# Método Práctico para la Población y Persistencia de un Modelo Semántico

José A. Gilliard[1], Omar R. Perín[1], Mariela Rico[1], and Ma. Laura Caliusco[1,2]

[1] Centro de Investigación y Desarrollo de Ingeniería en Sistemas de Información (CIDISI) - UTN - Fac. Regional Santa Fe, Lavaise 610 - S3004EWB - Santa Fe
jgilliard@frsf.utn.edu.ar
[2] Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)

**Resumen** 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 fácilmente replicadas. En este trabajo se presenta una estrategia para el poblado y persistencia de datos basados en un modelo semántico utilizando el lenguaje de mapeo D2RQ y el Triple Store Jena TDB.

**Keywords:** modelo semántico, datos abiertos, D2RQ, Jena TDB

## 1. Introducción

Hoy en día, la publicación de datos ha tomado importancia, siendo ésta, junto con la búsqueda de información, los objetivos principales de Internet. Si se relacionan los datos y además se les agrega interpretación, entonces se puede hablar no sólo de la publicación de información sino también de conocimiento [9].

Por otro lado, ha surgido un nuevo modelo de gobierno llamado Gobierno Abierto, el cual se basa en la premisa de que todos los datos que produce el gobierno son públicos [1]. 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 hace de forma abierta y transparente [2]. 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 estaba disponible, otras, como el caso de Gran Bretaña y España, realizaron un tratamiento previo de los datos 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 [5]:

- Utilizar URIs (*Uniform Resource Identifier*) como nombres únicos para los recursos.

- Incluir enlaces a otras URIs para localizar más Datos Enlazados.
- 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[3] y utilizar SPARQL[4], un lenguaje de consulta para RDF, como lenguaje de consulta de dichos datos.

En este último sentido existen diferentes arquitecturas y metodologías basadas en modelos semánticos compuestos por ontologías. Por ejemplo, en [8] se presenta un proceso para publicar datos enlazados de gobierno, pero cuando se quiere llevar a la práctica dicho proceso se carece de instrucciones detalladas sobre cómo hacerlo. El trabajo presentado en [7] se centra principalmente en el desarrollo de la ontología en un contexto de datos enlazados y en la evaluación del producto obtenido según criterios de la Ingeniería Ontológica y los principios de Datos Enlazados, pero no se explicitan reglas claras para el poblado de esta ontología. En [5] se presenta un framework de alto nivel para la publicación de datos basados en los principios de Datos Enlazados pero no se describe cómo aplicarlo. Estos inconvenientes hacen difícil replicar estas propuestas en otros casos, sobre todo cuando se deben manejar grandes volúmenes de datos provenientes de distintas fuentes de información y no existe una correspondencia directa entre los elementos de estas fuentes y los de las ontologías.

El objetivo de este trabajo es presentar una estrategia que permita la persistencia y población de grandes ontologías para dar soporte a aplicaciones de Datos Abiertos siguiendo los principios de Datos Enlazados. Dicha estrategia se definió luego de realizar varias iteraciones, siguiendo los lineamientos del método Investigación en Acción [10].

## 2. Pasos para Poblar y Persistir un Modelo Semántico

En esta sección se presentan los pasos de una estrategia para la población y persistencia de un modelo semántico que será utilizado para la publicación de datos de gobierno. El objetivo de dicha estrategia es generar las tripletas necesarias para poblar las ontologías del modelo semántico a partir de los datos almacenados en una base de datos (BD), cumpliendo con los requerimientos especificados en el Documento de Especificación de Requerimientos del Modelo Semántico (DERMS) que se desea poblar y las correspondencias entre los datos en las tablas de las BD y los elementos del modelo. Dichas correspondencias se encuentran especificadas en el Documento de Correspondencias entre la BD y las Ontologías (DCBDO) que se debe recibir como entrada.

Se considera como caso de estudio, para presentar la estrategia, la población y persistencia del modelo semántico del personal del Gobierno de la Provincia de Santa Fe [6]. Se recibieron como entrada el DERMS, el DCBDO, el modelo semántico implementado en el lenguaje OWL y la BD de la cual se debían extraer los datos. El modelo semántico recibido como entrada está compuesto por cuatro ontologías modulares: Lugares, Cargos, Organismos y Agentes.

