TRABAJO FINAL DE CARRERA

PROYECTO. LENGUAJES DE CONSULTA PARA DOCUMENTOS RDF.

TITULACIÓN INGENIERIA INFORMATICA.

ALUMNO. Maria Isabel Lamas Codesido

TUTOR. Oscar Celma Herrada

9 de enero 2006

A mis padres y a Isi por todo el tiempo que no puedo dedicarles y por todo el apoyo que me dan día a día.

Presentación.

Intenet fue concecibido para ser utilizado por humanos, si bien todo lo que hay la Web puede ser leído por lo agentes computacionales, no es entendible para ellos. El aumento continuo de documentos en la red y su estabilidad ocasionan que los directorios clasificados por humanos solo sean capaces de cubrir una parte de la red al mismo tiempo que son difíciles de actualizar (esto se traduce muchas veces en enlaces rotos, pérdida de información a buscar ..)

La Web Semántica no es más que una evolución de la Web actual que va a permitirnos catalogar la información para poder manipularla de una forma rápida y eficaz que hará posible que los agentes computacionales puedan entenderla evitando que el ser humano tenga que intervenir para seleccionarla. El objetivo es que estos agentes compartan, combinen e infieran información de manera sencilla y no ambigua.

Las distintas capas de la arquitectura propuesta por **Tim Berners-Lee**[1] han provocado que surjan diferentes lenguajes para cada una de estas capas.

RDF y **OW**L son ejemplos de estandarizaciones del consorcio W3C[2] que se estudiarán en este proyecto. Se analizarán además algunos lenguajes que permiten interrogar los documentos de la Web Semántica similares al SQL de los SGBD relacionales. Algunos de estos lenguajes son RQL, RDQL, SeRQL, RDFQL, SquishQL, Triple, Versa y la última estandarización de W3C **SPARQL**.

La Web Semántica necesita SGBDs que se adapten al nuevo modelo de datos de RDF, se analizará un estándar libre como es MySQL 5 y la última versión de oracle 10g. con características específicas para la persistencia de datos de modelos RDF.

El desarrollo práctico de este proyecto termina con un ejemplo de un buscador Web que utiliza Sesame y SeRQL para permitir hacer búsquedas en un repositorio creado en memoria que almacena el modelo RDF que se utilizará en todos los ejemplos de estas páginas.

[1] http://www.w3.org/DesignIssues/Metadata.html

[2] http://www.w3.org/

INDICE

EJEMPLOS

FIGURAS

TABLAS COMPARATIVAS Y DE ELEMENTOS

1. Introducción.

Tim Berners-Lee definió la Web Semántica como una evolución de la Web actual, donde la información tenga un significado bien definido y permita una mejor cooperación hombre-máquina. La interoperabilidad entre sistemas impone un entorno de trabajo en el que cada sistema emplea el significado de los datos en diferentes contextos. Este requisito queda cubierto por las tecnologías semánticas, puesto que ofrecen un marco de trabajo que permite conectar datos distribuidos y describir su contenido en función de su contexto. Por lo tanto, el uso de tecnologías semánticas posibilita la descripción de la naturaleza y contexto lógico de la información a intercambiar, mientras que permite que cada sistema mantenga su máxima independencia.

El objetivo es permitir la cooperación e intercambio de información de dos sistemas de información que inicialmente no se desarrollaron con esa finalidad, sin tener que modificar las aplicaciones ni los datos con los que trabaja cada sistema de información. Para ello, la Web Semántica hace uso de tecnologías precedentes de diferentes áreas, como son la inteligencia artificial, el área de lenguajes y resultados provenientes de la integración de aplicaciones empresariales.

Estudiaremos todas las tecnologías necesarias para poder desarrollar un proyecto que trabaje con modelos RDF. Esto implicará el estudio de conceptos relacionados con la semántica de los datos, ontologías y herramientas software necesarias para poder manipular la información de la ontología que se empleará para el desarrollo práctico final.

En algunos casos será necesario comparar las diferentes tecnologías para descartar aquellas que no se adapten completamente a las necesidades de los modelos de datos de la Web Semántica.

1.1. Justificación del TFC.

Ante el crecimiento desmesurado de información en Internet se demanda una evolución de la Web actual que permita que los agentes manejadores de información gestionen de una forma más acertada la información demandada. Los buscadores actuales no entienden la semántica de los datos esto hace necesario investigar entre las nuevas tecnologías que están disponibles en la Web para seleccionar aquellas que pueden obtener un mejor resultado ante esta problemática.

Una de las justificaciones de este trabajo es plantearnos que pueden ofrecer las nuevas tecnologías para conseguir interoperabilidad entre sistemas de información.

Trataremos de ver las limitaciones de los lenguajes Web Semánticos que se han propuesto así como descartaremos algunas herramientas frente a otras que se están utilizando actualmente para la consulta de datos semánticos. Para esto tendremos que estudiar los diferentes lenguajes de consulta RDF que existen en la actualidad. También se hace necesario el estudio de los frameworks que permiten realizar consultas con estos lenguajes de una forma rápida y sencilla.

Los gestores de datos relacionales también pueden permitirnos guardar la información contenida en estos grafos RDF. Veremos como podemos solucionar esta problemática con MySQL 5.0 y con oracle 10g.

En el desarrollo práctico se utilizarán algunas de las herramientas analizadas para realizar un buscador Semántico en grafos RDF.

1.2. Objetivos del TFC.

Los objetivos principales de este proyecto son analizar y descartar algunas de los lenguajes de consulta, gestores relacionales y repositorios que se están utilizando más habitualmente en el entorno de la Web Semántica. Con el estudio de las principales tecnologías implicadas en la Web Semántica se tratará de realizar una aplicación que permita el manejo de datos de un grafo RDF.

1.3. Enfoque y método seguido.

Para poder alcanzar los objetivos de este trabajo se hace necesario el estudio de las diferentes capas de la arquitectura de la Web Semántica.

El material utilizado para este desarrollo está disponible en la red. Los buscadores actuales nos han ayudado a encontrar mucha información relacionada con la Web Semántica. Hemos tenido que realizar las funciones de agentes humanos seleccionando la que se ha considerado más oportuna para este proyecto. Los principales documentos que se han consultado están en la página del consorcio W3C y en las de los desarrolladores de los productos ya mencionados (mySQL, oracle, Jena, Sesame, etc.).

Un documento importante para situarnos en este estudio ha sido el artículo de Tim Berners-Lee al que se hacemos referencias en las siguientes páginas.

Se han probado muchas herramientas para manejo de ficheros RDF, se han realizado consultas en varios lenguajes RDF y se han descartado las que han resultado ser menos eficaces para lo que se pretendía o las que han tenido una dificultad especial para proceder a su instalación.

1.4. Planificación del proyecto.

El proyecto se ha planteado en tres etapas diferentes, recolección de información, estudio de la misma, estudio de las diferentes herramientas que permiten desarrollar la parte práctica de este proyecto y desarrollo del buscador Web Music con las herramientas principales estudiadas. La arquitectura utilizada para el desarrollo de la parte práctica ha sido consecuencia del descarte de algunas tecnologías que presentaban problemas en las últimas versiones de su software[3].

1.5. Productos obtenidos.

Los productos obtenidos son una serie de comparativas relacionadas con la tecnología existente para el desarrollo de la Web Semántica y un proyecto que permite realizar búsquedas en sobre un modelo RDF que utiliza una ontología sobre el mundo de la *música.*

En concreto se han realizado comparativas siguientes:

- **Lenguajes de Consulta RDF.** En esta parte se ha concluido que tanto SeRQL como SPARQL son dos soluciones a lenguajes de consulta RDF que permiten realizar consultas en grafos RDF, los operadores y funciones de las que disponen permiten realizar casi cualquier consulta sobre grafos RDF. El resultado de esta comparativa puede verse en la Pág. 56.
- **Repositorios Jena y Sesame.** Dos arquitecturas diferentes que permiten trabajar con modelos RDF. Jena se hace más adecuada para sistemas locales, mientras

[3] Se ha descartado la versión 2 de Sesame que permite el manejo de SPARQL porque actualmente en la pàgina no existe documentación suficiente para poder utilizar Sesame con SPARQL.

que Sesame se defiende mejor en sistemas remotos. Algunas de las diferencias obtenidas pueden verse en la tabla de la Pág. 64.

- **Persistencia de datos en gestores relacionales.** Se ha utilizado Sesame para poder comparar los modelos persistentes que se crean con MySQL y con oracle 10g. Oracle 10g ha creado un módulo específico Espacial que facilita el manejo de grandes volúmenes de datos de modelos RDF. Sesame saca ventaja en el manejo de la persistencia respecto a Jena.

Además se ha desarrollado un buscador Web sobre grafos RDF, basados en la ontología sobre el mundo musical.
Se aporta también una presentación PowerPoint con los temas principales de este proyecto.

1.6. Descripción capítulos de la memoria.

1.6.1. Introducción.

Todos los capítulos de la memoria relacionados con la capa semántica y la capa ontológica mencionan un ejemplo de ontología relacionado con el mundo de la música.

Los ejemplos que se muestran se han probado con diferentes herramientas y sobre los ficheros RDF que se han aportado en la parte práctica de este proyecto. Estos ficheros se encuentran en la parte final de los anexos.

La ontología utilizada define dos clases, una de ellas para definir a los cantantes o grupos musicales llamada *artistas* y otra para definir la discografía MP3 de la que se dispone *canciones*[4]. La clase artistas define los atributos nombre, década, ciudad y nacionalidad y la clase canciones hace lo propio con la webMP3 y el artista/grupo que la interpreta.

1.6.2. Evolución de la Web.

Los buscadores actuales solo acceden a contenido estático, ignoran la parte dinámica de la Web generada a partir de bases de datos. Estos buscadores no entienden la semántica de los datos por lo que nos devuelven información irrelevante que usa una cierta palabra con un significado diferente al que pretendemos buscar. La Web semántica pretende construir documentos estructurados y dinámicos que puedan entender los agentes (humanos y computacionales).

1.6.3. Capa semántica.

La capa semántica propuesta por la arquitectura de Tim Berners Lee está formada por los lenguajes RDF y RDF Schema propuestos recientemente como estándares de World

[4] clase canciones " tracks".

Wide Web Consortium (W3C), en este capítulo se estudian las características de estos lenguajes así como algunas deficiencias. RDF establece los mecanismos para definir relaciones entre datos.

1.6.4. Capa ontológica.

La capa ontológica pertenece a un nivel superior de la arquitectura de la Web Semántica propuesta de Tim Berners-Lee incluye el concepto de Ontología. En este capítulo se estudia brevemente el lenguaje OWL, propuesta de W3C, que permite definir formalmente las relaciones entre términos. Las ontologías más típicas de la red constan de una taxonomía y de un conjunto de reglas de inferencia. OWL permite además la definición de restricciones sobre datos y sobre las relaciones en las que participan dichos datos.

1.6.5. Lenguajes de consulta para RDF.

Una vez que la información de métricas e indicadores está almacenada en repositorios RDF necesitamos un lenguaje de consulta para implementar la navegación semántica. Este capítulo se dedica al estudio de algunos lenguajes de consulta RDF. Se citan solo algunos de los que existen actualmente, la mayor parte de ellos nos recuerdan al SQL utilizado en los SGBDS existentes.

1.6.6. Depósitos para RDF.

Se han seleccionado al azar dos de los repositorios más citados en los foros de la Web Semántica, Sesame y Jena. En este capítulo se estudian sus principales características y las ventajas que nos ofrecen cada uno de ellos.

1.6.7. Persistencia de datos RDF con Sesame y MySQL.

Los dos repositorios de RDF analizados en el capítulo anterior nos permiten conectar con algún gestor de base de datos relacional (últimas versiones de MySQL, PostgreSQL y oracle) para guardar en una base de datos las tripletas RDF del modelo que se utiliza. En este capítulo se instala Sesame para poder trabajar con el gestor relacional MySQL se verá como automáticamente se crea una base de datos en la que se guardarán las tripletas del modelo en varias tablas “resources”, “literals”.
Se estudia también las nuevas características pensadas por los desarrolladores de oracle 10g. para solucionar las necesidades de almacenamiento de grandes cantidades de datos que requiere la tecnología implicada en la Web Semántica.

2. Evolución de la Web: Web semántica y conceptos relacionados.

Uno de los mayores problemas a los que nos enfrentamos hoy día en la sociedad de la información es la sobrecarga de información, un problema que se potencia por el enorme tamaño de la World Wide Web (WWW). La Web nos proporciona acceso a millones de recursos, independientemente de su situación física y lenguaje.

Para poder tratar esta enorme cantidad de información, han surgido nuevos modelos de negocio en la Web, como los buscadores comerciales (de los cuales Google[5] es el referente con diferencia).

Con el crecimiento continuo de la WWW, se espera que los buscadores tengan dificultades para mantener la calidad de recuperación de los resultados. Además, los buscadores sólo acceden a contenido estático, e ignoran la parte dinámica de la web (páginas generadas a partir de bases de datos). Los buscadores actuales no entienden la semántica de los datos. Estos buscadores basados en palabras clave suelen devolver información irrelevante que usa una cierta palabra con un significado diferente del que se pretende en la búsqueda, y pierden información cuando no reconocen palabras diferentes pero con el mismo significado que la buscada. Esto hace que sea necesaria la lectura humana para extraer información relevante de un origen determinado. Mantener orígenes textuales débilmente estructurados representa una tarea dificultosa, y consumidora de tiempo.

Tim Berners-Lee[6], el creador de la WWW tenía otra visión sobre lo que es hoy por hoy la web que conocemos. El pensó en una red de recursos que nos permitiera programar agentes que navegaran la infinitud de sitios pudiendo obtener la información que necesitamos sin tener que indicarle de donde obtenerla o que significado debe tener cada recurso, transformando finalmente esa información a un formato que sea fácilmente entendible por nosotros. Esa Web, que aún se encuentra en una fase de desarrollo, es lo que se conoce como la Web Semántica. Tim Berners-Lee autor del artículo Scientific American de Mayo del 2001 " The Semantic Web" define en ese documento el concepto de Web Semántica.

La Web Semántica es el futuro de la web actual, una web que facilitará la localización de recursos, la comunicación entre sistemas y programas, que nos ayudará a gestionar nuestro día a día, hasta llegar a niveles que hoy pueden ser considerados como de ciencia ficción. Se pretende ampliar la interoperabilidad entre los sistemas informáticos y reducir la mediación de operadores humanos en los procesos inteligentes de flujo de

[5] http://www.google.es

[6] Tim Berners-Lee creador de la WWW y director del Consorcio de la Web. Recibió el premio Príncipe de Asturias en 2002 como creador de la WWW junto con Lawrence Roberts, Vinton Cerf y Robert Kahn, considerados los "padres" de Internet.

informáticos. De todas formas, la Web Semántica no pretende reemplazar la web actual, sino más bien complementarla. En la siguiente tabla podemos ver los cambios que tendrá la Web actual hasta convertirse en la Web Semántica:

Característica	Primera Generación	Segunda Generación
Lenguaje principal	HTML	*XML*
Forma y estructura	Documentos no estructurados	Documentos estructurados
Semántica	Semántica implícita	Etiquetado explícito (*metadatos*, Web Semántica)
Relación entre contenido y forma	HTML = fusión de forma y contenido	estructura en capas de forma y contenido: XML + transformación (p.e., *XSL*) a HTML, WML, PDF, u otros formatos
Editabilidad	Documentos estáticos	Documentos dinámicos
Descomponibilidad y recomponibilidad	sitios web monolíticos, independientes	Bricolaje (agregación), sindicación, reasignación de contenido
Interactividad	Medio de difusión unidireccional	*Web editable*, bidireccional
Audiencias	Para consumo humano	Para humanos y ordenadores (p.e., *servicios web*)
Control de producción	Centralizado	Descentralizado (*peer-to-peer P2P*)

Tabla 1. Comparativa Web actual y Web Semántica (WS)[7].

Los principales componentes de la Web Semántica son los metalenguajes y estándares de representación XML, XML Schema, RDF[8] Schema [Pág.19] OWL[9][Pág.37]. Estas tecnologías se combinan para aportar descripciones explícitas de los recursos de la Web (ya sean estos catálogos, formularios, mapas u otro tipo de objeto documental). De esta forma el contenido queda desvelado, como los datos de una base de datos accesibles por la Web, o las etiquetas inmersas en el documento (normalmente en XHTML, o directamente en XML, y las instrucciones definidas en una hoja de estilos aparte).

