UNIVERSITAT OBERTA DE CATALUNYA

Ingeniería en Informática

Estudio de los lenguajes de consulta para documentos RDF

Alumno: Antonio Rodil Garrido
Dirigido por: Óscar Celma Herrada

Enero de 2006

Resumen

Estudio de los lenguajes de consulta para documentos RDF

Con este estudio se pretende introducir al lector en diversas tecnologías relacionadas con el concepto de Web Semántica. El estudio se inicia con una breve introducción sobre lo que hoy en día es la Web Semántica. Posteriormente, este trabajo se centra en un análisis de los distintos lenguajes de consulta para documentos RDF. La elección de uno u otro lenguaje, puede determinar también el uso de otras tecnologías, como los sistemas de gestión de bases de datos para documentos RDF. De ahí que una parte muy importante de este estudio sea la estructura y organización de dos de los más utilizados sistemas de gestión de bases de datos para documentos RDF, Sesame y Jena2. También es importante realizar una comparativa entre MySQL y PostgreSQL, dos sistemas de gestión de bases de datos muy utilizados cuando queremos trabajar con sistemas persistentes de datos RDF. Estos sistemas son gratuitos y pueden encontrarse en la Web. Se ha realizado pruebas entre MySQL, PostgreSQL y la API de Sesame con el objetivo de que esta comparativa sea más eficiente. Este trabajo contiene el desarrollo de una aplicación en la que se podrá realizar búsquedas semánticas en el dominio del mundo musical. Todas las tecnologías empleadas en este desarrollo han sido estudiadas en este trabajo. Para finalizar, se indica los pasos a seguir en la instalación de algunas de las tecnologías mencionadas en este estudio y puesta en práctica en la aplicación desarrollada.

Índice:

Índice de figuras:

Introducción

Actualmente, la World Wide Web (WWW) se ha convertido en un instrumento de uso cotidiano en nuestra sociedad, comparable con otros medios tan importantes como la radio, la televisión o el teléfono, a los que aventaja en muchos aspectos. La Web es hoy un medio extraordinariamente flexible y económico para la comunicación, el comercio y los negocios, ocio y entretenimiento, acceso a información y servicios, difusión de la cultura, etc. Paralelamente al crecimiento espectacular de la Web, las tecnologías que la hacen posible han experimentado una rápida evolución. Desde las primeras tecnologías básicas: HTML[1] y HTTP[2], hasta nuestros días, han emergido tecnologías como CGI[3], Java[4], JavaScript[5], ASP[6], JSP[7], PHP[8], Flash[9], J2EE[10], XML[11], por citar algunas de las más conocidas, que permiten una Web mejor, más amplia, más potente, más flexible, o más fácil de mantener. Estos cambios influyen y son al tiempo influidos por la propia transformación de lo que entendemos por WWW. La generación dinámica de páginas, el acoplamiento con bases de datos, la mayor interactividad con el usuario, la concepción de la Web como plataforma universal para el despliegue de aplicaciones, la adaptación al usuario, son algunas de las tendencias evolutivas más marcadas de los últimos años.

La evolución de la Web no termina aquí ni mucho menos. Son diversos los aspectos susceptibles de mejorar. A finales de los 90 surge la visión de lo que se ha denominado *Web semántica* [Berners-Lee 2001]. Ésta es una de las últimas tendencias que pueden repercutir en el futuro de la Web a medio plazo. Se trata de una corriente, promovida por el propio inventor de la Web y presidente del consorcio W3C[12], cuyo último fin es lograr que las máquinas puedan entender, y por tanto utilizar, el contenido de la Web. Esta nueva Web estaría poblada por agentes o representantes software capaces de navegar y realizar operaciones por nosotros para ahorrarnos trabajo y optimizar los resultados. Para conseguir esta meta, la Web semántica propone describir los recursos de la Web con representaciones procesables (es decir, entendibles) no sólo por personas, sino por programas que puedan asistir, representar, o reemplazar a las personas en tareas rutinarias o inabarcables para un humano. Las tecnologías de la Web semántica buscan desarrollar una Web más cohesionada, donde sea aún más fácil localizar, compartir e integrar información y servicios, para sacar un partido todavía mayor de los recursos disponibles en la Web.

---

[1] http://www.w3.org/Markup/
[2] http://www.w3.org/Protocols/
[3] http://hoohoo.ncsa.uiuc.edu/cgi/
[4] http://java.sun.com/
[5] http://www.mozilla.org/js/
[6] http://www.asp.net/
[7] http://java.sun.com/products/jsp/
[8] http://php.apache.org/
[9] http://www.macromedia.com/
[10] http://java.sun.com/j2ee/
[11] http://www.w3.org/XML/
[12] http://www.w3.org/

El objetivo de este trabajo es introducir al lector en el estudio de los lenguajes de consulta de documentos RDF, base de la Web Semántica. Algunos de estos lenguajes son RQL, RDQL, SPARQL y SeRQL, entre otros. Es importante realizar una comparativa de algunos de los lenguajes citados. Dentro de los objetivos de este trabajo, cabe destacar el estudio de la estructura y organización de algunos de los sistemas de gestión de bases de datos para RDF (Sesame y Jena2), así como para los sistemas persistentes de datos RDF (MySQL y PostgreSQL) y una comparativa de estos últimos. Otro objetivo importante es poner en práctica todo lo estudiado en este trabajo, mediante la instalación de algunas de estas tecnologías y el desarrollo de una pequeña aplicación.

Los métodos seguidos para la realización de este estudio son:

- Búsqueda de información en Internet, libros y revistas.
- Recopilación de esta información.
- Aportaciones propias.
- Experimentación con las distintas tecnologías estudiadas.
- Comparativas.

La planificación del proyecto se estructura en tres entregas parciales y una final con la totalidad del estudio realizado. En cada entrega se cumplió con una serie de objetivos definidos previamente. Las entregas se realizaron en las fechas siguientes:

- Primera entrega: 3 de octubre de 2005.
- Segunda entrega: 9 de noviembre de 2005.
- Tercera entrega: 23 de diciembre de 2005.
- Entrega final: 9 de enero de 2006.

La última parte de este trabajo consiste en obtener una aplicación que permita realizar búsquedas empleando las tecnologías estudiadas en este proyecto. El dominio de la aplicación será el mundo musical, donde podremos consultar diversos datos relacionados con canciones y artistas.

A parte de esta introducción, este trabajo se compone de 5 capítulos y un anexo. Seguidamente se realizará una breve descripción de los mismos.

En el Capítulo 1, se realiza una breve reseña histórica sobre la Web, así como el estado actual de la misma. Luego veremos conceptos relacionados con la Web semántica y para terminar el capítulo, veremos como encaja la Web semántica en la Web actual.

El Capítulo 2 estará dedicado al estudio de los distintos lenguajes de consulta para documentos RDF que existen actualmente. Entre los que se estudiarán, se destacan los siguientes lenguajes: SPARQL, RDQL, RQL y SeRQL. Para finalizar el capítulo, realizaremos una comparativa entre algunos de los lenguajes de consulta mas utilizados.

La siguiente parte de este estudio la encontraremos en el Capítulo 3, el cual se centra en conocer la estructura y organización de los SGDB que trabajan con información basada en RDF. Algunos de los SGBD que se evaluaran son: Sesame 2.0, Jena2 y Joseki.

En el Capítulo 4, realizaremos un análisis comparativo sobre algunos de los SGBD vistos en el Capítulo 3.

El Capítulo 5, se centra en un caso práctico, donde dispondremos de un conjunto de datos, en RDF, cuyo dominio será el mundo musical, en el cual se describirán los metadatos de las canciones: su título, el compositor, el estilo del grupo de música, el año en qué se grabó la canción, etc. Realizaremos una pequeña aplicación Web que nos permita hacer búsquedas sobre documentos RDF.

Terminaremos con el Capítulo 6, donde se describirá las conclusiones a las que se ha llegado tras el estudio realizado.

Al final, disponemos de un anexo, en el que se puede encontrar una guía para la instalación y puesta a punto de algunos de los SGBD vistos en este trabajo.

Capítulo 1

La Web Semántica

Historia de la Web

La aparición de la WWW se remonta al año 1989 [Abrams 1998, Connolly 2000], cuando Tim Berners-Lee presentó su proyecto de “World Wide Web” [Berners-Lee 1989] en el CERN (Suiza), con las características esenciales que perduran en nuestros días. El propio Berners-Lee completó en 1990 el primer servidor Web y el primer cliente, y un año más tarde publicó el primer borrador de las especificaciones de HTML y HTTP. El lanzamiento en 1993 de Mosaic, el primer navegador de dominio público, compatible con Unix, Windows, y Macintosh, por el National Center for Supercomputing Applications (NCSA), marca el momento en que la WWW se da a conocer al mundo, extendiéndose primero en universidades y laboratorios, y en cuestión de meses al público en general, iniciando el que sería su vertiginoso crecimiento. Los primeros usuarios acogieron con entusiasmo la facilidad con que se podían integrar texto y gráficos y saltar de un punto a otro del mundo en una misma interfaz, y la extrema sencillez para contribuir contenidos a una Web mundial.

Por estas mismas fechas se define la interfaz CGI para la generación dinámica de páginas Web, con lo que se consigue ofrecer información actualizada en tiempo real, enlazar con bases de datos, o tener en cuenta entradas del usuario, y más aún, servir como punto de acceso y plataforma para la ejecución de aplicaciones distribuidas. En 1994 miembros del equipo que creó Mosaic desarrollan Netscape, un navegador con sensibles mejoras que contribuye a impulsar la propagación de la Web. Este mismo año se celebra el primer congreso internacional de la WWW, y unos meses más tarde se constituye el consorcio W3C, que desde entonces y presidido por Tim Berners-Lee, se ha encargado de estandarizar las principales tecnologías Web. En 1995 Sun lanza

oficialmente la primera versión del lenguaje Java, y un año más tarde Netscape presenta JavaScript. Estos lenguajes y otros posteriores permiten que las propias páginas Web contengan programas enteros, dando opción a una mayor autonomía respecto del servidor, mayor eficiencia, capacidad dinámica y capacidad de interacción.

La Web actual

Hoy en día se estima que alrededor de 10^9 usuarios utilizan la Web, y que ésta contiene del orden de $4 \cdot 10^9$ documentos, lo que representa un volumen de información equivalente a entre 14 y 28 millones de libros [Bergman 2001]. Como dato comparativo, la asociación *American Research Libraries*, que agrupa unas 100 bibliotecas en EE.UU., tiene catalogados unos 3.7 millones de libros. La biblioteca de la Universidad de Harvard, la mayor de EE.UU., contiene en torno a 15 millones de libros. Estas cifras incluyen sólo lo que se ha dado en denominar la Web superficial, formada por los documentos estáticos accesibles en la Web. Se ha calculado que la llamada Web profunda, (constituida por bases de datos cuyos contenidos, no directamente accesibles, se hacen visibles mediante páginas generadas dinámicamente), puede contener un tamaño de información varios cientos de veces mayor, y de mucha mejor calidad, que la Web superficial, y crece a un ritmo aún mayor que ésta [O'Neill 2003]. Se estima que el tamaño de la Web profunda ha superado ya al volumen total de información impresa existente en todo el planeta.

Hoy casi todo está representado de una u otra forma en la Web, y con la ayuda de un buen buscador, podemos encontrar información sobre casi cualquier cosa que necesitemos. La Web se ha convertido en una enciclopedia universal del conocimiento humano. Por otra parte la Web nos permite realizar diferentes actividades de nuestra vida diaria con una comodidad, economía y eficiencia sin precedentes: sin movernos de casa podemos comprar todo tipo de productos y servicios, gestionar una cuenta bancaria, buscar un restaurante, consultar la cartelera, leer la prensa, localizar a una persona, matricularnos en la universidad, acceder a un callejero, o trabajar desde nuestro domicilio.

No obstante, hay espacio para mejoras. Por ejemplo, el enorme tamaño que ha alcanzado la Web, a la vez que es una de las claves de su éxito, hace que algunas tareas (por ejemplo encontrar la planificación óptima con transporte, alojamiento, etc., entre todas las posibles para un viaje bajo ciertas condiciones), requieran un tiempo excesivo para una persona o resulten sencillamente inabarcables. Desarrollar programas que realicen estas tareas en nuestro lugar es enormemente complicado, ya que es muy difícil reproducir -y más costoso aún mantener- en una máquina la capacidad de una persona para comprender los contenidos de la Web tal y como están codificados actualmente.

La asombrosa eficacia de los buscadores actuales tiene también sus límites. Por ejemplo, si queremos conocer la historia de Netscape, los resultados de

una consulta como “Netscape history”, nos informan sobre las herramientas de históricos de este navegador, pero no nos dicen nada sobre el origen y evolución de Netscape. Igualmente, para averiguar qué organismo se ocupa de estandarizar CGI, o en qué fecha apareció la primera versión de Java, necesitaremos realizar varias consultas y leer varios documentos y artículos hasta llegar indirectamente a la respuesta buscada. Si introducimos la palabra “Ketchup” para buscar información sobre el grupo de música del mismo nombre, obtendremos enlaces a restaurantes, recetas, fabricantes, distribuidores y clubes de aficionados al condimento, y finalmente lo que buscábamos (posiblemente ni siquiera esto si el grupo fuese menos popular).

Todos estos ejemplos son el síntoma de una causa común: la falta de capacidad de las representaciones en que se basa la Web actual para expresar significados. Los contenidos y servicios en la Web se presentan en formatos (p.e. HTML) e interfaces (p.e. formularios) comprensibles por personas, pero no por máquinas.

En estas condiciones es poco viable automatizar tareas mediante software en substitución del humano. Un programa puede llevar al usuario hasta lugares en la Web, generar, transportar, transformar y ofrecer la información a las personas, pero la máquina sencillamente no sabe lo que esta información significa, y por tanto su capacidad de actuación autónoma es muy limitada. Esta misma limitación expresiva hace que la noción de semántica que manejan los buscadores Web se limite a palabras clave con pesos, pero planas e inconexas, lo que no permite reconocer ni solicitar significados más elaborados.

La Web semántica

"*The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation*."[13] -- Tim Berners-Lee, James Hendler, Ora Lassila, The Semantic Web, Scientific American, May 2001.

La Web semántica[14] pretende ser una extensión de la Web actual. Está dotada de mayor significado en la que cualquier usuario en Internet podrá encontrar respuestas a sus preguntas de forma más rápida y sencilla gracias a una información mejor definida. Al dotar a la Web de más significado y, por lo tanto, de más semántica, se pueden obtener soluciones a problemas habituales en la búsqueda de información gracias a la utilización de una infraestructura común, mediante la cual, es posible compartir, procesar y transferir información de forma sencilla. Esta Web extendida y basada en el significado, se apoya en lenguajes universales que resuelven los problemas ocasionados por una Web carente de semántica en la que, en ocasiones, el acceso a la información se convierte en una tarea difícil y frustrante.

[13] http://www.w3.org/2001/sw/
[14] http://www.w3c.es/divulgacion/guiasbreves/WebSemantica

Ayuda a resolver dos importantes problemas: sobrecarga de información y heterogeneidad de fuentes de información con el consiguiente problema de interoperabilidad. Así, permite a los usuarios delegar tareas en software. Gracias a la semántica en la Web, el software es capaz de procesar su contenido, razonar con este, combinarlo y realizar deducciones lógicas para resolver problemas cotidianos automáticamente.

El objetivo de la Web semántica es construir una Web más útil. Se conseguirá facilitando el procesamiento del significado por parte de las máquinas. La Web semántica aboga por clasificar, dotar de estructura y anotar los recursos con semántica explícita procesable por máquinas.

Actualmente la Web se asemeja a un grafo formado por nodos del mismo tipo, y arcos (hiperenlaces) igualmente indiferenciados. Por ejemplo, no se hace distinción entre la página personal de un profesor y el portal de una tienda on-line del mundo musical, como tampoco se distinguen explícitamente los enlaces a las asignaturas que imparte un profesor de los enlaces a sus publicaciones. Por el contrario en la Web semántica cada nodo (recurso) tiene un tipo (profesor, tienda, pintor, libro), y los arcos representan relaciones explícitamente diferenciadas (pintor – obra, profesor – departamento, libro – editorial, canción - autor).

La forma en la que se procesará esta información no sólo será en términos de entrada y salida de parámetros sino en términos de su semántica. La Web semántica como infraestructura basada en metadatos aporta un camino para razonar en la Web, extendiendo así sus capacidades.

Se debe alcanzar un entendimiento entre las partes que han de intervenir en la construcción y explotación de la Web: usuarios, desarrolladores y programas de muy diverso perfil. La Web semántica rescata la noción de ontología del campo de la Inteligencia Artificial como vehículo para cumplir este objetivo.

Gruber, define ontología como "*a formal explicit specification of a shared conceptualization*" [Gruber 1993]. Una ontología es una jerarquía de conceptos con atributos y relaciones, que define una terminología consensuada para definir redes semánticas de unidades de información interrelacionadas. Una ontología proporciona un vocabulario de clases y relaciones para describir un dominio, poniendo el acento en la compartición del conocimiento y el consenso en la representación de éste. Por ejemplo, una ontología sobre el mundo musical podría incluir clases como *Autor, Canción* o *Estilo musical*, y relaciones como *autor* de una canción, grupos *pertenecientes* a un estilo musical o títulos *localizados* en una tienda.

La idea es que la Web semántica esté formada (al menos en parte) por una red de nodos tipificados e interconectados mediante clases y relaciones definidas por una ontología compartida por sus distintos autores. Por ejemplo, una vez establecida una ontología sobre títulos de compact disc y grupos musicales, una tienda virtual puede organizar sus contenidos definiendo instancias de títulos, estilos musicales, etc., interrelacionándolas y publicándolas en la Web semántica. La adopción de ontologías comunes es clave para que todos los

que participen de la Web semántica, contribuyendo o consumiendo recursos, puedan trabajar de forma autónoma con la garantía de que las piezas encajen. Así por ejemplo varios tiendas podrían colaborar para dar lugar a una gran meta-tienda que integre los contenidos de todas ellas. Un programa que navegue por una red como ésta puede reconocer las distintas unidades de información, obtener datos específicos o razonar sobre relaciones complejas. A partir de aquí sí podemos distinguir entre una canción escrita por un artista y una canción interpretada por ese mismo artista.

Por último, la Web no solamente proporciona acceso a contenidos sino que también ofrece interacción y servicios (comprar un compact disc, reservar una plaza en un vuelo, hacer una transferencia bancaria, simular una hipoteca). Los servicios Web semánticos son una línea importante de la Web semántica, que propone describir no sólo información sino definir ontologías de funcionalidad y procedimientos para describir servicios Web: sus entradas y salidas, las condiciones necesarias para que se puedan ejecutar, los efectos que producen, o los pasos a seguir cuando se trata de un servicio compuesto. Estas descripciones procesables por máquinas permitirían automatizar el descubrimiento, la composición, y la ejecución de servicios, así como la comunicación entre unos y otros.

¿Cómo encaja la Web semántica con la actual?

Para responder a ésta pregunta, debemos ver cómo accederá el usuario a la Web semántica, y sobre todo, cómo hacer la transición de la Web actual a la Web semántica. Para que la Web semántica pueda realizarse, es importante que guarde, al menos al principio, una compatibilidad con la tecnología actual. Es deseable por ejemplo mantener HTML (u otros lenguajes compatibles con los navegadores actuales, como por ejemplo XHTML) como vehículo de comunicación con el usuario. La asociación entre las instancias de la Web semántica y el código HTML se puede establecer de distintas maneras. Una consiste en conservar los documentos actuales, y crear las instancias asociadas anotando su correspondencia con los documentos. Esta posibilidad es la más viable cuando se parte de un gran volumen de material antiguo. Otra es generar dinámicamente páginas Web a partir de las ontologías y sus instancias. Esta última opción puede resultar factible cuando los documentos antiguos ya se estaban generando automáticamente a partir, por ejemplo, de una base de datos.

La transición de la Web actual a la Web semántica puede implicar un coste altísimo si tenemos en cuenta el volumen de contenidos que ya forman parte de la Web. Crear y poblar ontologías supone un esfuerzo extra que puede resultar tedioso cuando se agregan nuevos contenidos, pero directamente prohibitivo por lo que respecta a integrar los miles de gigabytes de contenidos antiguos. Las estrategias más viables combinan una pequeña parte de trabajo manual con la automatización del resto del proceso. Las técnicas para la automatización incluyen, entre otras, el mapeo de la estructura de bases de

datos a ontologías, el aprovechamiento, previa conversión, de los metadatos y estándares de clasificación presentes en la Web (y fuera de ella), y la extracción automática de metadatos a partir de texto y recursos multimedia.

