información relacionada al personal del Gobierno de la Provincia de Santa Fe [Lozano 2013]. Este modelo está compuesto por cinco ontologías, una de las cuales, denominada "Lugares", representa la semántica de la información de distribución geográfica relacionada con los organismos y los datos personales de los agentes (Figura 3).

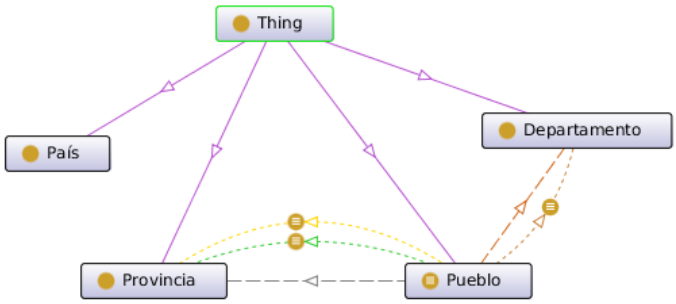

Figura 3. Ontología Lugares.

[4] http://jena.apache.org/documentation/tdb/

La ontología Lugares está compuesta por las clases País, Provincia, Departamento y Pueblo. Las instancias de estas cuatro clases poseen un nombre, definido por la propiedad homónima, y los pueblos tienen un código y sub-código postal.

Las instancias de la clase Departamento se deben obtener a partir de los datos en la tabla dptos de la BD, mientras que las instancias correspondientes a países, provincias y pueblos se deben extraer de las tablas paises, pcias y postales, respectivamente. La Figura 4 muestra un diagrama entidad-relación simplificado de la BD, que incluye sólo las tablas de utilidad para esta ontología.

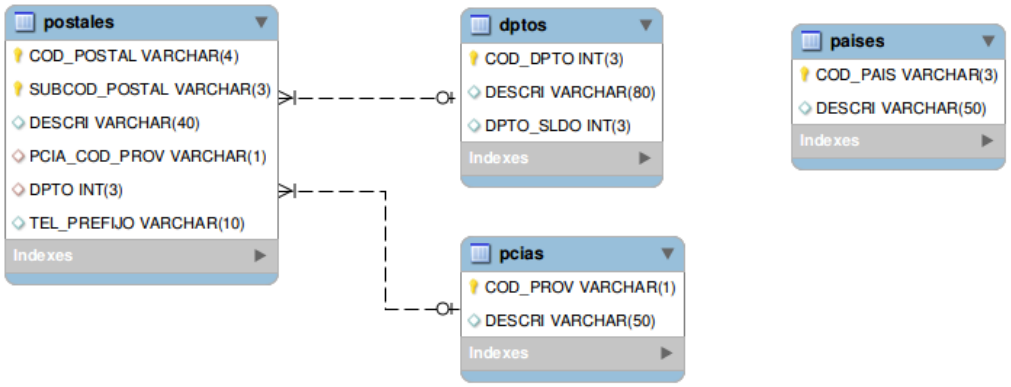

Figura 4. Diagrama entidad-relación de las tablas correspondientes a la ontología Lugares.

A continuación se describe cada uno de los procesos y actividades que componen el método propuesto.

3.1. Proceso 1. Análisis de las Fuentes de Información

Dado un modelo semántico y sus ontologías constitutivas, se debe afrontar la tarea de generar las tripletas necesarias para poblar las ontologías a partir de los datos almacenados en una BD relacional, cumpliendo con los requerimientos especificados en el Documento de Especificación de Requerimientos del Modelo Semántico (DERMS). Con este objetivo, se deben realizar las siguientes actividades:

Actividad 1.1: Analizar las fuentes de información.

El método propuesto requiere que las fuentes de información estén en el formato de BD relacional. Por ello, como primera medida se deben considerar las posibilidades tecnológicas para la extracción de la información primaria, en función de su formato original, y su adaptación al formato requerido. Luego, se debe verificar que las fuentes contengan toda la información necesaria para poblar las ontologías que componen el modelo semántico. Así, se establecen las correspondencias entre los datos de la BD y los elementos de las ontologías del modelo semántico para generar las tripletas necesarias.

