UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 3

Grupo de Investigación de Web Semántica

Método para la actualización incremental de grafos RDF

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autor:
Katerín Martínez Rojas

Tutor:

Ing. Yusniel Hidalgo Delgado

La Habana, junio de 2014

"Año 56 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro ser la autora de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____	_____
Katerín Martínez Rojas	Ing. Yusniel Hidalgo Delgado
Autor	Tutor

DATOS DE CONTACTO

Síntesis del Tutor

Yusniel Hidalgo Delgado es graduado con título de oro en el 2010 de la carrera de Ingeniería en Ciencias Informáticas por la Universidad de las Ciencias Informáticas (UCI) en La Habana, Cuba. Actualmente investiga sobre nuevos métodos para la publicación y consumo de datos enlazados en varios dominios de aplicación. Es profesor instructor del departamento de Técnicas de Programación de dicha facultad. Es afiliado de la Sociedad Cubana de Matemática y Computación y de la Asociación Cubana de Reconocimiento de Patrones. Revisor de artículos de la Serie Científica UCI y de la Revista Cubana de Ingeniería de la CUJAE.

A Lidia, la madre ejemplar, la amiga, la confidente.
A César, por su sabiduría y ejemplo.
A mi familia toda, por su amor.
Al grupo de Web Semántica.
A mis amigos.
A Cuba.

AGRADECIMIENTOS

Gracias a Lidia, mi mami, por la formación, el amor, la guía, la amistad, la comprensión, por todo.

Gracias a mi papá César, por el amor y el ejemplo sin importar distancias.

Gracias a mi tía Isa, mi segunda madre, mi guía desde el inicio.

Gracias a mi abuela Rosita, mi tío Tutu, mis primas, mis sobrinos, por la educación y el derroche de amor.

Gracias a mi familia toda, por todo el apoyo y el cariño.

Gracias a Ernesto, por su amor, por siempre estar ahí, en las buenas y en las malas.

Gracias a mis amigos de la vocacional, irremplazables, por enseñarme el verdadero valor de la amistad, esa que perdura aún en la distancia.

Gracias a mis amigos de la Universidad y a mis compañeros de aula, por sus lecciones y comprensión, en especial a Pupo, Rey, Dayana, Kety, Alieski por su cariño y aliento en todo momento.

Gracias a mi tutor, por sus conocimientos y por adentrarme en el camino de la investigación.

Gracias a los profesores que contribuyeron a mi formación.

Gracias a la FEU y a todos los que implica, por formarme integralmente, por los valores inculcados.

Gracias al Grupo de Web Semántica, por sus recomendaciones.

A todos, muchas gracias.

RESUMEN

El modelo de datos relacional constituye el paradigma frecuentemente utilizado por los sistemas operacionales que utilizan las empresas para gestionar sus procesos. En los últimos años han logrado un importante impulso las tecnologías de la Web Semántica. Se han desarrollado estándares como *Resource Description Framework* (RDF), *Web Ontology Language* y SPARQL, acrónimo recursivo del inglés *SPARQL Protocol and RDF Query Language*. RDF constituye el modelo de datos utilizado para la publicación y enlazado de datos estructurados en la Web. Estudios recientes demuestran que es posible trasladar el modelo de datos relacional al modelo de datos basado en grafos RDF. Por tal motivo es posible realizar operaciones de inserción, actualización y eliminación en ambos sentidos relacional - grafo y grafo - relacional. Una de las tareas comunes es la actualización de datos en la base de datos relacional, por lo que es necesaria la definición de métodos que detecten estas actualizaciones (cambios) y luego las trasladen al modelo de datos RDF. Las aproximaciones existentes solo se suscriben a la actualización total de los grafos RDF, es decir, un cambio generado en la base de datos relacional implica generar nuevamente el grafo RDF en su totalidad, incurriendo en altos costos de tiempo y recursos computacionales. En esta investigación se propone un método incremental para la actualización de grafos RDF a partir de datos almacenados en bases de datos relacionales. Está basado en tres fases que siguen un modelo basado en tuberías, donde la salida de una fase constituye la entrada de la próxima.

Palabras claves: Actualización Incremental, Alineación, Modelo Relacional, Grafo RDF, Web Semántica.

ABSTRACT

METHOD FOR INCREMENTAL UPDATING OF RDF GRAPHS

The relational data model is the paradigm frequently used by the operational systems used by companies for managing their processes. In recent years have made major push technologies of the Semantic Web. Researcher and practitioners have been developed standards such as Resource Description Framework (RDF), Web Ontology Language and SPARQL, recursive acronym for SPARQL Protocol and RDF Query Language. RDF is the data model used for publishing and linking structured data on the Web. Recent studies show that it is possible to transform data from relational data model to the data model based on RDF graphs. Therefore, it is possible to perform insert, update and delete operations in both ways, relational - graph and graph - relational. To update data from relational databases is a common task, therefore the definition of methods for detecting these updates (changes) and then transform the RDF data model is required. Recent approaches only perform update tasks in the whole RDF graphs. This mean that a change performed into the relational database involves the generation of whole RDF graph, incurring high costs in time and computational resources. In this research, we propose a method for incremental updating of RDF graphs from data stored in relational databases. It is based on three stages following model based on pipelines, where the output of one stage is the input of the next.

Keywords: *Incremental Updating, Mapping, Semantic Web, RDF Graph, Relational Model*

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

INTRODUCCIÓN

Con más de un billón de páginas y miles de millones de usuarios, la Web es uno de los artefactos de ingeniería más exitosos creado por los científicos (1). La Web actual (Web 2.0) es un concepto relacionado con las aplicaciones web que facilitan la interacción entre los usuarios. Se basa en sitios web dinámicos permitiendo que los internautas compartan y generen contenidos e información. Sin embargo, la Web actual posee un conjunto de limitaciones que se basan fundamentalmente en el formato, la integración y la recuperación de la información (2).

Al estar la mayoría de los contenidos de la Web en formato *Hypertext Markup Language* (HTML), la información puede ser consumida por los humanos, no siendo así para las máquinas, imposibilitando la extracción automática de su contenido semántico. En la Web actual los datos se encuentran dispersos, no existiendo relación explícita entre ellos, dificultando su descubrimiento y utilización por sistemas informáticos. Esta limitación hace referencia a la integración de los datos. Junto a las dos limitaciones anteriores, formato e integración, se ve afectada la recuperación de la información. Los resultados ofrecidos por motores de búsqueda resultan imprecisos al no ser capaces de recuperar información a partir de consultas expresadas en lenguaje natural, no satisfaciendo las necesidades de búsqueda de los usuarios (2).

En la Web actual la información es almacenada en bases de datos. El modelo de datos relacional constituye un paradigma frecuentemente utilizado por los sistemas operacionales que actualmente utilizan las empresas para gestionar sus procesos (3). En este modelo de datos, los objetos abstractos y del mundo real se convierten en relaciones (entidades) que luego son persistidas en una Base de Datos Relacional (BDR).

La Web Semántica ha surgido como una solución a las limitaciones de la Web actual. El promotor del concepto es Timothy Berners-Lee quien manifiesta la intención de dotar a la Web actual de información con un significado bien definido, enunciando: *"La Web Semántica no pretende sustituir la Web actual, sino que es una extensión de la misma en la que la información tiene un significado bien definido, posibilitando a los humanos y las computadoras trabajar en cooperación"* (4). La Web Semántica, como parte de las tecnologías que componen la Web 3.0, ha definido nuevos conceptos y herramientas para el trabajo con la misma.

El concepto de anotaciones semánticas manifiesta la intención de brindar una descripción semántica a los programas y recursos computacionales. Constituyen un proceso de adición, a los

recursos de la Web, de metadatos semánticos asociados con ontologías. Las anotaciones semánticas son consideradas la esencia de las tecnologías de la Web Semántica (1). Con estas es solucionada la limitación del formato de la Web actual. Sin embargo, la integración y la recuperación son limitaciones que aún se consideran vigentes.

La Web Semántica es considerada una de las áreas de investigación activa en la comunidad científica mundial. El cambio de modelo para la manipulación de la información es uno de los retos actuales. La base del modelo empleado por la Web Semántica son los datos enlazados (5) los que se refieren, según (6) a un *"conjunto de buenas prácticas para la publicación y enlazado de datos estructurados en la Web"*. A partir de este momento comienza a implementarse la concepción de publicar datos estructurados siguiendo los principios de los datos enlazados. El marco de descripción de recursos, por sus siglas en inglés RDF, el lenguaje de consultas SPARQL y las ontologías, son tecnologías que se destacan en el ámbito de la Web Semántica.

RDF constituye el modelo de datos utilizado para la publicación y enlazado de datos estructurados en la Web. Este modelo de datos está basado en un grafo dirigido, compuesto por tripletas del tipo sujeto – predicado – objeto, siendo el sujeto y el objeto los nodos del grafo y el predicado el arco que une a dichos nodos (7).

Estudios recientes demuestran que es posible trasladar el modelo de datos relacional al modelo de datos basado en grafos RDF. El grupo de trabajo *RDB2RDF*[1] perteneciente al *World Wide Web Consortium* (W3C) propone lenguajes para dicha alineación. En (3) se realiza una comparación entre los lenguajes de alineación existentes, clasificándolos de acuerdo a su propósito y características.

Por tal motivo es posible realizar operaciones comunes en ambos sentidos relacional-grafo y grafo-relacional. Una de las tareas más comunes consiste en la actualización de datos de la base de datos relacional, siendo necesaria la definición de métodos que detecten estas actualizaciones (cambios) y las trasladen al modelo de datos RDF.

Aproximaciones existentes solo se suscriben a la actualización total de los grafos RDF, es decir, un cambio generado en la base de datos relacional implica que sea necesario generar nuevamente el grafo RDF en su totalidad. Por tanto esto implica altos costos de tiempo y recursos computacionales.

[1] http://www.w3.org/TR/2012/NOTE-rdb2rdf-implementations-20120814/

Los grupos de investigación de Web Semántica de la Universidad de las Ciencias Informáticas (UCI) de Cuba y el grupo KHAOS[2] de la Universidad de Málaga de España están desarrollando en colaboración el proyecto DBJournal. El grupo KHAOS trabaja en la intersección entre las tecnologías de Bases de Datos y de la Web Semántica. Su principal objetivo es no sólo colaborar en el desarrollo de estas tecnologías y no sólo usarlas en entornos realistas sino aplicarlas en problemas reales. El proyecto DBJournal tiene como objetivo la publicación de metadatos bibliográficos siguiendo los principios de los datos enlazados. En el contexto de este proyecto se ha desarrollado la plataforma BM2LOD[3] (8). Esta plataforma integra fuentes de datos de revistas académicas que emplean el protocolo de intercambio de metadatos, *(OAI-PMH[4])* de inglés *Open Archive Initiative Protocol for Metadata Harvesting*. Dichos metadatos se almacenan en una BDR usando la herramienta Metharto (9). Dicha herramienta se encarga de recolectar metadatos bibliográficos provenientes de revistas científicas que utilizan el protocolo OAI-PMH. La BDR se alinea con las ontologías y vocabularios para posteriormente generar las tripletas RDF. La limitante fundamental de la plataforma BM2LOD se basa en que requiere realizar la actualización del grafo RDF, generándolo en su totalidad una vez que se detectan cambios en la BDR (9). Generar un grafo que contenga alrededor de 50000 tripletas puede consumir 15 segundos aproximadamente al generarse, un grafo de 1 millón de tripletas puede consumir 300 segundos y así aumentando según la cantidad de tripletas, un cambio en la BDR implicaría un consumo de tiempo superior al generar nuevamente el grafo RDF en su totalidad.

A partir de la situación descrita anteriormente, se plantea el siguiente **problema a resolver:** ¿Cómo actualizar grafos RDF a partir de bases de datos relacionales de manera que se reduzcan los costos de tiempo y recursos computacionales en la plataforma BM2LOD?

El **objeto de estudio** donde se enmarca la investigación está constituido por el modelo de datos RDF dentro del **campo de acción** de la actualización incremental en el modelo de datos RDF.

Para resolver el problema se identifica el siguiente **objetivo general:**

Diseñar un método y su correspondiente implementación para la actualización incremental de grafos RDF a partir de bases de datos relacionales en el contexto del proyecto DBJournal, de forma tal que se reduzcan los costos de tiempo y recursos computacionales.

[2] http://khaos.uma.es/
[3] *Bibliographic Metadata to Linked Open Data* (Metadatos Bibliográficos a Datos Abiertos Enlazados)
[4] http://www.openarchives.org/pmh/

A partir de lo planteado anteriormente se desglosan los siguientes **objetivos específicos:**

1. Realizar el marco teórico y el estado del arte de la investigación para identificar tendencias y adoptar una posición al respecto.
2. Proponer un método para la actualización de grafos RDF a partir de bases de datos relacionales.
3. Implementar una herramienta que utilice el método propuesto.
4. Validar que el método propuesto y la herramienta implementada resuelven el problema de la actualización incremental de grafos RDF a partir de bases de datos relacionales.

Se tiene como **idea a defender**:

Si se obtiene un método para la actualización incremental de grafos RDF a partir de bases de datos relacionales, entonces se disminuirán los costos de tiempo y recursos computacionales asociados a la actualización de dichos grafos.

De lo anterior se obtiene la **variable independiente:** método para la actualización incremental de grafos RDF y las **variables dependientes:** tiempo y recursos computacionales.

Tabla I: *Operacionalización de la variable independiente*

Variable independiente	Dimensión	Indicadores	Escala
Método para la actualización incremental de grafos RDF		Lógico	Sí o No

Tabla II: *Operacionalización de la variable dependiente*

Variable dependiente	Dimensión	Indicadores	Escala
Costo en tiempo	Tiempo	Si se reduce en: 0-20%: Mal 21%-40%: Regular >41%: Bien	Segundos (s)
		Si se reduce en: 0-5%: Mal 5%-10%: Regular	Porciento (%)

Costo en recurso computacionales	Uso de Memoria RAM[5]	>10%: Bien	
	Uso del Microprocesador	Si se reduce en: 0-10%: Mal 11%-20%: Regular >20%: Bien	Porciento (%)

Para el desarrollo de la investigación se parte de que se tiene como **población:** las bases de datos relacionales. Por otra parte la **muestra** seleccionada es: base de datos obtenida de la herramienta Metharto.

Durante la investigación se han empleado un conjunto de **métodos científicos** como procedimientos lógicos que se ha seguido para la obtención y el procesamiento de la información.

Métodos teóricos

El método **Analítico-Sintético** ha permitido realizar un análisis sobre la teoría y las tendencias de los componentes relacionados con las BDR y el modelo de datos RDF, de manera que se hayan podido estudiar a profundidad cada uno de ellos por separado, así como las técnicas o tecnologías involucradas en el proceso de alineación que se realiza entre ambos. Ha permitido además caracterizar cada uno de los componentes analizados previamente.

El método **Inductivo-Deductivo** ha permitido realizar una generalización de los procesos involucrados en la alineación entre BDR y grafos RDF teniendo como base los elementos comunes de las aproximaciones previamente analizadas. Luego, a partir de dichos conocimientos, se ha podido generar una propuesta de solución con elementos más generales a través del razonamiento lógico.

El método **Histórico-Lógico** ha permitido analizar la evolución, de forma cronológica, de los elementos relacionados al modelo RDF, así como la evolución de tecnologías usadas en la alineación entre BDR y grafos RDF, teniendo en cuenta el desarrollo de las aproximaciones realizadas.

Métodos empíricos

[5] *Random Access Memory* (Memoria de Acceso Aleatorio)

El método de la **medición** ha permitido la aplicación de métricas para garantizar la calidad de la propuesta de solución.

La investigación está estructurada en tres capítulos.

Capítulo I: Se definen los principales conceptos, relacionados en el trabajo, que pertenecen al ámbito de la Web Semántica. Se analizan un conjunto de investigaciones relacionadas. Se realiza un estudio de la literatura para identificar los elementos que formarán parte de la propuesta de solución y su impacto social.

Capítulo II: Se define el método para la actualización incremental de grafos RDF, seleccionando las ontologías que forman parte del modelo. Se exponen las características relevantes en cuanto a su implementación.

Capítulo III: Se describe el proceso de validación de la propuesta de solución. Son realizadas pruebas para validar el diseño y la herramienta. Se expone un caso de estudio para validar el método para la actualización incremental de grafos RDF, haciendo un análisis de las variables tiempo y recursos computacionales. Se ilustran ejemplos y resultados finales.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

En este capítulo se hace referencia a los fundamentos básicos del modelo RDF y a los elementos necesarios para la alineación entre BDR y RDF, así como la actualización de grafos RDF, haciendo énfasis en los conceptos, definiciones y características principales. Siendo el modelo RDF base de la Web Semántica, se realiza un análisis de la literatura para identificar los elementos que formarán parte de la propuesta de solución. Se establece una comparación entre las aproximaciones y herramientas utilizadas en los procesos de detección de cambios en la BDR, generación de las tripletas RDF a partir de los cambios identificados y actualización del grafo RDF. Se identifican la metodología, las herramientas, tecnologías y otros elementos utilizados en la confección de la solución.

1.2. Análisis bibliométrico y documental

En la presente investigación el análisis bibliométrico se realiza con el objetivo de mostrar la novedad de la revisión bibliográfica realizada, basándose en las fechas de las publicaciones consultadas. Las bases de datos utilizadas fueron: *IEEE, Google Scholar y Scielo*. Además, se relacionan los tipos de fuentes bibliográficas más citadas. Las fuentes bibliográficas son: artículos de revistas, libros, tesis (específicamente de maestrías o doctorados), artículos en congresos y sitios web. El análisis realizado se muestra en la siguiente tabla.

Tabla III: *Análisis bibliométrico y documental*

Tipo de fuente bibliográfica	Cantidad consultada	Cantidad publicada en los últimos cinco años (2009-2014)
Artículo de revista	15	11
Libro	10	5
Tesis (maestrías o doctorados)	2	1
Artículo en congreso	5	4
Sitio web	14	12
Total:	**46**	**33**

De la tabla anterior se obtiene el resultado mostrado en el **Gráfico I**, donde se muestra que el 79% de la literatura consultada pertenece a los últimos cinco años, evidenciando la actualidad de la bibliografía consultada.