[7] http://istpub.berkeley.edu:4201/bcc/Fall2001/feat.2ndgenweb.html
[8] RDF. http://www.w3.org/RDF/
[9] OWL. http://www.w3.org/TR/owl-features/

XML es un lenguaje para documentos semiestructurados, donde el orden de los elementos es importante, su modelo representa un árbol; en cambio RDF es un lenguaje para metadatos, el modelo es un grafo etiquetado y dirigido donde el orden no es relevante.

XML aporta semántica a los documentos al incorporar etiquetas que tratan sobre el significado de la información. Esto es un avance sobre HTML, donde las etiquetas expresan sólo la estructura del documento. Para un dominio particular, XML es una alternativa como lenguaje de marcado semántico, utilizando esquemas XML para definir vocabularios o bien una combinación de XML y RDF.

Las reglas de inferencia en la ontología proveen más poder aun. Una ontología puede expresar la regla "Si un código de ciudad está asociado a un código de estado, y si una dirección es el código de ciudad, entonces esa dirección tiene el código de estado asociado". Un programa podría entonces sin dificultad deducir que una dirección de la Universidad de la UOC, al estar en la ciudad de Barcelona, debe estar en el país de España, que queda en Europa, y debería por lo tanto estar formateado según los estándares de Europa. La computadora no "entiende" nada de lo que está procesando, pero puede manipular los términos de modo mucho mas eficiente produciendo ganancia para la inteligibilidad humana.

Las ontologías pueden mejorar el uso de la Web en muchos sentidos. Desde facilitar las búsquedas haciendo que los buscadores se fijen en páginas concretas y no en palabras ambiguas. Mucho más interesante es el caso en que las ontologías pueden vincular informaciones en una página a las estructuras de conocimiento y a las reglas de inferencia asociadas.

2.1. Arquitectura de la Web Semántica.

Tim Berners Lee propuso la siguiente arquitectura para la Web Semántica:

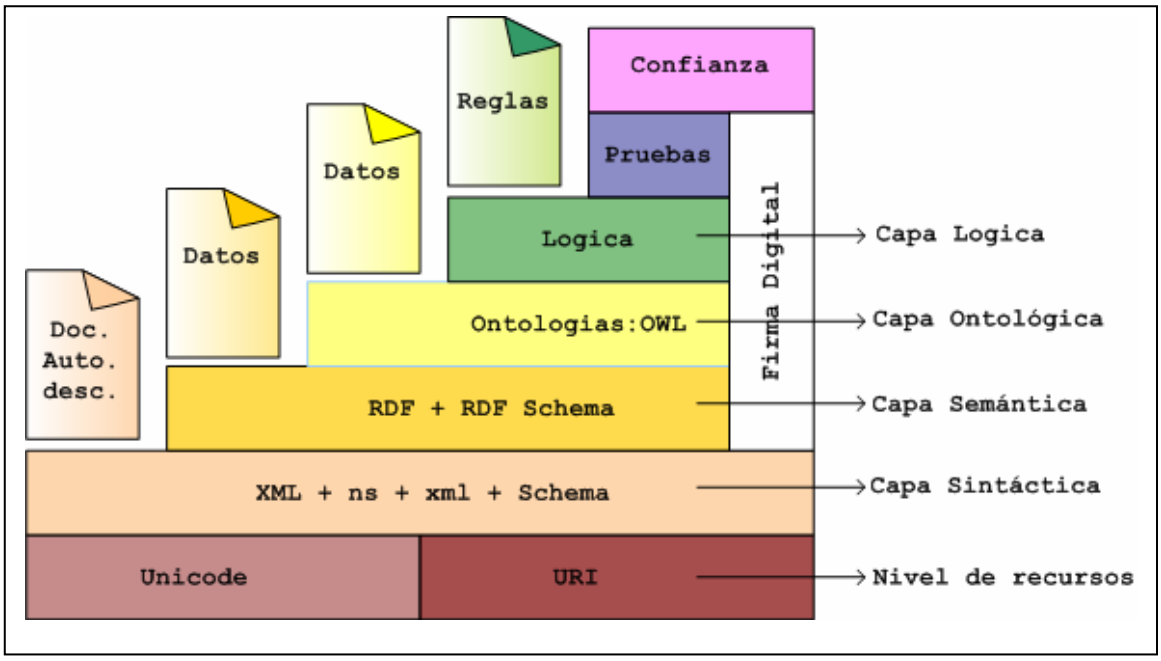


Figura 1 Arquitectura Web Semántica propuesta por Berners-Lee

Cada uno de estos niveles tiene el siguiente significado:

Nivel de Recursos

En el primer nivel se incluye la identificación de recursos Web, estableciendo así la capital importancia que tiene definir el conjunto de recursos distribuidos por la red. Los **URIs** (*Uniform Resource Identifier*)[Pág.19] identifican de forma inequívoca un recurso introducido en la red, este identificador además cumplirá la función de identificador de objetos en el mundo real.
Unicode[10] es una norma de codificación del texto que permite utilizar los símbolos de diferentes idiomas sin observar caracteres extraños. Su objetivo es asignar a cada posible carácter de cada posible lenguaje un número y nombre único. Esto permite expresar información en la Web Semántica en cualquier idioma.

Nivel Sintáctico

[10] http://www.unicode.org/

En este nivel se soluciona el problema de cómo definir distintos lenguajes de etiquetado para añadir contenido semántico a las páginas web. **XML** (*eXtensible Model Language*) es un metalenguaje, un lenguaje para escribir lenguajes. La característica que ha hecho de XML la elección más acertada es la capacidad de: formalizar y validar los documentos, la estructuración y la capacidad de ampliación. Los entusiastas de XML opinan que algún día todos los navegadores procesarán XML en vez de HTML, a través de una migración progresiva mediante XHTML[11], SVG[12] (Scalable Vector Graphic), Xlink[13], etc.

Nivel Semántico.
RDF (*Resource Description Framework*)[Pag.19] es una recomendación de W3C diseñada para estandarizar la definición y el uso de las descripciones de meta-datos de los recursos Web. Sin embargo, también se utiliza para representar simples datos. Como su nombre indica, el área donde está enmarcado es la descripción de los recursos de la red, siendo los recursos cualquier cosa: personas, dispositivos, páginas, etc. RDF es un lenguaje de etiquetado basado en XML. Además, RDF ofrece estructuras formales (no ambiguas) gracias al uso de URIs, permitiendo la codificación, el intercambio y el procesamiento automatizado de meta-datos normalizados.

Nivel de Ontologías.
Las ontologías son la piedra angular de la propuesta de Berners-Lee. Para que esto se pueda llevar a cabo, el conocimiento no sólo debe ser legible por una máquina, sino que debe ser consensuado y reutilizable. **OWL** [Pág.37] es oficialmente (según W3C) la forma más apropiada de compartir conocimiento en la Web. Técnicamente, OWL se descompone en tres sublenguajes incrementales para ser utilizado según las necesidades: OWL lite, OWL DL y OWL full.

El papel de los Agentes.
La Web semántica tiene como objetivo último ser utilizada por sus usuarios, pero, ¿quiénes son los usuarios de una Web semántica? Sin duda alguna, los agentes inteligentes. Entidades software autónomas que realizan acciones complejas para conseguir sus objetivos, teniendo un carácter social con otros agentes, reactivo o la capacidad de movilidad. Para que en la Web actual un agente inteligente pueda realizar una tarea debe tener la capacidad de procesar lenguaje natural. Además necesita realizar una planificación sobre qué hacer, y para ello asume que en esa planificación hay un completo conocimiento del entorno. En la Web actual eso es simplemente imposible ya que es demasiado grande, demasiado cambiante y demasiado inconsistente. En la Web semántica el agente podrá leer los contenidos semánticos de una página en un lenguaje formal (OWL), utilizando conocimiento compartido (ontologías), o descubrir exactamente donde están los recursos que requiere (URIs). Los agentes y aplicaciones Web que exploten este conocimiento anotado en páginas Web serán capaces de interpretar el conocimiento ontológico y axiomas para realizar tareas como búsquedas inteligentes, y el mantenimiento de ontologías y de los recursos, etc.

[11] http://www.w3.org/TR/xhtml1/
[12] http://www.w3.org/TR/SVG/
[13] http://www.w3.org/TR/xlink/

Capa lógica.
El nivel de la lógica pretende dar flexibilidad a la arquitectura para realizar consultas e inferir conocimiento a partir de las ontologías de la capa anterior.
Está compuesta por un conjunto de axiomas y reglas de inferencia que los agentes (humanos o computacionales) podrán utilizar para relacionar y procesar la información. Estas reglas ofrecen el poder de deducir nuevas sentencias a partir de los datos y estructuras que están descritos en las capas XML y RDF, usando además las relaciones entre estos datos y estructuras definidas en la capa Ontológica.

Resto de capas.
El nivel de seguridad permite asignar grados de confianza y seguridad a los distintos recursos distribuidos en la Web, a través de firmas digitales, redes de confianza u otras técnicas de autenticación por red. A día de hoy, no existe una recomendación oficial por parte de W3C acerca del resto de capas de la arquitectura propuesta.
Las firmas digitales son bloques encriptados de datos que los ordenadores y los agentes pueden utilizar para comprobar que la información agregada procede de una fuente identificada y digna de confianza. Parece obvio que los agentes deberían desconfiar de aserciones que leyeran en la Web cuya identidad no esté verificada.

3. Capa semántica.

La arquitectura propuesta por Tim Berners-Lee plantea la necesidad de crear una "capa semántica" basada en sistemas de metadatos "entendibles" por las máquinas y que sirvan para descubrir mejor la información que hay en la Web. Esta capa trata de dar un significado bien definido a la información, permitiendo un mejor trabajo de colaboración, tanto a ordenadores como a personas.
RDF es la propuesta para esta capa del W3C que veremos a continuación.

3.1. RDF (Resource Description Framework).

RDF[14] es el lenguaje recomendado por el W3C para representar los recursos y objetos en la Web Semántica. RDF ofrece una estructura semántica in-ambigüa. Proporciona interoperabilidad entre aplicaciones que intercambian información legible por la máquina en la Web. RDF destaca por la facilidad para habilitar el procesamiento automatizado de los recursos Web. RDF puede utilizarse en distintas áreas de aplicación; por ejemplo: en recuperación de recursos para proporcionar mejores prestaciones a los motores de búsqueda, en catalogación para describir el contenido y las relaciones de contenido disponibles en un sitio Web, una página Web, o una biblioteca digital particular, por los agentes de software inteligentes para facilitar el intercambio y para compartir conocimiento.

RDF no es independiente de la sentencia ya que está implementado sobre XML. Los esquemas XML pueden utilizarse para validar las sintaxis de las expresiones RDF/XML. Es un modelo de datos que se articula en tripletas objeto-atributo-valor que representan sentencias.

---

14 El lenguaje *Resource Description Framework (RDF)* está recogido en 6 recomendaciones del W3C: *Primer, Concepts, Syntax, Semantics, Vocabulary (Schema) y Test Cases.*

W3C. *RDF Primer*. http://www.w3.org/TR/rdf-primer/: *The Resource Description Framework (RDF)* es un lenguaje para referenciar la información de los recursos de la World Wide Web

W3C. *Resource Description Framework (RDF): Concepts and Abstract Syntax.* http://www.w3.org/TR/rdf-concepts/.

W3C. *RDF/XML Syntax Specification (Revised).* http://www.w3.org/TR/rdf-syntax-grammar/.

W3C. RDF Semantics. http://www.w3.org/TR/rdf-mt/: Especifica una semántica precisa y ofrece un completo sistema de reglas de inferencia para RDF y RDF Schema (RDFS)

W3C. RDF Vocabulary Description Language 1.0: RDF Schema. http://www.w3.org/TR/rdf-schema/

W3C. *RDF Test Cases.* http://www.w3.org/TR/rdf-testcases/ Este documento describe el *RDF Test Cases* ofrecido por el RDF Core Working Group.

Sentencia o afirmación.
El modelo utiliza para su formulación básica, las sentencias que son afirmaciones con sujeto, verbo y predicado. De esta forma podemos encadenar varias sentencias formando estructuras complejas.
Una sentencia sencilla sería *ABBA es de nacionalidad US* , si queremos añadir más información a esta sentencia podemos crear otra que permita saber por ejemplo de que década es ABBA esto podemos hacerlo con la frase ABBA es de la década de los 80.
Esta forma general de construir oraciones puede aproximarse más a una forma más centrada en propiedades y valores, tal y como maneja la documentación sobre RDF. Para ello cambiamos la construcción de sentencias a una forma más genérica:

X tiene una propiedad Y cuyo valor es Z

Las sentencias anteriores quedan de la forma:

ABBA tiene una nacionalidad cuyo valor es US.
ABBA pertenece a una década cuyo valor es 80.

Esta es la forma que tiene RDF de describir los recursos, establece afirmaciones sobre el valor que toma una determinada propiedad de un determinado objeto, esta es la base con la que se construye todo el modelo.

La terminología explícita para citar estos elementos es

- **Sujeto (Subject).** Ser sobre el cual se realiza la afirmación.
- **Predicado (Predicate).** Propiedad del objeto a la que se refiere la afirmación, en ocasiones es conveniente expresarlo como un verbo.
- **Objeto (Object).** Valor que toma una propiedad.

El conjunto de estos tres elementos forma una ***sentencia, afirmación o tripleta*** (en algún texto triple).

Una cuestión fundamental es que un objeto puede ser a su vez el sujeto de una nueva afirmación, creándose de esta forma estructuras que pueden llegar a ser muy complejas.

El conjunto de varias sentencias forma lo que se llama en terminología RDF ***grafo.***

URIs.

Para distinguir unívocamente cada elemento de las sentencias surge el concepto de URI (Uniform Resource Identifier), las URIs, a diferencia de las URLs (Uniform Resource Locator) no están orientadas a localizar recursos, sino a su identificación, por lo tanto no es necesaria su existencia física y puede ser utilizado para identificar conceptos abstractos utilizados dentro de un modelo RDF.

RDF utiliza un conjunto más genérico de URIs llamado URIrefs (URI referenciados) que continen una cadena al final del URI separadas por el carácter "#", como si se tratara de un ancla HTML.

La ventaja de las URIrefs es que dada la forma de gestión descentralizada, permite que cada organización pueda establecer independientemente su propio vocabulario sin tener riesgo a colisionar con el espacio de nombres de otras organizaciones.

En lo que queda de este texto utilizaremos indistintamente URI y URIrefs para referirnos en cualquier caso a este último.

A pesar de que una URI por lo general no se refiere a una localización física de un determinado recurso, muchas veces una URI va a corresponder con una URL que puede contener una descripción de ese vocabulario, probablemente un esquema RDF con la descripción formal de dicho vocabulario.

La especificación RDF utiliza una ***notación abreviada*** de los URIref, consistente en utilizar espacios de nombres cualificados (*qualified names*) al igual que los utiliza XML.

Un *nombre cualificado*[15] consiste en un prefijo y un sufijo separadazos por dos puntos ":". El prefijo es una cadena que substituye a una URI, de manera que para resolver el nombre cualificado basta con concatenar la URI que substituye el prefijo con el sujeto.

Para expresar una sentencia de las anteriores con un nombre cualificado:

MP3: http://www.mp3.com/search.php?stype=artist&query=ABBA&amp

MP3: ABBA MP3:nacionalidad "US" **.**

El final de la tripleta se marca con un punto **"."**

Queda claro entonces que:

- ***Un recurso (i.e. nodo, discos, canciones, autores)***
- ***Propiedad (i.e. predicado, relación binaria, editado por, interpretado por)***
- ***Sentencia (i.e. tripleta):*** **asignación de valores a propiedades de un recurso**

  Sujeto (recurso, URI) + predicado (propiedad, URI) + objeto (recurso, URI ó literal)

[15] Encontraremos el término QNAME en el capítulo dedicado a los lenguajes RDF.

Para representar este modelo disponemos de:

1. **Grafo dirigido y etiquetado.**

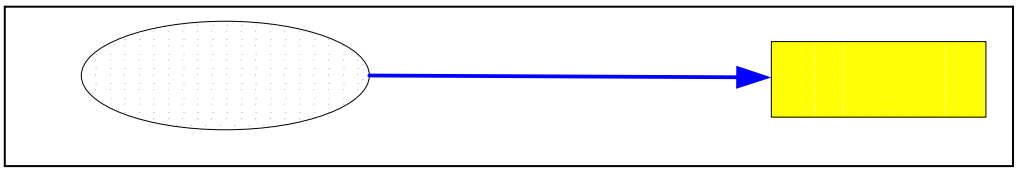


Figura 2. Grafo sentencia RDF (IsaViz)

La sentencia representa la construcción básica que establece el modelo de datos RDF. Los nodos dibujados con óvalos representan recursos y los arcos representan propiedades.