Actividad 1.2: Definir una URL base para el modelo semántico y, en base a ésta, una URL para cada ontología que compone el modelo.

Para publicar datos en la Web, los elementos de un dominio de interés primero deben ser identificados [Heath and Bizer 2011]. Datos enlazados utiliza sólo URIs

HTTP por dos razones: (1) proporcionan una forma sencilla de crear nombres únicos globales de manera descentralizada (cada propietario de un nombre de dominio, o el delegado del propietario del nombre de dominio, puede crear nuevas referencias URI), y (2) sirven como medio de acceso a la información que describe la entidad identificada.

Por ejemplo, la URL `http://www.frsf.utn.edu.ar/personto/` identifica globalmente al modelo semántico.

Actividad 1.3: Definir las colecciones para las instancias de cada ontología y sus respectivas URLs.

Cada clase definida en un modelo semántico da origen a una serie de recursos, sus instancias. Los recursos de una misma clase se agrupan en una colección. Para identificar estos recursos es necesario adoptar un criterio respecto de los nombres a asignar a las colecciones y, además, construir una cadena de texto que identifique de manera unívoca a cada recurso.

Para la nominación de colecciones se utiliza el plural del sustantivo que da nombre a la clase a la cual corresponden los recursos. Así, por ejemplo, las instancias de la clase Provincia se agrupan dentro de la colección provincias.

Para la generación de los identificadores se utilizan los valores correspondientes a las claves primarias de las tablas desde las cuales se extraen las instancias. En aquellos casos donde la clave primaria está compuesta por más de un campo, se concatenan los valores de cada uno.

Así, la URI correspondiente a cualquier recurso del modelo sigue la siguiente estructura: `<http://www.frsf.utn.edu.ar/personto/colección/id_recurso>`. Por ejemplo, para las instancias de la clase Departamento, la URI es `<http://www.frsf.utn.edu.ar/personto/departamentos/id_recurso>`, donde `id_recurso` es el valor correspondiente del campo COD_DPTO de la tabla dptos. En el caso de las instancias de la clase Pueblo, extraídas a partir de la tabla postales, la URI es `<http://www.frsf.utn.edu.ar/personto/pueblos/id_recurso>`, donde `id_recurso` es la concatenación de los campos COD_POSTAL y SUBCOD_POSTAL, haciendo uso del operador `d2rq:uriPattern` (el cual se explica más adelante).

3.2. Proceso 2. Población del Modelo Semántico

Con el objetivo de poblar las ontologías que componen el modelo semántico se hace uso de la plataforma D2RQ, que provee mecanismos para automatizar la conversión de datos relacionales a tripletas `<Sujeto; Propiedad; Objeto>`. Se realizan las siguientes actividades:

Actividad 2.1: Para cada ontología, identificar grupos de tripletas a generar.

El objetivo es identificar claramente qué patrones de tripletas se deben generar, según los elementos que componen la ontología considerada. Por ejemplo, para la ontología Lugares se deben generar tripletas para la declaración e instanciación de las clases Pais, Provincia, Departamento y Pueblo; tripletas para la declaración e instanciación de la propiedad nombre de las clases Pais, Provincia, Departamento y Pueblo; tripletas para la declaración e instanciación de las propiedades codigoPostal y subCodigoPostal de la clase Pueblo; tripletas para la declaración e instanciación de la

relación perteneceAlDepartamento entre intancias de las clases Pueblo y Departamento; y tripletas para la declaración e instanciación de la relación perteneceAProvincia entre instancias de las clases Pueblo y Provincia.

Actividad 2.2: Definir los patrones que seguirán las componentes sujeto, propiedad y objeto de las tripletas de cada grupo.

El objetivo de esta actividad es identificar la composición de las tripletas que se deben generar para representar las definiciones de clases, instancias y sus propiedades, y las relaciones entre instancias de las clases. Estas tripletas permitirán convertir la ontología modelada en una ontología poblada, una ontología con instancias.