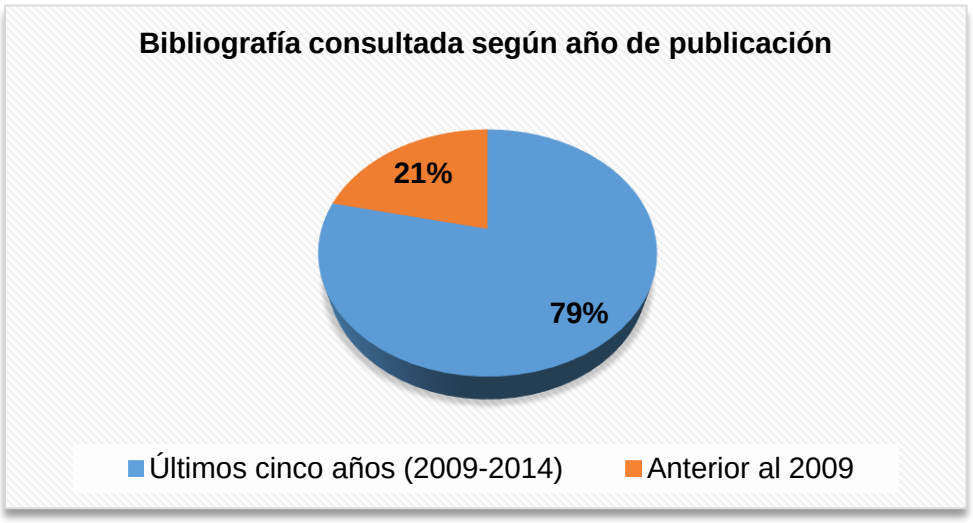

Gráfico II: *Actualidad de la bibliografía consultada*

1.3. Marco teórico. Conceptos y definiciones

En la Web actual la información es almacenada en bases de datos (3). Una **base de datos** es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su uso posterior. El modelo relacional para la gestión de una base de datos se basa en la lógica de predicados y en la teoría de conjuntos. En este modelo los datos son almacenados en correspondencia con sus relaciones, formando una **BDR**. El principio básico del modelo relacional es realizar una reducción de la redundancia de los datos (10).

Para el procesamiento del contenido almacenado en BDR se necesita una tecnología que interprete el significado de la información que en ella se encuentra. Con la Web Semántica la información contiene un significado bien definido (4). Para organizar la Web Semántica es indispensable el uso de ontologías. Varios autores han definido el término ontología para la Ingeniería del Conocimiento en los últimos años.

En 1991 Neches propone una definición de ontología, planteando que “define términos básicos y relaciones para definir extensiones al vocabulario” (11). En 1998 Studer plantea que: “una ontología es una especificación formal y explícita de una conceptualización compartida” (12). Esta última definición se convirtió en la más citada en la literatura y la más usada por la comunidad de

desarrolladores de ontologías (13). La evolución del término ha estado marcada por aportes al concepto dado por Studer según el área marcada en cada investigación. En este trabajo se adopta la definición dada por Studer, refiriéndose con “formal” a la necesidad de que sea comprensible por las computadoras, “explícita” a su descripción en un lenguaje, “conceptualización” a la forma de entender y describir un dominio y “compartida” a ser consensuada por un grupo, compartida por varias partes (14).

Las ontologías se pueden clasificar atendiendo a sus diferencias en cuanto a dependencias y relación con tareas específicas (11). Pueden ser de **Nivel Superior**, describiendo conceptos de forma general. De **Dominio**, siendo reutilizables en un determinado dominio específico, definiendo además, sus relaciones. De **Tareas**, proponiendo un vocabulario sistemático de términos utilizados para resolver problemas asociados a tareas que pueden ser parte o no de un mismo dominio. De **Aplicación**, dependen de la aplicación en particular, pueden extenderse en el vocabulario del dominio y de las ontologías de tareas. En esta investigación se hace uso de las ontologías de dominio enmarcadas en el ámbito de los metadatos bibliográficos.

Para publicar y compartir datos usando ontologías ha surgido el lenguaje de marcado **OWL**, del inglés (*Web Ontology Language*). Tiene como objetivo facilitar un modelo de marcado construido sobre RDF y codificado en *Extensible Markup Language* (XML). Actualmente, OWL tiene tres variantes: *OWL Lite, OWL DL, OWL Full*. Estas variantes incorporan diferentes funcionalidades. *OWL Lite* es útil para la creación de jerarquías, permitiendo solo valores de cardinalidad 0 y 1, perdiendo expresividad. *OWL DL* (*Description Logic*) es recomendado para usuarios que requieran el máximo de expresividad. *OWL Full* posee un nivel máximo de expresión y libertad sintáctica de RDF. Es el más completo, por lo que se necesita de un alto poder computacional para realizar inferencias, siendo esta su limitación. Un estudio realizado en (15), se plantea que *OWL Lite* es más sencillo que *OWL DL*, y *OWL DL* es más sencillo que *OWL Full*.

En la transición de la Web actual a la Web Semántica se debe cumplir con los **principios de los datos enlazados**, de los cuales Timothy Berners-Lee acuñó el término en el 2006. Los **datos enlazados** se refieren a un conjunto de principios o buenas prácticas para la publicación y enlazado de datos estructurados a través de *Uniform Resource Identifier* (URIs) desreferenciables en la Web. Dichos principios se enuncian a continuación:

1. Utilizar el identificador de recurso uniforme, del acrónimo en inglés *Uniform Resource Identifier* (URI), para identificar cada recurso[6] publicado en la Web. (Identificar el recurso).

2. Publicar los datos en una URI basada en *Hypertext Transfer Protocol* (HTTP) con el que puedan ser fácilmente localizados y consultados. (Publicar el recurso)

3. Proporcionar información útil, detallada o extra acerca del recurso cuando se acceda a esta URI basada en HTTP. (Describir el recurso).

4. Incluir enlaces a otras URIs relacionadas con los datos contenidos en el recurso, de manera que se potencie el descubrimiento de información en la Web. (Enlazar el recurso con otros recursos similares).

La aplicación de los principios de los datos enlazados se ha extendido por todo el mundo a través de su utilización en sistemas de todo tipo. Sin embargo, su empleo requiere compatibilidad y estandarización para la representación de los datos. Es por ello que se utilizan estándares establecidos por el W3C[7] como el Marco de Descripción del Recurso (RDF, por sus siglas en inglés). **RDF** es un modelo de datos basado en grafos dirigidos para la publicación y enlazado de datos estructurados en la Web. Con RDF se hace uso de las ontologías para su descripción formal**.** Su sintaxis está basada en tripletas del tipo sujeto-predicado-objeto.

Para la actualización de grafos RDF se hace necesario detectar los datos que han sido modificados en la BDR para luego convertirlos a tripletas RDF. Para generar las mismas se usan lenguajes de alineación. Los lenguajes de alineación son usados para convertir a tripletas RDF u ontologías, la información almacenada en las BDR. Se establecen diferencias entre los mismos en cuanto a la forma de conversión y a la realización del proceso de alineación.

Para la consulta de grafos RDF existe el lenguaje **SPARQL**. Se trata de un lenguaje estandarizado por el *RDF Data Access Working Group* (DAWG) del W3C. Es una tecnología clave en el desarrollo de la Web Semántica que se constituyó como recomendación oficial del W3C en 2008 (16).

Resulta necesario distinguir entre el lenguaje de consulta y el motor para el almacenamiento y recuperación de los datos. Por este motivo, existen múltiples implementaciones de SPARQL, generalmente ligadas a entornos de desarrollo y plataformas tecnológicas (16). En un principio

[6] El término de la arquitectura web “recurso” se refiere al contenido de interés en la Web que se publica mediante URIs HTTP
[7] http://www.w3c.org

SPARQL únicamente incorpora funciones para la recuperación de tripletas RDF. Sin embargo, algunas propuestas también incluyen operaciones para el mantenimiento (creación, modificación y eliminación) de datos como es el caso de *SPARQL Update 1.1*[8] (16).

1.4. Estado del arte

El modelo RDF puede considerarse en parte un modelo relacional, pero a su vez es usado también para otras funciones, se considera que RDF es más general. En el modelo RDF las relaciones son objetos de primera clase (7). Según (3) el modelo de datos de la Web Semántica se conecta directamente con el modelo relacional y viceversa.

La mayoría de los datos en los que se basa la Web actual están almacenados en bases de datos relacionales debido a su probada trayectoria, almacenamiento eficiente, ejecución de consultas optimizadas y fiabilidad (3). En comparación con el modelo de datos relacional, RDF es más expresivo y los datos representados en RDF pueden ser interpretados, procesados y razonados por los agentes de software[9].

En (17) se define un componente que brinda un servicio para la actualización de grafos RDF. Sin embargo, solo se describe su objetivo de llevar a cabo actualizaciones desde la base de datos corporativa de la organización hacia el almacén RDF. Otros procesos como la creación automática de nuevas tripletas debido a nuevos registros en la base de datos corporativa, o la eliminación de tripletas por obsolescencia son solamente tareas de este mecanismo. En esta aproximación la creación de las nuevas tripletas es una tarea que se realiza automáticamente, lo cual dificulta la solución al problema de la actualización si se desea seguir un enfoque incremental.

En (18) se propone un mecanismo de traslación bidireccional relacional–RDF y se describe la herramienta D2RQ++, la cual es una extensión de D2RQ. Anteriormente D2RQ generaba solamente las tripletas RDF y las mostraba. Mediante D2RQ++, es posible actualizar los datos de la base de datos relacional a partir de los cambios que existan en las tripletas RDF y viceversa. Esta herramienta muestra la posibilidad de la traslación tanto del modelo relacional al modelo basado en grafos RDF y de este último al modelo relacional, evidenciando la alineación relacional

[8] http://www.w3.org/TR/sparql11-update/
[9] Programa de computación que actúa para un usuario u otro programa en una relación de entidad.

– grafo, grafo – relacional. Sin embargo, este proceso de actualización no es incremental y además el lenguaje de alineación que usa la herramienta no es estándar del W3C (19).

Otro trabajo interesante es la herramienta Morph-RDB (20), que es un motor RDB2RDF desarrollado por el *Ontology Engineering Group*[10]. Morph-RDB genera un grafo RDF virtual a partir de una BDR de acuerdo con una alineación definida entre los dos modelos que se especifica con el lenguaje R2RML.

Entre los trabajos más relevantes se encuentra *Virtuoso RDF Views*, el cual realiza la alineación del modelo relacional con el modelo RDF, y brinda la posibilidad de que este sea personalizado. *Virtuoso* incluye un metalenguaje declarativo para definir la alineación de datos SQL (del inglés *Structured Query Language*) con ontologías RDF. La alineación es dinámica, lo que implica que los cambios en los datos subyacentes se reflejan inmediatamente en la representación RDF. En esta propuesta se trabaja con vistas de BDR que deben ser creadas dentro del propio Virtuoso (21).

1.4.1. Detección de cambios en las BDR

Los cambios en las BDR son inherentes al desarrollo y no se pueden evitar. Llevar un estricto control de cambios y una buena gestión de la configuración de cada BDR es tarea de primer orden y su incorrecto funcionamiento puede provocar retrasos en el desarrollo o consecuencias que afecten la consistencia de los datos cuando se realiza un cambio en el ambiente real (22). Para la detección de los cambios que se realizan en las BDR, los mecanismos pueden ser basados en el registro de transacciones (*logs)*[11] o en disparadores (*triggers)*[12]. Los sistemas de réplica de bases de datos son ejemplo en el uso de mecanismos para la detección de cambios en bases de datos.

1.4.1.1. Basada en *logs*

Mediante un modelo de recuperación[13] se leen las actualizaciones usando los *logs*, obteniéndose la lista serializada de cambios que utiliza la base de datos para su propia recuperación en caso

[10] http://www.oeg-upm.net/
[11] Registro oficial de eventos durante un rango de tiempo en particular.
[12] Procedimiento que se ejecuta cuando se cumple una condición establecida al realizar una operación.
[13] Propiedad de las bases de datos que controla la forma en que se registran las transacciones.

de un reinicio. Se debe realizar la rotación del *log*, supervisando su tamaño. Algunas operaciones se pueden registrar mínimamente para reducir su impacto sobre el tamaño del *log*.

En cuanto a la replicación basada en *log*, este tiene la sobrecarga de rendimiento más bajo de cualquier método de replicación, se ocupa de un conjunto más amplio de cambios y tiene menor impacto en la gestión. Sin embargo, se considera que es más difícil de implementar ya que requiere la lectura de formatos del *log* que son complejos y raramente documentados por completo (23). El replicador Tungsteno es un ejemplo que utiliza la replicación basada *logs* debido a su rendimiento y flexibilidad. Sin embargo, no todos los lectores de registro están disponibles en código abierto, algunos son extensiones comerciales (23).

Los *logs* son un componente esencial de las bases de datos y, si se produce un error del sistema, podría ser necesario para regresar la base de datos en un estado consistente. Los *logs* nunca se deben eliminar o mover, a menos que se conozcan totalmente las implicaciones de esas acciones (24).

1.4.1.2. Basada en *triggers*

Por otra parte, se conoce que el concepto general de *triggers* surge debido a restricciones de integridad de los datos en las bases de datos (25). Esta opción posibilita que sean creados una serie de *triggers* en la base de datos, que permiten capturar las operaciones inserción, actualización y eliminación realizadas sobre tablas especificadas.

SymmetricDS[14] es un software de replicación de datos. Utiliza tecnologías web para replicar tablas entre BDR, casi en tiempo real. Mediante el uso de *triggers* de base de datos, se garantiza que los cambios en los datos sean detectados conservando la atomicidad. El apoyo a los proveedores de bases de datos se proporciona a través de una capa de base de datos del dialecto, con implementaciones de *MySQL, Oracle, SQL Server, SQL Server Azure, PostgreSQL*, entre otros.

El uso de los *triggers* ha permitido una mejora en la integridad de los datos ya que fuerzan restricciones dinámicas de estos y de integridad referencial. Aseguran que las operaciones relacionadas se realicen juntas de forma implícita. Brindan una respuesta instantánea ante un

[14] SymmetricDS es un software de código abierto para la replicación multi-master de base de datos, la sincronización filtrada o de transformación a través de la red en un entorno heterogéneo.

evento auditado, ofreciendo un mayor control sobre la BDR. La tarea que ha de realizar el *trigger* debe ser definida con antelación, teniendo en cuenta que no han de usarse en tablas temporales.

Cuando son obtenidos los cambios que se realizan en la BDR, resulta necesaria la transformación de esos cambios en tripletas RDF. En este sentido existe el *RDB2RDF Working Group* (WG)[15]. Su misión se define en (26). Se basa en estandarizar los lenguajes para la alineación de esquemas de datos relacionales con RDF y OWL.

1.4.2. Lenguajes de alineación BDR-RDF

Se han estudiado varios lenguajes para la alineación del modelo relacional al modelo de datos basado en grafos RDF atendiendo a sus características y a la forma de realizar el procedimiento.

Estudios recientes demuestran la posibilidad de trasladar el modelo de datos relacional al modelo de datos basado en grafo RDF. El grupo de trabajo RDB2RDF[16] perteneciente al W3C propone lenguajes para establecer una alineación entre el modelo de datos relacional y las ontologías. En (3) se realiza una comparación entre los lenguajes de alineación existentes, clasificándolos de acuerdo a su propósito y características, trabajo que demuestra la posibilidad de realizar operaciones comunes en ambos sentidos tanto: relacional-grafo como grafo-relacional (3). El aporte fundamental de la comparación realizada en (3) es la posibilidad de poseer una guía para elegir un lenguaje de alineación según su propósito, ya sea de forma general o específico.

1.4.2.1. *Direct Mapping*

Direct Mapping hace referencia a una alineación directa entre la BDR y RDF (27). Se establece una correspondencia entre las tablas del modelo relacional con las clases y propiedades de las ontologías y vocabularios, generando luego el grafo RDF. Genera una ontología putativa, la cual no puede ser modificada una vez generada. Las relaciones entre las tablas generan propiedades de objeto. La alineación se hace de forma automática por lo que se pierde personalización al realizar la misma (3).

1.4.2.2. R2RML

R2RML según (28) es el lenguaje de alineación estándar del W3C RDB2RDF WG de BDR – RDF (3). Utiliza ontologías de dominio para modelar los datos existentes en la BDR. Parte del concepto

[15] http://www.w3.org/TR/2012/NOTE-rdb2rdf-implementations-20120814/
[16] http://www.w3.org/TR/2012/NOTE-rdb2rdf-implementations-20120814/

de tablas lógicas, las cuales pueden ser una tabla de la BDR, una vista o consulta SQL[17] válida. Soporta la transformación de datos, cálculo de datos o filtros antes de generar las tripletas. Permite a los usuarios definir manualmente las asignaciones. EL proceso es considerado semiautomático ya que se puede modificar la alineación, adquiriendo esta un nivel de personalización.

1.4.2.3. D2RQ

D2RQ es un lenguaje de alineación y una plataforma para el tratamiento de BDR como grafos RDF virtuales. Su objetivo se basa en exponer BDR en el ámbito de la Web Semántica para proporcionar acceso a través de consultas SPARQL y datos enlazados (29). D2RQ genera vistas en modo de solo lectura por lo que no ha de ser posible realizar modificaciones ni a la BDR ni al grafo RDF.

1.4.2.4. Triplify

Triplify utiliza un enfoque ligero para publicar datos enlazados a partir de BDR. Se basa en realizar peticiones HTTP-URI y consultas a la BDR, para luego alinear las relaciones resultantes en declaraciones RDF (30). Parte de la base de que la mayoría de la información en la Web ya está almacenada en forma estructurada. La alineación mediante Triplify está implementada como guiones PHP[18]. Genera vistas en modo de solo lectura por lo que no ha de ser posible realizar modificaciones ni a la BDR ni al grafo RDF.

Se han estudiado otros lenguajes de alineación que se basan en características semejantes a las de los lenguajes anteriormente descritos, como *eD2R, R2O, Relational OWL, Virtuoso, R3M*, de los cuales se puede obtener más información en (3).