[3] http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/
[4] http://www.w3.org/TR/2013/REC-sparql11-query-20130321/

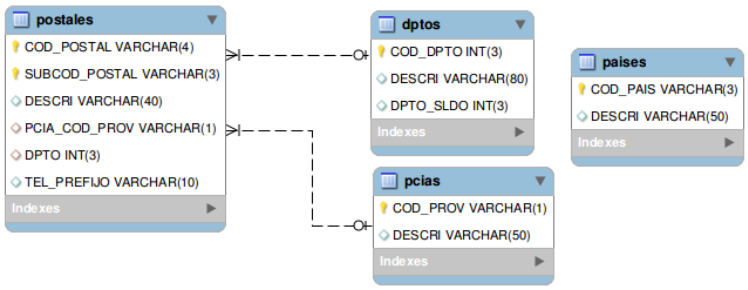


**Figura 1.** Diagrama entidad-relación simplificado para poblar la ontología *Lugares*

### 2.1. Análisis de las Fuentes de Información

Los objetivos de este paso son: (1) preparar las BDs origen para la extracción de los datos para generar las tripletas necesarias para poblar el modelo, y (2) analizar el modelo semántico para comprenderlo y definir las URLs a usar.

**Analizar las fuentes de información.** La información primaria se encuentra en una BD ORACLE de gran porte. Para independizar esta fuente de la publicación de datos, se hizo una réplica de la información a publicar en una BD de código abierto, MySQL, denominada *"personto"* (dicha BD cuenta con 47 tablas y un total de 7.439.605 registros) y se creó un archivo de texto plano conteniendo las sentencias SQL necesarias para la creación de las tablas y posterior carga de los datos. Para disponer de una BD operativa fue necesario montar un servidor de BD en los equipos de desarrollo, y configurar y poner en marcha el servicio.

Dado que no todas las tablas de la BD se usaron para la generación de tripletas, resultó de gran utilidad hacer diagramas de entidad-relación simplificados para cada una de las ontologías del modelo en los que sólo se desplegaron aquellas tablas que intervendrían en su población. Esta fragmentación surgió de los requerimientos definidos en el DERMS. A manera de ejemplo, se muestra en la Figura 1 el diagrama de tablas de Lugares que luego se utilizará en los ejemplos.

**Preparar los datos para generar las tripletas.** Para poder generar las tripletas fue necesario realizar las siguientes tareas sobre los datos de la BD:

*Tarea 1: Anonimizar datos.* Con el fin de no hacer referencia a personas reales, se reemplazaron sus nombres y apellidos por datos ficticios para lo cual se creó una rutina en lenguaje Java y se la ejecutó sobre la BD *"personto"*.

*Tarea 2: Tratar valores especiales.* Un aspecto importante a tener en cuenta es la existencia de valores nulos (NULL) en la BD. En algunos casos un valor nulo en un campo de una tabla significa ausencia o desconocimiento del dato. Por ejemplo, el campo que almacena el número de teléfono de una persona.

En otros casos, al momento de elaborar el modelo semántico se le dió un significado específico a ese valor nulo. Por ejemplo, el modelo semántico estudiado define la propiedad *funcionamiento* para la clase *OrganismoEnSARH*, que se origina a partir del campo *CLAUSURA* de la tabla *organismos*. Los valores almacenados en este campo son los caracteres *S* y *T*, que se traducen a los valores *En Funcionamiento* y *Cerrado Temporalmente* de la propiedad para las instancias de la clase. El modelo semántico, además de esta interpretación de los datos, agrega un tercer posible valor *Cerrado* que deberá utilizarse para los casos en que el valor del campo *CLAUSURA* sea *NULL*. Las herramientas adoptadas no dan soporte para este tipo de transformación con valores nulos, por lo que fue necesario modificar una serie de registros en la BD para incluir un nuevo valor al campo *CLAUSURA* en aquellos registros donde era nulo.

**Definir una URL base para el modelo semántico y, en base a ésta, una URL para cada ontología del modelo.** Para publicar datos en la Web, los elementos de un dominio de interés primero deben ser identificados[5]. *Datos Enlazados* utiliza sólo URIs HTTP por dos razones:

1. proporcionan una forma sencilla de crear nombres únicos globales de manera descentralizada
2. y sirven como medio de acceso a la información que describe la entidad identificada.