El objeto (el valor de la propiedad) puede ser otro recurso identificado entonces por una URI o bien un valor literal. Los nodos que representan cadenas de literales pueden dibujarse como rectángulos.

El sentido de las flechas es importante. El arco siempre empieza en el sujeto y apunta hacia el objeto de la sentencia.

El ejemplo siguiente muestra una sentencia que expresa la frase " El nombre del recurso que aparece en la Web http://www.mp3.com/search.php?stype=artist&query=ABBA es ABBA".

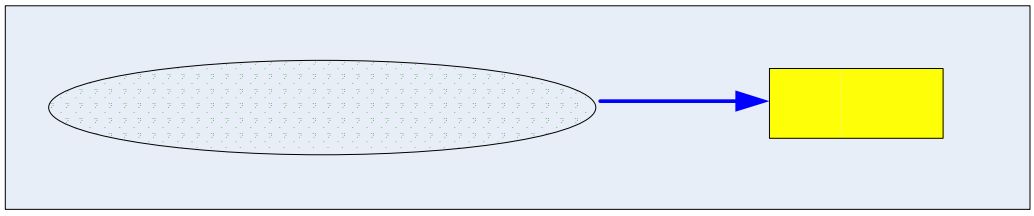


Figura 3. Ejemplo grafo RDF[16] con una sentencia.

SUJETO

[16] Se ha utilizado la herramienta IsaViz[16] para dibujar los grafos RDF.

Supongamos que queremos añadir la propiedad ciudad para indicar de que sitio es este grupo musical. Para esto tendremos:

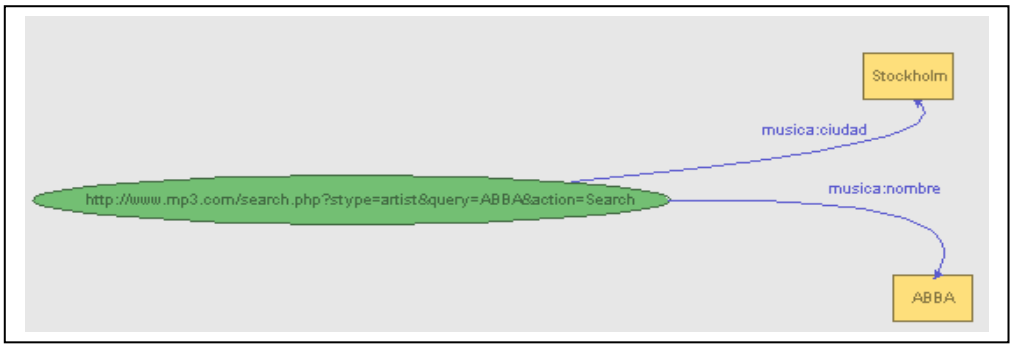


Figura 4. Grafo con 2 sentencias RDF.

Al añadir un arco más lo que tenemos son dos sentencias en vez de una sentencia que tenía el ejercicio anterior. Pasamos a tener un grafo con dos sentencias.

2. **Serialización RDF/XML** donde XML es utilizado como medio de transporte o sintaxis.

```
<rdf:RDF
        xmlns:rdf="&rdf;"
        xmlns:musica="&musica;">


  <rdf:Description
rdf:about="http://www.mp3.com/search.php?stype=artist&amp;query=ABBA&a
mp;action=Search">
    <rdf:type rdf:resource="&musica;Artist"/>
    <musica:nombre>ABBA</musica:nombre>
    <musica:ciudad>Stockholm</musica:ciudad>
  </rdf:Description>

</rdf:RDF>
```

Ejemplo 1. Serialización XML.

3. **N-TRIPLES** un listado de las tripletas del modelo que se representa.

Esta notación es muy simple, consiste en indicar el sujeto, predicado y objeto entre paréntesis angulosos y en este preciso orden. El final de cada sentencia o triple viene indicada por un punto "**.**" .

```
<http://www.mp3.com/search.php?stype=artist&query=ABBA&action=Search>
<http://www.semanticaudio.org/ontology/0.1#ciudad> "Stockholm" .
<http://www.mp3.com/search.php?stype=artist&query=ABBA&action=Search>
<http://www.semanticaudio.org/ontology/0.1#nombre> "ABBA" .
```

Ejemplo 2. Representación N-TRIPLES.

4. **Notation 3 (N3).**

```
# Base: file:/C:/IsaViz/export/artists22.txt
@prefix :        <#> .

<http://www.mp3.com/search.php?stype=artist&query=ABBA&action=Search>
      <http://www.semanticaudio.org/ontology/0.1#ciudad>
               "Stockholm" ;
      <http://www.semanticaudio.org/ontology/0.1#nombre>

               "ABBA" .
```

Ejemplo 3. Representación de sentencia RDF con Notation 3.

Así pues, una tripleta se representa mediante nodos conectados por líneas con etiquetas. Los nodos representan recursos y las líneas con etiquetas las propiedades de esos recursos. Los 3 elementos de una tripleta se representan mediante URIs.

Cada tripleta corresponde a un arco único en el grafo, que se completa con el comienzo del arco y el final de los nodos (el sujeto y el objeto de la declaración). A diferencia del grafo dibujado (aunque como la declaración original), la notación de tripletas requiere que un nodo se identifique separadamente para cada declaración que aparece.

3.1.1. Características de RDF

Algunas de las características de RDF son:

Independencia

Dado que una propiedad es un recurso, toda organización independiente o incluso cada persona puede inventarlas. Podemos inventar una propiedad llamada "Artista" y otros pueden inventar una propiedad llamada "Discográfica" que podría aplicarse, por ejemplo, a recursos asociados con discos.

Intercambio

Dado que las sentencias RDF se escriben en XML pueden ser fácilmente usadas para intercambiar información.

Escalabilidad

Las sentencias RDF son simples, registros con tres campos (Recurso, propiedad, valor) por lo que son fáciles de manejar y de usar para buscar objetos aun en volúmenes

realmente grandes. La Web ya es lo suficientemente grande y continúa creciendo constantemente, por eso la escalabilidad es importante.

Las propiedades son recursos

Las propiedades pueden tener sus propias propiedades y pueden ser encontradas y manipuladas como cualquier otro recurso. Esto es importante porque tendremos muchos recursos que manejar. Por ejemplo, podríamos tener que saber si alguien tiene definido una propiedad que describa el tipo de música que interpreta un determinado artista.

Los valores pueden ser recursos

Por ejemplo, la mayoría de las páginas Web podrían tener una propiedad llamada "home" que apunte al *home* del sitio. Por lo tanto los valores de sus propiedades que podrían incluir el titulo y autor de la página también tienen que incluir recursos

Las sentencias pueden ser recursos

Las sentencias también tienen propiedades. Dado que la Web es demasiado grande como para que cada uno provea el suyo tendremos que realizar búsquedas basadas en los metadatos de otras personas, tal y como se hace hoy con Yahoo[17]. Esto significa que querremos, dada una sentencia como "El tema de esta página es música" poder preguntar "Quien lo dice", "Cuando". Una forma útil de hacer esto es mediante metadatos y por ello las sentencias deben de tener sus propias propiedades.

Reificación. Sentencias sobre sentencias.

La reificación se basa en poder describir otras declaraciones utilizando RDF por ejemplo, para registrar información sobre cuándo se han hecho las declaraciones, quién las ha hecho, u otra información similar.
Puede verse un ejemplo detallado en el apartado [Pág. 30].

Jerarquía de clases

Las clases pueden organizarse en jerarquías, **subclassOf** define que una clase es subclase de otra. A es subclase de B si todo individuo de A pertenece a B. Una clase puede tener múltiples superclases.

[17] http://www.yahoo.com/

3.1.2. Por que no usar solo XML

Si bien XML permite definir datos de una forma estructurada, de manera que puedan ser compartidos y procesados automáticamente, no permite especificar la semántica de dicha estructura, de una manera formal que pueda ser interpretada por un ordenador.

Además, el orden en el cual los elementos aparecen en un documento XML es significativo y muchas veces necesario, en cambio en RDF el orden de las sentencias es irrelevante. El orden impuesto en XML no tiene tanto sentido en el mundo de los metadatos. Nadie se preocupa sobre si el *artista* o el *título* de la canción es listado primero siempre. Además mantener el orden correcto de millones de elementos de datos es caro y difícil.

3.1.3. Tipos de elementos en RDF

Para poder formar grafos RDF (conjuntos de sentencias) disponemos de una serie de elementos que nos permitirán relacionar y establecer propiedades entre los objetos que participan en el modelo que se pretende construir:

Tripleta y sus partes	Contenedores
rdf:Statement **rdf:subject** **rdf:predicate** **rdf:object** **rdf:value**	**rdf:Bag** **rdf:Seq** **rdf:Alt**
Propiedades y sus valores	**Listas**
rdf: Property **rdf: XMLLiteral** **rdf: type**	**rdf:List** **rdf:first** **rdf:rest** **rdf:nil**

Tabla 2. Elementos RDF.

3.1.3.1. Primitiva rdf:Statement.

Corresponde con el conjunto denominado sentencia. Si se necesita declarar quien es el autor de una declaración en particular, es necesario que la declaración sea un recurso, *rdf:Statement* sirve para tomar una declaración formulada anteriormente en un recurso. Estos recursos deben de tener por lo menos tres propiedades: *rdf:subject, rdf:object, rdf:Predicate.* Es muy utilizada en las reificaciones [Pág. 30].

3.1.3.2. Primitiva rdf:subject.

Corresponde con la primitiva "subject" sujeto del modelo RDF.

3.1.3.3. Primitiva rdf:predicate.

Corresponde con la propiedad *predicado* en el modelo RDF.

3.1.3.4. Primitiva rdf:object.

Corresponde con el elemento objeto del modelo RDF. Se usa para identificar el valor de la propiedad en una sentencia modelada.

```
  <rdf:Statement
rdf:about="http://www.mp3.com/search.php?stype=artist&amp;query=
ABBA&amp;action=Search">
    <rdf:subject rdf:resource="ABBA" />
    <rdf:predicate rdf:resource="&musica;Artist"/>
    <musica:object>http://www.mp3.com/search.php?stype=artist&amp;query=ABBA
</musica:object>
  </rdf:Description>
```

Ejemplo 4. rdf:subject, rdf:predicate, rdf:object, rdf:Statement

3.1.3.5. Primitiva rdf:value.

Corresponde con la propiedad value.

3.1.3.6. rdf:Property.

Representa el subconjunto de recursos RDF que son propiedades. El dominio de estos recursos se describe mediante la propiedad *rdfs:domain*, y el rango mediante *rdfs:range*, las jerarquías de propiedades se describen mediante *rdfs:subPropertyOf*.

rdf:range es una instancia de *rdf:Property* que se usa para establecer que los valores de una propiedad son instancias de una o más clases; rdfs:domain es una instancia de *rdf:Property* empleada para establecer que un recurso con una cierta propiedad es instancia de una o más clases.

3.1.3.7. Primitiva rdf:type.

El modelo RDF prevé algunas primitivas importantes para una mejor descripción de recursos, de ellas tenemos rdf:type para definir tipos (instancias)de recursos, y las primitivas para definir contenedores. Rdf:type indica una relación binaria entre dos elementos, estableciendo un mecanismo de instanciación, o sea especifica que un elemento es una instancia de otro. De esta forma se incluyen en una misma descripción datos y metadatos.

Por ejemplo si queremos crear una clase llamada “Artista” que contiene a todas las artistas tendríamos lo siguiente:

```
:Artista rdf:type rdfs:Class.
```

Ejemplo 5. rdf:type rdfs:Class

Ahora podríamos indicar que “ Artista “ es un tipo de Persona:

```
:Artista rdf:type rdfs:Persona.
```

Ejemplo 6. rdf:type

Además podemos definir los atributos de un tipo indicando rdf:Property y usarlas cuando necesitemos:

```
:nombre rdf:type rdf:Property.
:Artista :nombre  “ABBA”.
```

Ejemplo 7. rdf:Property

El esquema RDF nos permite definir además rdfs:subClassOf y rdfs:subPropertyOf de esta forma podemos indicar que *artista* es una subclase de persona. Y además que *actor* también es una subclase de persona.

```
:Artista rdfs:subClassOf :Persona
:Actor rdfs:subClassOf :Persona
```

Ejemplo 8.rdfs:subClassOf

Así podemos indicar que ABBA es un Artista y que Richard Gere es un actor:

```
:ABBA  rdf:type  :Artista
:Richard Gere rdf:type :Actor
```

Ejemplo 9. rdf:type

3.1.3.8. Contenedores RDF.

Los contenedores o repositorios se utilizan para mantener listas de recursos o literales. RDF define tres tipos de objetos contenedores:

Bag (rdf:Bag) Lista desordenada de recursos o literales. Los *bags* se utilizan para indicar que una propiedad tiene múltiples valores y que no es significativo el orden en el que se dan esos valores.

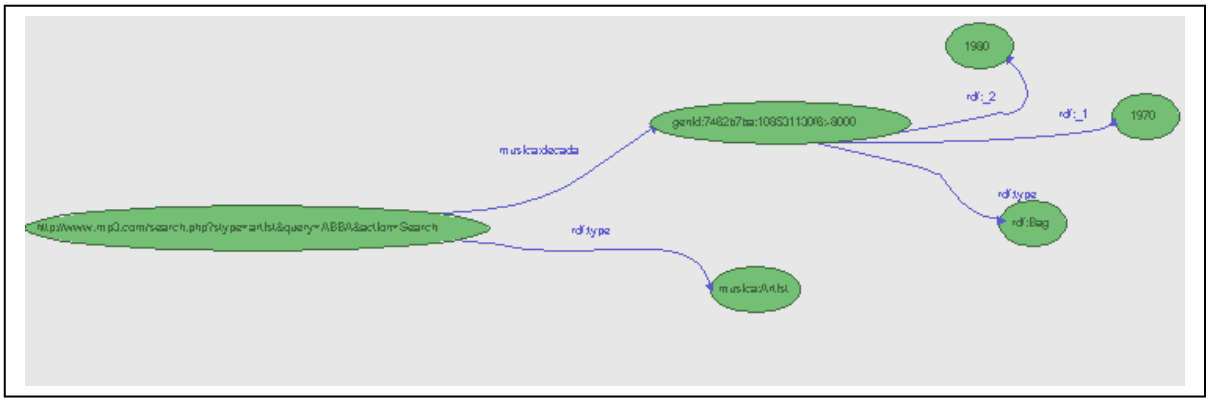
Sequence. Una lista ordenada de recursos o literales. *Sequence* se usa para manifestar que una propiedad tiene múltiples valores y que el orden de los valores es significativo.

Alternative. Una lista de recursos o literales que representan alternativas para un valor (individual) de una propiedad. Una aplicación que utiliza una propiedad cuyo valor es una colección alternativa, sabe que se puede elegir como correcto uno cualquiera de los ítems en la lista.

Una posible definición de contenedor sería:

```
 <rdf:Description
rdf:about="http://www.mp3.com/search.php?stype=artist&amp;query=ABBA&amp;action=Search">
   <rdf:type rdf:resource="&musica;Artist"/>
    <musica:decada>
     <rdf:Bag>
       <rdf:li resource="1970"/>
       <rdf:li resource="1980"/>
     </rdf:Bag>
   </musica:decada>
 </rdf:Description>
```

Ejemplo 10. Ejemplo de contenedor RDF.



Ejemplo 11. Grafo RDF con ejemplo de contenedor.

3.1.3.9. Reificación en RDF.

RDF permite definir sentencias sobre los recursos Web, sentencias sobre sentencias.

Para realizar declaraciones sobre declaraciones tenemos que construir un modelo de la sentencia original, este modelo es un nuevo recurso al que podemos anexar propiedades adicionales. Esta característica permite anidar declaraciones (reificación).

Formalmente, una reificación en RDF significa expresar una sentencia como un recurso de cuatro propiedades:

Subject. La propiedad *subject* identifica el recurso que describe la sentencia modelada: es decir, el valor de la propiedad *subject* es el recurso sobre el cual se hace la sentencia original.

Predicate. Esta propiedad identifica la propiedad original en la declaración moderada. El valor de la propiedad *predicate* representa la propiedad específica en la sentencia original.

Object. La propiedad *object* identifica el valor de la propiedad en la sentencia modelada. El valor de la propiedad *object* es el objeto en la sentencia original.

type. Este valor describe el tipo del nuevo recurso. Todas las sentencias transformadas reificadas son instancias de *rdf:statement*; es decir tiene una propiedad *type* cuyo contenido es *rdf:statement*.