Basado en la comparación que se establece en (3), los lenguajes de alineación pueden ser clasificados de acuerdo a cuatro categorías: alineación directa (*Direct Mapping*), alineación de propósito general de solo lectura (Virtuoso, D2RQ y R2RML), alineación de propósito general de lectura y escritura (R3M) y alineación de propósito especial (eD2R, R2O y Triplify). Virtuoso, Triplify, R2RML y hasta cierto punto D2RQ son lenguajes de alineación que dependen, en gran

[17] Lenguaje de consulta estructurado de acceso a bases de datos.
[18] Lenguaje de programación interpretado.

medida, de SQL para implementar dicha alineación. Además, otras especificaciones sobre *Direct Mapping* y R2RML pueden ser encontradas en (31).

Los dos lenguajes de alineación recomendados por el W3C son *Direct Mapping* y R2RML. Con *Direct Mapping* se realiza una correspondencia directa entre las tablas, realizando la alineación de forma automática. Las tablas y los atributos del modelo relacional son alineados con las clases y las propiedades de las ontologías respectivamente. De las relaciones entre las tablas se generan nuevas propiedades (*ObjectProperty*). Al realizarse la alineación de forma automática se pierde personalización. Por otra parte, con R2RML la alineación se realiza a partir de tablas lógicas (pueden ser tablas de la BDR, vistas o consultas SPARQL válidas). Con R2RML la alineación adquiere personalización al realizarse semiautomática.

Al obtener las tripletas RDF a partir de un lenguaje de alineación, se hace necesario actualizar el grafo RDF.

1.4.3. Actualizar tripletas en el grafo RDF

El lenguaje estandarizado para la consulta de grafos RDF, normalizados por el RDF *Data Access Working Group* (DAWG) del W3C es SPARQL. El mismo no contemplaba realizar modificaciones, pero con SPARQL 1.1 Update[19] es posible realizar ese tipo de consultas. SPARQL 1.1 Update es un lenguaje de actualización para grafos RDF. Utiliza una sintaxis derivada del lenguaje de consulta SPARQL. Es considerado un estándar del W3C desde el 2013.

En (32) se realiza una comparación entre algunos formatos para la actualización de grafos RDF tales como: *Talis Changeset, GUO, GRUF, Sesame RDF Transactions, SPARQL Update.*

Tabla IV: *Comparación de los formatos de actualización de RDF*

Formato	Ventajas	Desventajas
Talis Changeset	Metadatos de modificación	Requiere materializaciones Ineficiente para un gran número de tripletas No hay actualizaciones dinámicas Centrado en un solo recurso

[19] http://www.w3.org/TR/2013/REC-sparql11-update-20130321/

GUO	Ligero Metadatos de modificación	Centrado en un solo recurso Ineficiente para un gran número de tripletas
GRUF	Simplicidad Muy compacto	No existen metadatos de modificación No existen implementaciones en software
Sesame RDF Transactions	Ejecutado como una transacción	No existen metadatos de modificación No estandarizado No documentado
SPARQL Update	**Estándar del W3C** **Permite las operaciones CRUD** **Flexible** **Compacto**	**No existen metadatos de modificación**

De la comparación realizada en (32) se obtuvo como resultado que SPARQL 1.1 Update es un estándar eficiente y flexible para realizar consultas en grafos RDF. También permite todas las operaciones CRUD (acrónimo de Crear, Obtener, Actualizar y Borrar), mientras que la inducción del uso de la red es baja en comparación con otros formatos para la actualización de grafos RDF.

Para lograr la integración de las fases anteriormente descritas se hace necesario el uso de herramientas que permitan generar el grafo RDF y mantenerlo actualizado.

1.5. Metodologías de desarrollo de software

El desarrollo de todo software debe estar guiado por una metodología de desarrollo. De esta depende, en gran medida, que el software tenga la calidad requerida. Existen dos grupos de metodologías: ágiles y tradicionales. No existe una metodología universal para cada tipo de proyecto. Se define una metodología según las características del equipo de desarrollo, el dominio de aplicación, tipo de contrato, complejidad y envergadura del proyecto.

Las metodologías tradicionales centran su atención en llevar una completa documentación, planificación y procesos de todo el proyecto. Los resultados de los procesos no son siempre predecibles, existiendo altos costos al implementar un cambio. Las metodologías ágiles están orientadas a entornos variables. Exigen reducir los tiempos de desarrollo manteniendo una alta

calidad. Su prioridad es satisfacer al cliente mediante tempranas y continuas entregas, dando más valor a la creación del producto de software funcional sin escribir exhaustivamente la documentación. Lo anterior tributa a una elevada simplificación sin renunciar al aseguramiento de la calidad del producto (33). A continuación se presenta una tabla donde se refleja la comparación entre las metodologías con sus diferentes enfoques (tradicional y ágil).

Tabla V: *Comparación entre las metodologías de enfoque tradicional y de enfoque ágil*

Metodología tradicional	Metodología ágil
Más artefactos. El modelado es esencial.	Pocos artefactos. El modelado es prescindible.
Más roles, más específicos.	Pocos roles, más genéricos y flexibles
Existe un contrato prefijado.	No existe un contrato tradicional.
El cliente interactúa con el equipo de desarrollo mediante reuniones.	El cliente es parte del equipo de desarrollo.
Aplicables a proyectos de cualquier tamaño. Suelen ser efectivas en proyectos grandes y con equipos posiblemente dispersos.	Orientada a proyectos pequeños. Corta duración (o entregas frecuentes), equipos pequeños (menos de 10 integrantes) y trabajando en el mismo sitio.
Se promueve que la arquitectura se defina tempranamente en el proyecto.	La arquitectura se va definiendo y mejorando a lo largo del proyecto.
Énfasis en la definición del proceso: roles, actividades y artefactos.	Énfasis en los aspectos humanos: el individuo y el trabajo en equipo.
Se espera que no ocurran cambios de gran impacto durante el proyecto.	Se esperan cambios durante el proyecto.

Debido a la necesidad de desarrollar la propuesta de solución en un breve período de tiempo, garantizando la flexibilidad necesaria en cuanto a la variación de los requisitos, no existiendo un contrato tradicional, siendo el cliente parte del equipo de desarrollo, permitiendo reducir la generación de documentos y artefactos; se hace necesario optar por un enfoque ágil de desarrollo de software en lugar de un enfoque tradicional.

Existen varias metodologías que siguen un enfoque ágil de desarrollo de software, por ejemplo: *Scrum* (34), *Crystal* (35) y *Extreme Programming* (XP) (36). *Scrum* define un marco para la

gestión de proyectos. Está indicada para proyectos con un rápido cambio de requisitos. Una de sus características es la realización de reuniones a lo largo del proyecto. *Crystal* se trata de un conjunto de metodologías caracterizadas por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. XP se centra en potenciar las relaciones interpersonales como clave para el éxito del desarrollo de software, promoviendo el trabajo en equipo y preocupándose por el aprendizaje de los desarrolladores. XP se define adecuada para proyectos con requisitos imprecisos y cambiantes.

Scrum y XP son unas de las metodologías ágiles más utilizadas (37). La siguiente tabla muestra una comparación realizada entre ellas.

Tabla VI: *Comparación entre las metodologías Scrum y XP*

Scrum	XP
Las iteraciones de entrega son de 1 a 4 semanas.	Las iteraciones de entrega son de 1 a 3 semanas.
Metodología de desarrollo ágil basada en la administración del proyecto.	Metodología de desarrollo ágil centrada en la programación o creación del producto.
Cada miembro del equipo trabaja de forma individual.	Los miembros de equipo trabajan en parejas.

Un estudio realizado en (37) muestra la frecuencia de las metodologías ágiles utilizadas en una investigación. En esta investigación se analizan 109 proyectos, los cuales difieren en la localización donde fueron desarrollados (Europa, América, Asia y África). La metodología ágil más utilizada es XP con un 53.2%, como se muestra en la siguiente tabla.

Tabla VII: *Frecuencia del uso de metodologías ágiles*

Metodología	Frecuencia de uso	Porciento
XP	58	53.2
Scrum	23	21.1
Otras	26	23.9
Crystal	2	1.8
Total	**109**	**100.0**

Se selecciona la metodología XP, la cual es usada en el desarrollo de herramientas en el proyecto DBJournal. Pertenece al grupo de metodologías ágiles, utilizada para producir software en breves espacios de tiempo y con un equipo reducido que se enfrenta a requisitos cambiantes. Hace énfasis en la adaptabilidad del proceso. En el desarrollo vincula estrechamente al usuario final con el equipo, propiciando un buen clima de trabajo como clave para lograr el éxito (38).

Metodología XP

XP fue creada por Kent Beck y actualmente es muy utilizada (39). Se basa en la suposición de que es posible desarrollar software de gran calidad a pesar del cambio continuo. Se basa además, en la retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. Se asume que con un poco de planificación, un poco de codificación y unas pocas pruebas se puede decidir si se está siguiendo un camino acertado o equivocado, evitando así tener que echar marcha atrás demasiado tarde. Está basada en 5 valores (38):

Simplicidad: enfoque centrado en un diseño sencillo del código generando sólo la documentación indispensable.

Comunicación: potenciada por el desarrollo en pares y la presencia del cliente.

Retroalimentación: protagonismo del cliente que participa activamente y trabajo en ciclos cortos.

Coraje: toma de decisiones en ocasiones complejas, pudiendo afectar el tiempo de desarrollo y la calidad del producto.

Respeto: estimar en toda su magnitud el trabajo de los demás.

XP posee prácticas valiosas sustentadas en los 5 valores anteriores. Estas se complementan unas con otras ofreciendo una base sólida para el buen desempeño, alta productividad y beneficios. Se destacan entre ellas: pruebas antes de programar, diseño incremental, ciclo semanal, integración continua e implicación real del cliente.

El ciclo de vida ideal de XP se compone de seis fases: Exploración, Planificación de la Entrega (*Release*), Iteraciones (de no más de tres semanas), Producción, Mantenimiento y Muerte del Proyecto. Algunas ventajas que posee XP se refieren a la programación organizada existiendo menor tasa de errores debido a sus características.

Dentro de las características que posee XP se encuentra realizar un desarrollo iterativo e incremental. Se realizan pruebas unitarias continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión. Programación en parejas: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. Se centra en la frecuente integración del equipo de programación con el cliente o usuario. Son corregidos los errores antes de añadir una nueva funcionalidad. Se emplea la refactorización del código, es decir, rescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad, pero sin modificar su comportamiento. Existe la propiedad del código compartida: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos; este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Se basa también en la simplicidad en el código: es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir funcionalidad si es necesario. Algunas de estas características distinguen a XP entre todas las metodologías ágiles (33).

1.6. Estudio de herramientas y tecnologías

En el desarrollo de la investigación se utilizan diferentes herramientas y tecnologías. Por lo tanto se realiza una breve investigación y descripción de las mismas.

Sistema Gestor de Bases de Datos

Un Sistema Gestor de Bases de Datos (SGBD) es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Posee un lenguaje de consultas. Permite definir los datos a distintos niveles de abstracción y manipularlos (40).

PostgreSQL es un SGBD orientado a objetos, libre y multiplataforma, publicado bajo la licencia BSD (*Berkeley Software Distribution*). Utiliza un modelo cliente/servidor.

Sus características técnicas la hacen una de las bases de datos más potente y robusta del mercado. Su desarrollo comenzó hace más de 16 años, y durante este tiempo, estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta durante su desarrollo (40). En este trabajo se utiliza la versión PostgreSQL 9.3.

Lenguaje de Programación

Se utiliza el lenguaje de programación **Java**. Es utilizado además en el desarrollo del proyecto DBJournal. Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos. Es un lenguaje de programación de propósito general, concurrente, orientado a objetos y basado en clases que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo, lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser compilado otra vez para correr en otra. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso (41). En este trabajo se utiliza Java 1.7.

Entorno de desarrollo

Un Entorno de Desarrollo Integrado, por sus siglas en inglés (IDE[20]) es un programa compuesto por una serie de herramientas utilizadas por programadores para desarrollar aplicaciones (42).

NetBeans es un entorno de desarrollo integrado, por sus siglas en inglés (IDE). Consiste en un software cuyo principal objetivo es el desarrollo de otro software. NetBeans IDE permite desarrollar aplicaciones usando el lenguaje de programación Java, así como aplicaciones con HTML, JavaScript y CSS. Además proporciona un gran conjunto de herramientas para PHP y C / C + + (43). En este trabajo se utiliza la versión IDE NetBeans 7.4.

1.6.1. Marco de trabajo para aplicaciones de la Web Semántica

OpenRDF Sesame[21] es un marco de trabajo para el procesamiento de datos RDF. Incluye programas de análisis, almacenamiento, razonamiento y consultas, utilizando el lenguaje de consulta SPARQL. Ofrece una API (del inglés A*pplication Programming Interface*) en Java que se puede conectar a las soluciones de almacenamiento RDF. Soporta los formatos de archivos RDF, incluyendo RDF/XML, Turtle, N-Triples y TriX (44).

Apache Jena[22] que es un marco de trabajo para Java que permite la construcción de aplicaciones para la Web Semántica, ofreciendo una colección de herramientas y librerías Java. Entre sus principales características se encuentran (45):

[20] Integrated Development Environment
[21] http://www.openrdf.org
[22] http://jena.apache.org/

- Posee una interfaz de programación de aplicaciones o API para RDF, la misma que soporta la creación, manipulación y consulta de grafos RDF.
- Posee un API para el manejo de ontologías.
- Permite realizar lectura y escritura de documentos en formato RDF/XML, N3 y NTriples.
- Posee un almacenamiento persistente para poder recopilar gran cantidad de tripletas RDF.

Apache Jena 2.11 es el marco de trabajo utilizado en este trabajo. Incluye la utilización del lenguaje de consultas para grafos RDF SPARQL 1.1 Update, diferenciándolo de OpenRDF Sesame.

1.6.2. Almacén de tripletas RDF

Virtuoso[23] es un almacén de tripletas RDF conocido por su buen rendimiento, ejecutando consultas rápidas sobre conjunto de datos pequeños. Ha sido utilizado para almacenar importantes conjuntos de datos enlazados (por ejemplo *DBPedia*[24]). *Virtuoso* ofrece directamente un *Endpoint SPARQL* que permite realizar consultas a los recursos contenidos en dicho servidor. Sin embargo, no contempla el uso de SPARQL 1.1 Update.

Jena Fuseki[25] es un *Endpoint SPARQL*. Provee el uso de estándares tales como SPARQL 1.1 Query y SPARQL 1.1 Update usando el protocolo SPARQL para HTTP. En este trabajo se utiliza Jena Fuseki 1.0.1 debido a la posibilidad tanto de consultas como actualizaciones en el grafo RDF.

1.7. Conclusiones parciales

Las variantes para capturar cambios realizados en BDR están basadas en *triggers* o en registros de *logs*. Dada las características de ambos y que se pretende disminuir el costo en cuanto a recursos computacionales, los *triggers* constituyen una opción factible para la captura de los cambios que se realicen en la BDR.

Los lenguajes para la alineación de BDR-RDF difieren en cuanto a categorías y características en la forma de realizar la alineación. Al ser necesaria la manipulación de dicha alineación es

[23] http://virtuoso.openlinksw.com
[24] http://dbpedia.org
[25] https://jena.apache.org/documentation/serving_data/index.html

recomendable hacer uso del lenguaje R2RML, siendo este un estándar de la W3C, además de brindar la posibilidad de realizar la alineación de forma semiautomática.

Para la actualización incremental del grafo RDF existen varios formatos, los cuales difieren en características y simplicidad. No todos son recomendados por el W3C excepto SPARQL 1.1 Update.

Las herramientas y tecnologías a utilizar están compuestas por el SGBD PostgreSQL 9.3. Como lenguaje de programación se utiliza Java 1.7. Como entorno de desarrollo el NetBeans 7.4. Se utiliza además el marco de trabajo para aplicaciones de la Web Semántica Apache Jena 2.11, como lenguaje para la consulta de grafos RDF SPARQL 1.1 Update y el *Endpoit SPARQL* Jena Fuseki 1.0.1.

CAPÍTULO 2. DESCRIPCIÓN E IMPLEMENTACIÓN DEL MÉTODO

2.1. Introducción

El presente capítulo está compuesto por dos secciones principales. En la primera sección se describe la propuesta de solución desde un enfoque teórico, basada en un método compuesto por tres fases para la actualización incremental de grafos RDF. La salida de cada fase va a constituir la entrada del próximo. Para esto se utiliza una aproximación basada en el modelo arquitectónico de tuberías y filtros. En la segunda sección se describe el diseño de la propuesta de solución. Se especifican los principales artefactos generados por la metodología de desarrollo de software XP, así como la propuesta arquitectónica para la actualización incremental de grafos RDF. Por otra parte, serán abordadas en este capítulo las fases de Exploración, Planificación e Iteraciones que propone el ciclo de vida de XP.

2.2. Modelo teórico de la solución

Si se obtienen los cambios de la BDR, se alinean con el modelo ontológico y los vocabularios, se obtienen las tripletas RDF y con estas se actualiza el grafo RDF, no ha de ser necesario actualizar el grafo generándolo en su totalidad. Esta aproximación implicaría una reducción en cuanto al costo en tiempo y recursos computacionales de la actualización de grafos RDF.

El método propuesto consta de tres fases y sigue un enfoque incremental. Cada proceso depende, en gran medida del resultado de la fase anterior, siendo la salida de cada fase, la entrada de la próxima. Las fases propuestas son:

1. Detectar cambios en la BDR.
2. Generar tripletas RDF.
3. Actualizar las tripletas en el grafo RDF.

Según la revisión realizada en el capítulo anterior, en cada una de las tres fases mencionadas anteriormente intervienen varias herramientas y tecnologías para su desarrollo. La siguiente ilustración muestra el método propuesto para la actualización de grafos RDF. Se refleja el orden de las fases que lo componen, los resultados que serán obtenidos en cada una y las tecnologías que intervienen en las mismas.