La URL *http://www.frsf.utn.edu.ar/personto/* identifica globalmente al modelo semántico estudiado. Las demás URL definidas son las siguientes:

- Agentes: *http://www.frsf.utn.edu.ar/personto/ontoagentes#*
- Cargos: *http://www.frsf.utn.edu.ar/personto/ontocargos#*
- Lugares: *http://www.frsf.utn.edu.ar/personto/ontolugares#*
- Organismos *http://www.frsf.utn.edu.ar/personto/ontoorganismos#*

**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 utilizó 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 agruparon dentro de la colección provincias.

Para generar los identificadores se usaron los valores correspondientes a las claves primarias de las tablas desde las cuales se extraen los datos, asegurando así su unicidad. En los casos donde la clave primaria estaba compuesta por más de un campo, se concatenaron sus valores.

La estructura de la URI correspondiente a cualquier recurso del modelo es: <http://www.frsf.utn.edu.ar/personto/colección/id_recurso>

Algunos ejemplos de las correspondencias entre cada clase del modelo y la URI de la colección en la cual se agrupan sus instancias son:

- Pais: http://www.frsf.utn.edu.ar/personto/paises/
- Departamento: http://www.frsf.utn.edu.ar/personto/departamentos/
- Provincia: http://www.frsf.utn.edu.ar/personto/provincias/

### 2.2. Población del Modelo Semántico

Cada elemento de la ontología tendrá una correspondencia, directa o no, con elementos de las BDs. A partir de las BDs y en base a las definiciones establecidas en las ontologías, primero se deben establecer las correspondencias que permitan generar las tripletas necesarias para representar las definiciones de clases, instancias y sus propiedades, y las relaciones entre las instancias de esas clases. Estas tripletas convierten la **ontología modelada** en una **ontología poblada**.

**Para cada ontología, identificar grupos de tripletas a generar.** Esta actividad consiste en identificar qué grupos de tripletas se deben generar según los elementos que componen la ontología dada. Para cada ontología, se necesitan tripletas para la declaración e instanciación de cada una de sus elementos.

**Definir los patrones que seguirán las componentes sujeto, propiedad y objeto de las tripletas de cada grupo.** Partiendo de los grupos establecidos en la actividad anterior se procedió a especificar de qué manera debían estar compuestas las tripletas para cada grupo.

*Declaración de clases.* Para la declaración de clases se debe generar una tripleta que identifique a la clase como tal y dos tripletas opcionales para agregar comentarios y etiquetas, siguiendo los patrones que se muestran a continuación:

<URI_CLASE>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.w3.org/2000/01/rdf-schema#Class>.
<URI_CLASE>
<http://www.w3.org/2000/01/rdf-schema#label>“ETIQUETA” .
<URI_PROPIEDAD>
<http://www.w3.org/2000/01/rdf-schema#comment>“COMENTARIO” .

Cuando existe herencia de clases, para cada una de las subclases se debe incluir una tripleta extra indicando la relación, como se muestra a continuacin:

<URI_SUBCLASE>
<http://www.w3.org/2000/01/rdf-schema#subClassOf>
<URI_SUPERCLASE>.

*Instanciación de clases* Por cada instancia de clase se debe generar una tripleta según el siguiente patrón:

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

*Declaración de propiedades de instancias.* Las propiedades se declaran con una tripleta que identifique la propiedad como tal y, opcionalmente, dos o más para agregar comentarios y etiquetas, con los patrones que se muestran a continuación.

<URI_PROPIEDAD><http://www.w3.org/1999/02/22-rdf-syntax-ns#type><http://www.w3.org/1999/02/22-rdf-syntax-ns#Property>.
<URI_PROPIEDAD><http://www.w3.org/2000/01/rdf-schema#label>“ETIQUETA” .
<URI_PROPIEDAD><http://www.w3.org/2000/01/rdf-schema#comment>“COMENTARIO” .

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, las tripletas de etiqueta y comentario para cada clase incluida en el dominio.

*Instanciación de propiedades de instancias.* Se debe generar una tripleta por cada propiedad de cada instancia de cada clase siguiendo el siguiente patrón:

<URI_INSTANCIA><URI_PROPIEDAD>“VALOR_PROPIEDAD” .