Para cada ontología, se necesita crear los siguientes tipos de tripletas:

- Declaración de clases. Una tripleta que identifique cada clase como tal y dos tripletas, opcionales, para agregar comentarios y etiquetas. Tres tripletas en total por cada clase de la ontología. Por ejemplo, las siguientes tres tripletas definen la clase Pais:

```
<http://www.frsf.utn.edu.ar/personto/lugares#Pais>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.w3.org/2000/01/rdf-schema#Class> .

<http://www.frsf.utn.edu.ar/personto/lugares#Pais>
<http://www.w3.org/2000/01/rdf-schema#label>
"Pais" .

<http://www.frsf.utn.edu.ar/personto/lugares#Pais>
<http://www.w3.org/2000/01/rdf-schema#comment>
"Representa el pais de nacimiento de un Agente." .
```

  La primera de ellas establece que Pais es un recurso de tipo Clase para el esquema rdf y que corresponde a la ontología Lugares del modelo personto. La segunda y tercer tripleta, respectivamente, agregan una etiqueta y un comentario a la definición de la clase.

- Instanciación de clases. Una tripleta por cada instancia de cada clase de la ontología. Por ejemplo, la siguiente tripleta establece que la instancia localizada en la URI `http://www.frsf.utn.edu.ar/personto/paises/AR` es un objeto del tipo Pais, definido por el recurso `http://www.frsf.utn.edu.ar/personto/lugares#Pais`. "`AR`" es la identificación de la instancia correspondiente al país Argentina.

```
<http://www.frsf.utn.edu.ar/personto/paises/AR>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.frsf.utn.edu.ar/personto/lugares#Pais> .
```

- Declaración de propiedades de instancias. Una tripleta que identifique cada propiedad como tal y, opcionalmente, dos más para agregar comentarios y etiquetas. Tres tripletas en total por cada propiedad definida. En los casos en que el dominio de la propiedad está integrado por más de una clase, se genera una única tripleta para identificar a la propiedad, independientemente de la cantidad de clases que integren el dominio. Se deben crear igualmente las tripletas de etiqueta y comentario para cada clase incluida en el dominio. Por ejemplo, la propiedad nombre de la clase Pais se define por las siguientes tripletas:

```
<http://www.frsf.utn.edu.ar/personto/lugares#nombre>
```

```
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#Property> .

<http://www.frsf.utn.edu.ar/personto/lugares#nombre>
<http://www.w3.org/2000/01/rdf-schema#label>
"Nombre pais" .

<http://www.frsf.utn.edu.ar/personto/lugares#nombre>
<http://www.w3.org/2000/01/rdf-schema#comment>
"Nombre del pais" .
```

Un departamento también tiene un nombre, que se extrae del campo DESCRI de la tabla dptos. Dado que se trata de una única propiedad (nombre) con dominios múltiples (las clases Pais, Provincia, Departamento y Pueblo), no es necesario generar otra tripleta `<lugares:nombre; rdf:type; rdf:Property>` en el lote. Las tripletas para las etiquetas y comentarios, en cambio, sí deben agregarse para cada clase:

```
<http://www.frsf.utn.edu.ar/personto/lugares#nombre>
<http://www.w3.org/2000/01/rdf-schema#label>
"Nombre departamento" .

<http://www.frsf.utn.edu.ar/personto/lugares#nombre>
<http://www.w3.org/2000/01/rdf-schema#comment>
"Nombre del departamento" .
```

- Instanciación de las propiedades de instancias. Una tripleta por cada propiedad de cada instancia de cada clase. Por ejemplo, la siguiente tripleta agrega el valor "`Argentina`" a la propiedad nombre de la instancia "`AR`" de la clase Pais:

```
<http://www.frsf.utn.edu.ar/personto/paises/AR>
<http://www.frsf.utn.edu.ar/personto/lugares#nombre>
"Argentina" .
```

- Declaración de relaciones entre instancias. Una tripleta que identifique cada relación como tal y, opcionalmente, dos más para agregar comentarios y etiquetas. Tres tripletas en total por cada relación definida. Por ejemplo, la relación perteneceAlDepartamento entre instancias de la clase Pueblo e instancias de la clase Departamento se declara de la siguiente forma:

```
<http://www.frsf.utn.edu.ar/personto/lugares#perteneceAlDepartamento>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#Property> .

<http://www.frsf.utn.edu.ar/personto/lugares#perteneceAlDepartamento>
<http://www.w3.org/2000/01/rdf-schema#label>
"Pertenece al departamento" .

<http://www.frsf.utn.edu.ar/personto/lugares#perteneceAlDepartamento>
<http://www.w3.org/2000/01/rdf-schema#comment>
"El pueblo sujeto pertenece al departamento objeto" .
```