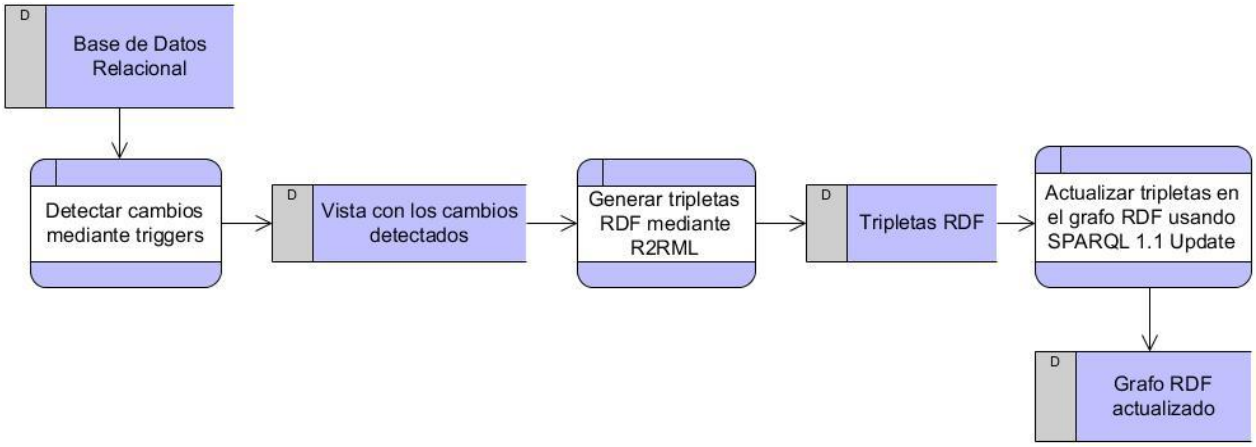

Ilustración I: *Método para la actualización incremental de grafos RDF*

El método representado en la **Ilustración I** se puede definir usando el álgebra relacional. Para definir la BDR Codd en 1970 propone un modelo para su descripción. El método queda definido de la siguiente forma.

- Sea (M) una base de datos relacional definida de la forma: $M = (R^M, A^M, att^M, pk^M, fk^M)$ donde:
  - $R^M = \{R_1^M, \dots, R_n^M\}$: Conjunto de tablas que contienen los mismos atributos.
  - $A^M = \{A_1^M, \dots, A_n^M\}$: Conjunto de atributos (columnas).
  - att^M: Conjunto de tuplas en la forma $(R_i^M, subconjunto(A^M))$, describe atributos de las tablas.
  - pk^M: Conjunto de tuplas en la forma $(R_i^M, subconjunto(A^M))$, describe la llave primaria de cada tabla.
  - fk^M: Conjunto de tuplas en la forma (R_1^M, A_1^M),(R_2^M, A_2^M) describe las llaves foráneas de las relaciones.
- Se genera, a partir de (M), un grafo RDF (G) mediante una implementación del lenguaje de alineación R2RML.
- En M existen *triggers* asociados a $(R_1^M, \dots, R_{n-1}^M)$ que activan una función F tras ocurrir operaciones de tipo i (O_i) donde:
  - i: Tipo de operación que se realiza en M. Valores: (1: *insert*, 2: *update*, 3: *delete*).
  - F: Función que inserta en la tabla R_n^M los datos referentes a los cambios realizados en M $\forall$ (O_i).
- La tabla en la que se encuentran los datos de los cambios se define como: R_n^M(id, *table_name, id_row, type_of_change*) donde:

- *id*: identificador (llave primaria).
- *table_name*: es R_i^M, donde i representa la tabla donde ocurrió el cambio.
- *id_row*: es pk^M que corresponde a R_i^M.
- *type_of_change*: O_i.

➢ Luego:

✓ Caso $O_i = (insert \text{ v } update)$

- $\sigma_{pk^M = id\_row}\ (R_i^M)$
- De la consulta anterior se obtiene un conjunto att_i^M correspondiente al conjunto att^M donde i representa el tipo de cambio correspondiente.
- Mediante una implementación del lenguaje de alineación R2RML se obtiene, a partir de att_i^M, el conjunto de tripletas RDF (T_i) correspondiente al tipo de cambio i.
- Usando consultas SPARQL correspondientes para cada i, se actualiza el grafo G.

✓ Caso $O_i = (delete)$

- Usando consultas SPARQL correspondiente para $i = 3$, y teniendo *table_name* e *id_row*, se conocen las tripletas T_3 a eliminar y se actualiza el grafo G.

➢ Luego el grafo actualizado (G_a) se define como: $G_a = (G \cup T_1 \cup T_2) \setminus T_3$

➢ Aclaraciones generales:

- Los triggers se asocian a las tablas de la BDR que se reflejarán en el grafo RDF, existiendo una personalización en la confección del grafo RDF.
- Las consultas SPARQL son realizadas en el mismo orden en que fueron detectados los cambios en la BDR.
- Una vez que son reflejados los cambios en el grafo RDF los datos de la tabla temporal son eliminados.
- Se utiliza SPARQL 1.1 Update.

2.2.1. Detección de cambios mediante *triggers*

Las fuentes de datos lo constituyen los metadatos bibliográficos de las revistas científicas y académicas que emplean el protocolo de intercambio de metadatos OAI-PMH. Una vez recolectados (los metadatos), son almacenados en una BDR, siendo el gestor de base de datos *PostgreSQL* el usado para realizar esta función.

Las principales entidades que se representan en el modelo de datos RDF son:

author: Almacena información sobre los autores: nombre y afiliación. Se intenta crear un registro único por cada autor, evitando la duplicidad.

record: Almacena información sobre los registros bibliográficos: título, resumen, año de publicación, fuente, entre otros. Un registro bibliográfico puede estar asociado a una colección dentro del proveedor de datos.

set: Almacena información relacionada con las colecciones de registros bibliográficos.

journal: Almacena información acerca de las revistas científicas y académicas de donde provienen los metadatos.

En la detección de los cambios de la BDR se necesita obtener los datos de dichos cambios. Para la obtención de los cambios se agrega una nueva tabla al modelo.

check_updates: Almacena información referente a los cambios realizados en las tablas de la BDR. Sus atributos son: *table_name, id_row y type_of_change*, los cuales hacen referencia al nombre de la tabla, el identificador de la fila donde se realizó el cambio, así como el tipo de modificación realizado (inserción, modificación o eliminación) respectivamente.

Para detectar los cambios en la BDR se emplean *triggers* encargados de capturar los datos involucrados en las operaciones inserción, modificación y/o eliminación realizadas sobre las tablas de la BDR.

A cada tabla, descrita anteriormente, deben estar asociados dos *triggers*, uno para las operaciones de inserción y modificación y otro para la eliminación. La diferencia entre ambos radica en capturar el identificador nuevo de la fila creada o el identificador de la fila eliminada, que ya no existiría. Ambos *triggers* han de invocar la misma función.

La función invocada por los *triggers* se encarga de insertar los datos descritos anteriormente en la nueva tabla creada. Cada vez que los cambios recogidos en la tabla creada sean alineados con el modelo de datos RDF se borrará la tabla evitando el cúmulo de información innecesaria en la BDR.

2.2.2. Generar tripletas RDF mediante R2RML

Para realizar la alineación mediante R2RML es necesario generar un documento de alineación que hace referencia a la estructura de las tablas de la BDR para obtener los datos contenidos en ellas (19).

En esta fase, la primera vez que se realiza la alineación, los datos son extraídos directamente de las tablas del modelo relacional. Cuando se proceda a realizar la actualización, las tripletas serán generadas a partir de la vista generada en la fase anterior.

La salida de esta fase está constituida por un grafo RDF en primera instancia generado en su totalidad. Además estará constituida por el conjunto de tripletas correspondientes a los cambios realizados en la BDR.

2.2.2.1. Modelo ontológico

El modelo ontológico constituye un paso fundamental en la alineación BDR - RDF. Su objetivo es la creación o reutilización de ontologías necesarias para la modelación de los datos. Esta tarea es esencial, a partir que un correcto modelado de los datos, permitirá posteriormente, la generación del grafo RDF. Cualquier detalle que se omita o cualquier ontología necesaria que se deje de plasmar en el modelo ontológico, influirá posteriormente en la publicación de los metadatos bibliográficos como datos enlazados, afectando el proyecto desarrollado.

El dominio de los metadatos bibliográficos ha sido ampliamente estudiado por ingenieros de ontologías y expertos en esta área del conocimiento. Existen algunas herramientas de búsqueda específicas (**Falcons, LOV, Watson, Sindice, motor de búsqueda semántica Web, Swoogle, Schemapedia**) que recopilan datos semánticos disponibles, así como ontologías para la descripción de datos en diferentes dominios. Para esta investigación se identificaron las ontologías que se describen en la **Tabla VIII**.

Tabla VIII: *Descripción de las ontologías utilizadas*

Nombre	Fuente de la ontología	Descripción
rdfs	http://www.w3.org/2000/01/rdf-schema#	Esquema de vocabularios RDF.

fabio	http://purl.org/spar/fabio/	Es una ontología para el registro y publicación de registros bibliográficos en la Web Semántica de trabajos académicos.
foaf	http://xmlns.com/foaf/0.1/	FOAF (Amigo del amigo). Integra tres tipos de red: las redes sociales de colaboración humana, la amistad y asociación.
bibo	http://purl.org/ontology/bibo/	Ontología bibliográfica. Puede ser usada como una ontología de clasificación de documentos o simplemente como una forma de describir cualquier tipo de documento en RDF.
dc	http://purl.org/dc/elements/1.1/	La Iniciativa de Metadatos Dublin Core es una organización para apoyar la innovación en el diseño de los metadatos y las mejores prácticas en toda la tecnología de metadatos.
swrc	http://swrc.ontoware.org/ontology#	SWRC (Web Semántica para Comunidades de Investigación). Es una ontología para el modelado de entidades tales como: personas, organizaciones, publicaciones (metadatos bibliográficos) y sus relaciones.

Estas ontologías/vocabularios permiten modelar los datos en función del tipo de información que se desea publicar. Estos vocabularios se pueden utilizar de manera combinada para crear modelos de datos más ricos que permiten representar la información de una manera más completa.

En la siguiente tabla (**Tabla IX**) se muestran las clases ontológicas que modelan las entidades fundamentales de la BDR.

Tabla IX: *Clases ontológicas que describen las entidades*

Nombre	Descripción
fabio: *Journal*	Conjunto de revistas.
foaf: *Person*	Una persona.
fabio: *JournalArticle*	Artículo de una revista.

fabio: *ItemCollection*	Conjunto de colecciones que agrupa a varios artículos.

En la siguiente tabla (**Tabla X**) se muestra la descripción de cada uno de los principales atributos de las principales entidades de la BDR. Para ello se utilizaron las propiedades que ofrecen las diferentes ontologías seleccionadas.

Tabla X: *Descripción de las entidades y sus atributos a través de las ontologías seleccionadas*

Nombre de la tabla	Clase	Atributos	Propiedades
autor	foaf: *Person*	name affiliation label	foaf: name swrc: affiliation rdfs: label
record	fabio: *JorunalArticle*	record_identifier title description date source url_identifier year_pub	dc: identifier dc: title fabio: abstract dc: date dc: source bibo: uri fabio: hasPublicationYear
journal	fabio: *Journal*	title base_url	dc: title bibo: uri
set	fabio: *ItemCollection*	set_name	foaf: name

Una vez descritas las entidades y sus atributos mediante las ontologías seleccionadas se procede a describir las propiedades de objeto (conocidas como *ObjectProperties*) que modelan las relaciones entre las entidades.

Tabla XI: *Clases ontológicas que describen relaciones entre entidades*

Entidad_Entidad	Propiedad
journal_record	swrc: *Journal*
record_set	fabio: *InCollection*
record_author	fabio: *hasCreator*

2.2.2.2. Alineación BDR – RDF

Con el modelo ontológico definido es posible generar el documento de alineación y modificarlo según las ontologías seleccionadas.

Para R2RML existen varias implementaciones (46), por ejemplo: *OpenLink Virtuoso*[26], *XSPARQL*[27], *db2triples*[28], *r2rml-parser*[29]. El principal obstáculo en el uso de R2RML radica en crear y mantener el documento de alineación (31). Estas implementaciones contribuyen a la generación y al mantenimiento de las tripletas RDF. Difieren en lenguajes de programación en los que han sido desarrolladas y gestores de bases de datos que soportan.

Haciendo uso del documento de alineación con el modelo ontológico definido, se generan todas las tripletas que forman el grafo RDF la primera vez. Cada vez que existan cambios en la BDR se generan solamente las tripletas correspondientes a los cambios sin generar el grafo RDF.

2.2.3. Actualizar tripletas en el grafo RDF usando SPARQL 1.1 Update

De la fase anterior, obtenidas las tripletas RDF y conociendo a priori el tipo de operación que implicó el cambio en la BDR, es posible realizar las consultas SPARQL correspondientes. Estas consultas de inserción, eliminación o modificación se encargarán de reflejar los cambios detectados en la BDR en el grafo RDF. Esta es la última fase que arroja como resultado el grafo RDF actualizado.

Al no tener que generar el grafo RDF completamente cada vez que se realicen cambios en la BDR ha de reducirse el costo en tiempo y recursos computacionales, empleando el método propuesto.

2.3. Diseño de la propuesta de solución

En esta investigación se pretende desarrollar una herramienta que utilice el método propuesto. Será la encargada de actualizar de forma incremental el grafo RDF. La herramienta tiene como nombre ***UpRDF*** (del inglés *Update RDF*). Su nombre contiene como significado la actualización de grafos RDF.

[26] http://virtuoso.openlinksw.com
[27] http://xsparql.deri.org
[28] https://github.com/antidot/db2triples
[29] https://github.com/nkons/r2rml-parser

2.3.1. Fase de exploración

En la fase de exploración en la que se divide el ciclo de vida XP, se plantea el alcance del proyecto, teniendo como objetivo el desarrollo y la entrega del producto requerido. Para ello el cliente define sus necesidades (requisitos tanto funcionales como no funcionales) a través de las Historias de Usuario (HU), a partir de las cuales los programadores definen el tiempo que se demorarán para desarrollar el sistema. Estas estimaciones se encuentran propensas a cambios. Se define una propuesta de arquitectura del sistema (38).

2.3.1.1. Propuesta de arquitectura

La propuesta de solución sigue una arquitectura de flujo de datos. Es aplicada cuando los datos de entrada se habrán de transformar en datos de salida mediante una serie de componentes para el cálculo o la manipulación. La estructura utilizada es de tuberías y filtros. Posee un conjunto de componentes, denominados filtros, conectados por tuberías que transmiten datos de un componente al siguiente. Cada filtro se diseña para esperar la entrada de datos con cierta forma y producir su salida (al siguiente filtro) de una forma específica (47).

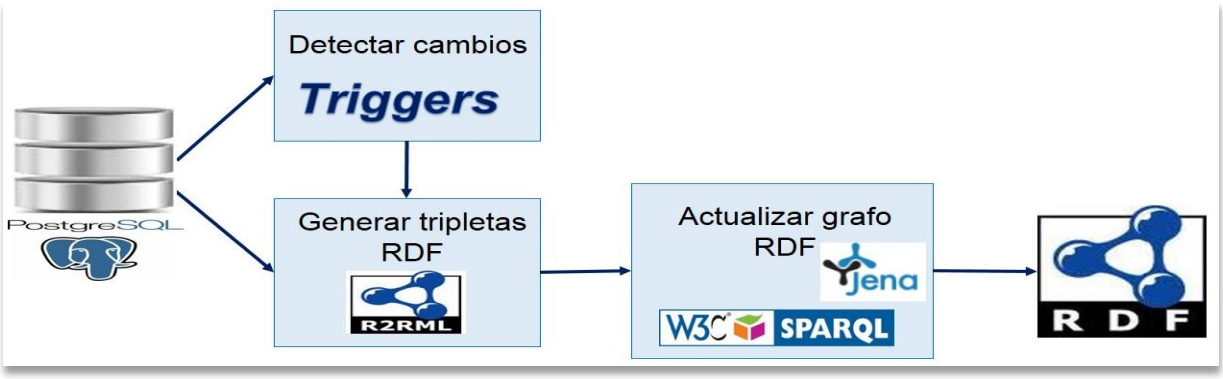

Ilustración II: *Arquitectura de la propuesta de solución*

En la **Ilustración II** se muestra la arquitectura de la propuesta de solución (UpRDF). En la BDR se realiza la detección de cambios. Se basa en la creación de la estructura adecuada para la detección y obtención de los cambios realizados en la BDR como se describe en la sección 2.2.1. Se crean funciones que inserten los datos de dicha tabla y se crean los *triggers* asociados a cada tabla que invocan a las correspondientes funciones. A cada tabla especificada se asocian dos *triggers*, uno para las operaciones de inserción y modificación y otro para la eliminación. La diferencia entre ambos radica en la captura del identificador de la fila relacionada al cambio. Uno

se utiliza para obtener los nuevos identificadores insertados, o los referentes a una modificación. El otro va a obtener el identificador de la fila eliminada, que deja de existir en la BDR.

Para la generación de las tripletas RDF se necesita el documento de alineación (*mapping*). Este es generado y configurado manualmente, realizando la alineación del modelo relacional con el modelo ontológico definido. Luego, la generación de tripletas RDF puede brindar dos salidas según se especifique. La primera genera todas las tripletas que conforman el grafo RDF en su totalidad. La segunda contempla la creación de tripletas RDF a partir de los cambios detectados en la BDR.

De lo descrito anteriormente se obtienen las tripletas RDF y se conoce a *priori* el tipo de operación que implicó el cambio en la BDR. Por lo tanto, es posible realizar las consultas SPARQL de inserción, eliminación o modificación correspondientes, que se encargarán de reflejar los cambios identificados en la BDR en el grafo RDF.

Con la nueva versión SPARQL 1.1 Update es posible realizar ese tipo de consultas. Con esto se obtiene como resultado el grafo RDF actualizado. Este grafo se genera completamente solo la primera vez, luego se actualiza de forma incremental.

2.3.1.2. Modelo de datos

El modelo de datos de la propuesta de solución está formado por cuatro recursos (*author, record, journal, set*), con sus propiedades y relaciones, ver **Ilustración III**.