*Declaración de relaciones entre instancias.* Se debe generar una tripleta que identifique cada relación como tal y, opcionalmente, dos o más para agregar comentarios y etiquetas. Los patrones para estas tripletas son:

<URI_RELACION><http://www.w3.org/1999/02/22-rdf-syntax-ns#type><http://www.w3.org/1999/02/22-rdf-syntax-ns#Property>.
<URI_RELACION><http://www.w3.org/2000/01/rdf-schema#label>“ETIQUETA” .
<URI_RELACION><http://www.w3.org/2000/01/rdf-schema#comment>“COMENTARIO” .

*Instanciación de relaciones entre instancias.* Se debe generar una tripleta por cada par de instancias relacionadas según el siguiente patrón:

<URI_INSTANCIA_SUJETO><URI_RELACION><URI_INSTANCIA_OBJETO>.

**Calcular la cantidad de tripletas a generar para cada grupo en base a consultas SQL a las BD de origen.** Se debe obtener, para cada grupo de tripletas y teniendo en cuenta los patrones necesarios para cada grupo, la cantidad exacta de tripletas que se deben generar para la declaración de todas las clases y propiedades del modelo semántico y la extracción de todas las instancias almacenadas en las BDs. Para estos cálculos se debe tener en cuenta lo siguiente:

- cada subclase en una relación de herencia requiere una tripleta adicional;
- la instanciación de una propiedad de una clase depende de la cantidad de valores no nulos en el campo origen de dicha propiedad;
- las propiedades con $n$ clases en su dominio ($n \geq 2$) requieren $3 + 2 \times (n - 1)$ tripletas

**Para cada ontología, redactar un archivo de mapeo D2RQ que permita generar los lotes correspondientes de forma automatizada.** Mediante el uso de un lenguaje específico, *D2RQ Mapping Language*, se pueden escribir bloques de código que establecen la asociación entre los campos de las tablas en las BDs y los elementos definidos en las ontologías, como clases y propiedades [4]. El

contenido base de un archivo de mapeo D2RQ consiste en la definición de prefijos, conexiones a BDs, bloques *d2rq:ClassMap*, uno por cada clase de la ontología, y bloques *d2rq:PropertyBridge* para generar propiedades entre las instancias de las clases. A continuación se detallan cada uno de ellos.

*Declaración de prefijos.* Los archivos de mapeo comienzan con la declaración de los prefijos necesarios, incluyendo uno para la URI base del modelo, uno para cada URI de las diferentes ontologías que lo forman y uno para cada colección de instancias. También, se deben incluir una serie de prefijos de espacios de nombres de uso común, como el de D2RQ y OWL. No es necesario incluir todos los prefijos específicos del modelo, sino sólo aquellos que se utilizan localmente.

*Conexión a BDs.* Se deben establecer las conexiones a las BDs con las que se va a trabajar para la extracción de tripletas. 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 el acceso a múltiples BDs diferentes. Una vez declarado este objeto, cualquier acceso a la BD se realizará haciendo referencia al mismo.

*Mapeo de clases y sus instancias.* D2RQ abarca las definiciones de clases y sus instancias con un solo bloque de código, a través de un objeto *d2rq:ClassMap*. Por ejemplo, el bloque de código siguiente realiza el mapeo entre la tabla *paises* y la clase *Pais* de la ontología *Lugares*, mediante la definición del mapeo *map:Paises*.

```
map:Paises a d2rq:ClassMap; d2rq:dataStorage map:persontoDB; d2rq:uriPattern
“paises/@@paises.COD_PAIS—encode@@”; d2rq:class ontolugares:Pais; d2rq:classDefinitionLabel
“Pais”; d2rq:classDefinitionComment “Representa el país de nacimiento de un Agente.”; .
```

Este mapeo da origen a las tripletas de definición de la clase *Pais*, una etiqueta (*d2rq:classDefinitionLabel*) y comentario (*d2rq:classDefinitionComment*) para la misma. La propiedad *d2rq:dataStorage* especifica que el objeto obtendrá datos de la BD definida anteriormente por el objeto *map:persontoDB*.

Cada registro de la tabla dará origen a una instancia de la clase, 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. Esto se logra haciendo uso de la propiedad *d2rq:uriPattern*. En tiempo de ejecución, a toda URI generada a partir de un mapeo D2RQ se le añade un prefijo global correspondiente a la URL base que identifica al modelo.

*Mapeo de propiedades y sus instancias.* El objeto D2RQ, que hace posible el mapeo de Data Properties y Object Properties es *d2rq:PropertyBridge*.