- Instanciación de las relaciones entre instancias. Una tripleta por cada par de instancias relacionadas. Por ejemplo, una instanciación de la relación anterior, donde "`3551-S`" es la identificación de la instancia correspondiente al pueblo "Los Amores" y "`149`" es la identificación del departamento "Vera", es:

```
<http://www.frsf.utn.edu.ar/personto/pueblos/3551-S>
<http://www.frsf.utn.edu.ar/personto/lugares#perteneceAlDepartamento>
<http://www.frsf.utn.edu.ar/personto/departamentos/149> .
```

Actividad 2.3: Calcular la cantidad de tripletas a generar para cada grupo en base a consultas SQL a las BD de origen. Estas cantidades se utilizan luego para verificar la cantidad de tripletas generadas por los lotes.

Por ejemplo en el caso de la ontología Lugares, la cantidad de tripletas de cada grupo a generar se detalla en la tabla siguiente:

Tabla 1. Cantidad de tripletas a generar para la ontología Lugares

	Tipo de tripletas	Cantidad
1	Declaración clase Pais	3
2	Instanciación clase Pais	240
3	Declaración propiedad nombre, clase Pais	3
4	Instanciación propiedad nombre, clase Pais	240
5	Declaración clase Departamento	3
6	Instanciación clase Departamento	25
7	Declaración propiedad nombre, clase Departamento	2
8	Instanciación propiedad nombre, clase Departamento	25
9	Declaración clase Provincia	3
10	Instanciación clase Provincia	25
11	Declaración propiedad nombre, clase Provincia	2
12	Instanciación propiedad nombre, clase Provincia	25
13	Declaración clase Pueblo	3
14	Instanciación clase Pueblo	1416
15	Declaración propiedad nombre, clase Pueblo	2
16	Instanciación propiedad nombre, clase Pueblo	1416
17	Declaración propiedad codigoPostal, clase Pueblo	3
18	Instanciación propiedad codigoPostal, clase Pueblo	1416
19	Declaración propiedad subCodigoPostal, clase Pueblo	3
20	Instanciación propiedad subCodigoPostal, clase Pueblo	1416
21	Declaración relación perteneceAlDepartamento entre intancias de las clases Pueblo y Departamento	3
22	Instanciación relación perteneceAlDepartamento entre instancias de las clases Pueblo y Departamento	1364
23	Declaración relación perteneceAProvincia entre instancias de las clases Pueblo y Provincia	3
24	Instanciación relación perteneceAProvincia entre instancias de las clases Pueblo y Provincia	1416
	TOTAL	9057

Actividad 2.4: Para cada ontología, redactar un archivo de mapeo D2RQ que permita generar los lotes correspondientes de forma automatizada y que contenga:

- Los prefijos necesarios, incluyendo uno para cada ontología y una URL base.

  Para la definición de todos los mapeos se hace uso del concepto de prefijo. Un prefijo es una cadena simple de texto que se puede reemplazar en el inicio de una URI, seguido de dos puntos. Esto permite simplificar la escritura y lectura de URIs complejas al reemplazar el prefijo completo por su abreviatura. Por ejemplo, en D2RQ se define el prefijo paises de la siguiente manera:

`@prefix lugares: <http://www.frsf.utn.edu.ar/persontо/lugares#>`. Luego, la URI `<http://www.frsf.utn.edu.ar/persontо/lugares#Pais>` se puede escribir como `<lugares:Pais>`.

- Un objeto `d2rq:database` por cada BD utilizada.

  Además de declarar abreviaturas de prefijos, se deben establecer las conexiones a las BD con las que se va a trabajar para la extracción de datos. Para ello, un objeto `d2rq:Database` define una conexión JDBC a una BD relacional local o remota. Un archivo de mapeo D2RQ puede contener más de un objeto `d2rq:Database`, permitiendo así el acceso a múltiples BD. Una vez declarado este objeto, cualquier acceso a la BD se realiza haciendo referencia al mismo.