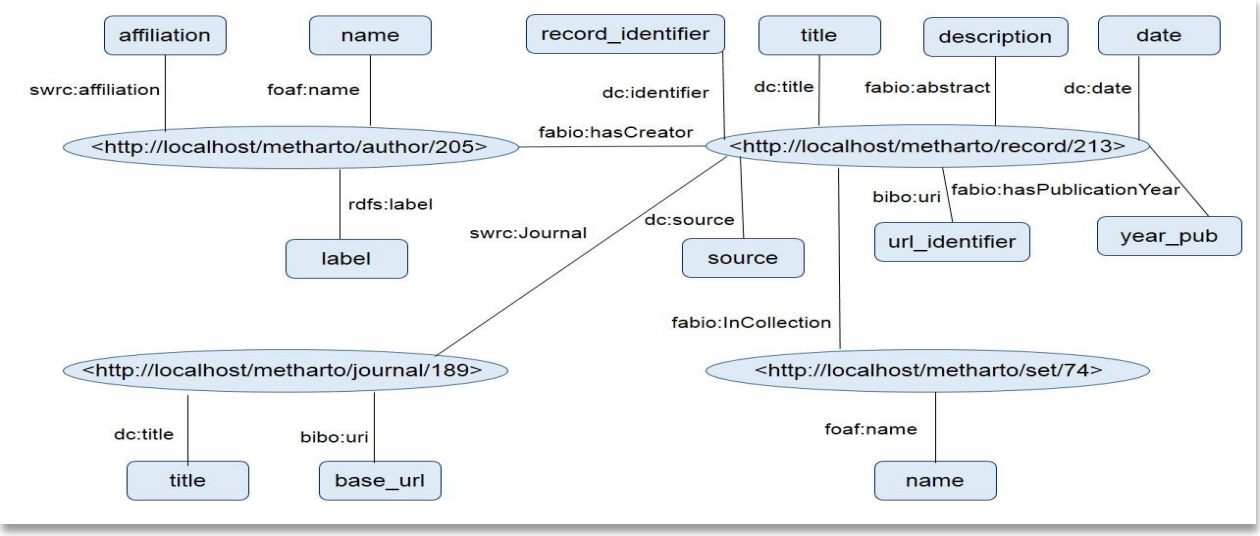

Ilustración III: *Modelo de datos de la propuesta de solución*

2.3.2. Fase de planificación

En esta fase el cliente establece la prioridad de cada HU y correspondientemente los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Las HU se escriben en lenguaje del cliente, es un requisito que debe satisfacer el sistema. Debe ser lo suficientemente sencillo como para que el cliente pueda comprenderlo y además el programador conozca qué implementar y poder llevar una traza de su trabajo sobre esta funcionalidad (39).

Si cuando el cliente escribe la HU, el programador entiende que no es lo completamente sencilla como para implementarla como una funcionalidad atómica, entonces se divide en dos o más historias.

Tienen varias clasificaciones en las que se pueden agrupar según las características del equipo de desarrollo y del proyecto, estas son: en base a su complejidad, en base a los riesgos que representan y en base a la prioridad. La prioridad es la clasificación más importante, ya que está directamente relacionada con los intereses del cliente.

2.3.2.1. Estimación de esfuerzo por Historias de Usuario

Según la prioridad asignada por el cliente a cada HU y teniendo presente la complejidad y riesgo determinado por el programador, se plasma la estimación de cada una de las HU identificadas, los resultados de la estimación se muestran en la siguiente tabla. La unidad de estimación es el punto, un punto equivale a una semana ideal de programación.

Tabla XII: *Estimación por HU*

Historias de Usuario	Puntos de estimación
Detectar cambios en la BDR	2
Generar tripletas RDF	3
Actualizar tripletas en el grafo RDF	2

2.3.2.2. Plan de iteraciones

Una vez descritas las HU y su estimación por parte del cliente para su posterior implementación por parte de los desarrolladores involucrados, se procede a la planificación de la entrega del sistema, agrupando a todas las HU que serán desarrolladas por cada iteración. Relacionado con

lo planteado anteriormente se decide desarrollar el sistema en dos iteraciones, las cuales se especifican a continuación.

Iteración#1

La presente iteración tiene como objetivo implementar las HU UpRDF-01 y UpRDF-02, descritas en las **Tablas XIII y XIV.** Estas permiten detectar cambios que se realicen en la BDR, generar el grafo RDF y las correspondientes tripletas RDF a partir de los cambios detectados.

Tabla XIII: *Historia de Usuario UpRDF-01*

HISTORIA DE USUARIO			
Código:	UpRDF-01	**Nombre:**	Detectar cambios en la BDR
Usuario:	Programador	**Actividad:**	Detectar cambios en la BDR
Riesgo:	Alto	**Prioridad:**	Alta
Iteración:	1	**Puntos estimados:**	2
Descripción:	Se crea una tabla en la BDR que recoge los cambios detectados mediante el empleo de *triggers* asociados a las tablas de la BDR.		

Tabla XIV: *Historia de Usuario UpRDF-02*

HISTORIA DE USUARIO			
Código:	UpRDF-02	**Nombre:**	Generar tripletas RDF
Usuario:	Programador	**Actividad:**	Generar tripletas RDF
Riesgo:	Alto	**Prioridad:**	Alta
Iteración:	1	**Puntos estimados:**	3
Descripción:	La primera vez se genera el grafo RDF utilizando una solución basada en una implementación del lenguaje de alineación R2RML. Se pasa como parámetro un documento de alineación generado sobre los metadatos almacenados en la base de datos. Luego se generan las tripletas RDF a partir de los cambios detectados en la Historia de Usuario UpRDF-01.		

Iteración#2

Esta iteración tiene como objetivo implementar la HU UpRDF-03, descrita en la **Tabla XV**. Esta permite realizar la actualización al grafo RDF mediante consultas SPARQL.

Tabla XV: *Historia de Usuario UpRDF-03*

HISTORIA DE USUARIO			
Código:	UpRDF-03	**Nombre:**	Actualizar tripletas en el grafo RDF
Usuario:	Programador	**Actividad:**	Actualizar tripletas en el grafo RDF
Riesgo:	Alto	**Prioridad:**	Alta
Iteración:	1	**Puntos estimados:**	2
Descripción:	Se generan consultas SPARQL dinámicas con las salidas generadas en las Historias de Usuario UpRDF-01 y UpRDF-02. Se realizan las consultas al grafo RDF generado en la Historia de Usuario UpRDF-02.		

A continuación (**Tabla XVI**) se muestra el plan de iteraciones para dar cumplimiento a cada una de las HU previamente definidas.

Tabla XVI: *Plan de iteraciones*

ITERACIÓN	HISTORIA DE USUARIO	DURACIÓN (en semanas)	
Iteración#1	UpRDF-01	2	5
	UpRDF-02	3	
Iteración#2	UpRDF-03	2	2

2.3.2.3. Plan de entrega

A continuación se presenta el plan de entrega elaborado para la fase de implementación, dada la duración de cada HU por iteración definida anteriormente.

Tabla XVII: *Plan de entrega*

Iteración	Iteración#1	Iteración#2
Cantidad de Historias de Usuario	2	1
Fecha de entrega	27/04/2014	10/05/2014

2.3.3. Fase de Iteraciones

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las HU que realicen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué HU se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción. Todo el trabajo de la iteración es expresado en tareas, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores.

2.3.3.1. Resumen de tareas de ingeniería

A continuación se muestran las tareas de ingeniería a desarrollar por cada HU en cada iteración (**Tabla XVIII**).

Tabla XVIII: *Resumen de tareas de ingeniería por HU*

Iteración	Historia de Usuario	No. de tarea	Tarea de ingeniería
Iteración#1	UpRDF-01	TI -01	Agregar tabla *check_updates* a la BDR
		TI -02	Crear funciones *insert_check_updates* y *insert_check_updates_delete*
		TI -03	Crear *triggers* asociados a las tablas de la BDR
		TI-04	Confeccionar el modelo ontológico
		TI -05	Generar documento de alineación (*mapping.n3 o mapping.ttl*)

	UpRDF-02	TI -06	Configurar documento de alineación (*mapping.n3 o mapping.ttl*)
		TI-07	Generar el grafo RDF
		TI -08	Obtener tuplas actualizadas en la BDR mediante consultas a la tabla *check_updates*
		TI -09	Generar tripletas RDF
Iteración#2	UpRDF-03	TI -10	Generar consultas SPARQL
		TI -11	Actualizar tripletas en el grafo RDF usando SPARQL

2.3.3.2. Tareas de ingeniería detalladas

A continuación se describen las tareas de ingeniería correspondientes a las HU descritas anteriormente y a las iteraciones.

Iteración#1

Tabla XIX: *Tarea de ingeniería de la Historia de Usuario UpRDF-01*

TAREA DE INGENIERÍA			
Código:	TI-01	**Historia de Usuario:**	UpRDF-01
Nombre de la tarea:	Agregar tabla *check_updates* a la BDR.		
Responsable:		Katerín Martínez Rojas	
Tipo de tarea:	Desarrollo	**Horas estimadas:**	5
Fecha inicial:	29/03/2014	**Fecha final:**	30/03/2014
Descripción:	Se agrega al modelo relacional la entidad *check_updates* encargada de recoger los datos de los cambios que se detecten en la BDR.		

Tabla XX: *Tarea de ingeniería de la Historia de Usuario UpRDF-01*

TAREA DE INGENIERÍA			
Código:	TI-02	**Historia de Usuario:**	UpRDF-01
Nombre de la tarea:	Crear funciones *insert_check_updates* y *insert_check_updates_delete.*		

Responsable:		Katerín Martínez Rojas	
Tipo de tarea:	Desarrollo	**Horas estimadas:**	8
Fecha inicial:	01/04/2014	**Fecha final:**	03/04/2014
Descripción:	Se crean dos funciones que guardan los datos descritos en TI-05. Una función hace referencia al tipo de operación *insert* o *update* y la otra a *delete*.		

Tabla XXI: *Tarea de ingeniería de la Historia de Usuario UpRDF-01*

TAREA DE INGENIERÍA			
Código:	TI-03	**Historia de Usuario:**	UpRDF-01
Nombre de la tarea:	Crear *triggers* asociados a las tablas de la BDR.		
Responsable:		Katerín Martínez Rojas	
Tipo de tarea:	Desarrollo	**Horas estimadas:**	10
Fecha inicial:	4/04/2014	**Fecha final:**	08/04/2014
Descripción:	A cada tabla de la BDR se le asocian dos *triggers* que invocan a las funciones correspondientes creadas en TI-06. Uno para las operaciones de *insert* y *update* y otro para *delete*.		

Tabla XXII: *Tarea de ingeniería de la Historia de Usuario UpRDF-02*

TAREA DE INGENIERÍA			
Código:	TI-04	**Historia de Usuario:**	UpRDF-02
Nombre de la tarea:	Confeccionar el modelo ontológico		
Responsable:		Katerín Martínez Rojas	
Tipo de tarea:	Diseño	**Horas estimadas:**	8
Fecha inicial:	09/04/2014	**Fecha final:**	10/04/2014
Descripción:	Se modela en términos de clases y propiedades de ontologías cada una de las entidades y atributos de la BDR, así como las relaciones entre sus entidades.		

Tabla XXIII: *Tarea de ingeniería de la Historia de Usuario UpRDF-02*

TAREA DE INGENIERÍA			
Código:	TI-05	**Historia de Usuario:**	UpRDF-02
Nombre de la tarea:	Generar documento de alineación (*mapping.n3 o mapping.ttl*)		

Responsable:		Katerín Martínez Rojas	
Tipo de tarea:	Desarrollo	**Horas estimadas:**	6
Fecha inicial:	11/04/2014	**Fecha final:**	12/04/2014
Descripción:	Se genera el documento de alineación (*mapping.n3 o mapping.ttl*) empleando el lenguaje de alineación R2RML y *"d2rq-r2rml-preview-v4"*.		

Tabla XXIV: *Tarea de ingeniería de la Historia de Usuario UpRDF-02*

TAREA DE INGENIERÍA			
Código:	TI-06	**Historia de Usuario:**	UpRDF-02
Nombre de la tarea:	Configurar documento de alineación (*mapping.n3 o mapping.ttl*)		
Responsable:		Katerín Martínez Rojas	
Tipo de tarea:	Desarrollo	**Horas estimadas:**	8
Fecha inicial:	13/04/2014	**Fecha final:**	14/04/2014
Descripción:	Se configura manualmente el *mapping* generado en la tarea de ingeniería anterior. Se agregan las ontologías seleccionadas para modelar la BDR.		

Tabla XXV: *Tarea de ingeniería de la Historia de Usuario UpRDF-02*

TAREA DE INGENIERÍA			
Código:	TI-07	**Historia de Usuario:**	UpRDF-02
Nombre de la tarea:	Generar el grafo RDF		
Responsable:		Katerín Martínez Rojas	
Tipo de tarea:	Desarrollo	**Horas estimadas:**	5
Fecha inicial:	15/04/2014	**Fecha final:**	16/04/2014
Descripción:	Se genera el grafo RDF haciendo uso del documento de alineación.		

Tabla XXVI: *Tarea de ingeniería de la Historia de Usuario UpRDF-02*

TAREA DE INGENIERÍA			
Código:	TI-08	**Historia de Usuario:**	UpRDF-02
Nombre de la tarea:	Obtener tuplas actualizadas en la BDR mediante consultas a la tabla *check_updates*.		

Responsable:		Katerín Martínez Rojas	
Tipo de tarea:	Desarrollo	**Horas estimadas:**	8
Fecha inicial:	17/04/2014	**Fecha final:**	18/04/2014
Descripción:	Se realizan consultas *select* a la tabla *check_updates* para obtener las tuplas correspondientes a los datos recogidos en ella.		

Tabla XXVII: *Tarea de ingeniería de la Historia de Usuario UpRDF-02*

TAREA DE INGENIERÍA			
Código:	TI-09	**Historia de Usuario:**	UpRDF-02
Nombre de la tarea:	Generar tripletas RDF.		
Responsable:		Katerín Martínez Rojas	
Tipo de tarea:	Desarrollo	**Horas estimadas:**	12
Fecha inicial:	21/04/2014	**Fecha final:**	26/04/2014
Descripción:	Con las tuplas obtenidas en TI-08 se generan las tripletas RDF correspondientes.		
Observaciones:	Es necesario tener almacenado el tipo de operación relacionado a la tripleta generada para posterior uso.		

Iteración#2

Tabla XXVIII: *Tarea de ingeniería de la Historia de Usuario UpRDF-03*

TAREA DE INGENIERÍA			
Código:	TI-10	**Historia de Usuario:**	UpRDF-03
Nombre de la tarea:	Generar consultas SPARQL.		
Responsable:		Katerín Martínez Rojas	
Tipo de tarea:	Desarrollo	**Horas estimadas:**	12
Fecha inicial:	28/04/2014	**Fecha final:**	03/05/2014
Descripción:	Dadas las tripletas generadas en TI-09 y los datos de la tabla *check_updates* se generan dinámicamente las consultas SPARQL.		

Tabla XXIX: *Tarea de ingeniería de la Historia de Usuario UpRDF-03*

TAREA DE INGENIERÍA			
Código:	TI-11	**Historia de Usuario:**	UpRDF-03
Nombre de la tarea:	Actualizar tripletas en el grafo RDF usando SPARQL.		

Responsable:		Katerín Martínez Rojas	
Tipo de tarea:	Desarrollo	**Horas estimadas:**	10
Fecha inicial:	06/05/2014	**Fecha final:**	10/05/2014
Descripción:	Se realizan las consultas generadas en TI-10 al grafo RDF generado en TI-04.		

2.4. Implementación de la herramienta *UpRDF*

2.4.1. Fase de Producción

En esta fase se describen las tarjetas Clase-Responsabilidad-Colaboración (CRC), definiendo el modelo de dominio y priorizando el enfoque orientado a objetos sobre el enfoque procedimental. La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Por esta razón en esta fase se realiza la planificación de la estrategia de pruebas a ejecutar en la próxima fase.

2.4.1.1. Tarjetas Clase-Responsabilidad-Colaboración (CRC)

Las tarjetas CRC son creadas para dar respuesta a una necesidad de documentar las decisiones de diseño de colaboración. Estas tarjetas se dividen en tres secciones que contienen la información del nombre de la clase, sus responsabilidades y sus colaboradores. Una clase es cualquier persona, cosa, evento, concepto, pantalla o reporte. Las responsabilidades de una clase son las cosas que conoce y las que realizan, sus atributos y métodos. Los colaboradores de una clase son las demás clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades (5).

Con otras metodologías de desarrollo de software es necesario definir un modelo de dominio (diagramas de clases *UML*[30]), que representa el conjunto de clases de la lógica de negocio y sus interrelaciones. En este caso, con XP, no es necesario definir un modelo de dominio, en su lugar se hace uso de tarjetas CRC (48).

A continuación se muestran las tarjetas CRC de las principales clases de la lógica del negocio.

Tabla XXX: *Tarjeta CRC clase DatabaseManager*

DatabaseManager

[30] Lenguaje unificado de Modelado (del inglés *Unified Modeling Language*)

Responsabilidad	Colaboración
Obtener la lista de cambios detectados en la BDR.	Connection ParserUtil
Obtener las tuplas correspondientes a los cambios detectados en la BDR.	

Tabla XXXI: *Tarjeta CRC clase Generator*

Generator	
Responsabilidad	**Colaboración**
Generar el grafo RDF a partir de una BDR dado el documento de alineación.	DatabaseManager ParserUtil MappingDocument
Generar tripletas RDF a partir de los cambios detectados en la BDR dado el documento de alienación y el grafo RDF generado previamente.	
Generar consultas SPARQL dadas las tripletas RDF, el tipo de operación y el grafo a actualizar generado previamente.	

Tabla XXXII: *Tarjeta CRC clase Parser*

ParserUtil	
Responsabilidad	**Colaboración**
Alinear las tablas de la BDR con el modelo ontológico definido en el documento de alineación.	DatabaseManager ParserUtil MappingDocument
Establecer las propiedades correspondientes del modelo ontológico con cada tabla de la BDR especificada en dicho modelo.	

Tabla XXXIII: *Tarjeta CRC clase MappingDocument*

MappingDocument	
Responsabilidad	**Colaboración**
Crear la estructura que va a tener la BDR según el modelo ontológico.	LogicalTableView LogicalTableMapping

2.4.1.2. Planificación de las pruebas

El proceso de pruebas de XP constituye una de sus fortalezas. Permite aumentar la calidad del sistema reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección.