Una Data Property 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. La definición de un *d2rq:PropertyBridge* está compuesta por la especificación del *d2rq:ClassMap*, que da origen a las instancias a las cuales agrega propiedades mediante la propiedad *d2rq:belongsToClassMap*,

la propiedad RDF que está generando, especificada por la propiedad *d2rq:property*, y el valor asignado.

Las Object Properties se extraen generalmente de alguna relación entre tablas y se mapean generando un bloque de tipo *d2rq: PropertyBridge* con el agregado de una o más propiedades *d2rq:join*, que fijen la relación de igualdad entre campos de tabla. Además, teniendo en cuenta la definición de una object property como una tripleta sujeto-propiedad-objeto, la propiedad *d2rq:belongsToClassMap* indica el mapeo que da origen a las instancias sujeto, mientras que *d2rq:refersTo ClassMap* indica las instancias objeto. Los signos < y > se utilizan para indicar, mediante la punta de flecha, cuál de los lados de la condición de igualdad corresponde a una clave primaria. Esto permite al motor D2RQ optimizar el mecanismo interno de resolución de consultas SQL, reduciendo los tiempos de procesamiento requeridos para la transformación de los datos.

En ciertos casos, el valor que de una propiedad puede estar incluido en una enumeración de posibles valores, por lo cual el rango de la propiedad corresponde a un tipo de datos enumerado. Pueden existir, además, casos en los que los valores almacenados en campos de tablas que dan origen a propiedades de instancias contengan valores que por diseño del modelo semántico deban ser traducidos o convertidos a otro formato al momento de la generación de las tripletas. Para estos casos se emplean características particulares del lenguaje D2RQ.

**Para cada ontología, generar un volcado de tripletas serializadas en formato n-triples.** Una vez redactado el archivo de mapeo para cada ontología, se generan volcados de tripletas para cada mapeo. La herramienta *dump-rdf* de la plataforma D2RQ permite generar estos volcados de tripletas serializadas en formato *n-triples*. Se utiliza este formato por ser el más adecuado para persistir, en el proceso siguiente, los lotes generados en un T-Store, ya que su procesamiento requiere menores niveles de recursos de CPU y memoria.

**Verificar si se generaron las cantidades de tripletas correctas.** Las herramientas de línea de comandos *grep* y *wc*[5] permiten contar la cantidad de líneas en un archivo que contengan un patrón de texto especificado. Dado que bajo el formato de serialización n-triples cada línea del archivo de volcado corresponde a una tripleta, si se define correctamente el patrón de texto a buscar mediante el uso de expresiones regulares, se puede saber con exactitud la cantidad de ocurrencias de un modelo de tripleta que responden a cada uno de los grupos particulares que existen en el volcado.

### 2.3. Persistencia del Modelo Semántico

El objetivo es generar un repositorio persistente en el que se encuentre almacenado el modelo semántico junto con las instancias generadas anteriormente, listo para ser consultado. Este proceso consta de las tres actividades siguientes.

[5] http://manpages.ubuntu.com/

**Crear un Triple Store.** Para crear la Triple Store se seleccionó la combinación de las herramientas D2RQ y el Framework Jena. Para la selección de dichas herramientas se buscó alcanzar el mayor grado de interoperabilidad entre ellas, teniéndose en cuenta, además, la disponibilidad de documentación y madurez de los proyectos de desarrollo correspondientes. De los dos repositorios ofrecidos por Jena, SDB y TDB[6], se eligió el segundo, debido a que SDB ya no se encuentra en desarrollo activo, y además presenta mayor eficiencia y escalabilidad.

**Cargar al Store cada una de las ontologías del modelo semántico.** La gestión del repositorio y el acceso al mismo pueden realizarse programáticamente mediante una serie de métodos ofrecidos por la API del framework Jena, o mediante el uso de herramientas de línea de comando provistas por la plataforma. Para la carga masiva de datos, la segunda alternativa resulta más eficiente.

La herramienta de línea de comandos *tdbloader* permite la carga masiva de archivos RDF al almacén Jena TDB, y genera con cada carga los índices necesarios para optimizar el acceso a los datos. Se deben tener en cuenta dos restricciones de esta herramienta. En primer lugar, las ontologías a cargar deben encontrarse serializadas en formato RDF/XML. La segunda restricción es que *tdbloader* no es transaccional, lo que implica el bloqueo del repositorio durante las operaciones de carga, denegándose el acceso simultáneo desde otro proceso.