Otra dificultad importante a la hora de realizar la Web semántica en la práctica es la de consensuar ontologías en una comunidad por poco amplia que sea. Converger a una representación común es una labor más compleja de lo que puede parecer, ya que típicamente cada parte del sistema conlleva peculiaridades necesarias, y un punto de vista propio que a menudo necesitan incidir en la propia ontología. La representación del mundo no es neutra respecto al uso que se le va a dar: tanto un dietista como un biólogo tienen conocimiento sobre las plantas, pero su representación de esa materia es muy distinta, y probablemente no sería adecuado imponer la misma representación para ambas perspectivas. Las vías para salvar esta dificultad consisten en compartir ontologías para las áreas comunes en que puede tener lugar una interacción o intercambio de información entre las partes, y establecer formas de compatibilidad con las ontologías locales, mediante extensión y especialización de las ontologías genéricas, o por mapeo y exportación entre ontologías.

Capítulo 2

Tecnologías relacionadas con la Web Semántica

El objetivo principal de este capítulo es realizar un análisis de las distintas propuestas de lenguajes de consulta para documentos RDF que existen actualmente. No se trata de hacer un estudio profundo de cada uno de estos lenguajes debido a problemas de espacio, por lo tanto queda fuera del ámbito de este trabajo, por lo que se presentará las características mas importantes, así como ejemplos de consultas de cada uno de ellos y terminaremos con una comparativa, lo cual puede resultar ser lo mas interesante del capítulo.

Previamente, para situar al lector, se realizará una visión general sobre las distintas tecnologías que hacen posible la Web semántica.

Es importante entender las características del lenguaje RDF. Para ello, se ha incluido una sección dedicada a este lenguaje.

Visión general

La tecnología que se ha creado para hacer posible la Web semántica incluye lenguajes para la representación de ontologías, parsers, lenguajes de consulta, entornos de desarrollo, módulos de gestión (almacenamiento, acceso,

actualización) de ontologías, módulos de visualización, conversión de ontologías, y otras herramientas y librerías.

El primer lenguaje para la construcción de la Web semántica fue SHOE[15], creado por Jim Hendler en la Universidad de Maryland en 1997. Desde entonces se han definido otros lenguajes y estándares con finalidad similar, como XML, RDF[16], DAML+OIL[17], y más recientemente OWL[18], por citar los más importantes.

XML

XML representa una primera aproximación a la Web semántica, y aunque no está expresamente pensado para definir ontologías, es el estándar más extendido hoy día en las aplicaciones de esta línea previa a la Web semántica. XML permite estructurar datos y documentos en forma de árboles de etiquetas con atributos. Con XML Schema[19] (XMLS) se pueden acordar de antemano las estructuras que se van a utilizar, así como manejar tipos de datos primitivos y derivados. Con el estándar XSLT[20] se pueden definir plantillas asociadas a las estructuras XML, que describen cómo generar código HTML para visualizar los contenidos en un navegador. Parsers como DOM[21] permiten moverse por las estructuras XML desde un programa Java o C++, y existen multitud de herramientas para facilitar la compatibilidad de XML con bases de datos, JavaBeans[22], etc.

Desde la aparición de XML en 1998, se han definido multitud de estándares para modelizar información en dominios específicos como las finanzas [Coates 2001] (XBRL, RIXML, FpXML, ebXML, etc.), el periodismo[23] (p.e. NewsML, XMLNews, PRISM[24]), la enseñanza (SCORM[25], IEEE LOM[26] y otros), o la medicina [Dudek 2001] (NLM Medline, SCIPHOX, CDA, etc.), entre otros muchos campos. XML es un primer paso en la dirección de avanzar hacia una representación explícita de los datos y la estructura de los contenidos de la Web, separada de su presentación en HTML. XML proporciona una sintaxis para hacerlo posible, pero ofrece una capacidad limitada para expresar la semántica. El modelo de datos XML consiste en un árbol que no distingue entre objetos y relaciones, ni tiene noción de jerarquía de clases.

[15] http://www.cs.umd.edu/projects/plus/SHOE/
[16] http://www.w3.org/RDF/
[17] http://www.w3.org/TR/daml+oil-reference/
[18] http://www.w3.org/2001/sw/WebOnt/
[19] http://www.w3.org/XML/Schema/
[20] http://www.w3.org/Style/XSL/
[21] http://www.w3.org/DOM/
[22] http://java.sun.com/products/javabeans/
[23] http://www.iptc.org/
[24] http://www.prismstandard.org/
[25] http://www.adlnet.org/
[26] http://ltsc.ieee.org/wg12/

RDF y RDF Schemas

En 1999 se publicó la primera versión de RDF (Resource Description Framework), un lenguaje para la definición de ontologías y metadatos en la Web. RDF es hoy el estándar más popular y extendido en la comunidad de la Web semántica. El elemento de construcción básica en RDF es la tripleta o sentencia, que consiste en dos nodos (sujeto y objeto) unidos por un arco (predicado), donde los nodos representan recursos, y los arcos propiedades. Por ejemplo una sentencia podría expresar el hecho de que el cantante (predicado) de una canción (sujeto) fue la persona Frank Sinatra (objeto). Encadenando estas tripletas se construyen grafos o redes semánticas para la Web.

Con RDF Schema (RDFS) se pueden definir jerarquías de clases de recursos, especificando las propiedades y relaciones que se admiten entre ellas. En RDF las clases, relaciones, y las propias sentencias son también recursos, y por lo tanto se pueden examinar y recorrer como parte del grafo, o incluso afirmar sentencias sobre ellas. Se han definido diferentes formas sintácticas para la formulación escrita de RDF, pero quizás la más extendida es la basada en XML. Es por ello que RDF se presenta a menudo como una extensión de XML.

Una de las realizaciones pendientes desde hace años en relación con RDF es la creación de un lenguaje de consulta, similar al SQL de las bases de datos, que permita expresar búsquedas complejas sobre un grafo RDF mediante una sintaxis declarativa sencilla. A falta de alcanzar un acuerdo sobre un estándar comúnmente aceptado, se han consolidado de facto distintas iniciativas particulares como la del RDF Query Language[27] (RDQL), por Hewlett Packard, posiblemente el más extendido; RDF Schema Query Language[28] (RQL) [Karvounarakis 2002], por el instituto ICS-FORTH de Grecia; y Sesame RDF Query Language (SeRQL), por la empresa holandesa Aidministrator. Por ejemplo, la siguiente consulta RDQL:

```
SELECT ?y
FROM <gente.rdf>
WHERE (?x, <edad>, ?z), (?x, <nombre>, ?y) and ?z > 17
```

devolvería la propiedad “nombre” de todos los recursos del grafo definido en el documento “gente.rdf” cuya propiedad “edad” tiene un valor mayor que 17.

Otras tecnologías relacionadas con la Web semántica

Cronológicamente, a RDF le siguieron OIL[29] (Ontology Inference Language), desarrollado en Europa, y DAML[30] (DARPA Agent Markup Language), en EE.UU., dos lenguajes ontológicos muy similares que de hecho se terminaron fundiendo en DAML+OIL. A partir de esta unión se definió el lenguaje OWL

[27] http://www.hpl.hp.com/semweb/rdql.htm
[28] http://139.91.183.30:9090/RDF/RQL/
[29] http://www.ontoknowledge.org/oil/
[30] http://www.daml.org/

(Web Ontology Language), con el propósito de reunir todas las ventajas de DAML+OIL y resolver los problemas de este lenguaje. OWL se puede formular en RDF, por lo que se suele considerar una extensión de éste. OWL incluye toda la capacidad expresiva de RDFS y la extiende con la posibilidad de utilizar expresiones lógicas. OWL permite, por ejemplo, definir clases mediante condiciones sobre sus miembros (p.e. la clase de los temas creados por grupos musicales españoles), mediante combinación booleana de clases (Mecano and Celtas Cortos and not Maná en una ontología del mundo musical), o por enumeración de las instancias que pertenecen a la clase (p.e. por extensión). Además OWL permite atribuir ciertas propiedades a las relaciones, como cardinalidad, simetría, transitividad, o relaciones inversas. Si bien RDF y OWL son hoy en día los lenguajes más consolidados, existen otros lenguajes interesantes, aunque su uso e implantación es mucho menor, como TopicMaps[31], OCML[32] o WebODE[33].

Escribir en lenguajes como RDF y OWL resulta sumamente difícil y propenso a errores. Afortunadamente se pueden utilizar entornos gráficos para visualizar y construir ontologías de forma mucho más razonable, como Kaon[34], WebODE o Protégé[35]. De todas ellas Protégé, desarrollada por el Stanford Medical Informatics de la Universidad de Stanford, es la herramienta de construcción de ontologías que más usuarios tiene actualmente. Con ella se puede fácilmente crear clases y jerarquías, declarar propiedades para las clases, crear instancias e introducir valores, todo ello en un entorno de menús, botones, cuadros de diálogo y representaciones gráficas fáciles de usar. Protégé tiene su propio lenguaje interno para definir ontologías, pero permite también trabajar con RDF y OWL de modo transparente. Protégé es un entorno abierto y fácil de extender, que ha generado en torno suyo toda una comunidad que contribuye activamente a ampliar el entorno con todo tipo de contribuciones en forma de plug-ins, y está haciendo de esta herramienta un entorno sumamente potente.

Estudio realizado

Durante la investigación realizada se analizaron varios lenguajes de consulta, encontrando que en la actualidad la mayoría de ellos se orientan a RDF. Dichos lenguajes pueden emplearse para explotar una ontología RDF o DAML, pero la sintaxis no cuenta con elementos específicos de la ontología sino que hay que utilizar ciertas convenciones, o definiciones RDF, que conviertan las consultas RDF en consultas propias de ontologías. Esto resulta en consultas, que a pesar de ser sencillas, se codifican en un formato extenso y complicado. Esto sin nombrar la necesidad de que la persona que codifica la consulta tiene que dominar RDF.

De los lenguajes de consulta que se han estudiado, uno de los más simples fue RQL, que aunque esta orientado a RDF, su sintaxis parecida a la de SQL hace

[31] http://www.topicmaps.org/
[32] http://kmi.open.ac.uk/projects/ocml/
[33] http://delicias.dia.fi.upm.es/webODE/
[34] http://kaon.semanticweb.org/
[35] http://protege.semanticweb.org/

que la codificación de consultas no sea tan difícil; sólo hay que conocer los principios de SQL y la noción de sentencia RDF para elaborar las consultas.

RDF

RDF (Resource Description Framework) es una de las tecnologías claves en el proyecto de la Web semántica. Es la propuesta del W3C para definir metadatos en el Web y la base para el procesamiento de metadatos: proporciona interoperabilidad semántica entre aplicaciones que intercambian información entendible por máquina. RDF es simplemente un modelo de datos que permite crear metadatos legibles y entendibles por máquina. La interoperabilidad semántica de sistemas de metadatos implica significados compartidos y gramáticas compartidas. Como con el lenguaje natural, traducir un sistema de los metadatos particular en los términos y gramática de otro requiere interpretación y puede involucrar pérdida o distorsión de significado. El reconocimiento y la aceptación de este límite inherente a la interoperabilidad es una marca constante en la filosofía de la Web semántica.

La Web semántica y RDF abordan estos problemas proporcionando una infraestructura que posibilite la automatización del descubrimiento de recursos (como los motores basados en robots) y la capacidad para indicar los recursos inteligentemente (como los motores basados en directorios). RDF lo logra a través del uso de semántica entendible por máquina. Se diseñan los metadatos en RDF específicamente para ser entendidos e intercambiados por procesos automatizados, como los agentes software y los sistemas de búsqueda. RDF intentará mejorar el descubrimiento de recursos proporcionando un mayor grado de precisión en los resultados de las búsquedas que los sistemas de búsqueda actuales.

Modelo de datos

Básicamente, RDF proporciona un modelo de datos para describir la semántica de los datos de una forma procesable por máquina. Un objeto de información o recurso se describe a través de un conjunto de propiedades. La esencia de RDF es pues un modelo formal para la representación de las propiedades y los valores de esas propiedades[36]. RDF usa los namespaces XML para calificar los recursos. Esto previene colisiones de nombre de elemento que podrían suscitarse cuando se combinan múltiples vocabularios. Los namespaces también se usan para identificar el esquema RDF.

El modelo de datos básico consiste en tres tipos de objetos:

[36] W3C World Wide Web Consortium. Resource Description Framework (RDF): Model and Syntax Specifica-tion. W3C Recommendation, 22 February 1999. [Ora Lassila y Ralph R. Swich, eds. Disponible en: http://www.w3.org/TR/1999/REC-rdf-syntax-19990222

- **Recursos:** Todo aquello que puede ser descrito por una expresión RDF es un "recurso". Puede ser una página Web, o una parte de una página Web, una colección entera de páginas o un sitio Web. Un recurso también puede ser un objeto que no es directamente accesible vía Web como por ejemplo un libro. Todos los recursos son identificados mediante URIs (entendidos como identificadores de objetos). Es importante destacar que cualquier cosa puede tener un URI y su extensibilidad permite la definición de identificadores para cualquier objeto imaginable. Un URI no necesariamente direcciona un recurso Web. Por ejemplo, el ISBN 0679405739, puede ser usado como un URI "ISBN:0679405739".

- **Propiedades:** Una propiedad es una característica, aspecto, atributo o relación usada para describir un recurso. Cada propiedad tiene un significado específico, define sus valores posibles, los tipos de recursos que puede describir y su relación con otras propiedades.

- **Sentencias:** Un recurso junto con una propiedad con nombre y con el valor de la propiedad para dicho recurso se denomina una "sentencia". Las sentencias representan relaciones binarias específicas entre dos objetos. Las tres partes de una sentencia se denominan respectivamente sujeto (recurso), predicado (propiedad) y objeto (recurso o literal). El objeto (el valor de la propiedad) puede ser otro recurso (especificado por una URI) o bien un valor literal.

El modelo RDF puede ser representado como un grafo dirigido etiquetado. Veamos un ejemplo concreto, extraído de la especificación Primer RDF[37] donde se muestran una serie de declaraciones o sentencias: "hay una persona identificada por http://www.w3.org/People/EM/contact#me, cuyo nombre es Eric Miller, cuya dirección de correo electrónico es em@w3.org, y cuyo título es Dr." que podría representarse como el grafo RDF de la figura 1.

Esta figura ilustra que RDF usa URIs para identificar:

- Individuos, por ejemplo, Eric Miller, identificado por http://www.w3.org/People/EM/contact#me.

- Clases de cosas, por ejemplo, Person, identificado por http://www.w3.org/2000/10/swap/pim/contact#Person.

- Propiedades de estas cosas, por ejemplo, mailbox, identificado por http://www.w3.org/2000/10/swap/pim/contact#mailbox.

- Valores de estas propiedades, por ejemplo, mailto:em@w3.org como el valor de la propiedad mailbox (RDF también usa cadenas de caracteres tales como "Eric Miller", y valores de otros tipos de datos como enteros y datos, o valores de propiedades).

[37] http://www.w3.org/TR/rdf-primer/

Figura 1 – Ejemplo de grafo RDF que describe a Eric Miller

En estos grafos, los nodos (representados como elipses) representan recursos y los arcos representan propiedades con nombre (el rótulo del arco es el nombre de la propiedad). Los nodos literales se dibujan como rectángulos.

Como ejemplo de un editor RDF con representación de grafos podemos destacar a IsaViz[38], un entorno para la visualización y edición de modelos RDF. Recursos y literales son los nodos del grafo (elipses y rectángulos), con propiedades representadas como arcos que conectan estos nodos.

RDF/XML

RDF sirve para describir la semántica. Sin embargo la semántica, en un sistema informático, sirve de poco sin una sintaxis asociada. RDF usa una serialización XML conocida como RDF/XML que proporciona la base sintáctica[39]. Se han desarrollado también varios lenguajes de consulta para RDF que soportan esta sintaxis.

[38] W3C IsaViz, http://www.w3.org/2001/11/IsaViz/
[39] W3C World Wide Web Consortium. RDF/XML Syntax Specification (Revised). W3C Working Draft 25 March 2002. Dave Beckett, ed. W3C, 25 de marzo de 2002. Disponible en: http://www.w3.org/TR/2002/WD-rdf-syntax-grammar-20020325

Una sentencia RDF raramente aparece en forma aislada. Lo normal es que varias propiedades de un recurso sean indicadas simultáneamente. La sintaxis RDF/XML ha sido diseñada para permitir agrupar varias sentencias sobre un mismo recurso en un elemento "Description". El elemento "Description" menciona en un atributo, "about", el recurso al que se aplican las sentencias. Si el recurso todavía no existe, el elemento "Description" puede asignarle un identificador en el momento, usando un atributo ID.

Un ejemplo básico en RDF/XML sería el siguiente:

```
<rdf:RDF>
        <rdf:Description about="http://www.w3.org/home/jperez">
                <s:creator>Juan Pérez</s:creator>
        </rdf:Description>
</rdf:RDF>
```

En un segundo ejemplo “El individuo cuyo nombre es Juan Pérez, email jperez@yahoo.com es el creador de http://www.w3.org/home/jperez” podría escribirse en RDF/XML de la forma:

```
<rdf:RDF>
        <rdf:Description about="http://www.w3.org/home/jperez">
                <s:creator rdf:resource="http://www.w3.org/staffId/37438"/>
        </rdf:Description>
        <rdf:Description about="http://www.w3.org/staffId/37438">
                <v:name>Juan Perez</v:name>
                <v:email>jperez@yahoo.com</v:name>
        </rdf:Description>
</rdf:RDF>
```