Estas 4 propiedades representan la sentencia original y puede usarse como objeto de otras declaraciones.

Supongamos que queremos expresar una sentencia como la que se visualizó en el ejemplo [Pág. 23]:

> “ ABBA es el nombre del recurso que aparece en http://www.mp3.com/search.php?stype=artist&query=ABBA&amp ”

Ahora queremos reificar esta sentencia, tomamos como nuevo objeto la sentencia original (sentencia reificada) y la usamos para crear nuevas sentencias.

Para expresar la frase: El grupo cuyo nombre es ABBA de nacionalidad SE es el creador de la Web

http://www.mp3.com/search.php?stype=artist&query=ABBA&action=Search

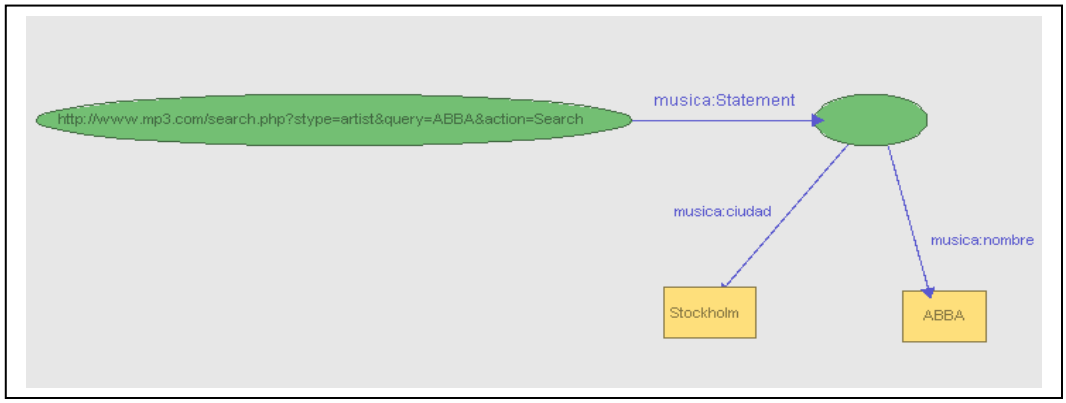


Figura 5. Grafo reificación.

3.2. RDFS (RDF Schema).

Con RDF hemos definido lo que se conoce en programación a objetos como instancias, es decir, ejemplares de una clase. Para definir las propias clases necesitamos utilizar RDF Schema (RDFS).

Con una ontología definida en RDFS, un conjunto de datos en RDF, y un motor de inferencia (*reasoner*) ya se pueden inferir nuevas relaciones que no se encontraban inicialmente en los datos y añadirlas a nuestros datos.

Los elementos más destacados que introduce con respecto a RDF son los que indicamos a continuación:

Clases de RDFS	Propiedades de RDFS
rdfs: Resource	**rdfs: domain**
rdfs:Class	**rdfs: range**
rdfs: Literal	**rdfs: subPropertyOf**
rdfs: Datetype	**rdfs: subClassOf**
rdfs: Container	**rdfs: member**
rdfs: ContainerMembershipProperty	**rdfs: seeAlso**
	rdfs: isDefinedBy
	rdfs: comment
	rdfs: label

Tabla 3. Elementos RDFS.

- **rdfs:Resource**. Es la clase más general en RDFS. Todas las clases son subclases de esta clase. Tiene dos subclases llamadas *rdfs:Class* y *rdfs:Property*. Cuando se especifica un esquema específico para un dominio RDFS, las clases y propiedades definidas en ese esquema se vuelven instancias de estos dos recursos. Todos los recursos RDF son instancias de esta clase.
- **rdfs:Class.** Define la clase de la ontología. Corresponde al concepto de clase en los lenguajes de programación orientados a objetos. Las clases RDF pueden representar páginas Web, personas, organizaciones, búsquedas en la web, etc.. Cada clase es miembro de *rdfs:Class*, clase de todas las clases; cuando se crea una nueva clase mediante un esquema RDF, la propiedad *rdf:type* del recurso representado por la clase adquiere como valor el recurso *rdfs:Class* o alguna subclase de *rdfs:Class*. Cuando un recurso tiene una propiedad rdf:type cuyo valor es una determinada clase, se dice que el recurso es una instancia de esa clase. Todas las clases son recursos. Una tripleta RDF/RDFS como (Persona, rdf:type, rdfs:Class) indica que Persona es una clase. Ver ejemplos [Pág 28] .
- **rdfs:subClassOf**. Atributo que indica que cierta clase es subclase de otra. Desde que *rdfs:subClassOf* es transitiva, las definiciones son especificadas por las clases mas específicas desde las mas generales, y recursos que son instancias de una clase son automáticamente instancias de todas las superclases de esa clase. En RDFS está prohibido que cualquier clase sea una *rdfs:subClassOf* de sí misma o de una de sus subclases. Solamente las instancias *rdfs:Class* pueden tener la propiedad *rdfs:subClassOf*. Ver ejemplos [Pág 28] .
- **rdfs:Literal**. Es la clase de los valores que pueden tomar las propiedades. Puede ser tipado (enteros o fechas, por ejemplo) o no tipado. Esta clase es subclase de *rdfs:Resource*.
- **rdfs:DataType**. Es la clase de los tipos de datos tipados. Es a la vez instancia y subclase de *rdfs:Class*. Todas las instancias serán subclases de *rdfs:Literal*.
- **rdfs:range.** Sirve para indicar que los valores que puede tomar una propiedad tienen que ser de la clase indicada.
- **rdfs:domain**. Sirve para indicar que los recursos que tengan este atributo asignado a cierta clase pertenecerán a la clase indicada. Es diferente de *rdf:type*, que se usa para indicar que un recurso es instancia de cierta clase.
- **rdfs:Container.** Clases base de los contenedores *rdf:Bag*, *rdf:Seq* y *rdf:Alt*.

Para los metadatos de la ontología (o definición de las clases), RDFS define varias propiedades:

- **rdfs:comment.** Proporciona la descripción en lengua natural de un recurso. Por ejemplo <*rdfs:comment*> " La música de ABBA se escucha en todo el territorio nacional" </*rdfs:comment*>.
- **rdfs:label**. Proporciona una versión legible para los humanos del nombre del recurso. Por ejemplo <*rdfs:label*> ABBA </*rdfs:label*>

- **rdfs:seeAlso**. Especifica un recurso que proporciona información adicional sobre el recurso principal. Por ejemplo <*rdfs:seeAlso*> http://www.ABBA.com </*rdfs:seeAlso*>.
- **rdfs:isDefinedBy** indica cuál es el recurso donde se define el recurso principal. Esta propiedad es una subpropiedad (*rdfs:subPropertyOf*) de la propiedad *rdfs:seeAlso*.

3.3. Limitaciones de RDF/ RDFS.

En este punto del estudio deberíamos considerar si no sería suficiente para la Web Semántica trabajar con XML y RDFS (considerando que RDF está incluido en RDFS). Tal y como ya se ha comentado XML, proporciona la interoperabilidad sintáctica; RDFS, la interoperabilidad semántica. La respuesta es negativa.

RDFS tiene las siguientes desventajas:

- Limitación a predicados binarios, complejidad del manejo de propiedades, complejidad de la reificación.
- No permite declarar restricciones de rango (rdfs:range) que sean válidas sólo para algunas clases. rdfs:range define el rango de una propiedad para **todas las clases**. Así, RDFS no permite expresar que los televisores sólo funcionan con corriente alterna, mientras que otros aparatos pueden funcionar con alterna o continua.
- No permite expresar **restricciones de cardinalidad.** Una propiedad no está restringida en cuanto al número de valores que puede tomar.
- No se pueden representar algunas características de las propiedades. En concreto, no se puede declarar que una propiedad es transitiva, simétrica , inversa o única.
- No se puede reflejar que unas determinadas clases **son disjuntas.** Por ejemplo pensemos en el típico ejemplo una clase vehículos del que hereda por ejemplo las motos y los coches, en este caso no podremos declarar que un coche no puede ser una moto y viceversa.
- Existen algunas expresiones cuya semántica no es estándar (es decir, no pueden expresarse mediante la lógica de primer orden). Esta característica es sumamente indeseable, pues causa que haya sentencias indecibles (sentencias de la que, dado un sistema de axiomas o premisas, no se puede afirmar o negar nada).

Frente al mantenimiento de un deseable equilibrio entre tratabilidad y expresividad de un lenguaje, RDFS se coloca en el extremo de la mínima expresividad, porque fue concebido para ser aplicable a toda la variada gama de recursos de la web. En contraste con los típicos lenguajes de representación de conocimiento, RDFS no ha sido concebido para ser una respuesta definitiva en la representación de conocimiento en un dominio particular, sino un núcleo pasible de ser extendido.

Aunque RDFS provee soporte para el modelado de conceptos ontológicos y relaciones, no lo provee para los axiomas, que son un ingrediente clave en la definición de una ontología.

La expresividad de RDF y RDFS para modelar ontologías completas es muy limitada: como hemos visto en el caso de RDF se limita a representar predicados binarios y RDFS está limitado a jerarquías de clases, jerarquía de propiedades con definición de restricciones de dominio y rango de dichas propiedades. RDF/RDFS carece de soporte para tipos de datos primitivos, carece de poder expresivo para representar axiomas (no hay negación, implicación, cardinalidad, cuantificación) no es posible definir propiedades (transitividad, simetría, etc.) no permite especificar condiciones necesarias y suficientes para establecer la pertenencia a una clase.

Con las limitaciones vistas sobre RDF/RDFS se hace necesario buscar una herramienta que nos permita **añadir lógica descriptiva** a RDFS y definir condiciones necesarias y suficientes para definir pertenencia a una clase.

Recientemente W3C ha especificado el lenguaje OWL[18] que añade mayor expresividad a RDF y RDFS con una semántica formal basada en lógica descriptiva. Este es el lenguaje recomendado por la W3C para la definición de ontologías Web [Pág.37].

3.4. Aplicaciones de RDF/RDFS.

Algunas de las áreas de aplicación de RDF se muestran a continuación:

- Recuperación de recursos.
- Catalogación.
- Agentes de software inteligentes.
- Calificación del contenido.
- Colecciones de páginas.
- Derechos de propiedad intelectual.
- Firmas digitales.

[18] http://www.w3.org/2004/OWL/

3.4.1. RSS.

Una de las aplicaciones más conocidas de RDF y XML es RSS[19]. Se trata de un vocabulario que se usa para describir información de tal manera que puede ser reutilizada. RSS es, actualmente, el acrónimo de *Real Simple Syndication*, aunque anteriormente lo fue de *Rich Site Summary* y de *RDF Site Summary*.

RSS es un formato para distribuir un conjunto de titulares organizados en forma de sumario o índice, llamados *canales* (channels o feeds), a cuyos contenidos se puede acceder a través de Internet sin necesidad de usar el navegador. Gracias a RSS, el usuario dispone de los titulares actualizados de muchos sitios webs sin que tenga necesidad de conectarse a cada uno de ellos. Muchas webs, sobre todo las que se actualizan constantemente como los *blogs* y los sitios web de los medios de comunicación, ya han sido adaptadas a RSS y permiten el acceso a una enorme cantidad de información de forma sencilla y automatizada. Un canal RSS siempre contiene el título de la cabecera, un breve resumen de la información del canal y un enlace que conduce al contenido o texto completo de la noticia. Los enlaces a canales se suelen indicar en los sitios Web utilizando un icono que contiene las siglas RSS.

Además de usar lectores específicos para leer los titulares de RSS, también es posible integrar estos titulares en páginas Web que no tengan relación con los autores de las noticias. Existen sitios, incluso, que están formados únicamente por titulares de otros. Si tenemos una página Web sobre un tema, podemos tomar los archivos RSS de otro sitio Web de nuestro interés e integrarlos en nuestra propia Web. De esta forma, dispondremos de los últimos titulares del Web generador de contenidos totalmente integrado dentro de nuestra Web y con los contenidos actualizados automáticamente.

Pero no sólo utilizan RSS los medios de comunicación (por ejemplo, El País, El Mundo, ABC, etc. ya cuentan con este servicio), también los ***principales buscadores*** como Yahoo y MSN Search utilizan RSS y recientemente, Ask Jeeves ha comprado el lector de noticias RSS Bloglines.

Una de las aplicaciones del RDF que más impacto está teniendo en la forma en que se consume y comparte la información no surge en las empresas, sino en los propios usuarios de la Web: los ***weblogs, blogs o bitácoras***. Muchas utilizan el formato RDF Site Summary (RSS) para difundir sus contenidos. A partir del interés desatado por los contenidos publicados en las bitácoras se están definiendo nuevas formas de compartir y consumir la información, que suponen una aplicación efectiva de los principios de la Web semántica y la definición de nuevos modelos de negocio: los proveedores de contenidos incorporan RSS para alcanzar mayor difusión (las propias bitácoras, periódicos, empresas para sus RRPP, buscadores para automatizar las consultas); surgen intermediarios que agregan los canales RSS, los sindican, los personalizan, permiten realizar búsquedas sobre dichos canales, o leerlos en el cliente de correo o por Web; y el usuario comienza a consumir la información a través de nuevos soportes (el cliente de correo, agregadores web, el webmail, aplicaciones personalizadas, ...), lo que le permite

[19] http://blogs.law.harvard.edu/tech/rss

hacerlo de forma más efectiva y personalizada, accediendo a muchas más fuentes en menos tiempo y con menor esfuerzo.

Otro de los servicios de RSS/RDF es poder sindicar las noticias de un sitio en mi sitio, es decir, si un sitio X permite la sindicación de noticias, con un Script en X lenguaje puedo hacer que esas mismas noticias aparezcan en mi Web.

3.4.2. Dublin core.

Uno de los primeros ejemplos de RDFS ha sido Dublin Core (DC). El formato Dublin Core nace para tratar de proporcionar metadatos para los materiales accesibles en red. El primer intento de normalizar el procesamiento técnico de la información de Internet fue rápidamente acogido, pues existía ya gran sensibilidad por las búsquedas y recuperaciones de los recursos de Internet.
DC está dividido en dos niveles diferentes: *DC simple* (simple Dublin Core) y *DC cualificado* (qualified Dublin Core). El primero es un conjunto de quince términos que define la semántica básica utilizada por DC y que cubre la mayoría de los aspectos relevantes de un recurso. DC cualificado es una ampliación de DC simple por el cual se define un nuevo término y todo un nuevo grupo de cualificadotes que restringen la semántica de los términos elementales de DC simple; de esta forma con DC cualificado se puede aportar información más precisa respecto a un recurso.

El objetivo original de Dublin Core fue definir un conjunto de elementos que puedan ser utilizados por autores para describir sus propios recursos en la Web. Enfrentando el hecho de a proliferación de recursos electrónicos y la incapacidad de los profesionales de las bibliotecas de catalogar todos estos recursos, la meta era definir unos pocos elementos y algunas reglas simples que pudieran ser aplicadas por autores inexpertos en catalogación. Los 15 elementos son:

Contenido	Propiedad intelectual	Aplicación
Coverage (cobertura*)* *Description (* descripcion *)* *Type (* tipo *)* *Relation (* relación *)* *Source (* fuente *)* *Subject (* materia o sujeto*)* *Title (* título *)*	*Contributor (* colaborador*)* *Creador (* creador *)* *Publisher (* editor *)* *Rights (* derechos *)*	*Date (* fecha *)* *Format (* formato *)* *Identifier (* identificador *)* *Language (* lengua *)*

Tabla 4. Elementos básicos de Dublin Core.

Dublin Core considera los documentos como objetos e integra todos los posibles tipos de materiales y su tratamiento técnico, ya sean bases de datos, imágenes digitales, bancos de imágenes, textos electrónicos, vídeos y películas en formato digital y no digital, objetos multimedia, grabaciones sonoras en formato digital y no digital. De esta forma se pretenden integrar los distintos catálogos en un dominio de acceso cruzado y múltiple a través de los enlaces hipertextuales.

Permite, además, que todos los campos sean repetibles, opcionales y puedan tener asociados cuantos enlaces se consideren necesarios.

La aplicación práctica del formato Dublin Core se ha proyectado en varias grandes bibliotecas, como la Biblioteca del Congreso de Washington, la Biblioteca Nacional de Australia y la Biblioteca Nacional de Nueva Zelanda, que han tratado de integrar los elementos del Dublin Core en sus catálogos.

Dublin Core es el formato de metainformación general más divulgado y más citado, al menos en el ámbito estrictamente bibliotecario. Fue diseñado para promover un estándar de propósito general, sencillo y descriptivo de los recursos electrónicos de cualquier materia.