**Cargar al Store cada uno de los lotes de tripletas generados.** El proceso de carga se lleva a cabo con la herramienta *tdbloader*, la cual primero realiza la carga de tripletas al almacén para luego pasar a la fase de indexado.

### 2.4. Consulta al Modelo Semántico Persistido y Poblado

Para cada una de las preguntas de competencia planteadas en el DERO para el modelo semántico estudiado[6], se realizó una traducción a consultas en lenguaje SPARQL. El texto de cada consulta SPARQL se guardó en un archivo de texto con extensión .sparql, para luego utilizarlo en la ejecución de la consulta.

Finalmente, sobre el repositorio Jena TDB se ejecutó cada una de las consultas SPARQL. Esta tarea se puede llevar a cabo ejecutando la herramienta de línea de comandos *tdb-query* de la plataforma Jena TDB. Otra forma de resolver consultas SPARQL es utilizando el motor de ejecución de consultas ARQ mediante la API provista por el framework Jena. Este método brinda mayor flexibilidad, a costas de un mayor consumo de recursos y tiempos de respuesta.

## 3. Lecciones Aprendidas y Trabajos Futuros

En este trabajo se presentó una estrategia para persistir y poblar grandes ontologías para dar soporte a aplicaciones de Datos Abiertos siguiendo los principios

[6] https://jena.apache.org/documentation/tdb/

de Datos Enlazados. La misma se probó en dos tesis de maestría que requerían la población de ontologías con datos almacenados en sistemas de gobierno. De estas pruebas se pudo concluir que la estrategia es útil cuando la BD origen está compuesta por varias tablas y los mapeos entre los elementos de la BD y los del modelo semántico no son directos. Cuando hay pocas tablas, con muchos registros, y la herramienta de ontology learning permite crear de forma directa el modelo semántico y sus instancias, puede resultar más adecuada una herramienta como RDB2Onto [3].

La estrategia definida requiere contar con desarrolladores con conocimientos en tecnologías semánticas y definir un gran número de tripletas (en el caso presentado se definieron más de 50.000.000 de tripletas). Por ello, se desarrolló un prototipo de aplicación que guía en la implementación de la estrategia definida, el cual están fuera del alcance de este trabajo.

La estrategia propuesta está basada en D2RQ y el T-Store Jena TDB, sin embargo podría extenderse a otros lenguajes y T-stores.

Otro tema para trabajo futuro es la actualización de datos. En los casos estudiados los mismos se modifican una vez al mes siendo necesario realizar los pasos definidos para actualizarlos. En otros casos, el tema de la actualización debe reveerse.

## Referencias

1. Brys, C., Montes, A.: Gobierno Electrónico 3.0. Aplicaciones de la Web Semántica a la Administración Pública. In: Anales del SIE 2011, Simposio de Informática en el Estado, pp. 15–30 (2011)
2. Calderón, C., Lorenzo, S.: Open Government: Gobierno Abierto. Algón Editores, España (2010)
3. Cerbah, F.,: Learning highly structured semantic repositories from relational databases: the RDBToOnto tool. In: Proc. 5th European semantic web conference on The semantic web: research and applications, pp. 777–781 (2008)
4. Cyganiak, R., Bizer, C., Garbers, J., Maresch, O., Becker, C.: The D2RQ mapping language, http://d2rq.org/d2rq-language (2012)
5. Heath, T., Bizer, C.: Linked Data: Evolving the Web into a Global Data Space. Morgan & Claypool, 1st ed. (2011)
6. Lozano, A.: 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, Argentina (2013)
7. Póveda-Villalón, M.: A Reuse-based Lightweight Method for Developing Linked Data Ontologies and Vocabularies. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) The Semantic Web: Research and Applications. LNCS, vol. 7295, pp. 833–8379. Springer, Heidelberg (2012)
8. Villazón-Terrazas, B., Vilches-Blázquez, L., Corcho, O., Gómez-Pérez, A.: Methodological Guidelines for Publishing Government Linked Data. In: Wood, D. (ed.) Linking Government Data, pp. 27–49. Springer New York (2011)
9. Zins, C.: Conceptual Approaches for Defining Data, Information, and Knowledge. J. Am. Soc. Inf. Sci. Technol. 58, 479–493 (2007)
10. Avison, D., Lau, F., Myers, M., Nielsen, P.: Action research. Commun. ACM 42, 94–97 (1999)