Una muestra de RDF en RDF/XML correspondiente al grafo de la figura 1 es la siguiente:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
           xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">
  <contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
    <contact:fullName>Eric Miller</contact:fullName>
    <contact:mailbox rdf:resource="mailto:em@w3.org"/>
    <contact:personalTitle>Dr.</contact:personalTitle>
  </contact:Person>
</rdf:RDF>
```

Frecuentemente es necesario referirse a una colección de recursos. RDF define tipos específicos de contenedor que representan multiconjuntos, secuencias, y alternativas. RDF define contenedores para manejar listas de recursos o literales:

- **Bag:** Un bag es una lista de recursos o literales sin orden. Se permiten valores duplicados y no hay diferencias en cuanto al orden en que aparecen los recursos.

- **Sequence:** Una secuencia es una lista ordenada de recursos o literales. Se permiten valores duplicados.

- **Alternative:** Es una secuencia de recursos o literales para un "único" valor o propiedad. De la lista de recursos o literales debe elegirse uno.

RDF Schema

Las primitivas de modelado de datos proporcionadas por RDF son muy básicas (identificación de objeto, relaciones binarias, reificación y contenedores). La RDF Schema Specification[40] es una extensión de RDF que proporciona primitivas adicionales para definir ontologías. Enriquece el modelo básico, proporcionando un vocabulario para RDF, que se asume tiene una cierta semántica. Permite a los diseñadores especificar una jerarquía explícita de clases de recursos y propiedades que describen estas clases, junto con las restricciones sobre las combinaciones permitidas de clases, propiedades y valores.

Comentamos en detalle algunas características relevantes de RDFS:

- Clases: rdfs:Resource, rdf:Property, y rdfs:Class. Cualquier cosa descrita por una sentencia RDF se considera una instancia de la clase rdfs:Resource. La clase rdf:Property es la clase de todas las propiedades utilizadas en la caracterización de las instancias de rdfs:Resource. Finalmente, rdfs:Class se usa para definir conceptos. Cada concepto debe ser una instancia de rdfs:Class.

- Propiedades: rdf:type, rdfs:subClassOf, and rdfs:subPropertyOf. La relación rdf:type modela interrelaciones del tipo instancia-de entre recursos y clases. Un recurso puede ser una instancia de más de una clase. Rdfs:subClassOf modela la jerarquía de clases, donde una clase puede ser subclase de otras subclases. Si una propiedad P2 es una subpropiedad de (rdfs:subPropertyOf) otra propiedad P1, y si un recurso R tiene una propiedad P2 con valor V, esto implica que el recurso R también tiene la propiedad P1 con valor V.

- Restricciones y clases: rdfs:ConstraintResource, rdfs:ConstraintProperty, rdfs:range, y rdfs:domain. rdfs:ConstraintResource define la clase de todas las restricciones. rdfs:ConstraintProperty es un subconjunto de

[40] W3C World Wide Web Consortium. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Work-ing Draft 30 April 2002. Dan Brickley y R. V. Guha, eds. W3C, 30 de abril de 2002. Disponible en: http://www.w3.org/TR/2002/WD-rdf-schema-20020430/

rdfs:ConstraintResource. Tiene dos instancias: rdfs:range y rdfs:domain que se usan para restringir el rango y el dominio de las propiedades. No se permite expresar más de una restricción de rango sobre una propiedad[41]. En dominios si se permite, y se interpreta como una unión de dominios.

El siguiente ejemplo se ha tomado de la especificación de RDFS 1.0.

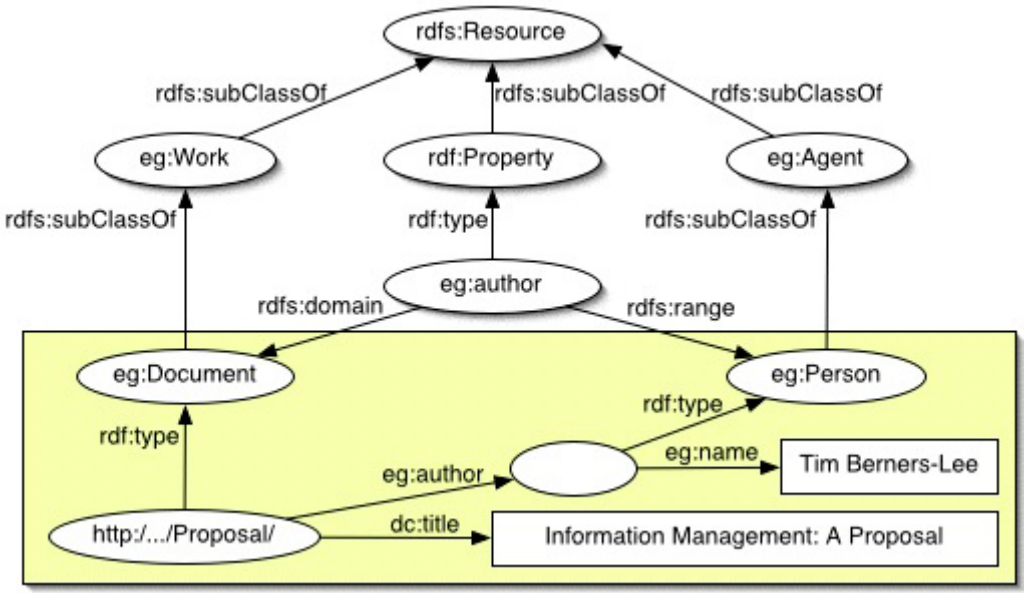

Figura 2 – Ejemplo de grafo RDFS

XML Schemas frente a RDF Schemas

Confusamente, el término “esquema” se asocia con dos especificaciones del W3C, el esquema XML y el esquema RDF. En un sentido amplio, un esquema XML se diseña para analizar y validar la sintaxis de la estructura de etiquetas de archivos de metadatos (en este sentido, un esquema XML es un “esquema de documento”). En contraste, para representar las relaciones de términos particulares con otros términos del esquema o con términos definidos en otros esquemas en el Web, es preferible un esquema RDF (en este sentido, un esquema RDF es un esquema semántico).

A pesar de la similitud en sus nombres, RDFS desempeña un papel diferente al de XMLS. XMLS, como las DTDs, prescribe el orden y combinación de las etiquetas en un documento XML. En contraste, RDFS proporciona información sobre la interpretación de las sentencias contenidas en un modelo de datos RDF, pero no restringe la apariencia sintáctica de una descripción RDF.

Desarrollados independientemente por dos grupos de trabajo paralelos a finales de los 90, la relación entre estas dos especificaciones es la causa de

[41] Aunque no está recogida en la propuesta original de RDFS, podemos encontrar una recomendación para implementadores que si la permite. El grupo RDF Core la incluirá en la especificación final.

muchas confusiones, y desde enero de 2002, el W3C está liderando los esfuerzos para combinar las funcionalidades de ambos en un lenguaje de esquema integrado.

Patel-Schneider y Siméon[42] señalan las diferencias entre RDFS y XMLS: XML es ordenado, RDF no; XML usa un modelo de árbol, RDF de grafos; RDF distingue entre clases y propiedades, y en XML, todos son elementos. Sin embargo, cree que es posible desarrollar un modelo unificado que sirva como base para las aplicaciones que trabajen tanto con documentos (XML) como con semántica (RDF).

RQL

RQL es un lenguaje tipado, que siguiendo un enfoque funcional, define un conjunto de consultas básicas e iteradores. RQL es soportado por el sistema Sesame (ver Capítulo 3). A continuación se presentan los elementos más importantes del lenguaje, así como algunos ejemplos de consultas.

Funciones

RQL define algunas funciones que le sirven para obtener resultados básicos de la ontología, al mismo tiempo que auxilian en la construcción de consultas más complejas. Dichas funciones pueden ejecutarse independientemente. A continuación se presentan algunas de las más representativas:

subClassOf(c)	Devuelve todas las subclases transitivas de c.
subClassOf ^(c)	Devuelve todas las subclases directas de c.
superClassOf(c)	Devuelve todas las superclases transitivas de c.
superClassOf ^(c)	Devuelve todas las superclases directas de c.
subPropertyOf(p)	Devuelve todas las subpropiedades transitivas de c.
subPropertyOf ^(p)	Devuelve todas las subpropiedades directas de c.
superPropertyOf(p)	Devuelve todas las superpropiedades transitivas de c.
superPropertyOf ^(p)	Devuelve todas las superpropiedades directas de c.
subClassOf(c,n)	Devuelve todas las subclases transitivas de c hasta el nivel n. Esta sintaxis se aplica también para superClassOf, subPropertyOf y superPropertyOf.
Topclass	Devuelve el nodo raíz de la jerarquía de clases.
Leafclass	Devuelve los nodos hoja de la jerarquía de clases.
topproperty	Devuelve el nodo raíz de la jerarquía de clases.
leafproperty	Devuelve los nodos hoja de la jerarquía de clases.
domain(p)	Devuelve las clases del dominio de la propiedad p.
range(p)	Devuelve las clases del rango de la propiedad p.
namespace(o)	Devuelve el namespace del esquema o.

[42] Patel-Scheneider, P.; Siméon, J. The Ying/Yang Web: XML syntax and RDF semantics. WWW2002, Honolu-lu, Hawai, mayo de 2002. Disponible en: http://www2002.org/CDROM/refereed/231/index.html

Class	Devuelve todas las clases definidas.
Property	Devuelve todas las propiedades definidas.

Consultas de esquema

Siguiendo una sintaxis similar a la de SQL se pueden construir consultas como la siguiente:

```
SELECT $C1, $C2
FROM {$C1}creates{$C2}
```

que pregunta por el dominio y el rango de la propiedad creates. El prefijo $ en el nombre de las variables denota variables de clases en el esquema. El resultado obtenido sería el producto cruz de las clases que pertenecen al dominio contra las que pertenecen al rango.

Si se desea conocer las propiedades definidas en la clase Painter, se hace una consulta como la siguiente:

```
SELECT @P
FROM {;Painter}@P
```

Donde @P denota una variable de Propiedad.

Ahora bien, si se pregunta por los rangos de la propiedad exhibited que pueden ser alcanzados desde una clase en el rango de la propiedad creates, se tendría:

```
SELECT $X, $Z
FROM creates{$X}, {$Y}exhibited{$Z}
WHERE $X = $Y
```

Consultas de descripciones o instancias

Se puede preguntar por las instancias de la clase Artist con la siguiente pregunta:

```
SELECT X
FROM Artist{X}
```

También se puede preguntar por las instancias que estén relacionadas a través del predicado creates:

```
SELECT X, Y
FROM {X}creates{Y}
```

Se pueden incluir restricciones o filtros en la búsqueda. Por ejemplo, se puede preguntar por los recursos que fueron modificados (last modified) después del año 2000:

```
SELECT X, Y
FROM {X}last_modified{Y}
WHERE Y >= 2000-01-01
```

Hay forma de restringir que las instancias buscadas pertenezcan a cierta clase. Por ejemplo, en la siguiente consulta se buscan específicamente las pinturas (Painting) exhibidas (exhibited) como recurso externo (ExtResource):

```
SELECT X, Y
FROM {X;Painting}exhibited{Y;ExtResource}
```

Estas son algunas de las consultas que se pueden construir, sin embargo, hay muchas otras características de RQL que no se muestran en esta reseña y que permiten hacer consultas mucho más complejas y obtener información acerca del metaesquema, esquema, e instancias. Cabe señalar que los resultados son codificados en XML o RDF.

RDQL

RDQL es una implementación de un lenguaje de consulta similar a SQL pero para RDF. Ha sido desarrollado por HP y es soportado por Jena2. De hecho, también podemos decir que RDQL es el lenguaje de consulta para grafos RDF de Jena2. El hecho de que sea similar a SQL, resulta familiar a muchos usuarios.

El siguiente es un ejemplo de RDQL desde Jena2:

```
QueryResults results =
        Query.exec ("SELECT ?x, ?y WHERE (?x, <name>, ?y)", model);

while (results.hasNext()) {
        ResultBinding res = (ResultBinding) results.next () ;
        Resource x = (Resource) res.get ("x") ;
        Literal y = (Literal) res.get ("y") ;
        System.out.println ("x = " + x + " y = " + y) ;
}

results.close() ;
```

RDQL trata RDF como datos y provee consultas con patrones de sentencias y restricciones sobre un modelo RDF. Se diseñó para ser usado en scripts y para experimentación en lenguajes de modelación de información. El lenguaje está derivado de SquishQL, el cual a su vez está basado en RdfQL, un lenguaje de

bases de datos escalables creadas para trabajar con ServiciosWeb semánticos.

La cláusula SELECT define las variables que se requieren mostrar en el conjunto resultante. La cláusula WHERE define un patrón de subgrafo en términos de variables y constantes. La cláusula AND introduce un filtro en las variables; solamente los resultados que pasan el filtro son incluidos en el conjunto resultante de la consulta. La cláusula USING define abreviaciones para espacios de nombre.

Consultas RDQL

Según lo anterior, podemos decir que una consulta RDQL tiene la siguiente forma:

```
SELECT vars
FROM documents
WHERE Expressions
AND Filters
USING Namespace declarations
```

Por ejemplo:

```
SELECT ?x,?y
FROM <http://example.com/sample.rdf>
WHERE (?x,<dc:name>,?y)
USING dc for <http://www.dc.com#>
```

Supongamos que tenemos el fichero people.rdf, con las siguientes sentencias RDF:

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:dt="http://foo.org#">

<rdf:Description about="http://foo.org/persons/john">
  <dt:name>John</dt:name>
  <dt:age>26</dt:age>
</rdf:Description>

<rdf:Description about="http://foo.org/persons/Peter">
  <dt:name>Peter</dt:name>
  <dt:position>foo2</dt:position>
  <dt:age>25</dt:age>
  <dt:friend rdf:resource="http://foo.org/persons/john" />
  <dt:friend rdf:resource="http://foo.org/persons/Carl" />
</rdf:Description>
```

```
<rdf:Description about="http://foo.org/persons/Micky">
  <dt:name>Micky</dt:name>
  <dt:position>foo</dt:position>
  <dt:age>16</dt:age>
</rdf:Description>

<rdf:Description about="http://foo.org/persons/Carl">
  <dt:name>Carl</dt:name>
  <dt:position>foo</dt:position>
  <dt:age>28</dt:age>
  <dt:friend rdf:resource="http://foo.org/persons/Peter" />
</rdf:Description>

<rdf:Description about="http://foo.org/team">
  <dt:members>
   <rdf:Bag>
    <rdf:li rdf:resource="http://foo.org/persons/Peter" />
    <rdf:li>Kim</rdf:li>
    </rdf:Bag>
  </dt:members>
</rdf:Description>

</rdf:RDF>
```

Si queremos obtener el nombre y la edad de aquellos que tienen más de 20 años:

```
SELECT ?y
FROM <people.rdf>
WHERE (?x,<dt:age>,?z),(?x,<dt:name>,?y)
AND ?z>20
USING dt for <http://foo.org#>, rdf for <http://www.w3.org/1999/02/22-rdf-
syntax-ns#>
```

Para mostrar todos los miembros de un equipo, las sentencias RDQL serían:

```
SELECT ?z
FROM
WHERE (?x,,?y),(?y,?w,?z)
AND ?z<>"http://www.w3.org/1999/02/22-rdf-syntax-ns#Bag" &&
?x=="http://foo.org/team"
USING dt for , rdf for
```

Diferencias entre RQL y RDQL

Para terminar esta sección, veremos dos ejemplos de sentencias en RQL y RDQL para poder determinar sus diferencias y similitudes:

RQL	RDQL
SELECT X, @P FROM {X} @P {Y} WHERE Y like "Vincent"	SELECT ?X, ?P WHERE (?X, ?P, ?Y) and ?Y == "Vincent"

RQL	RDQL
SELECT painter, painting, tech FRP, {PAINTER} cult:paints {painting}. cult:technique {tech} USING namespace cult = http://www.icom.com/schema.rdf#	SELECT ?painter, ?painting, ?tech WHERE (?painter, <cult:paints>, ?painting), (?painting, <cult:technique>, ?tech) USING cult for <http://www.icom.com/schema.rdf#>

SeRQL

SeRQL es un lenguaje declarativo de consulta de instancias y esquemas RDF, que aprovecha todas las características del modelamiento RDF. Fué desarrollado por la empresa holandesa Aidministrator. SeRQL es soportado por el sistema Sesame.

Una consulta SeRQL se construye típicamente con un conjunto de seis cláusulas. Éstas son: SELECT, FROM, WHERE, LIMIT, OFFSET y USING NAMESPACE. Es fácil reconocer las primeras cinco cláusulas del SQL, pero su uso es diferente. Para las llamadas “construct queries”, las cláusulas son las mismas, con la excepción de la primera, que sustituiremos por la cláusula CONSTRUCT. Por tanto, la primera cláusula será SELECT o CONSTRUCT. Con SELECT, podemos indicar que valores obtendremos y en que orden, en cambio, con CONSTRUCT, indicaremos que triples deseamos obtener.

La cláusula FROM es opcional y determina la ruta de la expresión, es decir, define el camino o ruta de un grafo RDF, el cual es relevante para la consulta. Hay que hacer notar que cuando la cláusula FROM no se incluye, se devuelven constantes especificadas en las cláusulas SELECT o CONSTRUCT.

La cláusula WHERE es opcional y puede tener operadores lógicos. Las cláusulas LIMIT y OFFSET también son optativas. Pueden ser usadas separada o conjuntamente en orden, para obtener un subconjunto de la consulta. Su uso es muy similar a las cláusulas LIMIT y OFFSET de SQL.

Finalmente, la cláusula USING NAMESPACE es también opcional y puede contener declaraciones de namespaces.

Algunos ejemplos de este lenguaje son los siguientes:

```
SELECT C
FROM {C} rdf:type {rdfs:Class}
```

```
SELECT *
FROM {S} rdfs:label {O}

SELECT DISTINCT *
FROM {Country1} ex:borders {} ex:borders {Country2}
USING NAMESPACE
   ex = <http://example.org/things#>

CONSTRUCT {Parent} ex:hasChild {Child}
FROM {Child} ex:hasParent {Parent}
USING NAMESPACE
   ex = <http://example.org/things#>

CONSTRUCT
   {Artist} rdf:type {ex:Painter};
         ex:hasPainted {Painting}
FROM
   {Artist} rdf:type {ex:Artist};
         ex:hasCreated {Painting} rdf:type {ex:Painting}
USING NAMESPACE
   ex = <http://example.org/things#>

CONSTRUCT *
FROM {SUB} rdfs:subClassOf {SUPER}

SELECT *
FROM {X} Y {Z}
WHERE FALSE

SELECT Country
FROM {Country} ex:population {Population}
WHERE Population < "1000000"^^xsd:positiveInteger
USING NAMESPACE
   ex = <http://example.org/things#>
```

SPARQL[43]

Es un lenguaje de consulta capaz de obtener información desde grafos RDF. Es la propuesta de estándar del W3C. Al igual que SeRQL, es soportado por Sesame y Jena. Proporciona facilidades para:

- Extraer información en forma de URIs, nodos blancos y literales.
- Extraer subgrafos RDF.
- Construir nuevos grafos RDF basados en los grafos incluidos en la consulta.

[43] http://www.w3.org/TR/2005/WD-rdf-sparql-query-20050721/

Al igual que los lenguajes de acceso a datos, SPARQL es adecuado para el uso en local y remoto.

SPARQL se basa en patrones de grafos. Los patrones más simples son las tripletas, los cuales son como las tripletas RDF pero con la posibilidad de tener una variable en el sujeto, predicado u objeto.

Ejemplo de consulta SPARQL

Veremos a continuación un ejemplo en el que se muestra una consulta SPARQL, la cual encontrará el título de un libro desde la información dada por un grafo RDF. La consulta consta de dos partes, la cláusula SELECT y la cláusula WHERE. La primera identifica las variables que aparecerán en el resultado de la consulta, y la cláusula WHERE tiene un patrón triple:

Sentencia RDF (en notación N3):

```
<http://example.org/book/book1> <http://purl.org/dc/elements/1.1/title>
"SPARQL Tutorial"
```

Consulta:

```
SELECT ?title
WHERE
{
  <http://example.org/book/book1> <http://purl.org/dc/elements/1.1/title> ?title .
}
```

Resultado:

title
“SPARQL Tutorial”

Las variables en las consultas SPARQL tienen un alcance global. Las variables se indican por “?”. Hay que tener en cuenta que “?” no forma parte de la variable. “$” es una alternativa a “?”. En una consulta que incluya $abc o ¿abc, se trata de la misma variable.

En SPARQL se pueden hacer con sintaxis distintas, pero con el mismo resultado. Los tres ejemplos siguientes expresan la misma consulta:

```
PREFIX  dc: <http://purl.org/dc/elements/1.1/>
SELECT  ?title
WHERE   { <http://example.org/book/book1> dc:title ?title }
```

```
PREFIX  dc: <http://purl.org/dc/elements/1.1/>
PREFIX  : <http://example.org/book/>
SELECT  $title
WHERE   { :book1  dc:title  $title }
```

```
BASE    <http://example.org/book/>
PREFIX  dc: <http://purl.org/dc/elements/1.1/>
SELECT  ?title
WHERE   { <book1>  dc:title  ?title }
```

Comparativa

Esta comparativa la realizaremos sobre seis lenguajes de consulta RDF. Estos lenguajes son: RQL, RDQL, SeRQL, TRIPLE, N3 y Versa. En este capítulo hemos visto algunas características de los tres primeros. Seguidamente realizaremos una breve introducción sobre los tres últimos lenguajes citados.

Lenguaje TRIPLE, N3 y Versa

El lenguaje TRIPLE se basa en consultas y reglas del lenguaje. El lenguaje es derivado de F-Logic. Los Triples RDF (S, P, O) son representados en F-Logic como expresiones S[P->0] y puede ser jerarquizado. Por ejemplo, la expresión S[P1->01, P2->02 [P3->03]] corresponde a tres triples RDF, (S, P1, 01), (S, P2, 02) y (02, P3, 03).

Triple no distingue entre consultas y reglas. Triple no codifica una semántica fija de RDF.

Notation3 (N3) provee texto basado en sintaxis RDF. Por lo tanto, el modelo de datos de N3 se forma con el modelo de datos RDF. Además, N3 define reglas, las cuales usan una sintaxis especial, por ejemplo: ?y rdfs:label “foo” => ?y a :QueryResult.

Versa esta diseñado para ser integrado en otros lenguajes de programación. Versa permite tratar y consultar nodos y arcos en RDF. Utiliza una sintaxis simple y expresiva. Utiliza constructores de los namespaces de XML.

El escenario

Llegados a este punto, estamos en disposición de comenzar la comparativa. Para ello, se presentará una serie de casos que nos servirán para evaluar los seis lenguajes propuestos para esta comparativa.