4. Capa Ontológica.

La Web semántica busca catalogar la información de los recursos web –páginas HTML, documentos PDF, vídeos, archivos de sonido– mediante ontologías (esto es, mediante el significado de las palabras), no mediante palabras clave.

Con las ontologías, los usuarios organizarán la información de manera que los agentes de software podrán interpretar el significado y, por tanto, podrán buscar e integrar datos mucho mejor que ahora. Gracias al conocimiento almacenado en las ontologías, las aplicaciones podrán extraer automáticamente datos de las páginas web, procesarlos y sacar conclusiones de ellos, así como tomar decisiones y negociar con otros agentes o personas. Por ejemplo, un agente inteligente que busque un CD musical que satisfaga las preferencias de un usuario, usará las ontologías musicales para elegir el CD (artista, género, década, precio…) y empleará las ontologías empresariales para encargarlo a alguna tienda musical.

De entro los principales lenguajes de ontologías podemos destacar los siguientes:

SHOE. *Simple HTML Ontology Extensions*. Fue el primer lenguaje de etiquetado para diseñar ontologías en la Web. Este lenguaje nació antes de que se ideara la Web Semántica. Las ontologías y las etiquetas se incrustaban en archivos HTML. Permite definir clases y reglas de inferencia, pero no negaciones o disyunciones.

OIL. *Ontology Inference Layer*. Este lenguaje, derivado en parte de SHOE, fue impulsado también por el proyecto de la Unión Europea On-To-Knoledge. Utiliza ya la sintaxis del lenguaje XML y está definido como una extensión de RDFS. Se basa tanto en la lógica descriptiva (declaración de axiomas) y en los sistemas basados en frames (taxonomías de clases y atributos). OIL posee varias capas de sub-lenguajes, entre ellas destaca la capa base que es RDFS, a la que cada una de las capas subsiguientes añade alguna funcionalidad y mayor complejidad. La principal carencia de este lenguaje es la falta de expresividad para declarar axiomas.

DAML-OIL[20].Este lenguaje nació fruto de la cooperación entre OIL y DARPA y unifica los lenguajes DAML (DARPA's Agent Markup Language) y OIL (Ontology Inference Layer). Se basa ya en estándares del W3C. El lenguaje DAML se desarrolló como una extensión del lenguaje XML y de Resource Description Framework (RDF) y para extender el nivel de expresividad de RDFS. DAML-OIL hereda muchas de las características de OIL, pero se aleja del modelo basado en clases (frames) y potencia la lógica descriptiva. Es más potente que RDFS para expresar ontologías. En la última revisión del lenguaje (DAML+OIL) ofrece ya un rico conjunto de elementos con los cuales se pueden crear ontologías y marcar la información para que sea legible y comprensible por máquina. Este lenguaje presenta algunas carencias debido a su complejidad conceptual y de uso, complejidad que se intentó solventar con el desarrollo de OWL.

OWL es el que veremos en el siguiente apartado necesario para poder concluir con la parte práctica de este trabajo.

4.1. OWL. Web Ontology Language

El OWL[21] o Lenguaje de Ontologías para la Web, se ha convertido en recomendación del W3C el 10 de febrero de 2004. Es un lenguaje de etiquetado semántico para publicar y compartir ontologías en la Web. Puede usarse para representar ontologías de forma explícita, es decir, permite definir el significado de términos en vocabularios y las relaciones entre aquellos términos (ontologías). En realidad, OWL es una extensión del lenguaje RDF y emplea las tripletas de RDF, aunque es un lenguaje con más poder expresivo que éste. Se trata de un lenguaje diseñado para usarse cuando la información contenida en los documentos necesita ser procesada por programas o aplicaciones, en oposición a situaciones donde el contenido solamente necesita ser presentado a los seres humanos. OWL surge como una revisión al lenguaje DAML-OIL y es mucho más potente que éste. Al igual que OIL, OWL se estructura en capas que difieren en la complejidad y puede ser adaptado a las necesidades de cada usuario, al nivel de expresividad que se precise y a los distintos tipos de aplicaciones existentes (motores de búsqueda, agentes, etc.) .

OWL posee más funcionalidades para expresar el significado y semántica que XML, RDF, y RDFS, OWL va más allá que estos lenguajes pues ofrece la posibilidad de representar contenido de la Web interpretable por máquina.

El lenguaje OWL tiene 3 sub-lenguajes que incrementan su expresión: *OWL Lite*, *OWL DL*, y *OWL Full*.

OWL Lite da soporte a aquellos usuarios que primordialmente necesitan una clasificación jerárquica y restricciones simples. Por ejemplo, soporta restricciones cardinales, pero solamente permite valores cardinales de 0 ó 1. Así pues, es más simple

[20] http://www.daml.org/2001/03/daml+oil-index

[21] http://www.w3.org/TR/owl-ref/

proveer herramientas de soporte para OWL Lite. OWL Lite ofrece una rápida ruta de migración para tesaurus y otras taxonomías. Introduce conceptos como la información de cabecera de la ontología (*priorVersion, backwardCompatibleWith)* En resumen, OWL Lite tiene una más baja complejidad formal que OWL DL.

OWL DL da soporte a aquellos usuarios que quieren la máxima expresividad mientras conservan completamente la computacionalidad (todas las conclusiones son garantizadas para ser computables) y resolubilidad (todas las computaciones terminarán en tiempo finito). OWL DL incluye todos los constructos del lenguaje OWL, pero pueden usarse solamente bajo ciertas restricciones (por ejemplo, mientras una clase puede usarse por una subclase de muchas clases, una clase no puede ser una instancia de otra clase). OWL DL se denomina así debido a su correspondencia con las descripciones lógicas *(DL),* un campo de investigación que han estudiado los lógicos para la fundación formal de OWL.

OWL Full da soporte a usuarios que requieren el máximo de expresividad y la libertad sintáctica de RDF sin garantías computacionales. Por ejemplo, en OWL Full una clase puede ser tratada simultáneamente como una colección de individuos y como un individuo por derecho propio. OWL Full permite a una ontología aumentar el significado del vocabulario predefinido (RDF ó OWL). Es poco probable que algún software racional pueda soportar por completo el razonamiento para cada característica de OWL Full.

OWL permite la definición de las siguientes características:

Conceptos. Se definen como clases (utilizando la etiqueta *Class*). Permite también herencia múltiple (etiqueta *subClassOf)*. Estas dos etiquetas se definen en la versión *Lite*. Permite a su vez la definición de particiones disjuntas (etiqueta *disjoinWith,* introducida en OWL DL y Full).

No existe una etiqueta OWL para la definición de particiones exhaustivas, sin embargo, este lenguaje implementa esta característica haciendo uso de las etiquetas *disjointWith* y *UncionOf* conjuntamente.

Las **propiedades** pueden ser de dos tipos *ObjectProperty* o *DatatypeProperty*. Existe también la etiqueta *subPropertyOf* (definida en OWL Lite). Se pueden definir relaciones simétricas y transitivas con las etiquetas *SynmetricProperty (* definidas en la versión Lite).

Las **restricciones** sobre propiedades se especifican con *allValuesFrom* y *someValuesFrom*.

OWL especifica información bastante completa sobre la ontología, incluyendo datos como versiones relacionadas, referencia a versiones anteriores, especificación de compatibilidad con versiones anteriores y términos desaprobados. Para esto se utiliza la etiqueta *imports, priorVersion, backwardCompatibleWith e incompatibleWith.*

Características esquema RDF	Igualdad	Características Property
Class rdfs:SubClassOf rdf:Property rdfs:SubPropertyOf rdfs:domain rdfs:range Individual	equivalentClass equivalentProperty sameAs differentFrom AllDifferent distinctMembers	ObjectProperty DatatypeProperty InverseOf TransitiveProperty SymmetricProperty FunctionalProperty InverseFunctionalProperty
Restricciones Property	**Restricciones cardinalidad**	**Información cabecera**
Restriction onProperty allValuesFrom someValuesFrom	minCardinality maxCardinality cardinality	Ontology Imports
Intersección de clases	**Versionado**	**Anotación de Propiedades**
intersectionClassOf	versionInfo backwardCompatibleWith incompatibleWith DeprecatedClass DeprecatedProperty	rdfs:label rdfs:comment rdfs:seeAlso rdfs:isDefinedBy AnntotationProperty OntologyProperty
Tipo de datos		
Xsd:datatypes		

Tabla 5. Tipos de elementos para construcción de Ontologías de OWL.

Algunos ejemplos de los elementos de OWL los tenemos en el fichero music.rdf que utilizaremos en la parte práctica de este proyecto.

```
<owl:Ontology rdf:about="">
  <owl:versionInfo>v 0.1 2005/04/17 ocelma</owl:versionInfo>
  <rdfs:comment>Simac Music Ontology (Version 0.1)</rdfs:comment>
</owl:Ontology>
<!-- CLASSES -->
<!-- Artist ( artistas ) -->
<owl:Class rdf:ID="Artist">
  <rdfs:subClassOf rdf:resource="http://xmlns.com/wordnet/1.6/Artist"/>
```

```
 </owl:Class>

 <!-- Track ( canciones ) -->
 <owl:Class rdf:ID="Track">
  <rdfs:subClassOf rdf:resource="http://xmlns.com/wordnet/1.6/Song"/>
 </owl:Class>
<!--------------------------->
<!-- Data Properties   -->
<!--------------------------->

<!------------->
<!-- Artist -->
<!------------->
 <owl:DatatypeProperty rdf:ID="name">
  <rdfs:domain rdf:resource="#Artist"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:label>name</rdfs:label>
 </owl:DatatypeProperty>

 <!-- decades -->
 <owl:DatatypeProperty rdf:ID="decades">
  <rdfs:domain rdf:resource="#Artist"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:label>decades</rdfs:label>
 </owl:DatatypeProperty>

 <!-- city -->
 <owl:DatatypeProperty rdf:ID="city">
  <rdfs:domain rdf:resource="#Artist"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:label>city</rdfs:label>
 </owl:DatatypeProperty>
```

```
<!-- nationality -->
<owl:DatatypeProperty rdf:ID="nationality">
  <rdfs:domain rdf:resource="#Artist"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:label>nationality</rdfs:label>
</owl:DatatypeProperty>

<!-- Track -->
 <!-- title ( titulo ) -->
 <owl:DatatypeProperty rdf:ID="title">
  <rdfs:domain rdf:resource="#Track"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:label>title</rdfs:label>
 </owl:DatatypeProperty>

 <!-- plays ( artista )-->
 <owl:ObjectProperty rdf:ID="played_by">
  <rdfs:domain rdf:resource="#Track"/>
  <rdfs:range rdf:resource="#Artist"/>
 </owl:ObjectProperty>
```

Ejemplo 12. Definición de ontología con OWL.

Escribir en lenguajes como RDF y OWL resulta sumamente difícil y propenso a errores. Se pueden utilizar entornos gráficos para visualizar y construir ontologías de forma mucho más razonable, como Kaon[22], WebODE[23] o Protégé[24]. De todas ellas Protégé, desarrollada en la universidad de Stanford, es la herramienta de construcción de ontologías que más usuarios tiene actualmente.

[22] http://kaon.semanticweb.org/
[23] http://kw.dia.fi.upm.es/wpbs/
[24] http://protege.semanticweb.org Herramienta que permite trabajar con RDF y OWL de forma transparente. Protegé es un entorno abierto.

5. Lenguajes de consulta para RDF.

Los documentos RDF se pueden analizar a tres niveles diferentes:

- A nivel sintáctico.
- A nivel semántico.
- A nivel de estructura.

A nivel sintáctico los documentos RDF y RDFS son a su vez documentos XML. Cualquier documento RDF podrá ser consultado usando un lenguaje de consulta para RDF (por ej. Xquery).
Las relaciones en los datos RDF que no son advertidas en la estructura de XML son muy difíciles de consultar. Las consultas están limitadas a expresiones que atraviesen la estructura de árbol de XML, del tipo:

> *Recuperar todos los elementos contenidos en un elemento Description cuyo atributo rdf:about tenga el valor "ABBA".*

A nivel estructural, los documentos RDF y RDFS son un conjunto de tripletas. Se han propuesto e implementado varios lenguajes de consulta RDF que buscan y recuperan tales tripletas de varias formas (ej. Squish). Tienen en cuenta el modelo de datos RDF, que es un grafo. Encontraremos consultas de este tipo en este capítulo.

En este sentido la siguiente parte del proyecto se centra en estudiar los distintos lenguajes de consulta a nivel semántico que han surgido: RQL[25], RDQL[26] (desarrollado por HP[27] en su API Jena2[28]), SeRQL (sucesor de RQL) de Sesame, SPARQL[29] (recomendación del W3C), etc.

5.1. RQL.

RQL permite el uso de variables para denotar los nombres de nodos (es decir, las clases) y los arcos (es decir las propiedades). Una de las principales características de RQL, y que lo distinguen de otros lenguajes de consulta RDF es su capacidad de consultar esquemas RDF y descripciones RDF (es decir, instancias) en una misma consulta.

[25] http://139.91.183.30:9090/RDF/RQL/index.html

[26] http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/

[27] http://www.hp.com/

[28] http://www.hpl.hp.com/semweb/jena.htm

[29] http://www.w3.org/TR/rdf-sparql-query/

RQL permite realizar consultas sobre **esquemas RDF**, tal como Class (recupera todas las clases), property (recupera todas las propiedades) o sobre **instancias** (de clases y propiedades).

El lenguaje RQL está definido por medio de un conjunto de consultas básicas, e iteradores que se usan para construir otras consultas a través de una composición funcional.

RQL usa tres cláusulas para construir una consulta: SELECT-FROM-WHERE:

-**SELECT(obligatoria)** : Define la proyección sobre las variables de interés (en el ejemplo : artista, artista_tipo)

-**FROM (obligatoria)**: Especificamos una parte del subgrafo RDF (un filtro).

-**WHERE** clause (optional): Una expresión formada por la comparación de variables (o del resultado de otra query.

```
select artista,$artista_genero
from cult:artista{artista:artista_genero}
where $artista_genero <= cult:musica
using= http://www.music.com/schema.rdf#
```

Ejemplo 13. Ejemplo sentencia RQL.

RQL provee los operadores lógicos estándares como <,<=,=,>=,!=, *"like" y "in"* como se puede ver en el ejemplo anterior. Todos estos operadores se pueden utilizar en valores literales (ej. Strings, enteros, reales, fechas ..) o recursos (URIs). Permite además los operadores lógicos *"and", "or" y "no"t*
Además de esto RQL provee de un conjunto de funciones como son *min, max, avg, sum* y *count.*

```
SELECT Artistas
FROM {artistas} music:name {ciudad}
WHERE (Artistas) in
    SELECT Artistas
     FROM {artistas} music:ciudad {ciudad}
        WHERE ciudad LIKE "*MADRID*"
USING NAMESPACE
   music=<http://www.semanticaudio.org/ontology/0.1#>
```

Ejemplo 14. Sentencia RQL.

RQL soporta también un conjunto de operadores comunes que se aplican a conjuntos de elementos de un mismo tipo *union, intersect, minus.*

5.2. RDQL.

La sintaxis RDQL (*RDF Data Query Language*) es similar a la de 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 Servicios Web Semánticos.

RDQL usa 5 cláusulas para construir una query: *SELECT-SOURCE-WHERE-AND-USING*. La sintaxis básica de una sentencia SELECT sería:

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

Ejemplo 15. Estructura básica de una sentencia RDQL.

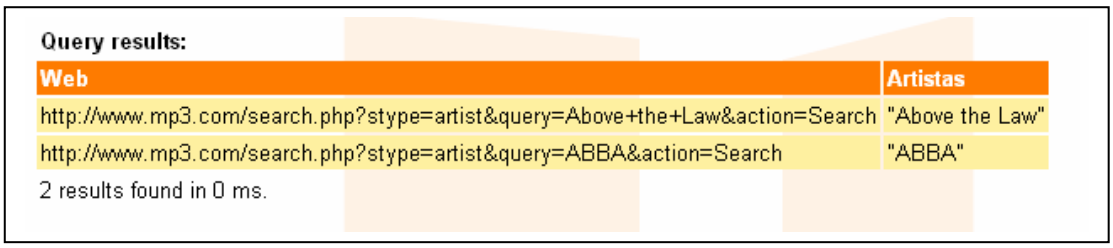
La cláusula WHERE define un patrón de subgrafo en términos de variables y constantes. Cada elemento del patrón con un subgrafo distinto del grafo consultado genera un conjunto de variables mostrables que es incluido en el conjunto resultante. La cláusula AND introduce un filtro en las variables: solamente los resultados que pasan el filtro son incluidos en el conjunto resultante de la consulta.
La cláusula USING permite definir abreviaciones para espacios de nombre NAMESPACES.