Las pruebas del sistema se dividen en dos grupos: pruebas unitarias, verifican el código, siendo diseñadas para los programadores, y pruebas de aceptación o pruebas del cliente, las especifica el cliente y se enfocan en las características generales y la funcionalidad del sistema, se derivan de las HU que se han implementado (47).

Los clientes escriben las pruebas funcionales para cada HU que deba validarse. En este contexto de desarrollo evolutivo y de énfasis en pruebas constantes, la automatización para apoyar esta actividad es crucial.

Pruebas de aceptación

Las pruebas de aceptación son creadas a partir de las HU. Durante una iteración la HU seleccionada en la planificación de iteraciones se convertirá en una prueba de aceptación. El cliente o usuario especifica los aspectos a probar cuando una HU ha sido correctamente implementada.

Una prueba de aceptación es como una caja negra. Cada una de ellas representa una salida esperada del sistema. Es responsabilidad del cliente verificar la corrección de las pruebas de aceptación y tomar decisiones acerca de las mismas (47).

Para llevar a cabo este proceso se utilizaran pruebas de caja negra, creando para cada HU uno o más casos de prueba en dependencia de las funcionalidades que involucre.

Cada caso de prueba debe contener un código, la HU a la que pertenece, el nombre, una breve descripción, la acción a probar, los datos de entrada, los resultados esperados y la evaluación de la prueba.

2.4.2. Diagrama de despliegue

El diagrama de despliegue es un tipo de diagrama que se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. Describen la topología del sistema, la estructura de los elementos de hardware y el software que ejecuta

cada uno de ellos. Los diagramas de despliegue representan a los nodos y sus relaciones. Los nodos son conectados por asociaciones de comunicación tales como enlaces de red, conexiones TCP/IP.

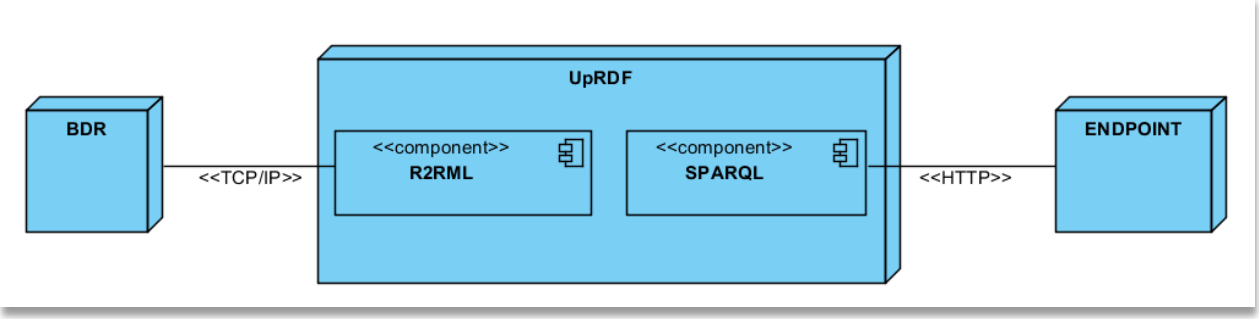

Ilustración IV: *Diagrama de despliegue*

De la BDR se obtienen los datos para construir el grafo RDF y los cambios detectados para actualizarlo. Estos datos se alinean con las ontologías utilizando el lenguaje de consulta R2RML y se generan tanto el grafo RDF inicial como las tripletas RDF a partir de los cambios detectados. Haciendo uso de SPARQL se publica el grafo en el *Endpoint SPARQL* Jena Fuseki donde se ejecutan las consultas SPARQL generadas previamente de forma automática.

2.5. Conclusiones parciales

El modelo teórico de la propuesta de solución para la actualización incremental de grafos RDF se basa en tres fases, siguiendo un enfoque basado en filtros y tuberías.

El método propuesto evita la generación total del grafo RDF, por lo que aporta una solución viable a los problemas de escalabilidad y rendimiento en la modificación y consulta de los datos procedentes de una BDR.

Las HU descritas por el cliente definieron los requisitos que debe cumplir el sistema, así como el nivel de detalle de cada funcionalidad.

La planificación de las iteraciones y el plan de entrega permitieron la organización de las HU según la prioridad del cliente y su posterior separación en tareas de ingeniería de acuerdo al tiempo requerido para su implementación.

El grado de acoplamiento del sistema se definió mediante la descripción de las tarjetas CRC, definiendo las clases, sus responsabilidades y la colaboración entre ellas.

CAPÍTULO 3. VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

3.1. Introducción

En este capítulo se realiza la validación de la solución propuesta en el capítulo anterior, en conjunto con la planificación de las pruebas de aceptación. Para cada HU se realizan las pruebas de aceptación que describe la metodología XP. Los errores detectados por los casos de prueba fueron mitigados tras un proceso iterativo. Se aplican métricas con el fin de validar el diseño realizado en el capítulo anterior. Se presenta un caso de estudio realizado para evaluar el costo en tiempo y recursos computacionales de la propuesta de solución para la actualización incremental de grafos RDF.

3.2. Validación del diseño

Para la validación del diseño de la solución propuesta, se estudiaron diferentes métricas de diseño y sus características (49). Con el objetivo de medir la complejidad de cada clase por separado se seleccionaron dos métricas que permiten realizar mediciones de este tipo: Tamaño de clase y Carencia de cohesión en los métodos. A continuación se muestra un ejemplo del resultado de la aplicación de estas métricas.

3.2.1. Métricas de diseño

Una métrica permite medir de forma cuantitativa la calidad de los atributos internos del software y de esta forma realizar una evaluación de la calidad del diseño durante el desarrollo del sistema. Las métricas se centran en cuantificar tanto la complejidad, como la funcionalidad y eficiencia inmersa en el desarrollo de software. Inclina sus objetivos a mejorar la comprensión de la calidad del producto, a estimar la efectividad del proceso y mejorar la calidad del trabajo (50).

3.2.1.1. Tamaño de Clase (TC)

Consiste en medir el tamaño general de una clase tomando el valor de la cantidad de operaciones que están encapsuladas dentro de dicha clase. Si el resultado obtenido indica valores grandes, significa que la clase posee un alto grado de responsabilidad, reduciendo la reutilización.

Será más difícil la implementación y la realización de pruebas de dicha clase. Mientras menor sea el valor para el TC será más fácil la reutilización de dicha clase dentro del sistema. A continuación se muestra la implicación de los valores altos de TC para los parámetros de calidad definidos para esta métrica.

- Reutilización: Para valores grandes de TC se reduce la reutilización de la clase.
- Complejidad de las pruebas: Para valores grandes de TC se hacen complejas las pruebas del sistema.
- Responsabilidad: Para valores grandes de TC la clase tiene mayor responsabilidad.

La métrica TC aplica umbrales definidos por Lorenz y Kidd para evaluar los resultados (49).

Tabla XXXIV: *Umbrales para la métrica TC*

TC	Umbral
Pequeño	<=20
Medio	>20 y <=30
Grande	>30

Un ejemplo de las medidas de los parámetros de calidad descritos anteriormente se muestra en la siguiente tabla (**Tabla XXXV**).

Tabla XXXV: *Resultados de la aplicación de TC*

Clase	Procedimientos	Responsabilidad	Complejidad	Reutilización
DatabaseManager	12	Media	Media	Media
Generator	15	Media	Media	Media
ParseUtil	9	Baja	Baja	Alta
MappingDocument	11	Baja	Baja	Alta

La métrica fue aplicada a un total de 4 clases para un total de 47 procedimientos, con un promedio de 12 aproximadamente. De las clases se obtiene un total de 4 de tamaño pequeño y ninguna de tamaño medio o grande.

Los resultados obtenidos para cada parámetro de calidad analizado, se muestran en los gráficos siguientes.

Gráfico II: *Parámetro Responsabilidad*

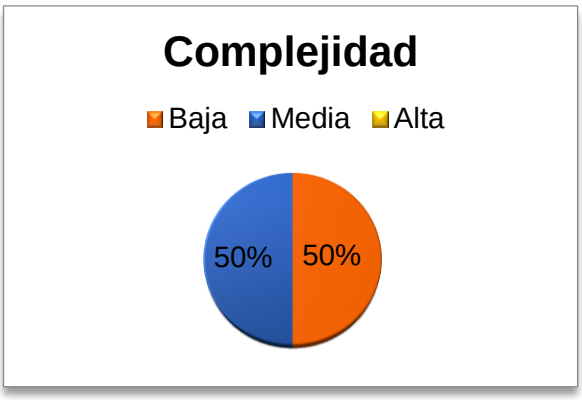

Gráfico III: *Parámetro Complejidad*

Gráfico III: *Parámetro Reutilización*

De los parámetros de calidad analizados se obtuvo que el 50% de las clases posee responsabilidad y complejidad media, mientras que el otro 50% posee bajos dichos parámetros. Para la reutilización se obtuvo que el 50% de las clases posee este parámetro en media, un 25% alta y el otro 25% baja. Estos datos arrojan un resultado positivo en la validación del diseño al no presentar valores altos de responsabilidad y complejidad, reflejándose la reutilización en índices mayormente medios y altos, favoreciendo la implementación de la solución.

3.2.1.2. Carencia de Cohesión en los Métodos (CCM)

La CCM se representa como la cantidad de métodos que acceden a un mismo atributo dentro de una clase. Cuando la CCM es alta los métodos deben acoplarse a otro por medio de los atributos, incrementando la complejidad del diseño de clases. En general, los valores de CCM altos implican que la clase debe ser rediseñada descomponiéndola en dos o más clases distintas. Todo método situado dentro de una clase C, accede a uno o más y en donde CCM es el número de métodos que acceden a uno o más de los mismos atributos. Si ningún método accede a los mismos atributos, entonces CCM será 0. En este caso el valor de CCM será aceptable mientras que no

exceda al valor que representa el 50% de la cantidad de métodos de una clase C. Un ejemplo de la aplicación de esta métrica se describe a continuación.

Tabla XXXVI: *Atributos de la clase Generator*

Atributo	Identificador
resultModel	r
db	db
util	u
properties	p
verbose	v
logModel	lm

Tabla XXXVII: *Métodos de la clase Generator*

Método	Atributos
incrementalUpdate	db, p
createGraph	rm, db, u, p, v, lm
generateTriples	rm, db, u, p, u
findFieldataType	u, v
deleteData	
insertData	
updateData	
getDb	db
setDb	db
getUtil	u
setUtil	u
getResultModel	rm
setResultModel	rm
getProperties	p
setProperties	p

Al aplicar esta métrica a la clase *Generator* se obtuvo que su nivel de CCM es de 5. Este valor representa el 33% que, según lo planteado anteriormente, se considera un valor aceptable. El CCM de las restantes clases posee un comportamiento similar, siendo aceptable su valor. Este resultado implica la disminución de la complejidad de las clases, favoreciendo la alta cohesión.

3.3. Pruebas de software

3.3.1. Pruebas de caja blanca

Las pruebas de caja blanca se aplicaron haciendo uso de la técnica del camino básico, con el objetivo de evaluar la complejidad lógica de un diseño procedimental y usar esta medida como guía para la definición de un conjunto básico de caminos de ejecución (47). Esta prueba permite garantizar que en los casos de prueba obtenidos a través del camino básico se ejecute cada sentencia del programa por lo menos una vez.

El uso de esta técnica es mostrado en el ejemplo siguiente. Se analizan y enumeran las sentencias de código del método ***deleteData*** contenidas en la clase ***Generator***.

```
public void deleteData(ResultSet tuple, String endpoint_query, String endpoint_update) {
    String query = null; //1
    try { //2
        query = "SELECT ?s WHERE { ?s ?p \"" + tuple.getString(2) + " #" + tuple.getString(3) + "\" }"; //3
    } catch (SQLException ex) { //4
        java.util.logging.Logger.getLogger(Generator.class.getName()).log(Level.SEVERE, null, ex); //5
    }
    QueryExecution qe = QueryExecutionFactory.sparqlService(endpoint_query, query); //6
    com.hp.hpl.jena.query.ResultSet execSelect = qe.execSelect();
    RDFNode s;
    while (execSelect.hasNext()) { //7
        QuerySolution qs = execSelect.next(); //8
        s = qs.get("s");
        String delete = "DELETE WHERE {<" + s.toString() + "> ?p ?o}";
        UpdateRequest updateInsert = UpdateFactory.create(delete);
        UpdateProcessRemote upr = (UpdateProcessRemote) UpdateExecutionFactory.createRemote(updateInsert, endpoint_update);
        upr.execute(); //9
    }

} //10
```

Ilustración V: *Función que elimina tripletas del grafo RDF mediante SPARQL*

Luego, es representado el grafo de flujo, asociado al código anterior, a través de nodos, aristas y regiones.

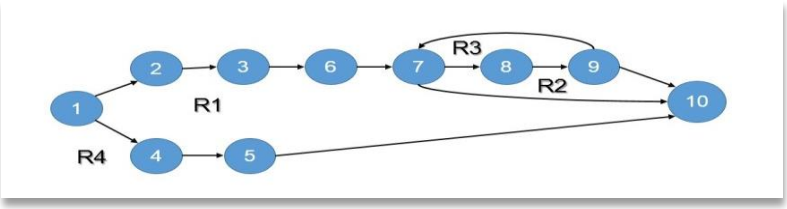

Ilustración VI: *Grafo de flujo asociado al método* ***deleteData***

Cálculo de la complejidad ciclomática para el grafo de flujo anterior

V(g)=(n-a)+2 *V(g)=(p+1)* *V(g)=r*

Donde "*V(g)*" es el valor de la complejidad ciclomática, "*a*" la cantidad total de aristas, "*n*" la cantidad total de nodos, "*p*" la cantidad total de nodos predicados (nodos de los cuales parten dos o más aristas) y "*r*" la cantidad total de regiones. Al evaluar las fórmulas anteriores se obtiene una complejidad ciclomática igual a 4. Este valor representa el número mínimo de casos de prueba para para el procedimiento tratado. Son mostrados además, los caminos básicos.

V(g)=(12-10)+2=4 V(g)=3+1=4 V(g)=4

Camino básico #1: 1-2-3-6-7-8-9-10 Camino básico #2: 1-2-3-6-7-8-9-7-10

Camino básico #3: 1-2-3-6-7-10 Camino básico #4: 1-4-5-10

Se ejecutan los casos de prueba para cada camino básico determinado en el grafo de flujo.

Tabla XXXVIII: *Caso de Prueba para el Camino Básico #1*

CASO DE PRUEBA CAMINO BÁSICO #1	
Descripción:	Los datos de entrada cumplirán los siguientes requisitos: -Variable "tuple" debe ser una tupla de la BDR -Variables "endpoint_query" y "endpoint_update" deben ser URL[31] correspondientes a los servicios de consultas y modificación respectivamente.
Condición de ejecución:	La tupla debe existir en la BDR y de ella solo se obtiene una tripleta RDF
Entrada:	endpoint_query: "http://localhost:3030/metharto/query" endpoint_update: "http://localhost:3030/metharto/update"
Resultados esperados:	Es eliminada del grafo RDF la tripleta correspondiente a la variable "tuple"
Resultado:	Satisfactorio

Tabla XXXIX: *Caso de Prueba para el Camino Básico #2*

CASO DE PRUEBA CAMINO BÁSICO #2	
Descripción:	Los datos de entrada cumplirán los siguientes requisitos: -Variable "tuple" debe ser una tupla de la BDR

[31] Localizador uniforme de recursos (del inglés *uniform resource locator*)

	-Variables “endpoint_query” y “endpoint_update” deben ser URL correspondientes a los servicios de consultas y modificación respectivamente.
Condición de ejecución:	La tupla debe existir en la BDR y de ella se obtiene más de una tripleta RDF
Entrada:	endpoint_query: “http://localhost:3030/metharto/query” endpoint_update: “http://localhost:3030/metharto/update”
Resultados esperados:	Son eliminadas del grafo RDF las tripletas correspondientes a la variable “tuple”
Resultado:	Satisfactorio

Tabla XL: *Caso de Prueba para el Camino Básico #3*

CASO DE PRUEBA CAMINO BÁSICO #3	
Descripción:	Los datos de entrada cumplirán los siguientes requisitos: -Variable “tuple” debe ser una tupla de la BDR -Variables “endpoint_query” y “endpoint_update” deben ser URL correspondientes a los servicios de consultas y modificación respectivamente.
Condición de ejecución:	La tupla debe existir en la BDR y de ella no se obtiene ninguna tripleta RDF
Entrada:	endpoint_query: “http://localhost:3030/metharto/query” endpoint_update: “http://localhost:3030/metharto/update”
Resultados esperados:	No se realiza ninguna eliminación en el grafo RDF
Resultado:	Satisfactorio

Tabla XLI: *Caso de Prueba para el Camino Básico #4*

CASO DE PRUEBA CAMINO BÁSICO #4	
Descripción:	Los datos de entrada cumplirán los siguientes requisitos: -Variable “tuple” debe ser una tupla de la BDR -Variables “endpoint_query” y “endpoint_update” deben ser URL correspondientes a los servicios de consultas y modificación respectivamente.
Condición de ejecución:	La tupla no debe existir en la BDR
Entrada:	endpoint_query: “http://localhost:3030/metharto/query”

	endpoint_update: “http://localhost:3030/metharto/update”
Resultados esperados:	No se realiza ninguna eliminación en el grafo RDF
Resultado:	Satisfactorio

3.3.2. Pruebas de aceptación

Estas pruebas las realiza el cliente. En este caso el cliente es el Grupo de Investigación de Web Semántica. Son pruebas funcionales, sobre el sistema, y buscan una cobertura de las HU. Estas pruebas no se realizan durante el desarrollo, pues no se podrían presentar al cliente; sino que se ejecutan sobre el producto terminado e integrado o pudiera ser una versión del producto o una iteración funcionad pactada previamente con el cliente.

La experiencia muestra que aún después del más cuidadoso proceso de pruebas por parte del desarrollador, quedan una serie de errores que aparecen cuando el cliente comienza a usarlo, por lo que las pruebas de aceptación tienen un mayor grado de importancia que las pruebas unitarias (51).

Una prueba de aceptación es como una prueba de caja negra. Cada una de ellas representa una salida esperada del sistema. Es responsabilidad del cliente verificar la corrección de las pruebas de aceptación y tomar decisiones acerca de las mismas.