Para realizarla, se usará como ejemplo, un conjunto de datos que describen un escenario simple del dominio de la investigación de la informática, modelando personas, publicaciones y una pequeña jerarquía.

```
df:RDF xml:base="http://www.aifb.uni-karlsruhe.de/WBS/pha/rdf-query/sample.rdf"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
```

```
 xmlns:acm="http://daml.umbc.edu/ontologies/topic-ont#"
 xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns="http://www.aifb.uni-
 karlsruhe.de/WBS/pha/rdf-query/sample.rdf#">
dfs:Class rdf:ID="Publication" />
dfs:Class rdf:ID="InProceedings">
dfs:subClassOf rdf:resource="#Publication" />
 </rdfs:Class>
dfs:Class rdf:ID="Person">
dfs:subClassOf rdf:resource="#Human" />
 </rdfs:Class>
dfs:Class rdf:ID="Human">
dfs:subClassOf rdf:resource="#Person" />
 </rdfs:Class>
dfs:Class rdf:ID="Topic" />
 <!--
need multiple inheritance, cycles (implies equivalence) and multiple instantiation
 -->
df:Property rdf:about="#author">
dfs:domain rdf:resource="#Publication" />
dfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq" />
 </rdf:Property>
df:Property rdf:about="#title">
dfs:domain rdf:resource="#Publication" />
dfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#STRING" />
 </rdf:Property>
df:Property rdf:about="#pages">
dfs:domain rdf:resource="#Publication" />
dfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#INTEGER" />
 </rdf:Property>
df:Property rdf:about="#year">
dfs:domain rdf:resource="#Publication" />
 <!--
please note the missing range restriction
 -->
 </rdf:Property>
df:Property rdf:about="#isAbout">
 <!--
please note the missing domain restriction
 -->
dfs:range rdf:resource="#Topic" />
 </rdf:Property>
df:Property rdf:about="#name">
dfs:domain rdf:resource="#Person" />
dfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#STRING" />
 </rdf:Property>
df:Property rdf:about="#email">
dfs:domain rdf:resource="#Person" />
dfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#STRING" />
 </rdf:Property>
erson rdf:about="http://www.aifb.uni-karlsruhe.de/WBS/rvo">
df:type rdf:resource="#Human" />
ame>Raphael Volz</name>
 </Person>
erson rdf:about="http://www.aifb.uni-karlsruhe.de/WBS/pha">
ame>Peter Haase</name>
 </Person>
uman rdf:about="http://www.aifb.uni-karlsruhe.de/WBS/aeb">
ame>Andreas Eberhart</name>
 </Human>
df:Description rdf:about="http://www.cs.vu.nl/~jbroeks/">
ame>Jeen Broekstra</name>
mail>jbroeks@cs.vu.nl</email>
 </rdf:Description>
Proceedings rdf:about="#Paper">
itle>An Overview of RDF Query Languages</title>
uthor>
df:Seq>
 <!--
Here we have both a blank node and a sequence
```

```
-->
df:li rdf:resource="http://www.cs.vu.nl/~jbroeks/" />
df:li rdf:resource="http://www.aifb.uni-karlsruhe.de/WBS/aeb" />
df:li rdf:resource="http://www.aifb.uni-karlsruhe.de/WBS/pha" />
df:li rdf:resource="http://www.aifb.uni-karlsruhe.de/WBS/rvo" />
 </rdf:Seq>
 </author>
ages>08</pages>
<!--
please note the lexical value of 08, the data value of 8
-->
ear>2004</year>
sAbout rdf:ID="reified_triple"
  rdf:resource="#ACMTopic/Information_Systems/Database_Management/Languages/Query_
  Languages" />
    - <!--
      please note that the isAbout statement is reified,
               stating that Peter has classified this publication
-->
 </InProceedings>
df:Description rdf:about="#reified_triple">
lc:creator rdf:resource="http://www.aifb.uni-karlsruhe.de/WBS/pha" />
 </rdf:Description>
cm:Topic rdf:about="#ACMTopic/Information_Systems">
dfs:label xml:lang="en">Information Systems</rdfs:label>
dfs:label xml:lang="de">Informationsysteme</rdfs:label>
cm:SubTopic rdf:resource="#ACMTopic/Information_Systems/Database_Management" />
 </acm:Topic>
cm:Topic rdf:about="#ACMTopic/Information_Systems/Database_Management">
dfs:label xml:lang="en">Database Management</rdfs:label>
dfs:label xml:lang="de">Datenbankmanagement</rdfs:label>
cm:SubTopic
  rdf:resource="#ACMTopic/Information_Systems/Database_Management/Languages" />
 </acm:Topic>
cm:Topic rdf:about="#ACMTopic/Information_Systems/Database_Management/Languages">
dfs:label xml:lang="en">Languages</rdfs:label>
dfs:label xml:lang="de">Sprachen</rdfs:label>
cm:SubTopic
  rdf:resource="#ACMTopic/Information_Systems/Database_Management/Languages/Query_
  Languages" />
 </acm:Topic>
cm:Topic
  rdf:about="#ACMTopic/Information_Systems/Database_Management/Languages/Query_La
  nguages">
dfs:label xml:lang="en">Query Languages</rdfs:label>
dfs:label xml:lang="de">Anfragesprachen</rdfs:label>
 </acm:Topic>
 </rdf:RDF>
```

El ejemplo cubre las características principales del modelo de datos de RDF. Incluye una jerarquía de clases con herencia múltiple, tipos de datos, recursos con instancias múltiples, reificación, colecciones de recursos, etc. Estas características se explotarán en los casos siguientes:

N3	Triple
{ ?y s:author ?z. ?z ?p ?x. ?x a s:Person; s:name ?result. } => { ?result a :QueryResult.}.	FORALL C,X,A,N <- (s:Paper[s:author->C] AND C[X->A] AND A[s:name->N])@rdfschema(s:ln).
RDQL	**Versa**
SELECT ?n WHERE <s:Paper>, <s:author>, ?c), (?c, ?collection, ?a), (?a, <s:name>,	QUERY=((s:Paper - s:author -> *) - properties(.) -> *) - s:name -> *

?n) USING s FOR ...

SeRQL	RQL
select PersonName from {X} <s:author> {} <rdfs:member> {} <s:name> {PersonName} using namespace s = <!...>	select PersonName from {X} s:author {y}. rdfs:member {z}. s:name {PersonName} using namespace s = ... rdfs = ...

El namespace *s* está limitado al de los datos de muestra.

Casos

Caso - Grafo

Debido a la naturaleza de los grafos basados en RDF, una de las características de la mayoría de los lenguajes de consulta es el soporte para emparejar el grafo.

- **Path expressions.** La característica principal usada para alcanzar este emparejamiento es una supuesta path expressions, que se utiliza típicamente para atravesar un grafo. Un path expressions puede ser descompuesta y a menudo implementada por partes. No es ninguna sorpresa que las path expressions son ofrecidas en varias formas sintacticas.

Query	RDQL	Triple	SeRQL	Versa	N3	RQL
Path	Sí	Sí	Sí	Sí	Sí	Sí

  En este caso, es soportado por los seis lenguajes. Dará como resultado los nombres de los autores de la publicación X.

- **Optional path expressions.** Un grafo RDF representa un modelo de datos semiestructurado. Su estructura es débil, por lo que la representación de la información es irregular e incompleta. Por lo tanto, los lenguajes de consulta de RDF deben proporcionar medios que se ocupen de las irregularidades y de la información incompleta. Una irregularidad particular, que será considerada en la pregunta siguiente, es que un valor dado puede o no estar presente.

  ¿Cuáles son los nombres, y en el caso de que se sepa, el e-mail de los autores de todas las publicaciones?.

Query	RDQL	Triple	SeRQL	Versa	N3	RQL
Optional Path	No	No	Sí	Sí	No	Parcial

  Desafortunadamente, solo dos lenguajes, SeRQL y Versa, proveen los medios necesarios para ocuparse de la información incompleta. Por

ejemplo, el lenguaje SeRQL proporciona las supuestas path expressions, denotadas por los corchetes, en caso de que sean irregulares:

```
SELECT PersonName, Email FROM
{X} <ns3:author> {} <rdfs:member>
{p} <ns3:name> {PersonName};
[<ns3:email> {Email}]
USING NAMESPACE
ns3 = <!...>
```

Generalmente, tales expresiones opcionales de las path expressions pueden ser simuladas, si un lenguaje proporciona la unión y la negación. Por ejemplo, RQL proporciona una respuesta correcta, unificando los resultados de dos consultas, donde el primer argumento de la unión recupera a todas las personas con una dirección de e-mail y el segundo recuperará a todas aquellas personas sin dirección de correo.

```
( select PersonName, Email
from {X} s:author {y}. rdfs:member {z}.
s:name {PersonName}, {z} s:email {Email}
) union (
select PersonName, NULL
from {X} s:author {y}. rdfs:member {z}.
s:name {PersonName}
where not ( z in select X from {X}
s:email {e} ) )
using namespace
s = ... , rdfs = ...
```

N3, Versa y Triple, provee otros operadores que pueden obtener resultados similares pero con algunas respuestas duplicados.

Caso - Relacional

RDF es frecuentemente usado como un modelo de estructuras relacionales. DE hecho, las tablas n-arias, tales como las que se encuentran en un modelo de datos relacional son fácilmente codificadas en triples RDF.

- **Operaciones básicas del álgebra relacional.** En el modelo de datos relacional, varias operaciones básicas de álgebra son consideradas, por ejemplo, selección, proyección, producto cartesiano, unión, etc. Estas operaciones pueden ser combinadas para expresar otras operaciones, tales como intersección, varias formas de unión, etc.

  Existen tres operaciones básicas, selección, proyección y producto, las cuales son soportadas por todos los lenguajes y son usados por las *path expressions*. Nosotros nos centraremos en otras dos operaciones básicas, la unión y la diferencia.

La **unión** es proporcionada por RQL. Versa también contiene un operador explícito, N3 y triple pueden simular la unión con reglas.

Query	RDQL	Triple	SeRQL	Versa	N3	RQL
Union	No	Sí	No	Sí	Sí	Sí

Devolverá los títulos de todas las publicaciones.

La **diferencia** es una forma especial de negación. Devolverá todos los registros que no son títulos de publicaciones.

Query	RDQL	Triple	SeRQL	Versa	N3	RQL
Difference	No	No	No	Parcial	No	Sí

Mientras la diferencia se describe en la documentación de Versa, ésta no está implementada. En cambio, RQL provee una implementación de este operador. La siguiente consulta, devuelve una respuesta correcta en RQL:

```
( select title
from s:Topic{T}. rdfs:label {title}
) minus (
select title
from s:Publication{P}. s:title {title} )
using namespace
s = ... , rdfs = ...
```

- **Quantification.** Un predicado universal es satisfecho si todos los valores del predicado han sido satisfechos. En este caso se devolverá las personas que son autores de todas las publicaciones.

Query	RDQL	Triple	SeRQL	Versa	N3	RQL
Universal	No	No	No	No	No	Sí

Solo el lenguaje RQL satisface el “universal quantification” necesario para responder a esta consulta:

```
SELECT person
FROM s:Person\{person}
WHERE FORALL z IN
(SELECT x FROM s:Publication\{x} )
SUCH THAT EXISTS p IN
(SELECT Y FROM \{z} s:author \{}. rdfs:member \{y})
SUCH THAT person = p
USING NAMESPACE s = ..., rdfs = ...
```

Caso - Agregación y agrupación

Las funciones agregadas computan un valor escalar de un conjunto de valores. Estas funciones son generalmente necesarias para contar el número de

valores. Por ejemplo, son especialmente necesarias para identificar el mínimo y máximo de un conjunto de valores. Adicionalmente, la agrupación permite agrupar los valores obtenidos.

- **Aggregation.** Un caso especial de agregación se probó en la consulta siguiente, la cuál es un simple conteo del número de elementos en un conjunto, como por ejemplo, el número de autores de una publicación.

Query	RDQL	Triple	SeRQL	Versa	N3	RQL
Counting	No	No	No	Sí	Sí	Sí

Counting es soportado por N3, Versa y RQL. La siguiente regla N3 nos dará un resultado satisfactorio.

```
{?y.sam:author math:memberCount ?result .} =>
{:Query :Result ?result}.
```

- **Groupping.** Ninguno de los lenguajes de consulta que estamos tratando en esta comparativa permite la agrupación, tal y como provee la cláusula GROUP BY de SQL.

Caso - Recursión

Las consultas recursivas aparecen a menudo en los sistemas de información, típicamente si la relación subyacente es transitiva. En el conjunto de datos de esta comparación, los topics se definen junto con sus subtopics, donde la característica del subtopic se puede considerar como transitiva.

En este caso, se retornaran recursivamente todos los subtopics del topic “Information Systems”:

Query	RDQL	Triple	SeRQL	Versa	N3	RQL
Recursion	No	Sí	No	Sí	Sí	No

Tanto Triple, como N3, son sistemas basados en reglas, las cuáles pueden apoyar naturalmente la repetición requerida a través de la definición de reglas auxiliares.

```
FORALL O,P,V O[acm:SubTopic->V] <-
EXISTS W (O[acm:SubTopic->W] AND W[acm:SubTopic->V])@default:ln.
FORALL Y <-
('...#ACMTopic/':Information_Systems[acm:SubTopic->Y])@default:ln.
```

Versa, no soporta la recursión, pero si dispone de la palabra reservada “traverse”, la cual efectúa una interpretación transitiva de la propiedad especificada.

```
traverse(@"...#ACMTopic/Information_Systems",
acm:SubTopic, vtrav:forward, vtrav:transitive )
```

Caso - Reificación

Agrega una meta-capa al grafo y permite tratar las declaraciones como recursos de ellos mismos, tales como declaraciones que se pueden hacer sobre otras declaraciones. En datos de muestra la reificación se utiliza para indicar quién incorporó los datos de la publicación.

Se retornará la persona que ha clasificado la publicación X.

Query	RDQL	Triple	SeRQL	Versa	N3	RQL
Reification	Parcial	Parcial	Sí	Parcial	No	Parcial

Solo Triple, soporta nativamente la reificación con una sintaxis especial, donde las declaraciones son encerradas entre corchetes y usándolo dentro de otra declaración.

```
FORALL V,W,X,Y,Z <- ( V[W-><X[Y->Z]>] ).
```

Sin embargo, podríamos solo utilizar esta característica en la sintaxis nativa de la F-Logic, puesto que las declaraciones de los datos deificados en la muestra, no fueron analizadas correctamente.

Se puede expresar una consulta en Versa y RDQL usando la característica rdf:subject de las declaraciones deificadas. Esto permite tratar el caso de la reificacion como cualquier otra consulta. El siguiente ejemplo en Versa lo demuestra:

```
(all() |- rdf:subject ->
@"../versa-sample.rdf#Paper") - dc:creator -> *
```

Semejantemente, SeRQL y RQL tartan declaraciones reificadas como nodos en un grafo, que se puede tratar como expresiones normales de la path expressions, confiando en la normalización de RDF. N3, sin embargo, no puede representar sintácticamente la reificación.

Caso - Colecciones y contenedores

RDF permite a grupos de entidades usar las colecciones (un grupo cerrado de entidades) y los contenedores, llamados *Bag, Sequence* y *Container*, que proporcionan un significado previsto de los elementos del contenedor. Un lenguaje de consulta debe poder recuperar elementos individuales así como colecciones y contenedores de elementos, junto con el orden de la información.

En nuestro caso, se retornara el primer autor de la publicación X.

Query	RDQL	Triple	SeRQL	Versa	N3	RQL
Sequences	Parcial	Parcial	Parcial	Parcial	Parcial	Parcial

Aunque ninguno de los lenguajes de proporcionan un apoyo explícito para el proceso de contenedores, en todos los lenguajes de consulta es posible

realizar una consulta para un elemento particular en un contenedor con la ayuda del predicado especial <rdf:_n>, el cuál permite para tratar el elemento nth en un contenedor. Sin embargo, este acercamiento no trabajaría para recuperar el último elemento de un contenedor (a menos que su tamaño se sepa antes).

Ninguno de estos lenguajes de consulta proporcionan la ayuda explícita para ordenar o clasificar de elementos, a excepción de Versa que ofrece el operador especial *sort*.

Caso - Namespaces

Los namespaces son una parte integral de cualquier lenguaje de consulta para los datos basados en Web. En los ejemplos presentados, se ha visto como los sistemas permiten introducir abreviaturas de los namespace para mantener las consultas. Este caso evalúa qué operaciones son posibles sobre los mismos namespaces. Dado un sistema de recursos, puede ser que sea interesante preguntar todos los valores de un cierto namespace o de un patrón sobre el mismo namespace.

En este caso se devolverán los recursos de los namespaces que comienzan por “http://www.aifb.unikarlsruhe.de/”.

Query	RDQL	Triple	SeRQL	Versa	N3	RQL
Namespace	Parcial	No	Sí	No	Sí	Sí

SeRQL, RQL y N3 permiten predicados concordantes con el modelo en URIs de manera semejante en cuanto a literales, que permite para realizar la consulta siguiente para RDQL:

```
select ?x, ?y, ?z
where (?x, ?y, ?z)
AND ?x =~ "http://www.aifb.uni-karlsruhe.de/*"
```

Caso - Lengua

RDF permite el uso de etiquetas al estilo del lenguaje XML. Las etiquetas XML que incluye un literal RDF puede llevar opcionalmente un atributo xml:lang. El valor de “lang” identifica la lengua usada en el texto del literal. Este caso examina si los distintos lenguajes tienen esta característica.

Query	RDQL	Triple	SeRQL	Versa	N3	RQL
Language	No	No	Sí	No	No	No

SeRQL proporciona una función especial para recuperar la información de la lengua de un literal:

```
select deLabel
from fg <rdfs:label> fdeLabel, enLabelg
where lang(deLabel) = "de" and lang(enLabel) = "en" and
```

```
label(enLabel) = "Database Management"
```

Caso - Literales y tipos de datos

Los literales se utilizan para identificar valores tales como números y fechas por medio de una representación léxica. Un lenguaje de consulta RDF debe apoyar los tipos de datos del esquema XML.

En este caso obtendremos la página '08' de todas las publicaciones.

Query	RDQL	Triple	SeRQL	Versa	N3	RQL
Lexical Space	Sí	Sí	Sí	Sí	Sí	Sí

Ahora, trataríamos de obtener la página 8 de todas las publicaciones

Query	RDQL	Triple	SeRQL	Versa	N3	RQL
Value Space	Parcial	No	Sí	No	No	No

Todos los lenguajes de consulta pueden consultar el espacio léxico, pero la mayoría de los lenguajes no poseen ayuda preliminar para los tipos de datos y no apoyan la distinción entre espacio léxico y valor.

Caso - Entailment

El vocabulario de un esquema RDF soporta el entailment si la información es implicita.

Con la consulta siguiente evaluamos la ayuda de los lenguajes de consulta para el entailment del esquema de RDF. Se espera que la consulta devuelva, no solo los recursos para los cuales la clase del miembro es proporcionada explícitamente, sino también la clase del miembro que puede ser dado de las reglas del entailment.

Se retornará todas las instancias de los miembros de la clase Publication.

Query	RDQL	Triple	SeRQL	Versa	N3	RQL
Entailment	No	Parcial	Sí	No	Parcial	Sí

RDQL y Versa no lo soporta, SeRQL y RQL lo soportan directamente. Tanto N3 como Triple lo soportan parcialmente. Estos últimos lenguajes requieren una axiomatización de la semántica de RDFS, por ejemplo, un conjunto de reglas.

En el siguiente ejemplo podemos observar como se realiza ésta consulta en RQL:

```
select publications
from ns3:Publicationfpublicationsg
using namespace
ns3 = <!http://www.aifb.uni-karlsruhe.de/WBS/pha/rdf-query/sample.rdf#>
```

Conclusiones

Es importante definir un lenguaje de consulta para documentos RDF. Esto debería ser una prioridad en términos de estandarización. Las consultas son fundamentales en cualquier aplicación de Web semántica. Juzgando el impacto de SQL en el terreno de las base de datos, la estandarización definitiva debe ayudar a la adopción de motores de consulta de RDF, hace el desarrollo mucho más fácil, y ayudará así a la Web semántica en general.

Se han evaluado seis lenguajes de consulta con metas y filosofías diferentes. Del análisis efectuado, se ha identificado un pequeño conjunto de criterios que difieren de unos lenguajes a otros.

Una distinción dominante es el soporte para la semántica del esquema RDF. Lenguajes como N3 y triple no hacen una distinción terminante entre las consultas y las reglas. Así, un programa lógico que, representando una semántica deseada, en este caso RDFS, puede suplir opcionalmente a una consulta. SeRQL y RQL soporta internamente semántica RDFS. RDQL ignora la semántica RDFS.

La característica “orthogonality” es muy deseable, puesto que permite combinar un sistema de operadores simples en construcciones de gran alcance. RQL, SeRQL, N3 y Versa soportan esta característica. Versa usa un conjunto de recursos como estructura de datos básica, mientras que RQL, N3 y SeRQL operan con grafos. Triple puede simular la característica “orthogonality” gracias a las reglas, mientras que RDQL no la soporta.

RQL y SeRQL aparecen como los lenguajes mas completos, luego tendríamos a Versa y N3, seguidos por Triple y finalmente RDQL.

Finalmente, consideramos la legibilidad y la utilidad de un lenguaje. Obviamente, esto depende mucho del gusto personal. Sintácticamente, RQL, RDQL, y SeRQL son muy similares, debido a su herencia, a SQL / OQL. Triple y N3 comparten la característica de las reglas. La sintaxis de Triple, permite algunas variantes sintácticas que pueden resultar agradables. El estilo de Versa es absolutamente diferente.

Capítulo 3

Estructura y organización de los SGBD adecuados para RDF

Este capítulo tiene como objetivo principal hacer un pequeño estudio sobre la estructura y organización de los SGBD que trabajan con información basada en RDF.

Introducción

Para desarrollar aplicaciones basadas en RDF, OWL o lenguajes similares se precisan librerías para leer y procesar las ontologías definidas en estos lenguajes. Basta dar un vistazo a esta lista de recursos RDF: http://www.ilrt.bristol.ac.uk/discovery/rdf/resources/#sec-tools, para comprobar la multitud de parsers y herramientas que se han desarrollado al efecto. Sin embargo, con diferencia, el parser de RDF y OWL más popular es Jena2[44], desarrollado por Hewlett Packard, que permite leer, recorrer y modificar grafos tanto RDF como OWL desde un programa Java. Jena2 permite además

[44] http://www.hpl.hp.com/semweb/jena2.htm

guardar las ontologías tanto en RDF textual como en formato de base de datos, lo que es importante para grafos muy grandes. Otro sistema muy conocido de similares características para RDF y OWL es Sesame[45], desarrollado en el proyecto europeo Ontoknowledge[46] y actualmente distribuido por Aidministrator. Jena2 incluye además un motor de consultas para RDQL, y Sesame ofrece lo propio para RQL y SeRQL. Las últimas versiones de Jena2 y Sesame han incorporado también motores de razonamiento para las expresiones lógicas de OWL.