Sesame nos permitirá ejecutar consultas RDQL [ver para instalación ANEXOS]. Una consulta que nos muestre los *artistas* (music:name) del fichero *artists.rdf* que estamos utilizando de ejemplo sería:

```
SELECT ?Web, ?Artistas
```

```
FROM <artists.rdf>

WHERE (?Web,<music:name>,?Artistas)

USING music for <http://www.semanticaudio.org/ontology/0.1#>
```

Ejemplo 16. Sentencia RDQL nombre y artistas del fichero RDF artists.rdf



Query results:

Web	Artistas
http://www.mp3.com/search.php?stype=artist&query=Above+the+Law&action=Search	"Above the Law"
http://www.mp3.com/search.php?stype=artist&query=ABBA&action=Search	"ABBA"

2 results found in 0 ms.

Figura 6. Resultado Query RDQL del ejemplo 15.

RDQL permite realizar relacionar las sentencias del grafo RDF en la cláusula WHERE para esto en el siguiente ejemplo se ha relacionado las 3 sentencias siguientes:

El nombre del grupo es ABBA con la sentencia

A*BBA pertenece a la década* * con la sentencia

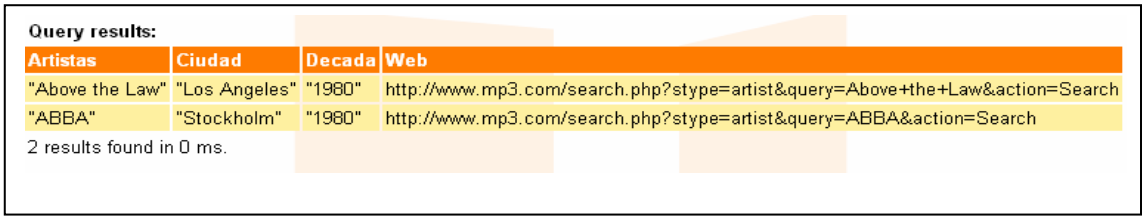
*ABBA es de la ciudad *.*

En este caso se establece un filtro sobre la década buscando solo los artistas que son de una década posterior a 1980.

```
SELECT ?Artistas, ?Ciudad, ?Decada,?Web

FROM <artists2.rdf>

WHERE (?Web,<music:decade>,?Decada),

      (?Web,<music:name>,?Artistas),

      (?Web,<music:city>, ?Ciudad)

AND ?Decada = "1980"

USING music for <http://www.semanticaudio.org/ontology/0.1#>
```

Ejemplo 17. Ejemplo consulta RDQL con unión de sentencias

La salida de esta consulta ejecutada en el entorno de Sesame sería:



Query results:

Artistas	Ciudad	Decada	Web
"Above the Law"	"Los Angeles"	"1980"	http://www.mp3.com/search.php?stype=artist&query=Above+the+Law&action=Search
"ABBA"	"Stockholm"	"1980"	http://www.mp3.com/search.php?stype=artist&query=ABBA&action=Search

2 results found in 0 ms.

Figura 7. Resultado consulta ejemplo 15

5.3. SeRQL.

Sesame RDF Query Language está siendo desarrollado por Aduna pertenece al grupo de Sesame, combina las mejores características de otros lenguajes de consulta RDF como son RQL, RDQL, N-Triples y añade alguna característica propia.

Entre sus ventajas está el que soporta consultas anidadas (RDQL no las soporta).

SeRQL tiene dos tipos de consultas: ***SELECT*** y ***CONSTRUCT.*** Las consultas SELECT devuelven una tabla de resultados, la cláusula CONSTRUCT devuelve un grafo RDF, que es parte del grafo que se examina.

La parte SELECT tiene el mismo significado que el visto en el lenguaje RQL. Una sentencia SELECT puede tener hasta 6 cláusulas:

```
SELECT (DISTINCT) campos
       FROM modelo de grafo
       WHERE
       LIMIT n
       OFFSET m
USING NAMESPACE.
```

Figura 8.Formato sentencia SELECT en SeRQL.

Las cláusulas LIMIT y OFFSET permiten recuperar solo una parte de los resultados, LIMIT indica el nº máximo de líneas que se recuperarán y OFFSET desde que nº de registro de salida se empieza a contar.
La cláusula NAMESPACE se puede utilizar para definir abreviadamente una URI. En todos los ejemplos de este trabajo se ha utilizado music: http://www.semanticaudio.org/ontology/0.1# como nombre corto de la URI indicada.

Las únicas cláusulas obligatorias son la SELECT y FROM. La estructura básica de una consulta SELECT sería:

```
select *
        from    {X} p {Y}
```

Ejemplo 18. Query SeRQL que trae todas las sentencias del ejemplo.

X	p	Y
http://www.mp3.com/search.php?stype=artist&query=Above+the+Law&action=Search	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.semanticaudio.org/ontology/0.1#Artist
http://www.mp3.com/search.php?stype=artist&query=Above+the+Law&action=Search	http://www.semanticaudio.org/ontology/0.1#name	"Above the Law"
http://www.mp3.com/search.php?stype=artist&query=Above+the+Law&action=Search	http://www.semanticaudio.org/ontology/0.1#decade	"1980"
http://www.mp3.com/search.php?stype=artist&query=Above+the+Law&action=Search	http://www.semanticaudio.org/ontology/0.1#decade	"1990"
http://www.mp3.com/search.php?stype=artist&query=Above+the+Law&action=Search	http://www.semanticaudio.org/ontology/0.1#city	"Los Angeles"
http://www.mp3.com/search.php?stype=artist&query=Above+the+Law&action=Search	http://www.semanticaudio.org/ontology/0.1#nationality	"US"
http://www.mp3.com/search.php?stype=artist&query=ABBA&action=Search	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.semanticaudio.org/ontology/0.1#Artist
http://www.mp3.com/search.php?stype=artist&query=ABBA&action=Search	http://www.semanticaudio.org/ontology/0.1#name	"ABBA"
http://www.mp3.com/search.php?stype=artist&query=ABBA&action=Search	http://www.semanticaudio.org/ontology/0.1#decade	"1970"
http://www.mp3.com/search.php?stype=artist&query=ABBA&action=Search	http://www.semanticaudio.org/ontology/0.1#decade	"1980"
http://www.mp3.com/search.php?stype=artist&query=ABBA&action=Search	http://www.semanticaudio.org/ontology/0.1#city	"Stockholm"
http://www.mp3.com/search.php?stype=artist&query=ABBA&action=Search	http://www.semanticaudio.org/ontology/0.1#nationality	"SE"

12 results found in 0 ms.

Tabla 6. Resultado de Query SeRQL realizado con Sesame ver. 1.2.3.

Supongamos que queremos saber el artista que canta la canción " Mamma mia" para esto tenemos que establecer un filtro sobre el campo título:

```
select DISTINCT Web, Titulo, Artista
from    {Web} rdf:type {};
music:title {Titulo};
music:played by {Artista}
where Titulo LIKE "*Mamma *" IGNORE CASE
using namespace
   music = <http://www.semanticaudio.org/ontology/0.1#>
```

Ejemplo 19. Sentencia SeRQL.

El resultado que nos ofrece Sesame 1.2.3. sería:

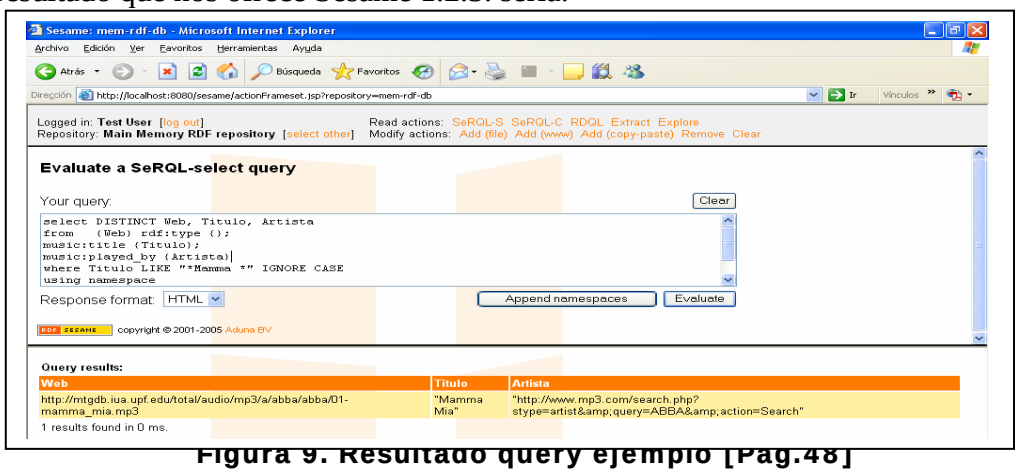


Figura 9. Resultado query ejemplo [Pag.48]

En la siguiente tabla podemos ver los operadores y funciones más utilizados dentro de las sentencias SeRQL.

Constantes
- *Booleanas* TRUE, FALSE

Lógicos.
- AND, OR y NOT
- *de comparación.* =, !=, <, >, <=, >=

Aritméticos.
- *Unarios* +A, -A, EXISTS
- *Binarios (A op B)*

Operador LIKE.
- **IGNORE CASE** permite ignorar mayúsculas/minúsculas en la comparación
- Podemos utilizar el comodin "*****" para relacionar subcadenas

Operadores de cuantificación.
- ANY (operador existencial)
- ALL (operador universal)

Funciones de comprobación booleanas.
- isRESOURCE(), isLITERAL()
- isURI(), isBNode()

Otras funciones.
- **label(), lang(), datatype().** Operan en todos los literales.
- **namespace(), localName().** Solo se pueden utilizar con URIs

Tabla 7. Operadores y funciones de SeRQL.

Otros operadores que permiten combinar resultados o grafos entre sí, son UNION, INTERSECT y MINUS.

Además podemos utilizar el predicado **IN** que nos permite hacer subquerys dentro de querys (similar a las de SQL).

Las sentencias CONSTRUCT se especifican de la siguiente forma:

```
CONSTRUCT (DISTINCT) claúsula
FROM modelo de grafo
LIMIT n
OFFSET m
```

Figura 10.Formato sentencia CONSTRUCT en SeRQL

Otra característica muy importante en la Web Semántica es que permite definir sentencias sobre sentencias es decir la **reificación.**

5.4. RDFQL

Es un lenguaje utilizado por RDF Gateway[30] en aplicaciones y agentes. Es un lenguaje de *scripting* derivado de JavaScript que incluye comandos similares a SQL.
El predicado de una sentencia debe de ser siempre un recurso y en general se refiere a una propiedad en un esquema. El sujeto debe de ser un recurso que define una persona, cosa o sentencia. El objeto debe de ser un recurso si la sentencia se refiere a otro recurso.

Recursos: [rdf:type]
[http://purl.org/rss/1.0/item]

Los recursos RDFQL soportan sintaxis namespaces XML para permitir una notación corta de URIs.
Un literal RDF es expresado como un string de caracteres delimitados por comillas dobles. Si el string de caracteres contiene un carácter de nueva linea, comillas o "/" debe de anteponerse una "\".
Una variable se denota con un símbolo "?". El nombre de la variable debe de empezar con una letra y solo puede contener letras, numeros y guiones "_". Ejemplos de variables: **`?x, ?author, ?doc_url`**

Una sentencia simple:

```
{
[rdf:value] ?s "ABBA"
}
```

Ejemplo 20. Sentencia RDFQL.

Una sentencia con join:

```
{
[rdf:value] ?s ?value} AND ?value LIKE "ABBA"}
```

Ejemplo 21. Sentencia join de RDFQL.

Una sentencia con funciones:

```
{
[rdf:value] ?s ?value} AND ?pos = instr(?value, ":") AND
strcmp(substr(?value, 0, ?pos), "http") = 0
}
```

Ejemplo 22.Sentencia RDFQL con funciones.

En muchas características es muy similar a RDQL.

[30] http://www.intellidimension.com

5.5. SquishQL

Es un lenguaje pensado para ser facil de aprender y de usar. RDQL es una versión posterior refinada en el *toolkit* de Jena.

Un ejemplo sencillo en SquishQL nos permite buscar la ciudad y el nombre de un artista determinado:

```
SELECT ?web, ?artista, ?ciudad
FROM http://anaya.es/libros
WHERE ( ?edicion,<rdf:type>,<music:web>),
           (?web,<rdf:artista>,?artista),
           (?web,<rdf:ciudad>,?ciudad),
USING  music FOR http://www.semanticaudio.org/ontology/0.1#
```

Ejemplo 23. Sentencia con SquishQL.

5.6. Triple.

El lenguaje deriva de F-Logic. Las tripletas del lenguaje no distinguen entre reglas y querys. Un ejemplo sencillo de Query sería:

```
FORALL X<- ( X [rdfs:label-> "disco"] ) @default:ln.
```

Devuelve todos los recursos que tienen una etiqueta " disco ". La salida es una tabla con variables y sus posibles valores.
Triple no codifica la semántica RDF. La semántica tiene que ser codificada como un conjunto de reglas con la Query. Triple no soporta tipos de datos y actualizaciones.

5.7. Versa

Versa está inspirado en XPath. De todos los lenguajes vistos hasta ahora ninguno se parece menos a SQL que Versa.
El lenguaje es muy sencillo y fácil de aprender, el único inconveniente es que no lo puedes utilizar con Jena o con Redland.

5.8. SPARQL

SPARQL[31] es el estándar en el que está trabajando actualmente W3C, un lenguaje que nos permite obtener información de grafos RDF. Se adapta totalmente a las necesidades de RDF. Algunas de sus características nos permiten:

- Extraer información de URIs, literales y nodos vacíos
- Obtener subgrafos RDF y
- Construir nuevos grafos RDF basados que información de grafos que devuelve una query.

Las *querys* más sencillas que podemos construir necesitan de la cláusula SELECT y de la cláusula WHERE.

SELECT. La cláusula SELECT identifica las variables que aparecen en el resultado de la Query.
WHERE. Está formada por tripletas.

Supongamos que necesitamos saber el título de un álbum de música.

```
        SELECT ?title
        WHERE
        {
                <http://example.org/album/album1>
<http://purl.org/dc/elements/1.1/title> ?title
        }
```

Para evitar tener que incluir todo el nombre de la URI cada vez que se necesita SPARQL nos permite definir QNames o prefijos, variables que guarda la URI para identificarla a lo largo de la query. En el siguiente ejemplo puede verse la definición de una variable *music* para traer la **descripción** del grupo ABBA, además se utilizan el operador *ORDER BY* para ordenar en sentido descendente y las cláusulas *LIMIT* para limitar el nº de registros que se obtendrán y *OFFSET* que nos trae los resultados a partir del 5º registro de salida.

```
PREFIX music: < 'http://www.semanticaudio.org/ontology/0.1#'>
SELECT DISTINCT ?Description
WHERE
{
        ?Description music:name "ABBA" .
}
ORDER BY DESC (?Description )
LIMIT 2
OFFSET 5
```

Para este ejemplo la salida ejecutada desde ARQ[32] sería:

[31] http://www.w3.org/TR/rdf-sparql-query/

Ejemplo 24.Ejemplo ejecución sentencia SPARQL.

La query anterior devolvería un grafo con todas las ocurrencias en las que aparece el grupo ABBA en el fichero *artist2.rdf.*

Para querys más complejas podemos utilizar operadores de la siguiente tabla que permiten filtrar los resultados de varias formas.

Lógicos.
- A||B, A&&B, ¡A, (A)
- *de comparación.* =, !=, <, >, <=, >=

Aritméticos.
- *Unarios* +A, -A
- *Binarios (A op B)*

Operadores y funciones RDF.
- *Boleanos.* BOUND(A), isURI(A), isBLANK(A), isLITERAL(A)
- *String.* STR(A), LANG(A), DATATYPE(A)

Operadores para relacionar Strings (paracido al LIKE de SQL).
- REGEX (String expression, pattern expression, [flags expresion])

Tabla 8.Tipos de datos en SPARQL

Soporta además tipos de datos RDF boolean, string, double, float, decimal, integer y dateTime.

Para traer todos los grupos de música de la ciudad de Oslo que tengan una “z” o “Z” en su nombre y que pertenezcan a una década posterior a 1980 podemos hacer la siguiente query:

```
PREFIX music: < http://www.semanticaudio.org/ontology/0.1#'>
SELECT ?name
WHERE
{
     ?city music:city Oslo . FILTER (?decade > 1980)
       FILTER regex(?name, “z”,”i”)
```

[32] ARQ es un módulo de Jena que permite probar sentencias SPARQL contra ficheros RDF locales. En la documentación de Jena tienes también la de ARQ : \doc\ARQ\documentation.html

```
}
```

Ejemplo 25. Ejemplo sentencia SPARQL.