  En el siguiente bloque de código, el mapeo `map:persontoDB` es un objeto de tipo `d2rq:Database` que establece una conexión con la BD relacional MySql local persontо haciendo uso del driver jdbc:mysql con usuario root definido mediante la propiedad `d2rq:username` y contraseña "proyecto" indicada por la propiedad `d2rq:password`.

```
map:persontoDB a d2rq:Database;
      d2rq:jdbcDriver "com.mysql.jdbc.Driver";
      d2rq:jdbcDSN
"jdbc:mysql://localhost/persontо?autoReconnect=true";
      d2rq:username "root";
      d2rq:password "proyecto";
      jdbc:keepAlive "3600";
      jdbc:keepAliveQuery "SELECT 1";
      .
```

- Un objeto `d2rq:ClassMap` por cada clase. Con este objeto se especifica de qué manera se generan las URIs para identificar los recursos instancias de la clase. Se sugiere formar los identificadores de instancia concatenando los valores de todos los campos que formen parte de la clave primaria de la tabla de origen. Cada `d2rq:ClassMap` está conectado a una `d2rq:Database` y tiene un conjunto de `d2rq:PropertyBridge`, que agregan propiedades a las instancias o recursos RDF.

  Un único mapeo `d2rq:ClassMap` genera tanto la declaración de una clase como sus instancias. Lo mismo sucede para el caso de las propiedades. Así, el siguiente bloque de código mapea la clase Pais de la ontología Lugares, mediante la definición del mapeo `map:Paises`, y genera las tripletas de definición de la clase, una etiqueta y comentario para la misma, y una tripleta por cada instancia.

```
map:Paises a d2rq:ClassMap;
      d2rq:dataStorage map:persontoDB;
      d2rq:uriPattern "paises/@@paises.COD_PAIS|encode@@";
      d2rq:class lugares:Pais;
      d2rq:classDefinitionLabel "Pais";
      d2rq:classDefinitionComment   "Representa   el   pais   de
nacimiento de un Agente.";
      .
```

  La propiedad `d2rq:dataStorage` especifica que el objeto obtendrá datos de la BD definida anteriormente por el objeto `map:persontoDB`, mientras que la propiedad `d2rq:classDefinitionLabel` establece una etiqueta del tipo

`rdfs:label` para la definición de la clase y `d2rq:classDefinitionComment` agrega a la misma un comentario de texto del tipo `rdfs:comment` que permite describir en forma breve la naturaleza de las instancias de la clase.

Cada registro de la tabla paises dará origen a una instancia de la clase Pais, que estará identificada y será posible acceder a la misma por medio de la URI compuesta por el infijo paises/, correspondiente a la colección, seguido del código del país, en base al criterio explicado anteriormente para la generación de identificadores de recurso. Esto se logra haciendo uso de la propiedad `d2rq:uriPattern`. Se dice que paises/ es un infijo, dado que a toda URI generada a partir de un mapeo D2RQ se le añade un prefijo global correspondiente a una URL base del modelo, que se establece en tiempo de ejecución al momento de generar un volcado de tripletas. En el caso del ejemplo, esta URL base es http://www.frsf.utn.edu.ar/personto/, que identifica globalmente al modelo.

- Un objeto `d2rq:propertyBridge` por cada propiedad de instancia.

Una propiedad de instancia puede generarse de varias formas. El valor de la misma puede obtenerse, por ejemplo, a partir de un valor simple de una columna de la misma tabla desde la cual se originó la instancia, una columna de otra tabla, un patrón generado a partir de la concatenación de varias columnas o una correspondencia entre valores enumerados.