Sesame

Sesame fue desarrollado por la empresa holandesa Aidministrator. Tiene una arquitectura desarrollada para un eficiente almacenamiento y consulta de grandes cantidades de metadatos en RDF y RDF Schema. En el caso de Sesame, el diseño y la implementación son independientes de cualquier dispositivo de almacenamiento, de forma que permite su empleo en una variedad de dispositivos de almacenamiento tales como bases de datos relacionales, almacenamiento de triples o bases de datos orientadas a objeto, sin tener que cambiar el motor de consulta u otros módulos funcionales. Sesame ofrece soporte de control de concurrencia, exportación independiente de RDF y RDFS y un motor de consulta para RQL.

Permite añadir y eliminar información escrita en RDF y puede almacenar esta información en cualquier base de datos. Sesame soporta como lenguajes de consulta a RQL, RDQL y SeRQL (Sesame RDF Query Language), para acceder a la información. Permite la interoperabilidad con un razonador de lógica descriptiva, el cual debe ser otro de los participantes en cualquier sistema emparejador.

Arquitectura

Uno de sus componentes principales es SAIL (Storage And Inference Layer) que es el nivel que se encarga de interactuar con los manejadores de bases de datos (DBMS) permitiendo de este modo mantener la arquitectura de Sesame independiente del DBMS. SAIL es una API que ofrece métodos RDF específicos a sus “clientes” y traduce estos métodos a llamadas a un DBMS específico. Los módulos funcionales de Sesame son “clientes” de SAIL, actualmente existen tres módulos: el motor de consultas RQL, el módulo de administración RDF y el módulo de exportación RDF.

Dependiendo del ambiente en el cual se implante, Sesame ofrece diversas formas de comunicación, ya sea vía HTTP para contexto WEB, o tal vez Remote Method Invocation (RMI) o Simple Object Acces Protocol (SOAP).

[45] http://sesame.aidministrator.nl/
[46] http://ontoknowledge.semanticweb.org/

Para permitir una máxima flexibilidad, la dirección real de estos protocolos, se ha puesto fuera del alcance de los módulos funcionales. En cambio, proporcionan protocolos intermediarios entre los módulos y sus clientes.

La adición de protocolos hace fácil la conexión de Sesame con diferentes entornos.

La arquitectura de Sesame se ha diseñado con extensibilidad y adaptabilidad. La posibilidad para utilizar otras clases de repositorios se ha mencionado antes. La adición de los módulos o protocolo adicionales también es posible.

Módulos funcionales de Sesame

- **Módulo de consulta RQL.** La versión de RQL de Sesame está caracterizada por cumplir con los estándares de la W3C, sin embargo no soporta la tipificación de datos. El proceso de consulta se describe en la siguiente figura, después del parsing y la construcción del árbol de consulta, pasa a un optimizador de consultas que trasforma la consulta en un modelo equivalente pero mas eficiente. La optimización consta de un conjunto de heurísticas, éstas preevaluaciones de optimización no dependen de ningún método de almacenamiento en particular.

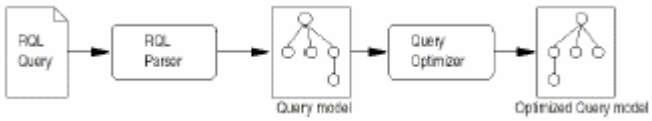

Figura 3 – Parsing y optimización del modelo

El modelo optimizado es subsecuentemente evaluado siguiendo la estructura de árbol generada, cada objeto representa una unidad básica en la consulta original y se evalúa así mismo extrayendo información por medio de SAIL cuando lo necesita. La gran ventaja de este enfoque es que los resultados se retornan de la misma forma subsecuente, en vez de construir primero la consulta completa en memoria.

El grueso de las consultas son resueltas por el motor RQL, por ejemplo cuando una consulta contiene una operación de semi-join, cada subconsulta es evaluada y la operación de semi-join es entonces ejecutada por el motor de consultas sobre los resultados.

Otra alternativa sería aprovechar la optimización a partir del DBMS, pero esto haría que el proceso de consulta fuera dependiente de un DBMS específico.

- **Módulo de administración.** Para posibilitar la inserción de datos RDF e información de esquemas en un repositorio, Sesame brinda el módulo de administración, ofreciendo dos operaciones fundamentales:

    - Adición incremental de datos y esquemas RDF.

- Limpiar un repositorio.

Aclarando que una opción borrado (por sentencias) funcional aun no esta disponible.

El modulo puede extraer información de una fuente RDFS, generalmente en XML serializado, y se hace un parsing usando ARP RDF, que entrega información al módulo de administración de la forma básica (objeto, atributo, valor), y luego al repositorio por medio de la capa SAIL reportando los errores que pudieron haber ocurrido.

- **Módulo de exportación RDF.** Permite exportar los contenidos de un repositorio en formato XML serializado. El objetivo de este módulo es combinar Sesame con otras herramientas RDF, dado que la mayoría reconoce el formato.

  Además de permitir la exportación de datos o también esquemas, dependiendo de si es necesario la semántica o no.

API SAIL

La API SAIL consta de un conjunto de interfaces Java que han sido específicamente diseñados para el almacenamiento y extracción de información basada en RDFS, a continuación se presenta los principios mas relevantes de la API de SAIL:

- Definir una interfaz básica para almacenar, borrar y recuperar RDF y RDFS.

- Una abstracción del mecanismo actual de almacenamiento, debiendo ser aplicable a RDBMS, sistema de archivos, almacenamiento en memoria, etc.

- Aplicable a hardware como PDAs, pero ofreciendo suficiente libertad para la optimización en el caso de gran cantidad de información, por ejemplo : clusters.

- Ser extensible a otros lenguajes basados en RDF como DAML+OIL.

Una característica importante es que la API SAIL ofrece métodos para consulta de clases y propiedades, así como restricciones de dominio y rango. Lo que no ocurre con otros productos que están enfocados al conjunto de triplas, dejando el trabajo de interpretación al usuario. Esta ventaja se basa en la fuerte relación entre la eficiencia de la inferencia y el modelo de almacenamiento empleado.

Otra característica de SAIL, se relaciona a su tamaño “ligero”, dado que solamente implementa cuatro interfaces predefinidas, ofreciendo almacenamiento y recuperación, así como funcionalidad y soporte transaccional.

La implementación actual se llama SQL92SAIL que se base en SQL92 considerando los alcances necesarios para el modelo RDF.

Jena2

Jena2 es un marco de trabajo desarrollado en java (y para java), para construir aplicaciones para la Web semántica. Provee un ambiente de programación para RDF, RDFS y OWL, incluyendo un motor de inferencia basado en reglas. Es un proyecto OpenSource y crece con la ayuda del programa de Web semántica de "HP Labs"[47].

El framework de Jena2 incluye:

- Un API para RDF.
- Lectura y escritura de documentos en formato RDF/XML, N3, y N-Triples.
- Un API para OWL.
- Almacenamiento persistente y en memoria.
- RDQL, un lenguaje de consulta para RDF.

El corazón de Jena la API RDF, la cual soporta la creación, manipulación y consulta de grafos RDF. La API también soporta diferentes tecnologías de almacenamiento. Plugins que permiten la lectura y escritura automática para diferentes lenguajes que los desarrolladores pueden usar para representar grafos RDF.

RDF API

Desde una perspectiva de API, hay dos formas distintas de ver un grafo RDF. En una vista centrada en afirmaciones, un grafo RDF es un conjunto de triples; donde cada triple identifica el nodo al comienzo del arco, el arco en si, y el nodo al final del arco. Esta vista es conveniente para manipular grafos como un todo, tal como cuando se lee, escribe, o mezclan. En una vista centrada en frames, un grafo RDF es una colección de recursos, cada uno de los cuales tiene propiedades. Esta vista es análoga al paradigma de programación de orientación a objetos que tiene objetos con atributos.

Algunos recursos RDF tienen un comportamiento implícito. Por ejemplo, los contenedores usualmente tienen métodos tales como insert, delete, tamaño, etc. Jena permite a los desarrolladores añadir el comportamiento apropiado a recursos de un tipo específico. La API actual de RDF API provee una primitiva de consulta, un método para extraer subconjuntos de todos los triples que un "objeto selector" selecciona desde un grafo. Una clase selector simple retorna todos los triples que calzan con un patrón dado.

[47] http://www.hpl.hp.com/semweb/

Parser

Los desarrolladores de Jena también mantienen ARP, un parser que lee RDF/XML, el lenguaje estándar para la representación de grafos RDF. Mientras que muchos parser RDF/XML son codificados a mano, un compilador de compiladores (Javacc) genera el parser ARP desde la gramática RDF/XML. ARP usa un XML parser estándar como preprocesador léxico. El parser generado entonces procesa los símbolos que, en efecto, son eventos en SAX, la API estándar para XML.

Writers

La sintaxis de RDF/XML es flexible, permite lo mismo que los grafos RDF a ser escritos en muchos formatos distintos. Aquí se introduce la noción de estilo. El writer RDF/XML básico de Jena toma ventaja de las características de RDF/XML que permiten expresiones más compactas y elegantes, pero puede escribir gráficos de tamaño arbitrario. Un PrettyWriter, de forma distinta, toma una mayor ventaja de la sintaxis disponible para producir salida compacta.

Almacenamiento

Jena contiene tres implementaciones de la API: uno almacena los datos en memoria, otro almacena los datos en bases de datos relacionales, y un tercero usa la base de datos incrustada Berkeley DB, un Software opensource de Sleepycat. Un SPI permite introducir nuevos sistemas de almacenamiento fácilmente. La implementación de bases de datos relacionales usa cualquier base de datos que soporte JDBC. La configuración de tablas permite la especialización de consultas a una base de datos específica. La estructura de tablas de una base de datos es también configurable, una característica que permite realizar ajustes de desempeño para aplicaciones especificas. El almacenamiento en Berkeley DB, al igual que el almacenamiento relacional, es persistente.

Query

RDQL es el lenguaje de consulta para grafos RDF en Jena. Diseñado para ser familiar a muchos usuarios, la sintaxis de RDQL es similar a SQL. RDQL se relaciona de cerca con SquishQL, el cual a su vez está basado en RdfQL, un lenguaje de bases de datos escalables creadas para trabajar con ServiciosWeb semánticos. La cláusula SELECT define las variables que se requieren mostrar en el conjunto resultante. La cláusula WHERE define un patrón de subgrafo en términos de variables y constantes. Cada calce de el patrón con un subgrafo distinto del grafo consultado genera un conjunto de variables mostrables que es incluido en el conjunto resultante. La cláusula AND introduce un filtro en las variables; solamente los resultados que pasan el filtro son incluidos en el conjunto resultante de la consulta. La cláusula USING define abreviaciones para espacios de nombre.

Capítulo 4

Comparativa entre PostgreSQL y MySQL

Hoy en día existen muchas empresas y sitios Web que necesitan mantener de forma eficiente un gran volumen de datos. Muchos de ellos optan por soluciones comerciales, aunque muchas otras confían en el software libre optando por una solución como PostGreSQL o MySQL.

En este capitulo se tratará de hacer una comparativa entre los sistemas de gestión de bases de datos libres más importantes y más usados en la red, los cuales proporcionan soluciones a miles de personas, de forma totalmente gratuita, sin pérdida de eficiencia alguna.

La mayoría de sistemas persistentes de datos RDF acaban usando uno de los sistemas de gestión de bases de datos que vamos a estudiar en el presente capitulo. Estos sistemas persistentes de datos RDF soportan tanto PostgreSQL como MySQL.

Para terminar, se ha realizado una serie de pruebas entre PostgreSQL, MySQL y la API SAIL de Sesame (ver Capítulo 3).

Introducción

Común es la pregunta entre las personas que se adentran por primera vez en el mundo de las bases de datos libres: ¿MySQL o PostGreSQL? En realidad no es una pregunta asociada específicamente a los "novatos", ya que incluso los profesionales dedicados a este campo se realizan muchas veces esta misma pregunta. La verdad es que no es una pregunta fácil de responder, y no carente de grandes controversias.

Veremos las características de estos dos magníficos sistemas de gestión de bases de datos, haciendo una pequeña comparativa entre ellas, con el fin de conducir a la elección más adecuada para cada situación.

PostGreSQL

¿Qué es PostGreSQL?

PostGreSQL es un sistema de gestión de bases de datos objeto-relacional (ORDBMS) basado en el proyecto POSTGRES, de la Universidad de Berkeley. El director de este proyecto es el profesor Michael Stonebraker, y fue patrocinado por Defense Advanced Research Projects Agency (DARPA), el Army Research Office (ARO), el National Science Foundation (NSF), y ESL, Inc.

PostGreSQL es una derivación libre (OpenSource) de este proyecto, y utiliza el lenguaje SQL92/SQL99, así como otras características que comentaremos más adelante.

Fue el pionero en muchos de los conceptos existentes en el sistema objeto-relacional actual, incluido, más tarde en otros sistemas de gestión comerciales. PostGreSQL es un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. A pesar de esto, PostGreSQL no es un sistema de gestión de bases de datos puramente orientado a objetos.

Historia de PostGreSQL

PostGreSQL (llamado también Postgres95) fue derivado del proyecto Postgres, como ya se ha comentado. A sus espaldas, este proyecto lleva más de una década de desarrollo, siendo hoy en día, el sistema libre más avanzado con diferencia, soportando la gran mayoría de las transacciones SQL, control concurrente, teniendo a su disposición varios "language bindings" como por ejemplo C, C++, Java, Python, PHP y muchos más.

La implementación de Postgres DBMS comenzó en 1986, y no hubo una versión operativa hasta 1987. La versión 1.0 fue liberada en Junio de 1989 a unos pocos usuarios, tras la cual se liberó la versión 2.0 en Junio de 1990 debido a unas críticas sobre el sistema de reglas, que obligó a su reimplementación. La versión 3.0 apareció en el año 1991, e incluyó una serie de mejoras como una mayor eficiencia en el ejecutor de peticiones. El resto de versiones liberadas a partir de entonces, se centraron en la portabilidad del sistema. El proyecto se dio por finalizado con la versión 4.2, debido al gran auge que estaba teniendo, lo cual causó la imposibilidad de mantenimiento por parte de los desarrolladores.

En 1994, Andrew Yu y Jolly Chen añadieron un intérprete de SQL a este gestor. Postgres95, como así se llamó fue liberado a Internet como un proyecto libre (OpenSource). Estaba escrito totalmente en C, y la primera versión fue un 25% más pequeña que Postgres, y entre un 30 y un 50% más rápida. A parte de la corrección de algunos bugs, se mejoró el motor interno, se añadió un nuevo programa monitor, y se compiló usando la utilidad GNU Make y el compilador gcc sin necesidad de parchearlo (como había hecho falta en versiones anteriores).

En 1996, los desarrolladores decidieron cambiar el nombre al DBMS, y lo llamaron PostGreSQL (versión 6.0) para reflejar la relación entre Postgres y las versiones recientes de SQL. Se crearon nuevas mejoras y modificaciones, que repercutieron en un 20-40% más de eficiencia, así como la incorporación del estándar SQL92.

Características de PostGreSQL

A continuación se enumeran las principales características de este gestor de bases de datos:

- Implementación del estándar SQL92/SQL99.
- Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP ...), cadenas de bits, etc. También permite la creación de tipos propios.
- Incorpora una estructura de datos array.
- Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes, etc.
- Permite la declaración de funciones propias, así como la definición de disparadores.
- Soporta el uso de índices, reglas y vistas.

- Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.

- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.

¿Qué es lo que le falta?

PostGreSQL es un magnífico gestor de bases de datos, capaz de competir con muchos gestores comerciales, aunque carezca de alguna característica casi imprescindible. Ésta es, bajo mi punto de vista, un conjunto de herramientas que permitan una fácil gestión de los usuarios y de las bases de datos que contenga el sistema. Por otro lado, la velocidad de respuesta que ofrece este gestor con bases de datos relativamente pequeñas puede parecer un poco deficiente, aunque esta misma velocidad la mantiene al gestionar bases de datos realmente grandes, cosa que resulta loable.

Opinión Personal

Tiene prácticamente todo lo que tienen los gestores comerciales, haciéndolo una muy buena alternativa GPL. A pesar de ello, el primer encuentro con este gestor es un poco "duro", ya que la sintaxis de algunos de sus comandos no es nada intuitiva. También resulta engorroso las pequeñas variaciones que presenta este gestor en algunos de los tipos de datos que maneja, siendo el problema más comentado el referente al tipo "serial".

Una vez nos hayamos hecho con su sintaxis y fijándonos en estos pequeños detalles (que por otro lado están totalmente documentados), PostGreSQL es un gestor magnífico, que posee una gran escalabilidad, haciéndolo idóneo para su uso en sitios Web que posean alrededor de 500.000 peticiones por día.

MySQL

¿Qué es MySQL?

MySQL es un sistema de gestión de bases de datos relacional, licenciado bajo la GPL de la GNU. Su diseño multihilo le permite soportar una gran carga de forma muy eficiente. MySQL fue creada por la empresa sueca MySQL AB, que mantiene el copyright del código fuente del servidor SQL, así como también de la marca.

Aunque MySQL es software libre, MySQL AB distribuye una versión comercial de MySQL, que no se diferencia de la versión libre más que en el soporte

técnico que se ofrece, y la posibilidad de integrar este gestor en un software propietario, ya que de no ser así, se vulneraría la licencia GPL.

Este gestor de bases de datos es, probablemente, el gestor más usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración.

Historia de MySQL

MySQL surgió como un intento de conectar el gestor mSQL a las tablas propias de MySQL AB, usando sus propias rutinas a bajo nivel. Tras unas primeras pruebas, vieron que mSQL no era lo bastante flexible para lo que necesitaban, por lo que tuvieron que desarrollar nuevas funciones. Esto resultó en una interfaz SQL a su base de datos, con una interfaz totalmente compatible a mSQL.

Se comenta en el manual[48] que no se sabe con certeza de donde proviene su nombre. Por un lado dicen que sus librerías han llevado el prefijo 'my' durante los diez últimos años. Por otro lado, la hija de uno de los desarrolladores se llama My. No saben cuál de estas dos causas (aunque bien podrían tratarse de la misma), han dado lugar al nombre de este conocido gestor de bases de datos.

Características de MySQL

Las principales características de este gestor de bases de datos son las siguientes:

- Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo.
- Soporta gran cantidad de tipos de datos para las columnas.
- Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP, etc).
- Gran portabilidad entre sistemas.
- Soporta hasta 32 índices por tabla.
- Gestión de usuarios y passwords, manteniendo un muy buen nivel de seguridad en los datos.

[48] http://www.mysql.com/documentation/index.html

¿Qué es lo que le falta?

MySQL surgió cómo una necesidad de un grupo de personas sobre un gestor de bases de datos rápido, por lo que sus desarrolladores fueron implementando únicamente lo que precisaban, intentando hacerlo funcionar de forma óptima. Es por ello que, aunque MySQL se incluye en el grupo de sistemas de bases de datos relacionales, carece de algunas de sus principales características:

- Subconsultas: tal vez ésta sea una de las características que más se echan en falta, aunque gran parte de las veces que se necesitan, es posible reescribirlas de manera que no sean necesarias.

- SELECT INTO TABLE: Esta característica propia de Oracle, todavía no está implementada.

- Triggers y Procedures: Se tiene pensado incluir el uso de procedures almacenados en la base de datos, pero no el de triggers, ya que los triggers reducen de forma significativa el rendimiento de la base de datos, incluso en aquellas consultas que no los activan.

- Transacciones: a partir de las últimas versiones ya hay soporte para transacciones, aunque no por defecto (se ha de activar un modo especial).

- Integridad referencial: aunque sí que admite la declaración de claves ajenas en la creación de tablas, internamente no las trata de forma diferente al resto de campos.

Los desarrolladores comentan en la documentación que todas estas carencias no les resultaban un problema, ya que era lo que ellos necesitaban. De hecho, MySQL fue diseñada con estas características, debido a que lo que buscaban era un gestor de bases de datos con una gran rapidez de respuesta. Pero ha sido con la distribución de MySQL por Internet, cuando más y más gente les están pidiendo estas funcionalidades, por lo que serán incluidas en futuras versiones del gestor.

Opinión Personal

Tras haber probado la PostGreSQL, y viendo las carencias que poseía MySQL, pensé que no merecería la pena ni tan siquiera probarlo, aunque por otro lado, creía que algo debía tener para que hubiera tanta gente que lo use, cuando está a merced de cada uno elegir la base de datos que quiere usar. La verdad es tras haber hecho unas pocas pruebas, mi impresión sobre este gestor mejoró considerablemente.

Para comenzar, el shell de comandos muestra una interfaz más amena y los comandos para gestionar la base de datos son más intuitivos, siendo muchos de ellos sentencias SQL (hay que decir que no dispone de ayuda en línea

sobre las palabras clave de SQL). Por otro lado, la API de PHP para acceder a MySQL era muchísimo más sencilla de usar, teniendo un estilo mucho más natural.

Impresiones en contra, la imposibilidad de usar subconsultas, así como también la definición de vistas, aunque según la documentación oficial, éstas dos características serán incluidas en próximas versiones (en las versiones actuales, se incluyen dos comandos, LEFT JOIN y RIGTH JOIN, que son capaces de suplir las subconsultas en gran parte de los casos, obteniendo, por otra parte, una mayor eficiencia).