En este trabajo son sometidas a pruebas de aceptación las tres HU definidas en el capítulo anterior, UpRDF-01, UpRDF-02 y UpRDF-03. Para la HU UpRDF-02 se realizan dos casos de pruebas, por ser esta HU la de más complejidad. Los casos de pruebas se muestran a continuación en las siguientes tablas.

Tabla XLII: *Caso de Prueba de Aceptación CPA-01*

CASO DE PRUEBA DE ACEPTACIÓN	
Código: CPA-01	**Historia de Usuario:** UpRDF-01
Nombre: Caso de Prueba Detectar cambios en la BDR	
Descripción: Prueba la funcionalidad de detectar cambios en la BDR al ser ejecutada la herramienta *Metharto*, que recolecta metadatos bibliográficos en la BDR. Los datos de los cambios detectados deben estar almacenados en la tabla *check_updates*.	
Condiciones de Ejecución: Debe estar creada la estructura de la tabla *check_updates* en la BDR.	

Deben estar creados los *triggers*, en las tablas del modelo que se representarán en el modelo basado en grafos RDF.
Entrada/Pasos de Ejecución: Se ejecuta la herramienta Metharto.
Resultado Esperado: La tabla *check_updates* de la BDR contiene los datos relacionados con las tuplas que fueron insertadas, modificadas o eliminadas.

Evaluación de la Prueba:	Satisfactoria

Tabla XLIII: *Caso de Prueba de Aceptación CPA-02*

CASO DE PRUEBA DE ACEPTACIÓN	
Código: CPA-02	**Historia de Usuario:** UpRDF-02
Nombre: Caso de Prueba Generar grafo RDF	
Descripción: Prueba la funcionalidad de generar el grafo RDF a partir del documento de alineación (*mapping.ttl*), y los datos de la BDR.	
Condiciones de Ejecución: Se debe especificar el archivo *mapping.ttl* generado. Se debe especificar el formato del grafo a generar (*RDF/XML, rdf*). Se deben especificar los datos para establecer la conexión con la BDR.	
Entrada/Pasos de Ejecución: Se ejecuta en un terminal el código de ejecución que realiza esta tarea.	
Resultado Esperado: El grafo es generado en la ruta especificada, con el formato especificado.	
Evaluación de la Prueba:	Satisfactoria

Tabla XLIV: *Caso de Prueba de Aceptación CPA-03*

CASO DE PRUEBA DE ACEPTACIÓN	
Código: CPA-03	**Historia de Usuario:** UpRDF-02
Nombre: Caso de Prueba Generar tripletas RDF a partir de los cambios detectados en la BDR	
Descripción: Prueba la funcionalidad de generar tripletas RDF a partir del documento de alineación (*mapping.ttl*), y los datos referentes a los cambios detectados en la BDR.	
Condiciones de Ejecución: Se debe especificar el archivo *mapping.ttl* generado.	

Debe estar generado un grafo RDF con todos los datos de la BDR, los cuales no contemplan los cambios. Se deben especificar los datos para establecer la conexión con la BDR. Debe estar creada la estructura para la detección de cambios en la BDR.
Entrada/Pasos de Ejecución: Se ejecuta en un terminal el código de ejecución que realiza la tarea de generar el grafo RDF.
Resultado Esperado: Se genera un conjunto de tripletas en un grafo temporal.
Evaluación de la Prueba: Satisfactoria

Tabla XLV: *Caso de Prueba de Aceptación CPA-04*

CASO DE PRUEBA DE ACEPTACIÓN	
Código: CPA-04	**Historia de Usuario:** UpRDF-03
Nombre: Caso de Prueba Actualizar grafo RDF	
Descripción: Prueba la funcionalidad de actualizar el grafo RDF de forma incremental.	
Condiciones de Ejecución: Debe estar generado un grafo RDF con todos los datos de la BDR, los cuales no contemplan los cambios. Debe estar generado un grafo RDF temporal con las tripletas generadas a partir de los cambios detectados en la BDR.	
Entrada/Pasos de Ejecución: Se ejecuta en un terminal el código de ejecución que realiza la tarea de generar el grafo RDF.	
Resultado Esperado: Se actualiza el grafo RDF anteriormente generado sin ser generado nuevamente en su totalidad.	
Evaluación de la Prueba:	Satisfactoria

3.3.2.1. Resultados

Se realizaron un total de 20 casos de prueba de aceptación (caja negra), de los cuales 7 resultaron no satisfactorios, representando el 35% de total de los casos de prueba. Los errores detectados por los casos de prueba fueron mitigados luego de dos iteraciones de prueba.

Gráfico IV: *Resultados de las Pruebas de Aceptación*

3.4. Caso de estudio

Con el objetivo de validar que se ha dado solución al problema de investigación se diseñan casos de estudio. En esta sección se ponen de manifiesto las variables y su operacionalización, elementos definidos anteriormente en la introducción. Las variables hacen referencia al costo en tiempo y en recursos computacionales (uso del CPU[32] y memoria RAM), como variables dependientes y a la actualización incremental como variable independiente. Las variables dependientes hacen referencia al rendimiento del sistema. El tipo de experimento a utilizar es el experimento puro, ya que se distingue de los preexperimentos y los cuaxiexperimentos del control de la situación experimental. Existe una manipulación intencional de la variable independiente, midiendo su efecto sobre las dependientes (52). Para evaluar el rendimiento del sistema se hace uso de la herramienta VisualVM[33].

VisualVM es una herramienta que, de forma gráfica, permite monitorizar la máquina virtual que se esté utilizando, así como los distintos procesos java que se estén ejecutando sobre ella. La utilidad de esta herramienta se centra en conocer el uso de recursos tales como memoria RAM y CPU. Estas operaciones pueden ser medidas y visualizadas ya que la herramienta genera gráficas en tiempo real (53).

3.4.1. Descripción

La BDR utilizada para esta prueba es *Metharto*. Contiene los metadatos bibliográficos obtenidos de 7 revistas científicas. Para el desarrollo del caso de estudio se cuenta con una computadora con las siguientes prestaciones:

[32] *Central Processing Unit* (Unidad Central de Procesamiento)
[33] http://visualvm.java.net/

- Tipo de CPU: Mobile DualCore Intel Core i3-380M, 2533 MHz.
- Memoria del sistema: 5814 MB (DDR3-1333 DDR3 SDRAM).

Se proponen los siguientes escenarios para la evaluación.

1. Generar grafo RDF en su totalidad sin usar el método propuesto.
2. Actualizar grafo RDF de forma incremental usando el método propuesto.

Se tienen en cuenta dos variantes.

1. La cantidad de cambios es menor que la mitad de datos contenidos en la BDR (menor que un 50%).
2. La cantidad de cambios es mayor que la mitad de datos contenidos en la BDR (mayor que un 50%).

Primeramente se genera el grafo RDF en su totalidad. Para cada escenario se evalúan la actualización del grafo, que implica generarlo nuevamente en su totalidad y la actualización incremental, utilizando la solución propuesta. Se miden las variables definidas: tiempo (milisegundos), uso del CPU (porciento) y uso de memoria RAM (porciento). Además, se tienen en cuenta la cantidad de tripletas generadas. La herramienta VisualVM genera gráficas que van mostrando las lecturas del uso del CPU y la memoria RAM cada 1000 milisegundos. Para el resultado final se calculó el promedio de la cantidad de lecturas que realiza la herramienta.

Escenario 1

En esta evaluación no se tiene en cuenta la cantidad de cambios pues se genera el grafo RDF en su totalidad obteniendo los datos directamente de la BDR. Se obtuvieron los siguientes resultados.

Tabla XLVI: *Evaluación del Escenario 1*

	Tiempo de respuesta (ms)	Uso del CPU (%)	Uso de la memoria RAM (%)	Cantidad de tripletas generadas
Escenario 1	13173	58.5	9	51329

El tiempo de respuesta y la cantidad de tripletas generadas se obtiene de la consola donde se ejecuta la aplicación. Las siguientes ilustraciones muestran los datos recogidos en la tabla anterior (uso del CPU y uso de la memoria RAM).

En la **Ilustración VII** a la izquierda se observa en color naranja la lectura del uso del CPU realizada por la herramienta. Mientras que a la derecha se observa el uso de la memoria RAM en color azul, el color naranja representa la memoria asignada de un máximo de 1 524 629 504B.

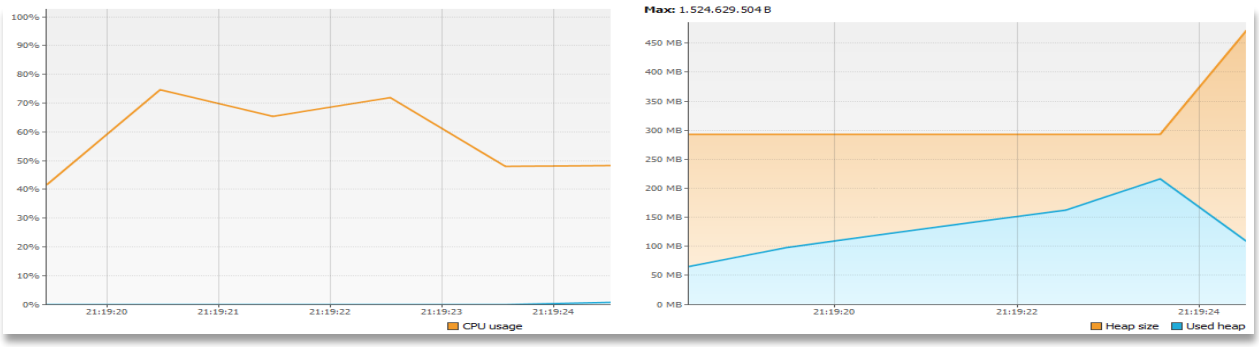

Ilustración VII: *Uso del CPU y memoria RAM del Escenario 1.*

Escenario 2 – Variante 1

En esta evaluación se cuenta con el grafo generado en su totalidad previamente. Se actualizará de forma incremental utilizando el método propuesto en este trabajo.

Tabla XLVII: *Evaluación del Escenario 2 - Variante 1*

	Tiempo de respuesta (ms)	Uso del CPU (%)	Uso de la memoria RAM (%)	Cantidad de tripletas generadas
Escenario 2 – Variante 1	5844	35.1	2.7	5927

Se puede observar que el tiempo de respuesta, con respecto al Escenario, 1 se redujo en un 56%, el uso del CPU se redujo en un 40%, el uso de la memoria RAM se redujo en un 70%. Nótese que la cantidad de cambios es menor que el 50% con respecto al Escenario 1. Las siguientes ilustraciones muestran los datos recogidos en la tabla anterior (uso del CPU y uso de la memoria).

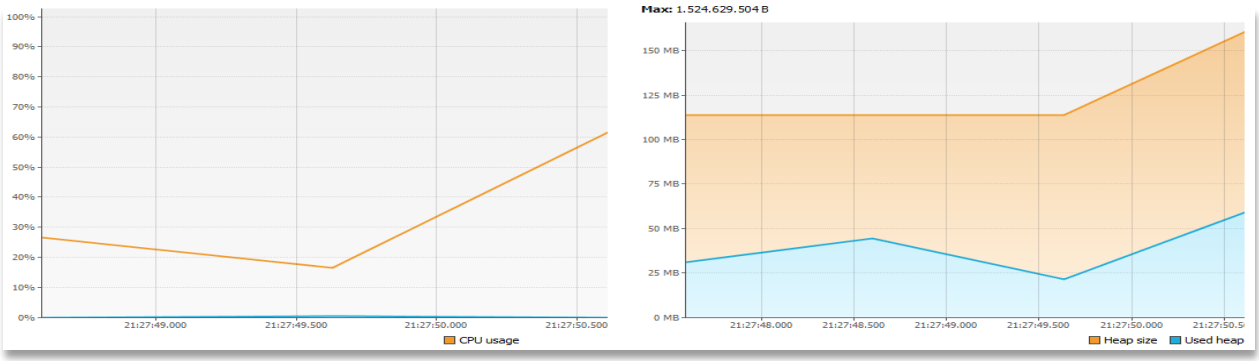

Ilustración VIII: *Uso del CPU y memoria RAM del Escenario 2 - Variante 1*

Escenario 2 – Variante 2

En esta evaluación no se cuenta con el grafo generado en su totalidad previamente. Se actualizará de forma incremental, utilizando el método propuesto en este trabajo, un grafo RDF en su totalidad.

Tabla XLVIII: *Evaluación del Escenario 2 - Variante 2*

	Tiempo de respuesta (ms)	Uso del CPU (%)	Uso de la memoria RAM (%)	Cantidad de tripletas generadas
Escenario 2 – Variante 2	29329	14.22	7.16	44886

Se puede observar que el tiempo de respuesta, con respecto al Escenario 1, aumentó en un 122%, mientras que el uso del CPU se redujo en un 76% y el uso de la memoria RAM se redujo en un 16%. Nótese que la cantidad de cambios supera el 50% con respecto al Escenario 1. Las siguientes ilustraciones muestran los datos recogidos en la tabla anterior (uso del CPU y uso de la memoria RAM).

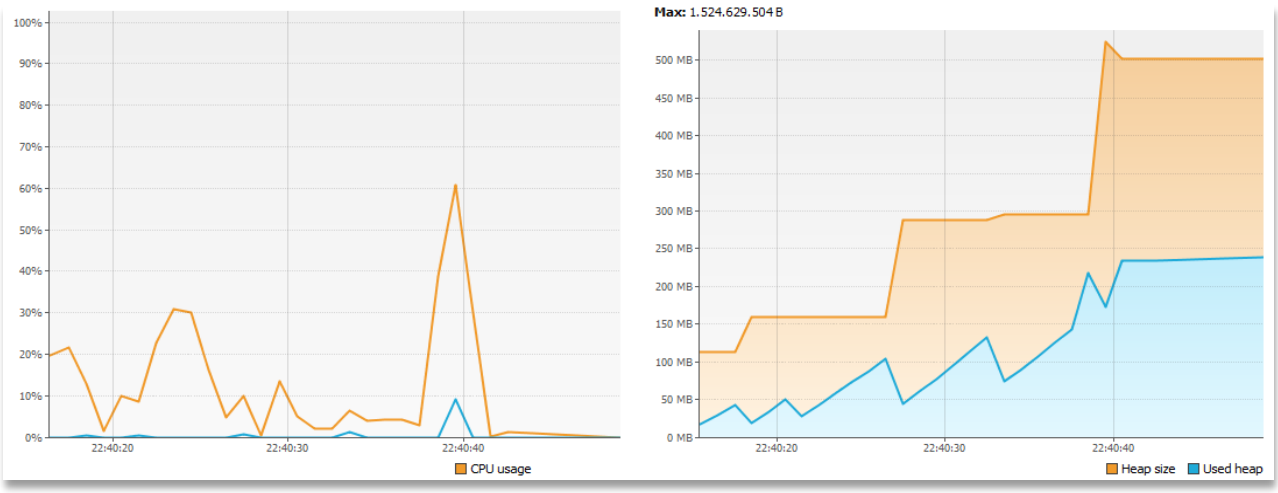

Ilustración IX: *Uso del CPU y memoria RAM del Escenario 2 - Variante 2*

3.4.2. Análisis de los resultados

El siguiente gráfico muestra un resumen de las medidas de la variable tiempo de respuesta, medida en milisegundos, utilizada en el caso de estudio.

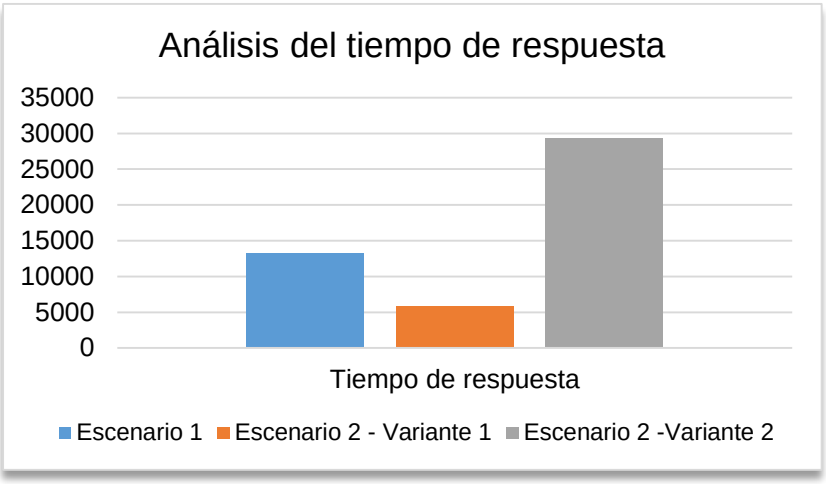

Gráfico V: *Análisis de la variable tiempo de respuesta*

Utilizando la actualización incremental propuesta en el método de solución se obtiene una reducción del 56% del consumo del tiempo. Según lo planteado en la operacionalización de las variables en la introducción, se considera que, al ser mayor que el 41%, la propuesta de solución es válida para para la actualización incremental, cuando la cantidad de cambios no superan el 50% de la cantidad de datos contenidos en la BDR. Por otra parte, si la cantidad de cambios superan el 50%, el consumo de tiempo es mucho mayor. Estos aspectos son considerados satisfactorios, ya que una vez generado el grafo en su totalidad, las actualizaciones en el mismo no superan, usualmente, el 50%.

El siguiente gráfico muestra un resumen de las variables usos del CPU y memoria RAM, medidas en porciento, utilizadas en el caso de estudio.