El siguiente fragmento de código especifica el objeto `map:Pais_nombre` que agrega a las instancias de la clase Pais, definida por el objeto `map:Paises`, la propiedad `lugares:nombre`, obteniendo los valores a partir de la columna DESCRI de la tabla paises mediante el uso de la propiedad `d2rq:column`.

```
map:Pais_Nombre a d2rq:PropertyBridge;
      d2rq:belongsToClassMap map:Paises;
      d2rq:property lugares:nombre;
      d2rq:column "paises.DESCRI";
      d2rq:propertyDefinitionComment "Nombre del país";
      d2rq:propertyDefinitionLabel "Nombre país";
```

El siguiente mapeo genera las tripletas necesarias para la definición de la clase Departamento y sus instancias:

```
map:Departamentos a d2rq:ClassMap;
      d2rq:dataStorage map:persontoDB;
      d2rq:uriPattern "departamentos/@@dptos.COD_DPTO|encode@@";
      d2rq:class lugares:Departamento;
      d2rq:classDefinitionLabel "Departamento";
      d2rq:classDefinitionComment   "Refiere   a   la   división
geográfica de la Provincia de Santa Fe.";
```

El modificador `|encode` a continuación de la indicación de cada columna es opcional y permite convertir cualquier caracter no válido, como espacios en blanco o caracteres especiales, a la notación estándar de porcentajes definida por la W3C[5].

- Un objeto `d2rq:propertyBridge` por cada relación entre instancias.

[5] http://www.w3.org/TR/2012/REC-r2rml-20120927/#dfn-iri-safe

Una relación entre instancias se mapea generando un bloque de tipo `d2rq:PropertyBridge` con el agregado de una o más propiedades `d2rq:join`, que fijan la relación de igualdad entre campos de tabla, y las propiedades `d2rq:belongsToClassMap` y `d2rq:refersToClassMap`. Pensando una relación entre instancias como una tripleta <sujeto, propiedad, objeto>, la propiedad `d2rq:belongsToClassMap` indica el mapeo que da origen a las instancias sujeto, mientras que `d2rq:refersToClassMap` indica las instancias objeto. Los signos < y > se utilizan en los casos que sea posible, para indicar mediante la punta de flecha cuál de los lados de la condición de igualdad corresponde a una clave primaria. Esto posibilita al motor D2RQ optimizar el mecanismo interno de resolución de consultas SQL, permitiendo reducir los tiempos de procesamiento requeridos para la transformación de los datos.

El siguiente mapeo D2RQ genera la relación perteneceAlDepartamento entre instancias de las clases Pueblo y Departamento. Las instancias relacionadas serán aquellas que surjan de registros de las tablas postales y dptos, donde los valores del campo DPTO de la primera tabla coincidan con los del campo COD_DPTO de la segunda.

```
map:Pueblo_perteneceAlDepartamento_Departamento                a
d2rq:PropertyBridge;
      d2rq:belongsToClassMap map:Pueblos;
      d2rq:property lugares:perteneceAlDepartamento;
      d2rq:refersToClassMap map:Departamentos;
      d2rq:join "postales.DPTO => dptos.COD_DPTO";
      d2rq:propertyDefinitionComment  "El  pueblo  pertenece  al
departamento";
      d2rq:propertyDefinitionLabel "Pertenece al departamento";
      .
```

Actividad 2.5: Generar un volcado de tripletas serializadas en formato n-triples.

Una vez redactado el archivo de mapeo para cada ontología, se utiliza la herramienta dump-rdf de la plataforma d2rq para generar un volcado de tripletas serializadas en formato n-triples. Se utiliza este formato por ser el más adecuado para persistir los lotes generados en un T-Store.

El siguiente comando ejecuta la herramienta dump-rdf utilizando como path base `http://www.frsf.utn.edu.ar/personto/`, a partir del archivo de mapeo lugares.ttl y almacenando el resultado en el archivo lugares.nt:

```
# ./dump-rdf -f N-TRIPLE -b http://www.frsf.utn.edu.ar/personto/ -o
lugares.nt --verbose lugares.ttl
```

El resultado de la ejecución es un archivo de texto conteniendo el lote completo de tripletas generadas a partir de los mapeos redactados en el archivo lugares.ttl.

Actividad 2.6: Verificar si se generaron las cantidades de tripletas correctas.