La verdad es que aunque estas diferencias son agradables, no llegan a tener una importancia suficiente como para cambiar el gestor que habitualmente solemos usar. Este tipo de cambios deberían estar basados en diferencias en el rendimiento que se nos ofrece, que es lo que se tratará en el siguiente apartado.

Pruebas

PostgreSQL y SAIL

En las primeras pruebas de Sesame se ha usado PostgreSQL, empleando las características de ser un DBMS relacional orientado a objeto, de modo que permite relaciones entre sus tablas. Nuevas tablas son adicionadas a la base de datos cada vez que una nueva clase o propiedad es adicionada al repositorio. En la figura 4 se muestra un ejemplo de RDF Schema y la figura 5 muestra el contenido de la base de datos para este ejemplo.

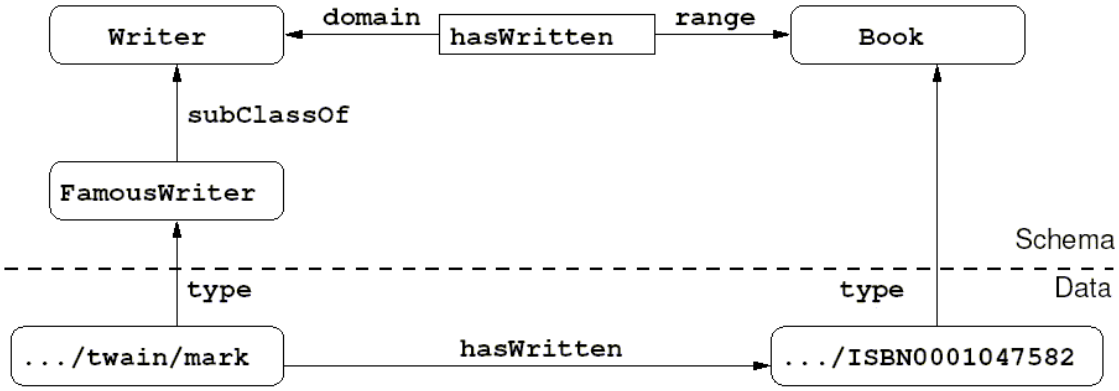

Figura 4 – Ejemplo de RDF Schema

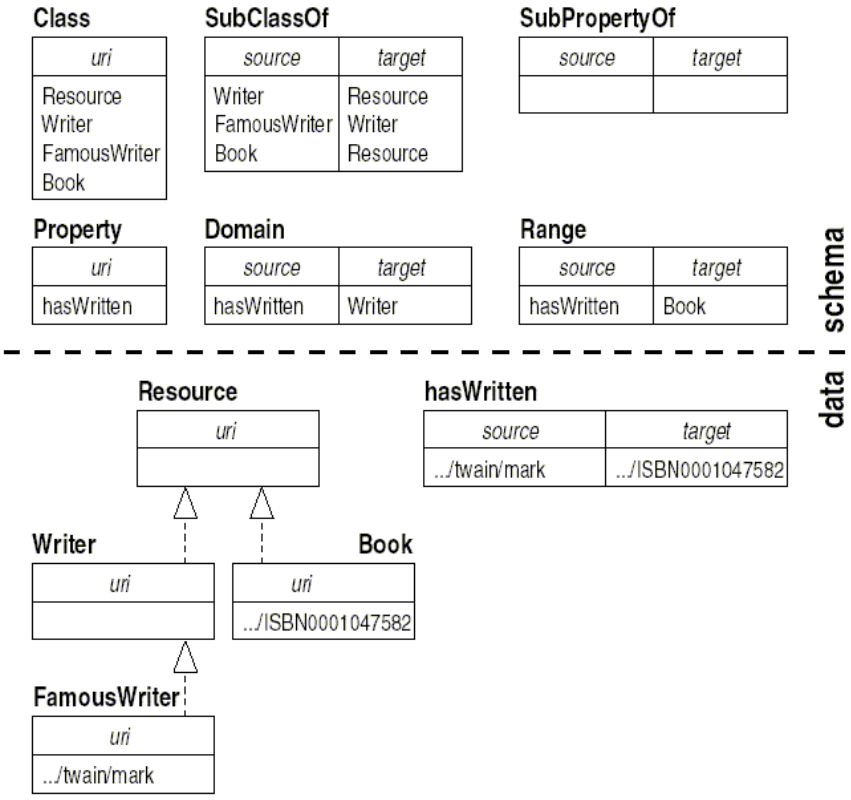

Figura 5 – Modelo implementado en PostgreSQL

Las pruebas en PostgreSQL no han sido muy satisfactorias, debido a que la inserción de datos no es tan rápida como se quisiera, especialmente la carga incremental de esquemas puede ser muy lenta, dado que la creación de nuevas tablas toma mucho tiempo en PostgreSQL, y mas aun cuando se trata de insertar una relación de subclase entre dos clases ya existentes.

Planteando una nueva configuración se uso SQL92SAIL (ver Capítulo 3) para conectarse, además de insertar en una tabla simple las sentencias RDF que tenia tres columnas (objeto, atributo, valor). Mostrando mejor desempeño en escenarios donde los RDFS cambian frecuentemente.

MySQL y SAIL

En las primeras pruebas con MySQL se implemento SAIL con un esquema estrictamente relacional, como se observa en la figura 6.

La gran diferencia con la implementación en PostgreSQL, es que en esta configuración el esquema de la base de datos no cambia cuando un RDFS cambia, de modo que se podría afirmar que insertar registros en tablas de una base de datos requiere menos tiempo y recursos que agregar nuevas tablas.

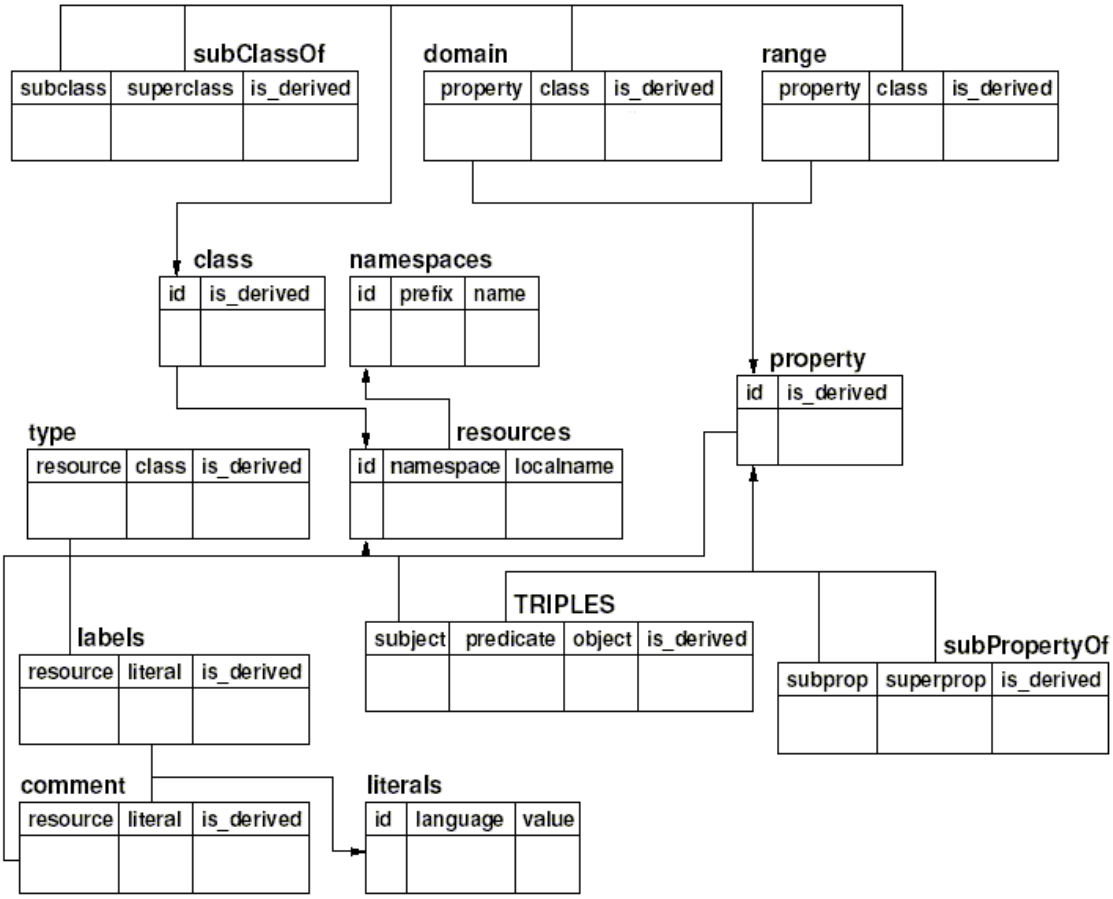

Figura 6 – Implementación en MySQL

Comparativa

Introducción

Son muchos los benchmarks que se han publicado sobre estos gestores de bases de datos, aunque muchos de ellos tienen una clara tendencia hacia uno de los dos bandos. Es por esto que hay que saber extraer bien las conclusiones a partir de un benchmark. Por ejemplo, es importante que las comparaciones entre los dos gestores se realicen en igualdad de condiciones, cosa que por otra parte, muchas veces resulta complicado debido a las distintas implementaciones que poseen estos gestores.

Además, resulta muy complicado diseñar un buen benchmark, que tenga en cuenta todas las posibles mejoras de rendimiento que pueda aplicar cada uno de estos gestores. Es lógico pues, que haya algunas pruebas que se le apliquen a un gestor, y que no tenga ningún sentido realizárselas al otro.

A continuación se resumen las conclusiones obtenidas a partir de diversos benchmark's, intentando hacer un descarte de los que tenían una clara tendencia hacia uno de los bandos.

Lo mejor y peor de PostGreSQL

Las características positivas que posee este gestor según las opiniones más comunes en Internet, son:

- Posee una gran escalabilidad. Es capaz de ajustarse al número de CPUs y a la cantidad de memoria que posee el sistema de forma óptima, haciéndole capaz de soportar una mayor cantidad de peticiones simultáneas de manera correcta (en algunos benchmarks se dice que ha llegado a soportar el triple de carga de lo que soporta MySQL).

- Implementa el uso de rollback's, subconsultas y transacciones, haciendo su funcionamiento mucho más eficaz, y ofreciendo soluciones en campos en las que MySQL no podría.

- Tiene la capacidad de comprobar la integridad referencial, así como también la de almacenar procedimientos en la propia base de datos, equiparándolo con los gestores de bases de datos de alto nivel, como puede ser Oracle.

Por contra, los mayores inconvenientes que se pueden encontrar a este gestor son:

- Consume gran cantidad de recursos.

- Tiene un límite de 8K por fila, aunque se puede aumentar a 32K, con una disminución considerable del rendimiento.

- Es de 2 a 3 veces más lento que MySQL.

Lo mejor y peor de MySQL

Es evidente que la gran mayoría de gente usa este gestor en Internet, por lo que encontrar opiniones favorables no ha resultado en absoluto complicado:

- Sin lugar a dudas, lo mejor de MySQL es su velocidad a la hora de realizar las operaciones, lo que le hace uno de los gestores que ofrecen mayor rendimiento.

- Su bajo consumo lo hacen apto para ser ejecutado en una máquina con escasos recursos sin ningún problema.

- Las utilidades de administración de este gestor son envidiables para muchos de los gestores comerciales existentes, debido a su gran facilidad de configuración e instalación.

- Tiene una probabilidad muy reducida de corromper los datos, incluso en los casos en los que los errores no se produzcan en el propio gestor, sino en el sistema en el que está.

- El conjunto de aplicaciones Apache-PHP-MySQL es uno de los más utilizados en Internet en servicios de foro (Barrapunto.com) y de buscadores de aplicaciones (Freshmeat.net).

Debido a esta mayor aceptación en Internet, gran parte de los inconvenientes se exponen a continuación, han sido extraídos de comparativas con otras bases de datos:

- Carece de soporte para transacciones, rollback's y subconsultas.

- El hecho de que no maneje la integridad referencial, hace de este gestor una solución pobre para muchos campos de aplicación, sobre todo para aquellos programadores que provienen de otros gestores que sí que poseen esta característica.

- No es viable para su uso con grandes bases de datos, a las que se acceda continuamente, ya que no implementa una buena escalabilidad.

Conclusiones

Después de haber leído diversos artículos sobre estos gestores de bases de datos, FAQ's y benchmarks (algunos con un curioso "modus operandi"), la conclusión que se puede sacar es que en realidad no hay que sacar ninguna conclusión sobre el tema. Cada uno de estos gestores es idóneo para ciertos campos, e intentar utilizar el otro acarrearía una pérdida de productividad del programa, como también grandes quebraderos de cabeza.

Ninguno de estos dos gestores son totalmente perfectos, por lo que no hay que obcecarse en la elección única y fanática, como se suele hacer en muchos casos de alguno de ellos. Simplemente se trata de escoger el más conveniente en cada caso. Éstos son los grandes inconvenientes y a la vez las grandes maravillas que conlleva el mundo OpenSource.

Capítulo 5

Análisis de requisitos para una Web de búsqueda semántica

En este capitulo, se presenta la especificación de requisitos para el desarrollo de una Web de búsqueda semántica. Por tanto, emplearemos un método para la especificación de requisitos en el proceso de desarrollo de aplicaciones Web que, basándose en la detección de las tareas que deben realizar los usuarios, y su posterior descripción, permite capturar tanto los aspectos funcionales y de información como los aspectos de navegación de este tipo de aplicaciones.

Las técnicas propuestas para ello, permiten obtener especificaciones intuitivas y amigables para el cliente (con el fin de facilitar su posterior validación), y a la vez, claras y concisas, para su correcta interpretación por los diseñadores Web.

Introducción

Uno de los aspectos más importantes en el proceso de desarrollo de software es determinar qué necesidades del cliente debe cubrir el sistema.

Según el estándar IEEE 610.12-1990 un requisito se define como una condición o capacidad que un sistema software debe alcanzar para satisfacer un contrato, estándar, especificación o cualquier otro documento formal que se imponga (necesidades del cliente).

Las capacidades (requisitos) que los clientes exigen a las aplicaciones tradicionales (no Web) cubren necesidades referentes básicamente a funcionalidad (qué debe hacer el sistema), almacenamiento de información (qué información se va a manejar en el sistema) u otras características no funcionales como la facilidad de uso y el rendimiento. Estos requisitos son ampliamente soportados por multitud de técnicas existentes en el campo de requisitos de la ingeniería del software, las cuales permiten su especificación de forma clara y concisa. Esto no es así en el ámbito de la Ingeniería Web, donde, aunque existen gran cantidad de aproximaciones metodológicas para la construcción de aplicaciones Web, sus esfuerzos se centran mayoritariamente en la etapa de modelado conceptual, tratando con menor profundidad la especificación de requisitos. Una aplicación Web no deja de ser una aplicación software, y por tanto, debe dar soporte también a requisitos funcionales, de almacenamiento de información o de calidad. Además, debido a sus orígenes hipermediales[49] debe satisfacer un tipo específico de requisitos: los requisitos navegacionales. Estos requisitos definen las necesidades navegacionales a través del hiperespacio[50] que presentan cada uno de los usuarios que pueden conectarse a la aplicación Web.

En este sentido, gran parte de los métodos capturan los requisitos navegacionales diseñando directamente su estructura navegacional. Esta estrategia para la especificación de requisitos navegacionales puede resultar cómoda e intuitiva para el diseñador Web, el cual posee un gran conocimiento y soltura con la aproximación utilizada. Sin embargo, el cliente no está familiarizado con dicha aproximación y puede resultarle costoso validar los requisitos especificados. El cliente no entiende de nodos o enlaces, de páginas Web o menús de opciones, ni de motores de búsqueda o mecanismos de acceso. El cliente únicamente conoce los aspectos relacionados con su ámbito de trabajo, los cuales le plantean ciertas necesidades que deben ser cubiertas por la aplicación Web. De este modo, estas necesidades deben ser especificadas en términos que: (1) el cliente conozca y entienda, con el fin de obtener una comunicación más fluida entre analista y cliente, y (2) que sean lo suficientemente claros y concisos para su correcta interpretación por parte de los programadores de la aplicación Web.

Para ello, en este capitulo se presenta una aproximación para la captura de requisitos navegacionales donde se especifican las necesidades del cliente a partir de las tareas que desea realizar interactuando con la aplicación Web. Una tarea se define como un trabajo que usuario y sistema realizan

[49] Los sistemas hipermediales fueron definidos como un enfoque para manejar y organizar información, a partir de una red de nodos conectados por enlaces. Estos nodos pueden contener textos, gráficos, imágenes, audio, animaciones y video. El usuario tiene la flexibilidad de acceder a dichos nodos de manera no secuencial lo que genera la noción de navegación.

[50] Término que describe el número total de localizaciones y todas sus interconexiones en un ambiente hipermedial.

conjuntamente para obtener un resultado determinado en un tiempo limitado. En este sentido, a diferencia de las aplicaciones tradicionales, las aplicaciones Web no se desarrollan con el fin de proporcionar únicamente una funcionalidad específica, sino que también se diseñan para proveer al usuario un acceso estructurado a una determinada información. De este modo, las tareas a realizar a través de una aplicación Web, no están relacionadas únicamente con la ejecución de operaciones (funcionalidad pura), sino también con la consulta de información. Si, por ejemplo, tenemos una aplicación Web de venta de productos “on-line”, una tarea puede constituir tanto la compra de un producto determinado (ejecución de una funcionalidad específica) como la búsqueda de dicho producto (consulta de información). Por otro lado, además de especificar las tareas que el cliente necesita realizar con la aplicación Web, se debe describir cómo se llevan a cabo. Para ello, se propone la extensión de las descripciones utilizadas tradicionalmente en la especificación de requisitos funcionales (secuencia de operaciones que realiza el sistema) introduciendo aspectos relacionados con la interacción sistema-usuario, la cual define a partir del intercambio de información que ambos realizan. Este intercambio de información se describe mediante términos familiares al ámbito de trabajo del cliente (la información que maneja diariamente), lo que permite obtener un mayor entendimiento por parte de éste, de la especificación de requisitos realizada. A lo largo de este trabajo veremos cómo este tipo de descripciones, además de facilitar al cliente su comprensión, permiten la especificación de requisitos funcionales, de almacenamiento de información y de navegación de las aplicaciones Web.

La aproximación para la captura de requisitos que se presenta en este trabajo se divide en dos etapas:

1. Identificación y especificación de tareas: Se describen las necesidades del usuario a partir del conjunto de tareas que puede llevar a cabo interactuando con la aplicación Web.

2. Descripción de tareas: Cada tarea se describe a partir de la secuencia de acciones que realiza el sistema introduciendo aspectos relacionados con la interacción entre sistema y usuario.

Descripción del sistema

El propósito general del sistema es el de proporcionar soporte para la búsqueda on-line de canciones. Por tanto el sistema a desarrollar es una Web de búsqueda semántica cuyo dominio de los datos será el mundo musical, dónde se describirán los metadatos de las canciones; su título, el compositor, el estilo del grupo de música, el año en qué se grabó la canción, etc. Se dispondrá de un conjunto de datos, en RDF.

Cuando el usuario se conecte al sistema Web, debe poder visitarlo de forma anónima. El usuario debe tener siempre accesible la posibilidad de hacer búsquedas.

El sistema será capaz de localizar la información en función de las clases artists y tracks, así como por las propiedades de cada una de estas clases descritas en la ontología “Simac Music Ontology (Version 0.1)”.

Las propiedades de las clases son las siguientes:

- **Artists**
  - Nombre.
  - Décadas. Puede ser más de una.
  - Ciudad.
  - Nacionalidad.
- **Tracks**
  - Título.
  - Cantante.

Especificación de requisitos

Identificación de tareas

Propósito general. El propósito general del sistema es el de proporcionar soporte para la realización de búsquedas.

Descripción de tareas

Búsqueda de un artista o grupo

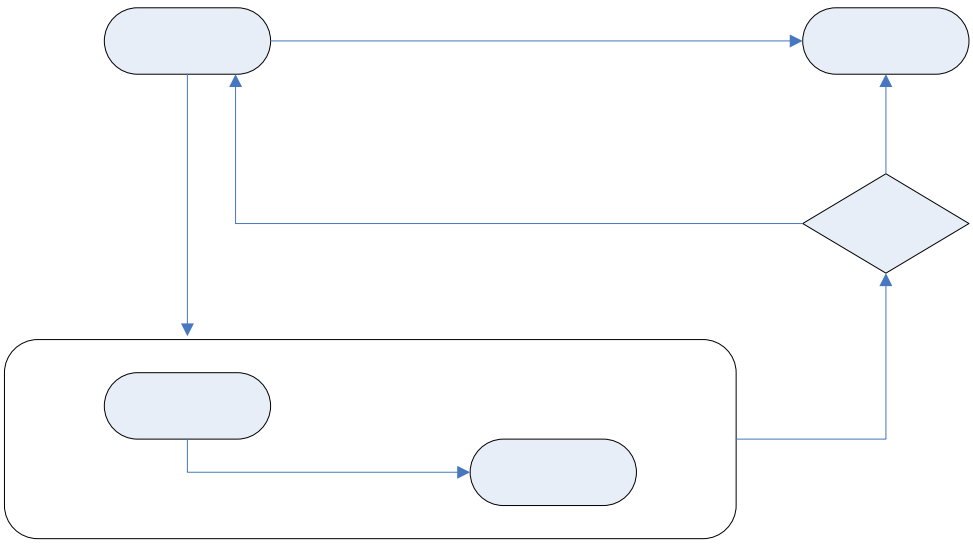

Figura 7 – Descripción de la tarea elemental *búsqueda de artista o grupo*

La figura 7 muestra la descripción de la tarea búsqueda de artista o grupo. El sistema le proporciona al usuario el conjunto de artistas o grupos del catálogo a partir del cual puede seleccionar uno para consultar la totalidad de sus datos, redirigiéndolo a una URL determinada.