La claúsula "i" indica que no sea sensitivo a mayúsculas (case- insensitive).

SPARQL permite la definición de querys con la directive **UNION:**

```
PREFIX dc10:  <http://purl.org/dc/elements/1.0/>
PREFIX dc11:  <http://purl.org/dc/elements/1.1/>
SELECT ?title
WHERE  { { ?book dc10:title  ?title } UNION { ?book dc11:title
?title } }
```

Ejemplo 26. Ejemplo sentencia SELECT de SPARQL.

La cláusula UNION es un operador binario que permite combinar dos grafos.

Las querys anteriores utilizan la cláusula SELECT, SPARQL permite utilizar la cláusula **ASK** simplemente para devolver YES/NO. También se puede utilizar **DESCRIBE** que devuelve un grafo con la información relacionada con los nodos del grafo o bien todos los metadatos asociados a un recurso (URI)..

```
PREFIX music: <http://www.semanticaudio.org/ontology/0.1#>
DESCRIBE music:music0
```

Ejemplo 27. Ejemplo sentencia DESCRIBE de SPARQL.

Figura 11. Resultado Query DESCRIBE del ejemplo.

Para ordenar resultados de las querys SPARQL ofrece la misma directiva que SQL: ORDER BY.
SPARQL tiene además una característica que es especialmente interesante **CONSTRUCT,** que permite construir un grafo de respuesta en vez de devolvernos la típica tabla de variables y valores de SQL.

La ejecución de la sentencia CONSTRUCT nos devuelve la siguiente salida:

```
CONSTRUCT
  { ?s ?p ?o .
  }
WHERE
  { _:b0 rdf:subject ?s .
    _:b0 rdf:predicate ?p .
    _:b0 rdf:object ?o .
  }
```

Ejemplo 28. Ejemplo sentencia CONSTRUCT de SPARQL

Figura 12. Resultado Query DESCRIBE de SPARQL.

La siguiente sentencia nos permite obtener el primer miembro de un contenedor tipo lista:

```
PREFIX  music:  <http://www.semanticaudio.org/ontology/0.1#>
PREFIX     :  <http://example.org/>
SELECT ?name ?list
{ ?name :p ?list .
  FILTER music:listMember(?list, 1)
}
```

Ejemplo 29. Ejemplo consulta 1º elemento de una lista en SPARQL.

El resultado que devuelve es nulo ya que en el fichero tracks.rdf no se ha definido ningún objeto contenedor de tipo lista.

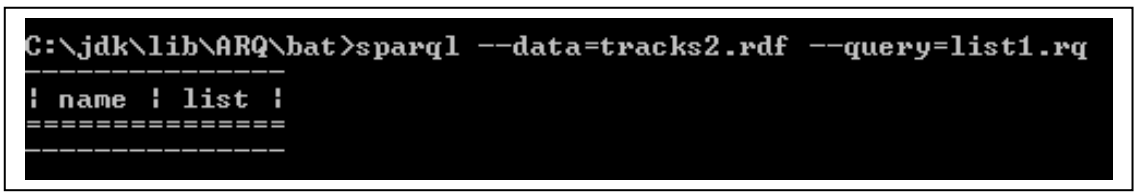


Figura 13. Resultado obtenido de consultar el primer elemento de una lista.

5.9. Comparativa lenguajes RDF

	RQL	RDQL	SeRQL	RDFQL	SquishQL	Triple	Versa	SPARQL
Tipos de datos	Strings,integers ,reals, dates, URI, enumerated, thesauri			Strings, integers, reals, dates and URI types	Strings, integers,reals, dates, URI types	Strings, integers	Strings, numbers, URIS, lists, sets, booelans	String, double, float, decimal, integer, dateTime,IRI,literal, bNode
Path Expression	Si	Si	Si	Si	No	Si	Si	Si
Optional Path	Restringido	No	Si	Si		No	Si	Si
Union	Si	No	Si	Si		Si	Si	Si
Difference	Si	No	Si	Si		No	Restringido	Si
Quantification	Si	No	Si	Si		Restringido	No	Si
Aggregation	Si	No	No	Si	No	No	Si	Si
Recursion	No	No	No	Si		Si	Si	Si
Reification	Restringido	Restringido	Si	Si	No	Restringido	Restringido	Si
Collection and containers	Restringido	Restringido	Restringido	Restringido	Si	Restringido	Restringido	Si
Namespace	Si	Restringido	Si	Si	Si	No	No	Si
Language	No	No	Si	Si		No	No	Si
Lexical Space	Si	Si	Si	Si		Si	Si	Si
Value Space	Si	Restringido	Si	Si		No	No	Si
Entailment	Si	Restringido	Si	Si		Restringido	No	Si

Tabla 9. Una comparativa similar se puede encontrar en: http://www.aifb.uni-karlsruhe.de/WBS/pha/rdf-query/

6.Depósitos de RDF (RDF repositories).

Los depósitos RDF se construyen por encima de los *analizadores lingüísticos* (o parsers). Presentan funciones adicionales para almacenar grandes cantidades de tuplas en bases de datos relacionales, proporcionan un análisis eficiente de RDFs e intercambio de funciones.

Para desarrollar aplicaciones basadas en RDF, OWL o lenguajes similares se precisan librerías para leer y procesar ontologías definidas en estos lenguajes. Basta dar un vistazo a la lista de recursos RDF:
http://www.ilrt.brisol.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 **Jena** que permite leer, recorrer y modificar grafos tanto RDF como OWL desde un programa java. Jena permite además guardar las ontologías tanto en RDF textual como en formato de base de datos, lo que es importante para grafos muy grandes. Otra librería conocida de similares características para RDF y OWL es **Sesame**, desarrollado en el proyecto europeo Ontoknowledge y actualmente distribuido por Administrator. Jena incluye además un motor de consultas RDQL, SPARQL.
Sesame ofrece lo propio en al versión 2 para RQL, SeRQL y SPARQL. Las últimas versiones de jena y Sesame han incorporado también motores de razonamiento para las expresiones lógicas de OWL.

En este capítulo veremos estos dos ejemplos de depósitos RDF: Sesame[33] y Jena2[34].

6.1. Arquitectura SESAME

Sesame[35] es una arquitectura genérica, implementada con software abierto, que permite almacenamiento persistente de datos y esquemas RDF y su posterior consulta en línea. La consulta se puede realizar con los lenguajes: RQL, RDQL y SeRQL. En la última versión de Sesame (Sesame 2.0-alpha-1) ltambién se acepta el lenguaje SPARQL para la consulta de los datos. La documentación que ofrece Sesame en este momento no incluye como debemos instalar correctamente esta versión. En este trabajo se ha intentado instalar Sesame con las indicaciones de la documentación de la versión 1.2.3. sin embargo Sesame versión 2 alpha no llega a funcionar correctamente.

[33] http://www.openrdf.org/index.jsp

[34] http://www.hpl.hp.com/semweb/jena.htm

[35] http://www.openrdf.org/

En la última versión de Sesame 2.0-ALPHA-1 se han añadido algunas características:

- Revisión del API del repositorio.
- Soporte adecuado para transacciones y recuperaciones.
- Incluye además soporte para el lenguaje SPARQL.

En la página oficial de Sesame hay una demo[36] en la que se pueden ejecutar sentencias SPARQL y SeRQL.

Almacenamiento.
Para el almacenamiento persistente de los datos Sesame está diseñado para trabajar con un repositorio escalable. Además para que sea independiente de los DBMS usados (Oracle[37], PostgreSQL y MySQL) en la administración del repositorio, todo el código que hace referencia a un DBMS específico, fue concentrado en una única capa de la arquitectura Sesame: la capa SAIL (Storage and Interface Layers).

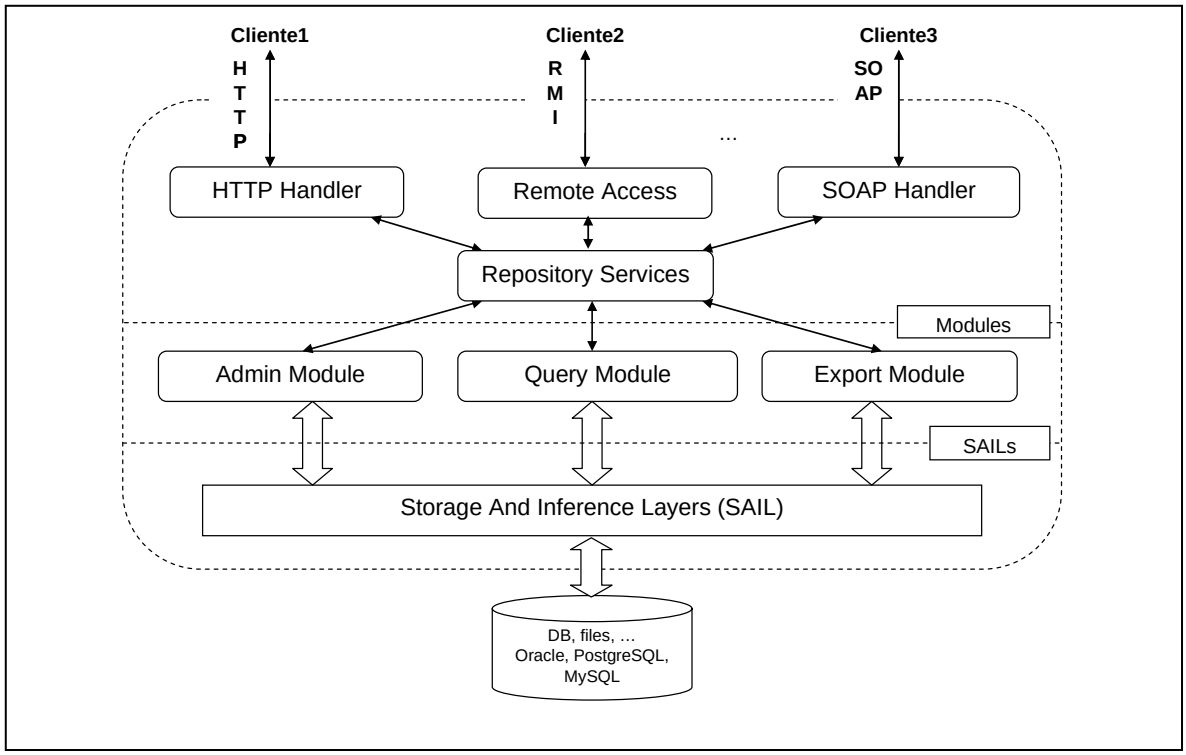


Figura 14. Arquitectura diseño Sesame

La capa SAIL es una API que ofrece métodos específicos RDF a sus clientes y los traduce en invocaciones a distintos métodos de los DBMS apropiados. Cada DBMS

36
http://demo.openrdf.org/webclient/select.view?ql=SPARQL&repository=http://demo.openrdf.org/sesame2/repositories/mem-rdf&descr=Sesame%202%20Demonstration%20Store

37 http://www.oracle.com/

tiene su propio dialecto SQL, por ejemplo el tipo carácter en oracle se define como nvarchar(2) y en Mysql como varchar(n). Esto puede significar que para cada DBMS tengamos que cambiar las sentencias SQL (o las querys realizadas con cualquiera de los lenguajes RDF actuales).Una ventaja importante de contar con dicha capa separada(SAIL), es la posibilidad de implementar Sesame en una gran variedad de repositorios en distintos DBMS.

Los módulos implementados actualmente en la arquitectura Sesame son:

- **Módulo de consultas RQL (querys).** Este módulo permite evaluar consultas RQL sobre el repositorio persistente.
- **Modulo de administración RDF.** Este módulo permite la actualización incremental de los datos RDF e información de esquemas, como también su eliminación.
- **El módulo de exportación RDF**. Permite la exportación de esquemas completos y/o datos desde repositorios con formato RDF serializados XML.

Sesame puede ser utilizado de diferentes formas según las necesidades de la situación. En el desarrollo de una web se puede acceder a Sesame a través de http, en otro tipo de aplicaciones puede ser necesario utilizar RMI[38] o SOAP[39]. Es por lo que los manejadores de protocolos están fuera del núcleo de Sesame, estos conforman una capa intermedia entre los módulos principales de Sesame y sus clientes, cada uno maneja un protocolo específico, traduciendo las solicitudes del cliente a invocaciones a método de la API de Sesame adecuadas.

La búsqueda y consulta de instancias y esquemas de la ontología son implementadas por el sistema Sesame, soportando para tal fin el lenguaje SeRQL, una versión ligeramente ampliada de RQL, que conserva sus principales características.

6.1.1. Modelo físico de datos de Sesame.

El modelo físico de datos de Sesame tiene claves foráneas que garantizan una confiabilidad de la información que se almacena. Es un modelo más seguro que el que nos ofrece Jena tal y como veremos en la siguiente sección.
La estructura física se puede ver en la siguiente figura. Decir que el modelo que se presenta es el que se tendría en caso de trabajar con MySQL.
Cada recurso o literal se almacena como un valor entero (campo ID) que permite realizar búsquedas de datos más rápidas.
Este tipo de configuración no cambia aunque cambie el esquema RDF lo cual permite que MySQL resuelva las consultas en mucho menos tiempo.

[38] http://java.sun.com/products/jdk/rmi/index.jsp

[39] http://www.w3.org/TR/soap/

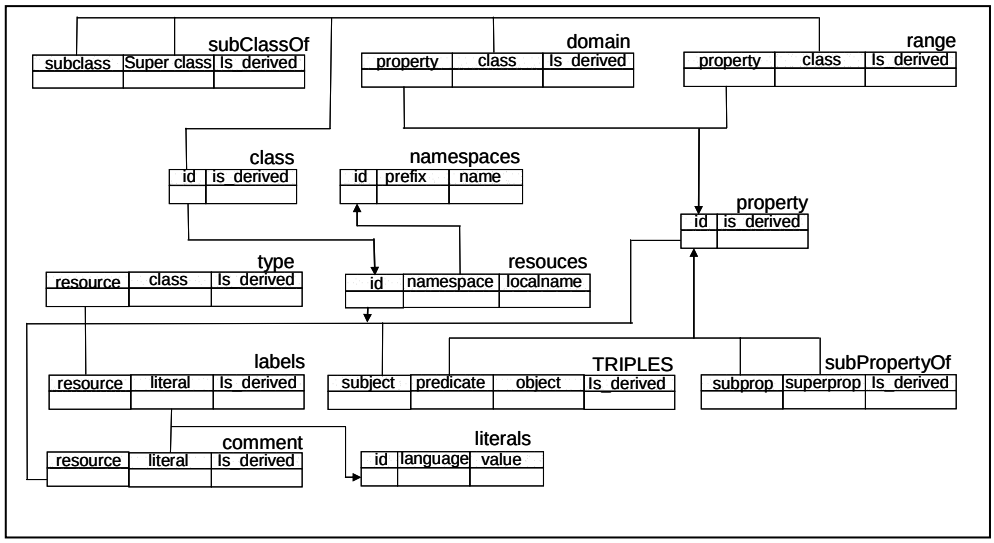


Figura 15. Almacenamiento de estructuras RDF en Sesame (Mysql).

Este modelo se explica con más detalle en la Pág. 67 en la que Sesame ha creado automáticamente estas tablas para poblarlas con datos RDF.

6.2. Arquitectura JENA 2

Jena es un framework desarrollado por los laboratorios Hewlett Parckard[40]. Su fin es manipular metadatos desde aplicaciones escritas en java. En su versión 2, JENA incluye un analizador sintáctico para RDF, una API para RDF, una API de ontologías que permite trabajar con OWL, DAML y RDFS, un subsistema de razonamiento y un sistema de persistencia. Por si fuera poco, permite trabajar con los lenguajes de consultas RDF (RDQL, SPARQL ..).

Los enunciados del modelo se almacenan en un grafo base. El razonador o motor de inferencia puede utilizar el contenido de dicho grafo junto con las reglas semánticas del lenguaje para mostrar un conjunto completo de enunciados. Esto implica incluir también aquellos enunciados que se pueden inferir a partir del grafo base. La interfaz Graph puede ser utilizada para acceder a los datos del modelo.

[40] http://www.hp.com/

Joseki[41] es un servidor para publicar modelos RDF en la Web. Los modelos tiene URLs y pueden ser accedidos por querys usando HTTP GET. Joseki es parte del toolkit de Jena.