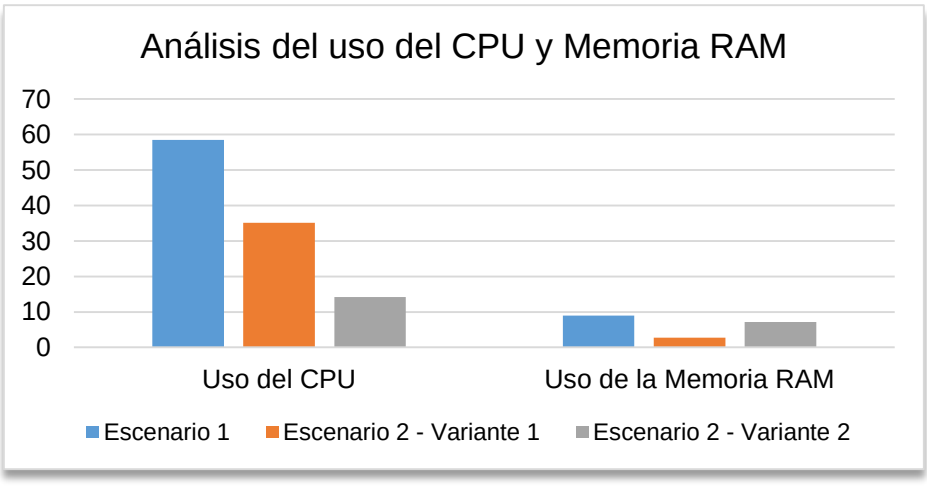

Gráfico VI: *Análisis de las variables uso del CPU y Memoria RAM*

Utilizando la actualización incremental propuesta en el método de solución se obtiene una reducción del 40% del uso del CPU. Según lo planteado en la operacionalización de las variables en la introducción, se considera que, al ser mayor que el 20%, la propuesta de solución es satisfactoria para la actualización incremental, cuando la cantidad de cambios no supera el 50% de la cantidad de datos contenidos en la BDR. Por otra parte, si la cantidad de cambios supera el 50%, el uso del CPU se reduce en un 76%, siendo, la propuesta de solución, satisfactoria también en este caso. En el caso del uso de la memoria RAM la reducción es de un 70% y un 16% para las variantes 1 y 2 respectivamente. Para el uso de la memoria RAM, la propuesta de solución se considera satisfactoria ya que la reducción supera el 10%. Estos aspectos son considerados positivos, ya que con la solución propuesta se evidencia la reducción del consumo de recursos computacionales en cualquier escenario (uso del CPU y memoria RAM).

Análisis de la actualización incremental

Utilizando el método propuesto para la actualización incremental de grafos RDF, se reducen los costos en tiempo y recursos computaciones, cuando las actualizaciones no están asociadas a una cantidad de cambios mayor que el 50% de la cantidad de datos contenidos en la BDR. Para esta variante, las variables tiempo y recursos computacionales, son directamente proporcionales: una reducción en el consumo del tiempo implica una reducción en el consumo de recursos computacionales. En caso contrario las variables son inversamente proporcionales: un aumento en el consumo del tiempo implica una reducción en el consumo de recursos computacionales.

El método propuesto permite utilizar la actualización incremental o no, según el escenario en que se encuentre. Esta variante es utilizada por el cliente haciendo uso de la propiedad "*incremental*" del archivo "*r2rml.properties*". En cualquier variante se reduce el costo de recursos computacionales, siendo este un aspecto positivo.

Se recomienda generar el grafo RDF en su totalidad directamente de la BDR, sin hacer uso de los cambios detectados, la primera vez (Escenario 1). Luego, los cambios detectados en la BDR, usualmente no superan el 50%. Esto valida la propuesta de solución de forma satisfactoria.

3.5. Conclusiones parciales

Las pruebas realizadas detectaron casos de prueba no satisfactorios, siendo mitigados tras dos iteraciones de pruebas.

El análisis de la actualización incremental demostró que el método propuesto reduce el costo de recursos computacionales. Además reduce el costo en tiempo cuando los cambios detectados no superan el 50% de los datos contenidos en la BDR.

CONCLUSIONES GENERALES

La revisión de la literatura evidenció que las aproximaciones existentes para la actualización incremental de grafos RDF poseen características tales como: no brindan un método genérico para su desarrollo, no utilizan lenguajes de alineación estandarizados por el W3C, altos consumos de tiempo y recursos computacionales.

Existen dos variantes para detectar cambios en BDR (*logs* o *triggers*), dos lenguajes de alineación BDR – RDF estandarizados por el W3C (*Direct Mapping* y R2RML) y un lenguaje de consulta para grafos RDF, estandarizado por el W3C. Lo anterior permitió determinar que el método a desarrollar debía utilizar *triggers* para la detección de cambios, el lenguaje R2RML para la alineación y SPARQL 1.1 Update para la consulta de grafos RDF.

A diferencia de las aproximaciones revisadas, la solución propone un método basado en tres fases para la actualización incremental de grafos RDF. Su implementación utiliza estándares del W3C haciendo uso de una arquitectura basada en tuberías y filtros.

La propuesta de solución disminuye el consumo de recursos computacionales con respecto a la actualización total. Lo anterior fue corroborado por el análisis del uso del CPU y memoria RAM.

Existen dos escenarios para la actualización incremental de grafos RDF. En el primero, los cambios detectados no superan el 50% de la cantidad total de datos de la BDR. En este escenario los costos de tiempo y recursos computaciones son directamente proporcionales. En el segundo escenario, los cambios detectados superan el 50% de la cantidad total de datos de la BDR. El costo de tiempo y recursos computacionales son inversamente proporcionales.

RECOMENDACIONES

El método de solución propuesto depende de la estructura de la BDR que se utilice y requiere la manipulación de la misma. Por esta razón, se recomienda la implementación de un componente que permita crear la estructura de la tabla que contiene los datos referentes a los cambios detectados y la creación de los *triggers* asociados a las tablas que se quieran representar en el modelo ontológico de forma automática.

REFERENCIAS BIBLIOGRÁFICAS

1. DOMINGUE, John, FENSEL, Dieter y HENDLER, James A. *Handbook of semantic web technologies*. Springer, 2011.

2. HIDALGO, Yusniel y RODRÍGUEZ, Rafael. La web semántica: una breve revisión. *Revista Cubana de Ciencias Informáticas*. 2013. Vol. 7, no. 1, p. 76–85.

3. HERT, Matthias, REIF, Gerald y GALL, Harald C. A comparison of RDB-to-RDF mapping languages. En: *Proceedings of the 7th International Conference on Semantic Systems* [En línea]. ACM, 2011. p. 25–32. Disponible en: http://dl.acm.org/citation.cfm?id=2063522

4. BERNERS-LEE, Tim, HENDLER, James y LASSILA, Ora. The semantic web. *Scientific American*. 2001. Vol. 284, no. 5, p. 28–37.

5. HEATH, Tom y BIZER, Christian. Linked data: Evolving the web into a global data space. *Synthesis lectures on the semantic web: theory and technology*. 2011. Vol. 1, no. 1, p. 1–136.

6. BERNERS-LEE, Tim. Linked Data - Design Issues. [En línea]. 2006. Disponible en: http://www.w3.org/DesignIssues/LinkedData.html

7. HAYES, Patrick J. y PATEL-SCHNEIDER, Peter F. RDF 1.1 Semantics. [En línea]. 25 February 2014. Disponible en: http://www.w3.org/TR/2014/REC-rdf11-mt-20140225/

8. HIDALGO, Yusniel, REYES, Liudmila, LEIVA, Amed, ROLDÁN, María del Mar y ALDANA, José F. BM2LOD: Platform For Publishing Bibliographic Data As Linked Open Data. *Proceedings of 7th IADIS International Conference on Information Systems 2014, IADIS Press* [En línea]. 2014. Disponible en: http://www.bibsonomy.org/bibtex/22ad1f39200ebf13fa9b3454b781383ab/yhdelgado

9. HIDALGO, Yusniel, RODRÍGUEZ PUENTE, Rafael, ORTIZ MUÑOZ, Ernesto y ALONSO SIERRA, Luis E. HERRAMIENTA PARA LA RECOLECCIÓN DE METADATOS BIBLIOGRÁFICOS MEDIANTE EL PROTOCOLO OAI-PMH. [En línea]. 2013. Disponible en: http://www.researchgate.net/publication/235934068_HERRAMIENTA_PARA_LA_RECOLECCIN_DE_METADATOS_BIBLIOGRFICOS_MEDIANTE_EL_PROTOCOLO_OAI-PMH__TOOL_FOR_BIBLIOGRAPHIC_METADATA_HARVESTING_THROUGH_OF_THE_OAI-PMH_PROTOCOL/file/79e4151473f74eadc0.pdf

10. BARZDINS, Janis y KIRIKOVA, Marite. *Databases and Information Systems VI: Selected Papers from the Ninth International Baltic Conference, DB&IS 2010* [En línea]. IOS Press, 2011. Disponible en: http://books.google.com/books?hl=es&lr=&id=7fYS29re860C&oi=fnd&pg=PR1&dq=++Databases+and+Information+Systems++Janis+Barzdins,+Marite+Kirikova&ots=6ZYTXggnSM&sig=z-y84G39fWJiMc6DpicYwaLIKWg

11. FERNÁNDEZ, Mariano y CORCHO, Oscar. *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. First Edition*. Springer, 2004. ISBN 9781852335519.

12. STUDER, Rudi y BENJAMINS, V. Richard. *Dieter Fense "Knowledge Engineering Principles and Methods."*2007.

13. RODRÍGUEZ, Eilys Pacheco, ESTRADA, Yuney Martínez y MARTÍNEZ, Nemury Silega. PROPUESTA METODOLÓGICA PARA EL DESARROLLO DE UNA ONTOLOGÍA PARA LA CLASIFICACIÓN DE MERCANCÍAS EN LA ADUANA GENERAL DE LA REPÚBLICA DE CUBA. *Iberoamerican Journal of Project Management*. 20 December 2013. Vol. 4, no. 2, p. 01–14.

14. GUARINO, Nicola. *Formal Ontology in Information Systems: Proceedings of the First International Conference (FOIS'98), June 6-8, Trento, Italy*. IOS Press, 1998. ISBN 9789051993998.

15. MCGUINNESS, Deborah L. y VAN HARMELEN, Frank. OWL Web Ontology Language Overview. W3C Recommendation, February 2004. *See http://www.w3.org/TR/owl-features*. 2004.

16. DUCHARME, Bob. *Learning Sparql* [En línea]. O'Reilly Media, Inc., 2013. Disponible en: http://books.google.com/books?hl=es&lr=&id=j2kXeNeZ00YC&oi=fnd&pg=PR7&dq=+DuCharme,+Bob.+Learning+SPARQL&ots=-F3FhelkPo&sig=O9v1Cqmj_RtVebdxx6TSSYdPf5Q

17. CIFUENTE, Francisco Adolfo. Publicación de datos abiertos enlazados en el ámbito legislativo. *Trabajo de fin de Máster* [En línea]. Universidad de Oviedo. 2011. Disponible en: http://www.slideshare.net/francisco.cifuentes/presentacion-tfm

18. RAMANUJAM, Sunitha, KHADILKAR, Vaibhav, KHAN, Latifur, SEIDA, Steven, KANTARCIOGLU, Murat y THURAISINGHAM, Bhavani. Bi-directional translation of relational data into virtual RDF stores. En: *Semantic Computing (ICSC), 2010 IEEE Fourth International Conference on* [En línea]. IEEE, 2010. p. 268–276. Disponible en: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5629266

19. DAS, S., SUNDARA, S. y CYGANIAK, R. *R2RML: RDB to RDF Mapping Language, W3C Recommendation 27 September 2012*. 2013.

20. PRIYATNA, Freddy, CORCHO, Oscar y SEQUEDA, Juan. Formalisation and Experiences of R2RML-based SPARQL to SQL Query Translation Using Morph. En: *Proceedings of the 23rd International Conference on World Wide Web* [online]. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2014. p. 479–490. WWW '14. ISBN 978-1-4503-2744-2. Disponible en: http://dx.doi.org/10.1145/2566486.2567981

21. VirtuosoRdfViews - semanticweb.org. [En línea]. Disponible en: http://semanticweb.org/wiki/VirtuosoRdfViews

22. DE LA VEGA, Erik. Control de cambios y gestión de la configuración de la base de datos. *Serie Científica* [En línea]. 2008. Vol. 1, no. 8. Disponible en: https://publicaciones.uci.cu/index.php/SC/article/view/77/0

23. KANG, Lee, JIN, Eun Sook, YUN, Dongsik y JEONG, Jaewoo. A Replication Architecture of Central Inventory Database. [En línea]. 2006. Disponible en: http://doi.ieeecomputersociety.org/10.1109/AICT-ICIW.2006.29

24. El registro de transacciones (SQL Server). [En línea]. Disponible en: http://msdn.microsoft.com/es-es/library/ms190925.aspx

25. CERI, Stefano, COCHRANE, Roberta y WIDOM, Jennifer. Practical applications of triggers and constraints: Successes and lingering issues. En: *VLDB* [En línea]. 2000. p. 10–14. Disponible en: http://dbpubs.stanford.edu/pub/showDoc.Fulltext?lang=en&doc=2000-26&format=pdf&compression=&name=2000-26.pdf

26. HALPIN, H. y HERMAN, I. *RDB2RDF Working Group Charter*. 2011.

27. A Direct Mapping of Relational Data to RDF. [En línea]. Disponible en: http://www.w3.org/TR/rdb-direct-mapping/

28. JALALI, Vahid, ZHOU, Mo y WU, Yuqing. A study of RDB-based RDF data management techniques. En: *Web-Age Information Management* [En línea]. Springer, 2011. p. 366–378. Disponible en: http://link.springer.com/chapter/10.1007/978-3-642-23535-1_32

29. BIZER, C., CYGANIAK, R., GARBERS, J., MARESCH, O. y BECKER, C. *The D2RQ Platform v0. 7-Treating Non-RDF Relational Databases as Virtual RDF Graphs-User Manual and Language Specification*. 2009.

30. AUER, Sören, DIETZOLD, Sebastian, LEHMANN, Jens, HELLMANN, Sebastian y AUMUELLER, David. Triplify: light-weight linked data publication from relational databases. En: *Proceedings of the 18th international conference on World wide web* [En línea]. ACM, 2009. p. 621–630. Disponible en: http://dl.acm.org/citation.cfm?id=1526793

31. SEQUEDA, Juan F. On the Semantics of R2RML and its Relationship with the Direct Mapping. [En línea]. Disponible en: http://m.iswc2013.semanticweb.org/sites/default/files/iswc_poster_4.pdf

32. DABROWSKI, Maciej, GRIFFIN, Keith y PASSANT, Alexandre. Approaches for real-time integration of semantic web data in distributed enterprise systems. En: *Semantic Computing (ICSC), 2011 Fifth IEEE International Conference on* [En línea]. IEEE, 2011. p. 47–50. Disponible en: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6061435

33. LETELIER, Patricio. Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). [En línea]. 15 April 2006. Disponible en: http://www.cyta.com.ar/ta0502/b_v5n2a1.htm

34. CERVONE, H. Frank. Understanding agile project management methods using Scrum. *OCLC Systems & Services*. 15 February 2011. Vol. 27, no. 1, p. 18–22. DOI 10.1108/10650751111106528.

35. WILLIAMS, Laurie. Agile Software Development Methodologies y Practices. En: ZELKOWITZ, Marvin V. (ed.), *Advances in Computers* [En línea]. Elsevier, 2010. p. 1–44. Advances in Computers. ISBN 0065-2458. Disponible en: http://www.sciencedirect.com/science/article/pii/S0065245810800014

36. DINGSØYR, Torgeir, NERUR, Sridhar, BALIJEPALLY, VenuGopal y MOE, Nils Brede. A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software*. June 2012. Vol. 85, no. 6, p. 1213–1221. DOI 10.1016/j.jss.2012.02.033.

37. CHOW, Tsun y CAO, Dac-Buu. A survey study of critical success factors in agile software projects. *Journal of Systems and Software*. June 2008. Vol. 81, no. 6, p. 961–971. DOI 10.1016/j.jss.2007.08.020.

38. PENADÉS, Mª Carmen y TORRES, Patricio Orlando Letelier. Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). *Técnica administrativa*. 2006. Vol. 5, no. 26, p. 1–.

39. BECK, Kent. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional, 2000. ISBN 9780201616415.

40. PostgreSQL: The world's most advanced open source database. [En línea]. Disponible en: http://www.postgresql.org/

41. Welcome to JavaWorld.com. [En línea]. Disponible en: http://www.javaworld.com/

42. BCK, Heiko. *The Definitive Guide to NetBeans Platform 7*. 1st. Berkely, CA, USA : Apress, 2011. ISBN 1430241012, 9781430241010.

43. NetBeans IDE - Overview. [En línea]. Disponible en: https://netbeans.org/features/index.html

44. OpenRDF.org: Home. [En línea]. Disponible en: http://www.openrdf.org/

45. Apache Jena - An Introduction to RDF and the Jena RDF API. [En línea]. Disponible en: https://jena.apache.org/tutorials/rdf_api.html

46. RDB2RDF Implementation Report. [En línea]. Disponible en: http://www.w3.org/TR/rdb2rdf-implementations/

47. PRESSMAN, Roger S. *Ingeniería del Software: Un enfoque práctico*. 5a edición. España: McGraw Hill, 2002.

48. BECK, Kent. A Laboratory for Teaching Object-Oriented Thinking. En: *OOPSLA'89 Conference Proceedings* [En línea]. New Orleans, Louisiana: SIGPLAN Notices, October 1989. Disponible en: http://c2.com/doc/oopsla89/paper.html

49. LORENZ, Mark y KIDD, Jeff. *Object-oriented Software Metrics: A Practical Guide*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1994. ISBN 0-13-179292-X.

50. GEORGE, Soumya, AVINASH, Sivapriya y ABRAHAM, John T. Object Oriented Design Metrics. [En línea]. 21 January 2014. Disponible en: http://csidl.org/xmlui/handle/123456789/603

51. HEREDIA, Javier, ALMANZA, Lilián Álvarez y PONS, Naryana Linares. Comparación y tendencias entre metodologías ágiles y formales. Metodología utilizada en el Centro de Informatización para la Gestión de Entidades. *Serie Científica* [En línea]. 11 October 2011. Vol. 4, no. 10. Disponible en: https://publicaciones.uci.cu/index.php/SC/article/view/484

52. DZUL ESCAMILLA, Marisela. Diseños de la investigación. [En línea]. December 2013. Disponible en: http://repository.uaeh.edu.mx/bitstream/handle/123456789/14915

53. MINELLA, Michael T. Scaling and Tuning. En: *Pro Spring Batch* [En línea]. Apress, 2011. p. 387–445. ISBN 978-1-4302-3452-4, 978-1-4302-3453-1. Disponible en: http://link.springer.com/chapter/10.1007/978-1-4302-3453-1_11