Búsqueda de títulos

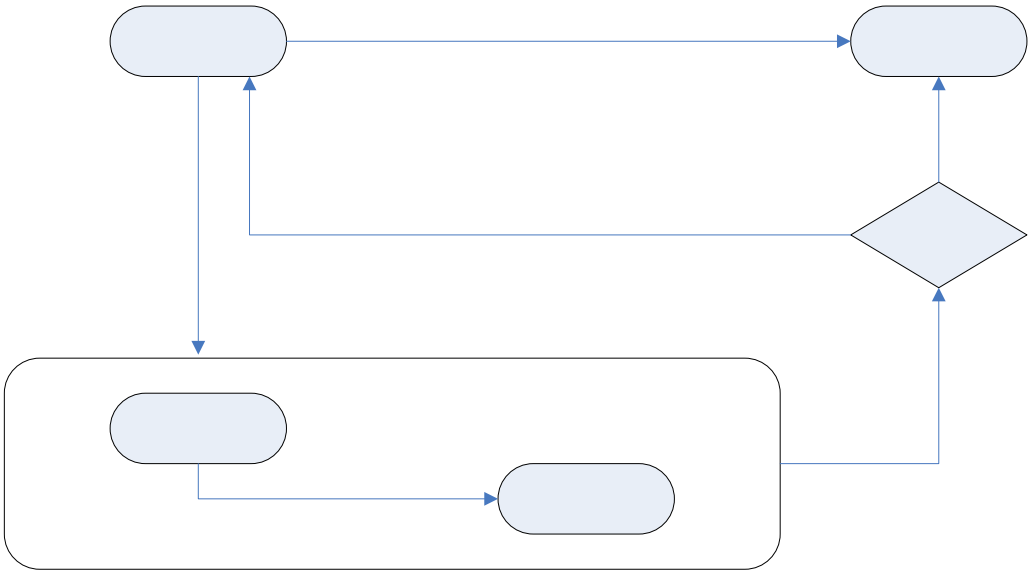

Figura 8 - Descripción de la tarea elemental *búsqueda de títulos*

La figura 8 muestra la descripción de la tarea búsqueda de títulos. El sistema le proporciona al usuario el conjunto de canciones del catálogo a partir del cual puede seleccionar una para consultar sus datos. También podrá descargar el fichero mp3 correspondiente a la canción deseada.

Modelo navegacional

<<output>>
TRACKS

A continuación se muestra un modelo navegacional básico de la aplicación Web presentada en este capítulo. Este modelo se compone de un mapa navegacional, puesto que solo tendremos un tipo de usuario, el anónimo.

Mapa anónimo

En la figura 9 se muestra el mapa del usuario Anónimo que se compone de dos contextos de exploración: contexto artista y contexto título mediante los cuales se proporciona información sobre los artistas y los títulos.

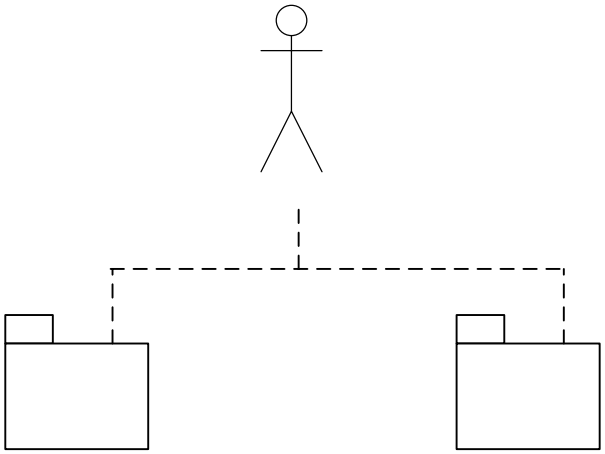

Figura 9 – Mapa anónimo

Herramientas

Se instalará la versión de Java J2EE, cuya versión será la 1.5.0_02.

El sistema a emplear para el acceso a los datos RDF será Sesame (ver Capítulo 3). La versión que se utilizará será la 1.2.3.

Sesame se instalará bajo la versión 5.5.12 de Apache/Tomcat[51]. Esta versión implementa Servlet 2.4 y JSP 2.0. Los datos RDF serán almacenados en un repositorio Sesame, el cual se creará en memoria principal, por lo que no se empleará ningún sistema de gestión de bases de datos como MySQL o PostgreSQL (ver Capítulo 4).

Como lenguaje de consulta, se utilizara SeRQL – Sesame Rdf Query Language (ver Capítulo 2), el cual está considerado como un lenguaje de consulta RDF de segunda generación.

Para el desarrollo de la aplicación, se emplearan páginas HTML y JSP. Para ello se utilizará la herramient DreamWeaver.

Diseño e Implementación

La página principal de la aplicación será *index.html*. Está página contendrá un formulario con una caja de texto donde el usuario podrá introducir el texto a localizar. Este texto podrá pertenecer al título de una canción o al nombre de un artista o banda de música. También dispondrá de un botón, el cual pondrá

[51] http://tomcat.apache.org/

en marcha el proceso de búsqueda. En la figura 10 podemos observar el diseño que tendrá esta primera pagina.

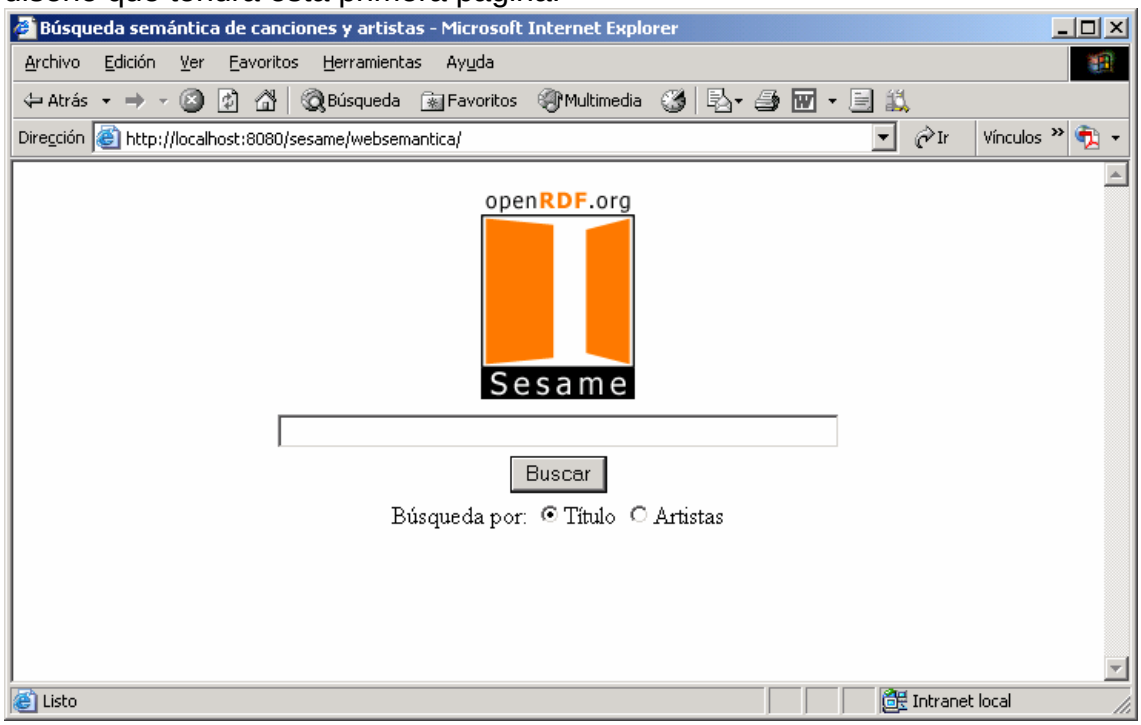

Figura 10 – Página inicial *index.html*

Una vez el usuario introduce el texto a localizar y hace clic en el botón *Buscar*, la aplicación se redirige a la página *realizarBusqueda.jsp*. Podemos ver un ejemplo de la misma en la figura 11. El ejemplo consiste en localizar aquellos títulos que contengan el texto *been*.

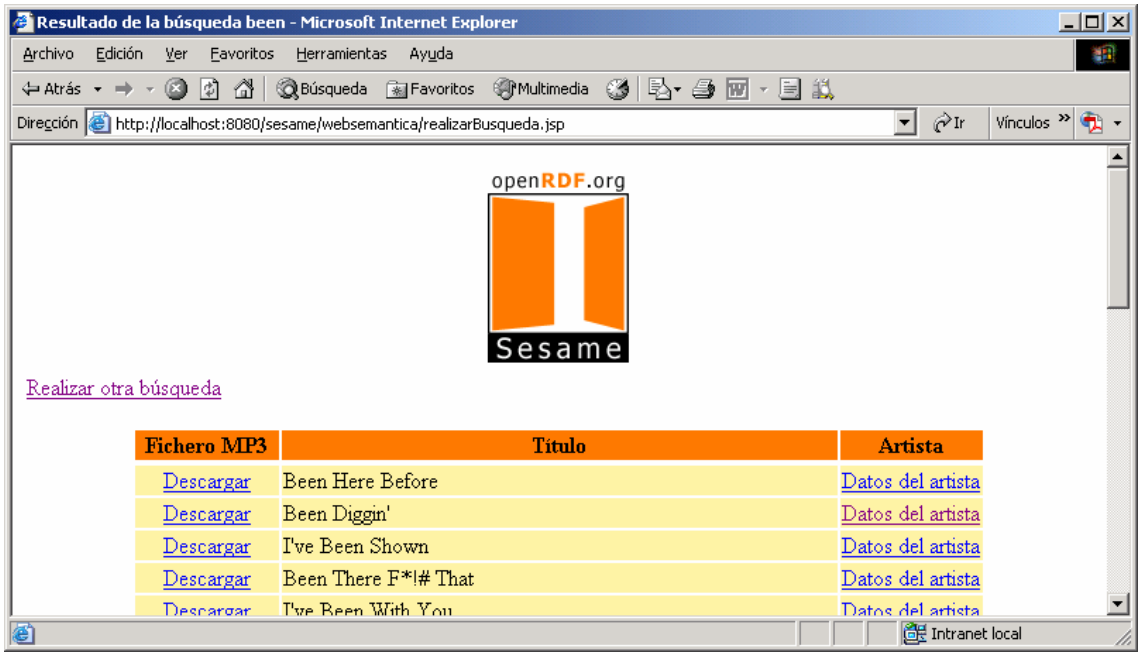

Figura 11 – Ejemplo de consulta por título

Podemos ver que la consulta genera varias filas cuyo título contiene el texto *been*.

Si seleccionamos un título y hacemos clic en *Descargar*, la canción sonará en el navegador (si está activa esta opción) o será descargado en la ubicación que el usuario decida. Podemos ver el resultado de esto en la figura 12.

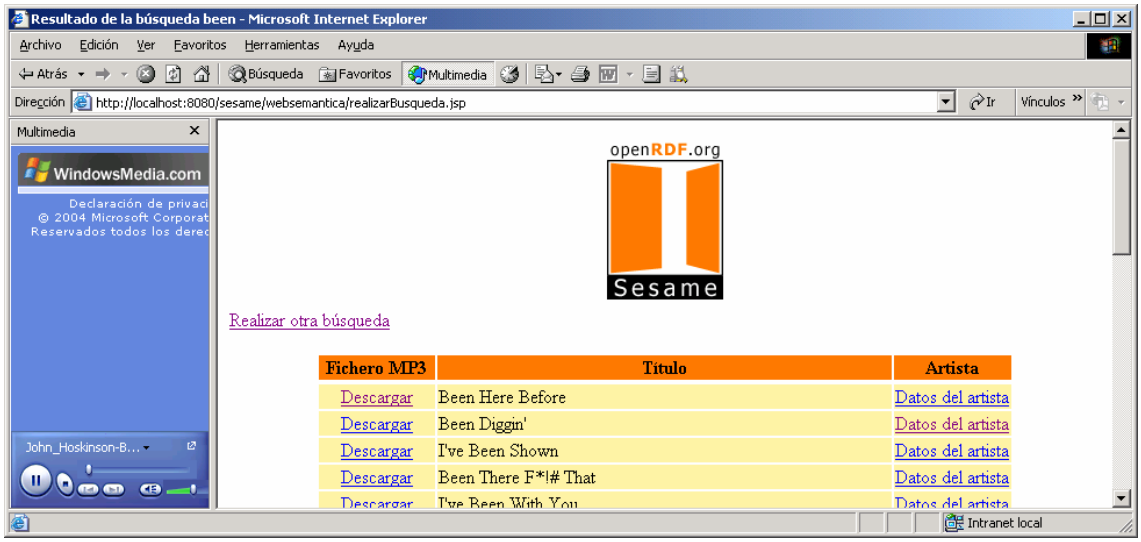

Figura 12 – Título elegido sonando en el navegador Web

También tenemos la posibilidad de consultar los datos del artista si hacemos clic sobre el enlace *Datos del artista*. Este enlace nos redirigirá a la Web www.garageband.com y accederá directamente a los datos que buscamos. Figura 13.

Figura 13 – Datos del artista

Otra opción que ofrecerá la aplicación es la búsqueda por artista. En la figura 14 podemos ver el ejemplo de buscar a artistas cuyo nombre comienza por *Andrew W*. Se puede observar que aparece el número de resultados obtenidos.

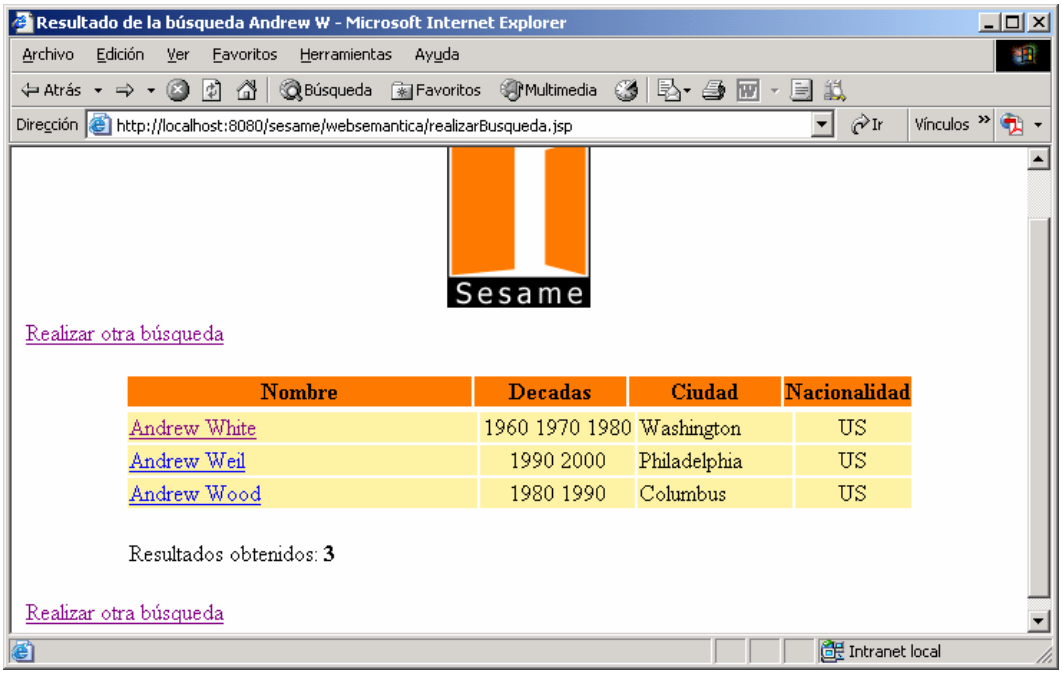

Figura 14 – Ejemplo de consulta por artista

En este caso, podemos ver varios de sus datos, pero si queremos ampliar la información, podemos hacer clic sobre el enlace de su propio nombre, el cual nos redirigirá a la página www.mp3.com, con lo que nos llevará directamente a los datos del artista. Figura 15.

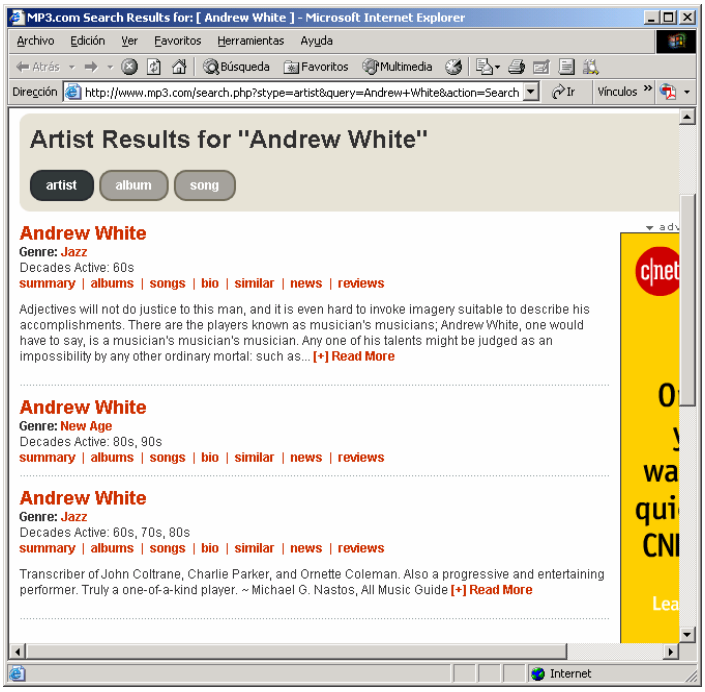

Figura 15 – Ampliación de datos del artista

La aplicación detectará si el repositorio esta creado y los datos cargados en memoria. En caso de que no lo este, se creará y los datos que contienen los ficheros *artists.rdf* y *tracks.rdf* se añadirán.

Las consultas se realizarán en función de las opciones seleccionadas por el usuario en la pantalla principal *index.html*.

En el caso de encontrar datos con los criterios de búsqueda introducidos por el usuario, estos aparecerán en pantalla. En el caso de que la búsqueda no haya sido satisfactoria, aparecerá la pantalla de la figura 16.

Figura 16 – Pantalla mostrada cuando la búsqueda no es satisfactoria

Se puede ver que el texto introducido para la búsqueda es *Frank Sinatra*, pero la búsqueda no ha producido ningún resultado, por lo que se avisa al usuario del hecho, indicándole el texto buscado y ofreciéndole la posibilidad de seguir realizando búsquedas.

Capítulo 6

Conclusiones

La Web Semántica se revela como una nueva Era para los contenidos Web, para la que la accesibilidad por parte de los ciudadanos con necesidades especiales no le es ajena. En su implantación el uso de metadatos se revela como clave.

Su infraestructura está evolucionando aunque aún tiene problemas no resueltos. Las tecnologías de la Web Semántica todavía están surgiendo y hay poco consenso en sus características.

La Web Semántica proporcionará un salto cualitativo sobre el potencial de la Web. Aplicaciones:

- Comercio electrónico.
- Búsqueda de información en la Web.
- Procesamiento del lenguaje natural.
- Librerías digitales.
- Turismo, etc.

Los pasos a seguir para que se lleve a cabo son:

- Establecimiento de estándares que permitan añadir tags descriptivos explícitos (metadata) al contenido que hay en la Web.
- Desarrollo de programas que sean capaces de convertir y acceder a los metadatos existente en los distintos sitios Webs.
- Desarrollo de aplicaciones específicas capaces de lograr hacer inferencias de los datos obtenidos, permitiendo actuar de forma dinámica generando acciones.

La consistencia en el acceso a la información por conceptos y su representación en RDF mejorará la accesibilidad de la información en dos sentidos:

- La capacidad de los metadatos de anticiparse al contenido/condiciones de acceso.
- La posibilidad de crear un perfil específico de aplicación de los metadatos a la accesibilidad.

Se hace necesaria una mayor investigación e inversión en metadatos y en organización de conocimiento.

Los principales problemas y a la vez objetivos a cumplir son la accesibilidad y la interoperabilidad.

RDF es a la Web Semántica, lo que HTML a la Web. El futuro de la Web Semántica está aún por llegar.

Sesame, constituye un importante avance que va mas allá de la forma actual de almacenar y consultar en RDF, dado que es la primera implementación pública para la semántica de RDFS.