Las herramientas de línea de comandos `grep` y `wc`, combinadas de manera adecuada, permiten contar la cantidad de líneas en un archivo que contengan un patrón de texto especificado. Mediante el uso de expresiones regulares se puede contar qué cantidad de tripletas responden a cada uno de los grupos particulares generados en el volcado. Luego se comparan estas cantidades con las calculadas en la Actividad 2.3.

Así, por ejemplo, se obtiene la cantidad de instancias de la clase Pais generadas, con el siguiente comando:

```
# grep "^<http://www.frsf.utn.edu.ar/personto/paises/[A-Z]\{1,3\}>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.frsf.utn.edu.ar/personto/lugares#Pais> .$" lugares.nt | wc
-l
```

La expresión regular indica que el código del país es una secuencia de entre 1 y 3 letras mayúsculas (`[A-Z]\{1,3\}`).

La ejecución del comando retorna como resultado el número 240, indicando que el patrón pasado como primer parámetro aparece 240 veces dentro del archivo pasado como segundo parámetro. Comparando este resultado con la cantidad correspondiente en la Tabla 1 se verifica que la cantidad de tripletas generada es correcta.

3.3. Proceso 3. Persistencia del Modelo Semántico

Actividad 3.1: Crear un Triple Store Jena TDB y cargar cada uno de los lotes de tripletas generados.

El componente Jena TDB, del framework de desarrollo Jena, es un almacén RDF de alta performance. Mediante el comando `tdbloader` se puede crear el almacén (si el directorio destinado al almacén está vacío) y/o agregar tripletas, pasando como parámetros el directorio que contendrá los nodos e índices y el archivo de volcado a cargar. En el siguiente ejemplo se realiza la carga de las tripletas y la generación de índices para el archivo Lugares.ttl, en el directorio destinado al almacén /opt/SemanticIDE/proyecto1/store.

```
apache-jena-2.11.1/bin/tdbloader -loc /opt/SemanticIDE/proyecto1/store
/opt/SemanticIDE/proyecto1/volcados/Lugares.ttl
```

4. Conclusiones

En este trabajo se presentó una arquitectura para la publicación de datos abiertos siguiendo los principios de datos enlazados. A diferencia de trabajos anteriores, dicha arquitectura propone la definición de un modelo semántico de los datos a publicar, en lugar de publicar los datos en crudo. De esta manera, se obtiene un conjunto de datos descriptos semánticamente que facilita la búsqueda de información.

Además, se presentó un método práctico para la población y persistencia de dicho modelo semántico basado en el lenguaje D2RQ y el Triple Store Jena TDB. La descripción de dicho método se realizó utilizando un caso real de publicación de datos del personal del Gobierno de la Provincia de Santa Fe.

Como trabajo futuro se propone el desarrollo de una aplicación que de soporte al concepto de datos enlazados sobre la información almacenada en el modelo semántico propuesto tanto para clientes HTML como RDF.

5. Referencias

Brys, C. y Aldana Montes, J. (2011) “Gobierno electrónico 3.0. Aplicaciones de la Web Semántica a la Administración Pública”. En: Anales del SIE 2011, Quinto Simposio de Informática en el Estado. Córdoba, Argentina, pp. 15-30.

Calderón, C. y Lorenzo, S. (Eds.) (2010) “Open government: Gobierno abierto”. Algón Editores, Alcalá la Real, España.

Heath, T. y Bizer, C. (2011) “Linked Data: Evolving the Web into a Global Data Space”. Synthesis Lectures on the Semantic Web. Morgan & Claypool Publishers.

Lozano, A. (2013) “Enfoque basado en ontologías para brindar acceso a la información del Personal del Gobierno de la Provincia de Santa Fe”. Tesis de Maestría, Universidad Tecnológica Nacional, Facultad Regional Santa Fe, Santa Fe, Argentina.

Póveda-Villalón, M. (2012) “A Reuse-based Lightweight Method for Developing Linked Data Ontologies and Vocabularies”. In: 9ª Extended Semantic Web Conference (ESWC2012), 27 - 31 May, 2012, Heraklion, Grecia.

Villazón-Terrazas, B., Vilches-Blázquez, L., Corcho, O. and Gómez-Pérez, A. (2011) “Methodological Guidelines for Publishing Government Linked Data”, Springer.