Es necesario resaltar la característica de la arquitectura Sesame que permite su implementación en una gran variedad de repositorios, ya sea en bases de datos relacionales, almacenamiento de Triplas RDF o a nivel remoto, debido a la independencia en relación al DBMS.

Sesame por si mismo es una aplicación basada en servidor, y por lo tanto puede ser usado como un servicio remoto de almacenamiento y consulta en la Web Semántica. Permitiendo además la abstracción del protocolo de comunicación, de manera que no esta condicionado a un protocolo en particular.

Las nuevas implementaciones propuestas por los investigadores de Sesame pueden ser consideradas como el paso necesario para lograr un servicio de almacenamiento y consulta para la Web Semántica.

De ser necesario el uso de sistemas de datos RDF persistentes, dos de los mejores sistemas de gestión de bases de datos son MySQL y PostgreSQL.

Glosario

A

Aidministrator

Empresa holandesa dedicada a la ingeniería del conocimiento. Esta empresa desarrollo Sesame (*véase* Sesame).

Apache

Servidor de páginas Web desarrollado por la Apache Software Fundation, organización formada por miles de voluntarios que colaboran para la creación de software de libre distribución.

API

Interfaz de programación de aplicaciones (del ingles Application Programming Interface - Interfaz de Programación de Aplicaciones, interfaz de programación de la aplicación): una serie de funciones que están disponibles para realizar programas para un cierto entorno. Representa un método para conseguir abstracción en la programación, generalmente (aunque no necesariamente) entre los niveles o capas inferiores y los superiores del software.

ASP

Active Server Pages. Una especificación que permite crear dinámicamente páginas Web mediante HTML, scripts, y componentes de servidor ActiveX reutilizables.

B

Benchmark

Programa especialmente diseñado para evaluar el rendimiento de un sistema, de software o de hardware.

Bugs

Un error de software o computer bug, que significa bicho de computadora, es el resultado de una falla de programación introducida en el proceso de creación de programas de ordenador. El término bug fue acreditado erróneamente a Grace Murray Hopper, una pionera en la historia de la computación, pero Thomas Edison ya lo empleaba en sus trabajos para describir defectos en sistemas mecánicos por el año 1870.

C

CDA

Estándar para modelizar información en el dominio de la medicina.

CGI

Common Gateway Interface (en inglés “Pasarela de Interfaz Común”, abreviado CGI) es una importante tecnología de la World Wide Web que permite a un cliente (explorador Web) solicitar datos de un programa ejecutado en un servidor Web. CGI especifica un estándar para transferir datos entre el cliente y el programa.

D

DAML

DARPA Agent Markup Language. Es un lenguaje para modelar ontologías creado con una extensión de RDF. Está siendo desarrollado por el DARPA en colaboración con W3C.

DAML+OIL

DARPA Agent Markup Language + Ontology Inference Layer. Es un lenguaje de codificación de ontologías.

DOM

Document Object Model, literalmente "Modelo de Objetos de Documento". Interfaz independiente de la plataforma y del lenguaje que permite a programas y scripts acceder y actualizar dinámicamente los contenidos, la estructura y el estilo de los documentos HTML y XML

DTD

Document Type Definition. Un documento que describe la estructura de una página Web escrita en XML.

E

ebXML

Estándar para modelizar información en el dominio de las finanzas.

F

FLASH

Software de Macromedia para crear pequeñas animaciones vectoriales reproducidas en la Web. El navegador de un usuario necesita el plug-in Flash Player para interpretar las animaciones Flash.

FpXML

Estándar para modelizar información en el dominio de las finanzas.

Framework

es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Tipicamente, un

framework puede incluir soporte de programas, librerias y un lenguaje de scripting entre otros softwares para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

G

GPL

La GNU General Public License (Licencia Pública General) es una licencia creada por la Free Software Foundation y orientada principalmente a los términos de distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software Libre.

Grafo

Estructura de datos utilizada en algunos lenguajes de programación, en la cual cada elemento puede tener uno o varios predecesores y uno o varios sucesores.

H

Hipermediales

Enfoque para manejar y organizar información, a partir de una red de nodos conectados por enlaces. Estos nodos pueden contener textos, gráficos, imágenes, audio, animaciones y video. El usuario tiene la flexibilidad de acceder a dichos nodos de manera no secuencial lo que genera la noción de navegación.

HTML

Acrónimo inglés de Hyper Text Markup Language (lenguaje de marcación de hipertexto), es un lenguaje de marcas diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web. Gracias a Internet y a los navegadores del tipo Explorer o Netscape, el HTML se ha convertido en uno de los formatos más populares que existen para la construcción de documentos.

HTTP

Es el protocolo de la Web (WWW), usado en cada transacción. Las letras significan Hyper Text Transfer Protocol, es decir, protocolo de transferencia de hipertexto. El hipertexto es el contenido de las páginas Web, y el protocolo de transferencia es el sistema mediante el cual se envían las peticiones de acceder a una página web, y la respuesta de esa web, remitiendo la información que se verá en pantalla

I

IEEE

IEEE corresponde a las siglas del Institute of Electrical and Electronics Engineers, Instituto de Ingenieros Eléctricos y Electrónicos, una asociación estadounidense dedicada a la estandarización. Es una asociación internacional sin fines de lucro formada por profesionales de las nuevas tecnologías, como ingenieros de telecomunicaciones, ingenieros electrónicos, Ingenieros en informática, etc.

IEEE LOM

Estándar para modelizar información en el dominio de la enseñanza.

IsaViz

Entorno para la visualización y edición de modelos RDF.

J

J2EE

J2EE son las siglas de Java 2 Enterprise Edition que es la edición empresarial del paquete Java creada y distribuida por Sun Microsystems. Comprenden un conjunto de especificaciones y funcionalidades orientadas al desarrollo de aplicaciones empresariales. Debido a que J2EE no deja de ser un estándar, existen otros productos desarrollados a partir de ella aunque no exclusivamente.

Java

Java es una plataforma de software desarrollada por Sun Microsystems, de tal manera que los programas creados en ella puedan ejecutarse sin cambios en diferentes tipos de arquitecturas y dispositivos computacionales.

JavaBeans

JavaBeans es un modelo de componentes creado por Sun Microsystems para la construcción de aplicaciones en Java.

JavaScript

Lenguaje desarrollado por Netscape. Aunque es parecido a Java se diferencia de él en que los programas están incorporados en el archivo HTML.

Jena

Framework desarrollado por HP Labs para manipular metadatos desde una aplicación Java.

Joseki

Servidor para la publicación, en la Web, de modelos RDF.

JSP

Java Server Pages (JSP) es la tecnología para generar páginas web de forma dinámica en el servidor, desarrollado por Sun Microsystems, basado en scripts que utilizan una variante del lenguaje java

K

Kaon

Entorno gráfico para visualizar y construir ontologías.

M

Metadato

Los metadatos son datos sobre los datos, esto es, información sobre la información misma. En esencia, intentan responder a las preguntas quién, qué, cuándo, cómo, dónde y porqué, sobre cada una de las facetas relativas a los datos que se documentan.

MySQL

MySQL es una de las bases de datos más populares desarrolladas bajo la filosofía de código abierto.

N

N3

Lenguaje de programación equivalente a RDF.

Netscape

Navegador Web, el cual estaba basado, en un principio, en el programa Mosaic. Ha crecido en sus características rápidamente y ahora se reconoce mundialmente como el mejor y más popular navegador. La corporación Netscape también produce otro tipo de software.

NewsML

Estándar para modelizar información en el dominio del periodismo.

NLM Medline

Estándar para modelizar información en el dominio de la medicina.

O

OCML

Lenguaje de programación.

Ontologia

Hace referencia al intento de formular un exhaustivo y riguroso esquema conceptual dentro de un dominio dado, con la finalidad de facilitar la comunicación y la compartición de la información entre diferentes sistemas.

Oracle

Sistema de administración de base de datos (o RDBMS por el acrónimo en inglés de Relational Data Base Management System), fabricado por Oracle Corporation.

OWL

Acronym de “Web Ontology Language”, un lenguaje de marcado para la publicación y compartición de datos usando ontologías en la Web.

P

Parser

Software especializado en reconocer marcas en un documento.

PHP

PHP (acrónimo recursivo de "PHP: Hypertext Preprocessor", originado inicialmente del nombre PHP Tools, o Personal Home Page Tools) es un lenguaje de programación interpretado. Aunque fue concebido en el tercer trimestre de 1994 por Rasmus Lerdorf. Se utiliza entre otras cosas para la programación de páginas Web activas, y se destaca por su capacidad de mezclarse con el código HTML

Plugin

Un plugin (o plug-in) es un programa de ordenador que interactúa con otro programa para aportarle una función o utilidad específica, generalmente muy específica. Los plugins típicos tienen la función de reproducir determinados formatos de gráficos, reproducir datos multimedia, codificar/decodificar emails, filtrar imágenes de programas gráficos, etc.

PostgreSQL

PostgreSQL es un servidor de base de datos relacional libre, liberado bajo la licencia BSD. Es una alternativa a otros sistemas de bases de datos de código abierto (como MySQL, Firebird y MaxDB), así como sistemas propietarios como Oracle o DB2.

PRISM

Estándar para modelizar información en el dominio del periodismo.

R

RDBMS

Un RDBMS es un Sistema Administrador de Bases de Datos Relacionales. RDBMS viene del acrónimo en ingles Relational Data Base Manager System.

RDF

RDF son las siglas de Resource Description Framework, la especificación de un modelo de metadatos, (normalmente implementado como una aplicación de XML) que ha sido desarrollada por el World Wide Web Consortium (W3C).

RDFS

Lenguaje para describir vocabularios RDF.

RDQL

Lenguaje de programación para manejar datos RDF.

Reificacion

Sentencias realizadas sobre otras sentencias.

RIXML

Estándar para modelizar información en el dominio del periodismo.

Rollback

Comando para descartar los cambios realizados en una base de datos y que aun no han sido confirmados.

RQL

Lenguaje de programación para manejar datos RDF.

S

SAIL

Acrónimo de “Storage And Inference Layer”, es el nivel que se encarga de interactuar con los manejadores de bases de datos (DBMS) permitiendo de este modo mantener la arquitectura de Sesame independiente del DBMS. Es una API que ofrece métodos RDF específicos a sus “clientes” y traduce estos métodos a llamadas a un DBMS específico.

SCIPHOX

Estándar para modelizar información en el dominio de la medicina.

SCORM

Estándar para modelizar información en el dominio de la enseñanza.

SeRQL

Lenguaje de programación para manejar datos RDF.

Servlet

Objetos que corren dentro del contexto de un servidor de aplicaciones (por ejemplo Tomcat) y extienden su funcionalidad.

Sesame

Sesame fue desarrollado por la empresa holandesa Aidministrator. Tiene una arquitectura desarrollada para un eficiente almacenamiento y consulta de grandes cantidades de metadatos en RDF y RDF Schema.

SGBD

Los Sistemas Gestores de Bases de Datos son un tipo de software muy específico, dedicado a servir de interfaz entre las bases de datos y las aplicaciones que la utilizan. En los textos que tratan este tema, o temas relacionados, se mencionan los términos SGBD y DBMS, siendo ambos equivalentes, y acrónimos, respectivamente, de Sistema Gestor de Bases de Datos y DataBase Management System, su expresión inglesa.

Shell

Parte fundamental de un sistema operativo encargada de ejecutar las órdenes básicas para el manejo del sistema. Suelen incorporar características tales como control de procesos, redirección de entrada/salida y un lenguaje de órdenes para escribir programas por lotes o (scripts).

SPARQL

Lenguaje de programación para manejar datos RDF.

SQL

El Lenguaje de Consulta Estructurado (Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Aúna características del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar información de interés de una base de datos, de una forma sencilla.

T

Tomcat

Tomcat (Jakarta Tomcat) funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Sun Microsystems. Se le considera un servidor de aplicaciones.

TopicMaps

Lenguaje de programación.

Triple

Lenguaje de programación para manejar datos RDF.

U

URI

Acrónimo de “Uniform Resource Identifier”, literalmente "Identificador Uniforme de Recurso"). Conjunto de caracteres que identifica un recurso de red, mediante un nombre o utilizando sus datos de localización

URL

Acrónimo de “Universal Resource Locutor” (Localizador Universal de Recursos / Identificador Universal de Recursos). Sistema unificado de identificación de recursos en la red. Es el modo estándar de proporcionar la dirección de cualquier recurso en Internet.

V

Versa

Lenguaje de programación para manejar datos RDF.

W

W3C

El World Wide Web Consortium, abreviadamente W3C, es una organización que produce estándares para la World Wide Web (o Telaraña Mundial). Está dirigida por Tim Berners-Lee, el creador original de URL (Uniform Resource Locator, Localizador Uniforme de Recursos), HTTP (HyperText Transfert Protocol, Protócolo de Transferencia de HiperTexto) y HTML (Lenguaje de Marcado de HiperTexto) sobre el que se basa la Web, y por extensión su inventor.

WebODE

Lenguaje de programación.

WWW

La World Wide Web (del inglés, Telaraña Mundial), la Web o WWW, es un sistema de hipertexto que funciona sobre Internet. Para ver la información se utiliza una aplicación llamada navegador Web para extraer elementos de información (llamados "documentos" o "páginas Web") de los servidores Web (o "sitios") y mostrarlos en la pantalla del usuario.

X

XBRL

Estándar para modelizar información en el dominio de las finanzas.

XHTML

XHTML, acrónimo inglés de eXtensible Hyper Text Markup Language (lenguaje extensible de marcado de hipertexto), es el lenguaje de marcado pensado para sustituir a HTML como estándar para las páginas Web. XHTML es la versión XML de HTML, por lo que tiene, básicamente, las mismas funcionalidades, pero cumple las especificaciones, más estrictas, de XML.

XML

XML es el acrónimo del inglés eXtensible Markup Language (lenguaje de marcado ampliable o extensible) desarrollado por el World Wide Web Consortium (W3C).

XMLNews

Estándar para modelizar información en el dominio del periodismo.

BIBLIOGRAFIA

[Abrams 1998] M. Abrams, ed., World Wide Web - Beyond the Basics, Prentice Hall, 1998. Versión on-line disponible en http://ei.cs.vt.edu/~wwwbtb/book/.

[Article_MySQL-PostGreSQL] Artículo comparativo, http://www.phpbuilder.com/columns/tim20000705.php3?page=1

[Bergman 2001] M. K. Bergman. The Deep Web: Surfacing Hidden Value. The Journal of Electronic Publishing. Volume 7, Issue 1, August, 2001. Disponible en http://www.press.umich.edu/jep/07-01/bergman.html.

[Berners-Lee 1989] T. Berners-Lee. Information Management: A Proposal. Internal Project Proposal, CERN, 1989. Disponible en http://www.w3.org/History/1989/proposal.html.

[Berners-Lee 2001] T. Berners-Lee, J. Hendler, O Lassila. The Semantic Web. Scientific American, May 2001.

[Coates 2001] A. B. Coates. The Role of XML in Finance. XML Conference & Exposition 2001. Orlando, Florida, December 2001.

[Connolly 2000] D. Connolly. A Little History of the World Wide Web. W3C Consortium, 2000. Disponible en http://www.w3.org/History.html.

[Gruber 1993] T. R. Gruber. A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, 5(2), pp. 199-220, 1993.

[Karvounarakis 2002] G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, M. Scholl. RQL: A Declarative Query Language for RDF. 11th International World Wide Web Conference (WWW2002), 2002.

[MySQL_Manual] Manual de MySQL, http://www.mysql.com/documentation/index.html

[O'Neill 2003] E. T. O'Neill, B. F. Lavoie, R. Bennett. Trends in the Evolution of the Public Web. D-Lib Magazine, Volume 9 Number 4, April 2003. Disponible en http://www.dlib.org/dlib/april03/lavoie/04lavoie.html.

[PostGreSQL_Manual] Documentación de PostGreSQL, http://www.postgresql.org/docs

Anexo A

Instalación y puesta a punto de los SGBD adecuados para RDF

Este anexo tratará de explicar la instalación y puesta a punto de algunos de los SGBD estudiados en este trabajo. Puesto que se trata de software libre, éstos han sido descargados de sus correspondientes páginas Web.

Instalación de Sesame

Desde la página oficial de Sesame[52], se puede descargar el software necesario para su instalación.

Sesame se puede instalar como una librería Java o como servidor:

[52] http://www.openrdf.org/download.jsp

Librería Java

Necesitamos el fichero sesame.jar. Este fichero será encontrado en la carpeta /lib. Simplemente hay que incluir este fichero .jar en la variable de entorno classpath. Con esto obtendremos toda la funcionalidad de Sesma en nuestras aplicaciones Java.

Sesame requiere Java 2, versión 1.4 o superior para un correcto funcionamiento.

Servidor

El software requerido es el siguiente:

- Sesame.
- Un contenedor de Java servlet.

Sesame tiene varias opciones para el almacenamiento de datos RDF. Puede almacenar datos en memoria principal, directamente en ficheros o en bases de datos relacionales. Para ésta última opción, la instalación de Sesame requiere la presencia de un DBMS con driver JDBC. Actualmente, PostgreSQL, MySQL, MS SQL Server y Oracle son soportados.

Instalación bajo Tomcat 4 o 5

Los siguientes pasos describe un procedimiento simple para instalar Sesame en Tomcat 4.x o 5.x.

1. Instalar Tomcat. Consiste en descargar la distribución de Tomcat desde http://jakarta.apache.org/tomcat e instalar en nuestro disco duro.

2. En la carpeta de aplicaciones Web ([TOMCAT_DIR]/webapps/), crear una carpeta “sesame”.

3. Extraer el fichero sesame.war, (puede ser localizado en la carpeta lib de la distribución de Sesame) en la carpeta “sesame” creada en el paso 2. Para extraer el fichero, se ejecutará el comnado “jar –xf [PATH/TO/]sesame.war”. Para esto también podemos usar el programa WinZip.

4. Si se desea usar bases de datos con Sesame, se debe copiar el driver JDBC apropiado al directorio [SESAME_DIR]/WEB-INF/lib/.

5. Copiar el fichero [SESAME_DIR]/WEB-INF/system.conf.example a [SESAME-DIR]/WEB-INF/system.conf.

6. Arrancar el servidor Tomcat. Ahora se puede acceder a la interfaz de Sesame en http://[MACHINE_NAME]:8080/sesame.

Notas para PostgreSQL

- Sesame ha sido probado a partir de la versión de PostgreSQL 7.0.2. La compatibilidad con versiones anteriores no está garantiza.

- El driver JDBC para PostgreSQL se puede encontrar en http://jdbc.postgresql.org.

- Hay que asegurarse que el servidor de PostgreSQL está funcionando con conexiones TCP/IP. Si TCP/IP no esta disponible, el driver JDBC no podrá ser utilizado por el servidor.

- Sesame necesita una cuenta en PostgreSQL. Se puede crear una nueva cuenta en PostgreSQL con el comando **createuser**.

- Se debería crear una base de datos "testdb". Esto se puede realizar con el comando **createdb testdb**. Esta base de datos servirá para asegurarnos del correcto funcionamiento de la instalación y podrá ser borrada una vez lo hayamos confirmado.

Notas para MySQL

- Sesame ha sido probado a partir de la versión de MySQL 3.23.47. La compatibilidad con versiones anteriores a 3.23.0 no está garantizada.

- El driver JDBC para MySQL se puede encontrar en http://www.mysql.com/downloads/api-jdbc.html.

- Sesame necesita una cuenta en MySQL. Se puede crear una nueva cuenta en MySQL con el comando **mysql –u root –p mysql**. Posteriormente, habrá que establecer los privilegios al usuario recientemente creado mediante el comando **GRANT ALL ON <DATABASE>.* TO <USER>@localhost;**.

- Se debería crear una base de datos "testdb". Esto se puede realizar con el comando **mysqladmin –u root –p create testdb**. Esta base de datos servirá para asegurarnos del correcto funcionamiento de la instalación y podrá ser borrada una vez lo hayamos confirmado.

Instalación de Jena2

Desde la página oficial de Jena2[53], podemos descargar el software necesario para su instalación. Se trata solo de un fichero .zip.

La instalación de Jena2 es muy simple. Solo debemos descomprimir el fichero descargado de la Web oficial de Jena2. Debemos incluir en la variable de entorno classpath, la ruta de la carpeta /lib de Jena2.

Instalación de Joseki

Desde la página oficial de Joseki[54], se puede descargar el fichero .zip que contiene todo el software.

Los pasos a seguir son los siguientes:

1. Debemos descomprimir el fichero .zip que ha sido descargado de su Web oficial. El fichero de la configuración por defecto de Joseki es etc/joseki.n3. Necesitamos escribir los modelos que queremos publicar. La instalación se puede comprobar con la configuración del fichero por defecto, siendo los modelos “testdata” y “books”.

2. En la variable de entorno classpath, se debe añadir la carpeta lib/ para que se tenga acceso a todos los ficheros .jar.

[53] http://jena.sourceforge.net/downloads.html
[54] http://www.joseki.org/downloads.html