La arquitectura de Jena se puede ver en el siguiente gráfico:

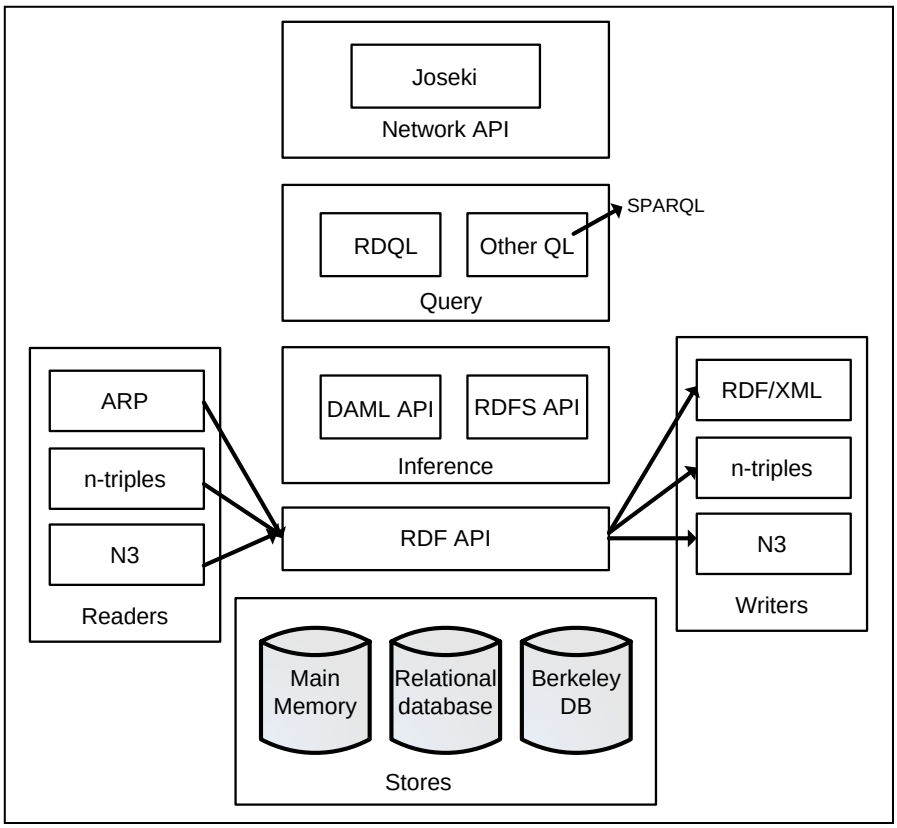


Figura 16. Arquitectura Jena

El núcleo principal es la API RDF, la cual soporta la creación, manipulación y consulta de grafos RDF. La API soporta tecnologías de almacenamiento. Plug-in interfaces permite lectura y escritura automática para diferentes lenguajes que los desarrolladores puedan usar para representar gráficos RDF.

RDF API.
Desde una perspectiva de API, hay dos formas diferentes de ver un grafo RDF. En una vista centrada en afirmaciones, un grafo RDF con un conjunto de tripletas; donde cada tripleta 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 se mezclan.

[41] http://www.joseki.org/

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 Pretty Writer, de forma distinta, toma una mayor ventaja de la sintaxis disponible para producir una salida compacta.

Joseki

Joseki[42] soporta las siguientes operaciones:

- **GET** (querys simples)
- **Query de varios lenguajes** (RDQL, SPARQL, ..)
- **Operaciones de actualización**
    a. Añadir
    b. Eliminar.
- **Ping**

En este caso no es buena idea utilizar el método GET para hacer peticiones en la base de datos ya que el resultado podría traer resultados demasiados grandes.

Almacenamiento.
JENA permite 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 en Berkeley DB un software open-source de Sleepycat[43]. 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 las tablas permite la especialización de consultas a una base de datos específica. La estructura de una base de datos es también configurable, una característica que permite realizar ajustes de desempeño para aplicaciones específicas. El almacenamiento en Berkeley DB, al igual que el almacenamiento relacional, es persistente.

[42] **http://www.joseki.org/protocol.html**

[43] http://www.sleepycat.com/

6.2.1. Modelo físico de datos en Jena.

El modelo físico de datos utilizado por Jena trabaja básicamente con el concepto de tripleta para almacenar los datos. De este modo, el sistema utiliza prefijos internos para identificar cada uno de los componentes de los modelos soportados por Jena, no tiene una estructura física que permita el almacenamiento normalizado de cada información. A través de estos prefijos se pueden hacer encadenamientos lógicos de los datos. Es importante saber que la estructura está modelada de esta forma para obtener un mejor alcance en los diferentes formatos de ontologías soportados por Jena, además del soporte al modelo RDF. Esta estructura proporciona flexibilidad para la inclusión de nuevas tecnologías al proyecto pero también dificulta la persistencia debido a la necesidad de localizar cada prefijo para cada inclusión o alteración de datos en el modelo físico.

Estructura de las tablas de sentencias (statements).
Jena 2 utiliza dos tipos de tablas para guardar los grafos: una tabla para guardar las **sentencias** *statements* y otra para guardar las ***reificaciones*** *reifications.* El beneficio que aporta separar estas dos tablas es que las reificaciones se guardan de forma optimizada. Pensemos que la *reificación* en RDF corresponden con 4 *statements* (sentencias).Almacenar una reificación en una tabla normal de tripletas puede requerir 4 filas mientras que utilizando una tabla aparte solo se necesita una fila. Para aplicaciones que utilicen gran número de reificaciones esto es una ventaja importante.

Estructura de tablas del sistema.
Las tablas del sistema de Jena 2 almacenan metadatos (datos acerca de los datos) como valores long para los literales y recursos en las tablas de sentencias (statements). Los campos *Varchar(n)* denotan strings de n caracteres de longitud máxima. Los campos *blob* denotan string de caracteres cuya longitud depende del motor de la base de datos que se esté utilizando. Normalmente el motor de base de datos que se utilice no permitirá utilizar un campo *blob* como campo clave.
En la siguiente figura podemos ver el esquema de la base de datos que utiliza Jena 2 para guardar URIs:

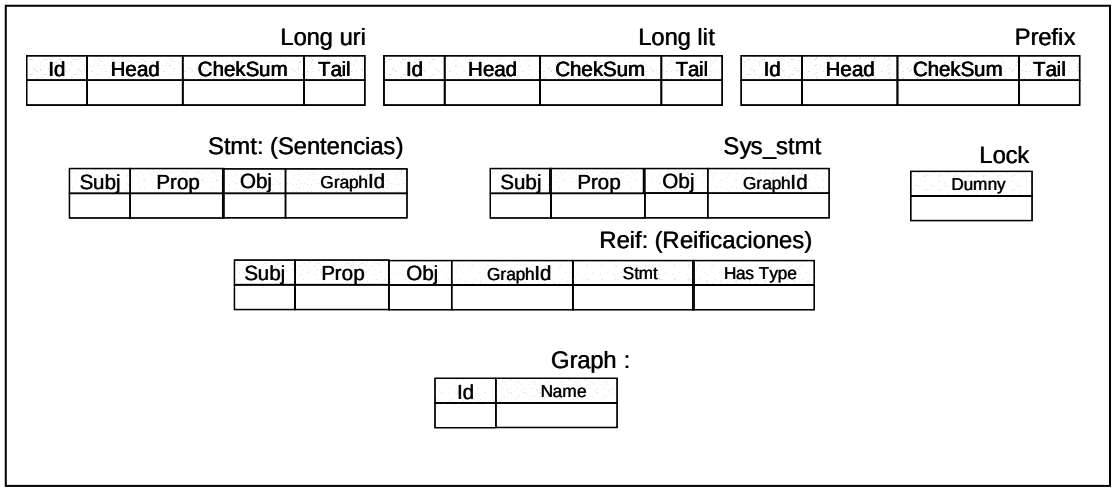


Figura 17. Estructura de datos de Jena

El modelo físico de datos se genera automáticamente a partir de la ejecución de Jena. El proceso de generación automática del modelo no genera ninguna clave foránea (de ahí que no exista ninguna relación entre las tablas del modelo).

6.3. SESAME versus JENA 2.

Los criterios que nos permiten comparar los dos repositorios vistos son:

Escalabilidad. Se necesita que esos repositorios sean escalables.

Query Language. Necesitamos saber que tipo de lenguajes de querys soportan cada uno de los repositorios analizados.

Eficiencia. Consumo de tiempo de ejecución de una Query.

Optimización. Se permite manipular la reificación.

Organización. Esquema de las tablas que guardan la información del esquema RDF/RDFS.

En la siguiente tabla se muestra las principales diferencias encontradas:

Características	**SESAME**	**JENA**
Query Language	RQL, RDQL, SPARQL	RDQL, SPARQL
Implementación	Java	Java
Almacenamiento BD	Mejor MySQL que PostgreSQL y oracle pero permite los tres	MySql, oracle, PostgreSQL
soporta actualización (ontología y datos)	Si	Si
Soporte API (querying+updating)	HTTP/SOAP	Java
Escalabilidad	Si	Si
Formato de exportacion de datos	RDF	Tripletas en ASCII
Soportan Inferencia	Si	Si
Operaciones	Union, Interseccion, Diferencia	Union, Interseccion, Diferencia

Tabla 10. Sesame vs Jena 2.

En el proyecto " SIMILE[44] " se estudian algunas de estas propiedades. En dicho estudio, se concluye que cuando ejecutamos Jena con MySQL los resultados se visualizan antes y en caso de trabajar con sistemas remotos Sesame es más adecuado.

Los modelos de datos de estos dos repositorios son bastante diferentes. Jena no es una implementación específica para manipular RDF, necesita de una gran flexibilidad de datos para persistir en diversas tecnologías (Oracle, Postgresql y MySQL), por este motivo su estructura de datos es sencilla y está basada en el concepto de tripleta.
Sesame es una implementación más específica para manipular grafos RDF, posee una estructura de datos más elaborada, cumpliendo claramente con las características de RDF. Utiliza índices del propio banco de datos que garantizan un almacenamiento consistente.
Vistas estas características sería lógico trabajar con Sesame, sin embargo como la estructura del ejemplo que se pretende construir es bastante sencilla se podría utilizar satisfactoriamente cualquiera de los dos.

7.Persistencia de datos RDF en Sesame.

La persistencia de datos nos va a permitir utilizar los ficheros RDF que tenemos para crear una base de datos en la que se van a guardar las tripletas del modelo RDF. Sesame 1.2.3. puede trabajar con PostgreSQL, Mysql y con oracle.
Para este estudio se han seleccionado un gestor de bases de datos abierto Mysql.

7.1. MySQL version 5.

MySQL es un servidor de bases de datos **rápido**, estable, multi-thread, multi-usuario, robusto y gratuito.

MySQL es un sistema de administración de bases de datos utilizado especialmente en entornos web. MySQL utiliza el lenguaje SQL, un estándard que utilizan también otros muchos sistemas de este tipo.

MySQL soporta múltiples lenguajes, es posible conectarse a una base de datos de este tipo a través de cualquiera de ellos: c, c++, PHP, Java, Perl, Python, TCL, Eiffel.
La versión 5 ha añadido nuevas características que le permiten competir en un mercado cada vez más amplio. Algunas de estas características son:

- Consultas anidadas o Sub-Select
- Definición de Vistas, triggers, procedimientos almacenados y cursores.

[44] http://simile.mit.edu/reports/stores/

- Creación de funciones definidas por el usuario UDF y no solo en un lenguaje particular, es posible crear funciones actualmente en C, SQL, Pl-PgSQL (Lenguaje nativo PostgreSQL), Perl, PHP, Java entre otros.
- Crear uniones de tablas a través de JOINs (INNER, OUTER, FULL, LEFT, RIGHT)
- Permitir la integridad referencial entre claves primarias / foráneas con CASCADE o RESTRICT según se desee programar
- Triggers o disparadores, eventos programables al momento de insertar, modificar o actualizar datos.
- Establecer condiciones o CHECKs para validar las entradas de datos
- Permitir transacciones, es decir, múltiples operaciones de tabla o registros de manera segura.
- Bloqueos de tablas y/o de registros, útil en entornos hoy por hoy multiusuario.
- Administración de Grupos de usuarios, y soporte nativo SSL.

Le queda pendiente añadir características Object relacionales para una futura versión.

La instalación de MySQL puedes verla en los anexos en la Pág.82.
Inicialmente Mysql tiene 3 bases de datos, nosotros crearemos una base de datos "testDB" para que Sesame pueda crear automáticamente la estructura de tablas que dará soporte al modelo RDF que utilicemos. Creamos un usuario "testuser" con contraseña "openSesame" con todos los permisos posibles para acceder a mysql (en concreto a la base de datos "testDB"):

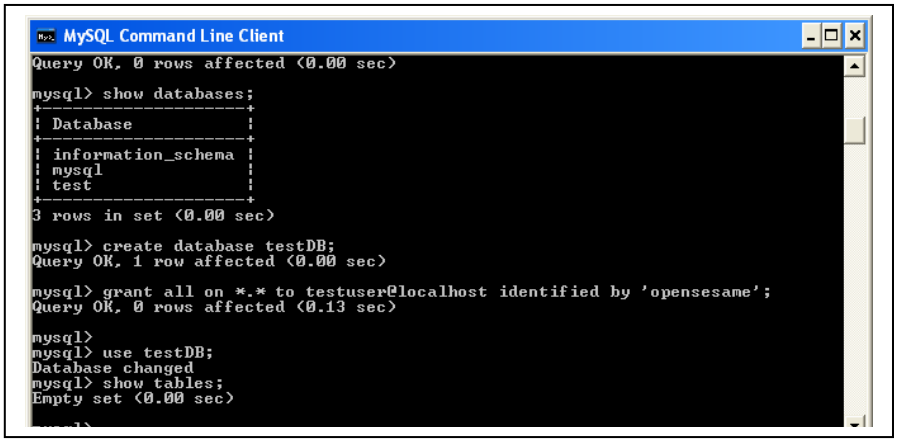


Figura 18. Estructura inicial de tablas de Mysql.

Una vez instalado Tomcat [ANEXO III], Sesame[ANEXO II] (en el directorio "..\Tomcat5.5\webApps\sesame) y el driver de conexión/J JDBC[45]abrimos desde el navegador la página http://localhost:8080/sesame con esto accedemos a la pantalla que nos permitirá ver como trabaja los distintos repositorios que permite Sesame y su funcionamiento:

[45] La instalación del driver JDBC de MySQL consiste en copiar el fichero mysql-connector-java-3.0.17-ga-bin.jar en el directorio "..\Tomcat5.5\webApps\sesame\lib". La última versión del driver la puedes bajar de http://dev.mysql.com/downloads/

El primer paso es hacer login con el usuario *testuser* contraseña *opensesame*[46], después tenemos que seleccionar el tipo de repositorio que queremos utilizar, en este caso vamos a utilizar el repositorio de MySQL RDFS TestDB[47].

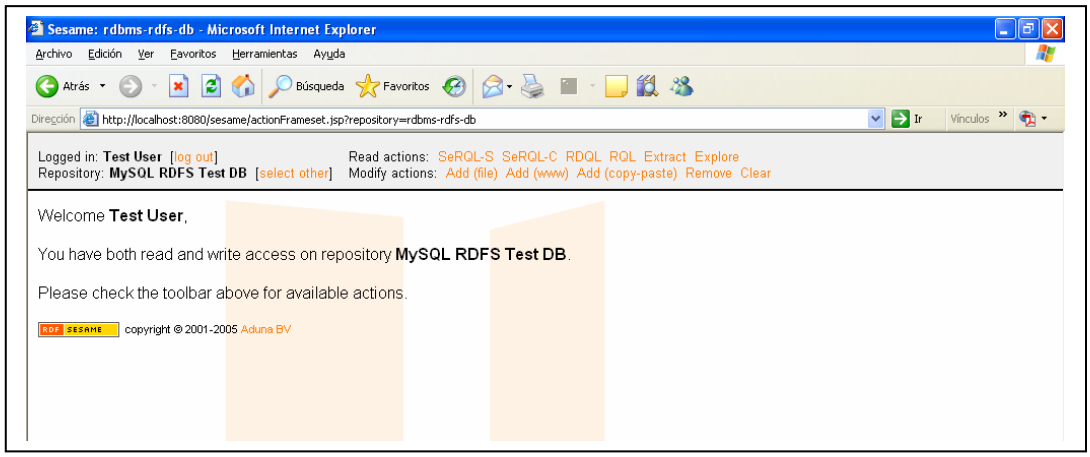


Figura 19. Persistencia con Sesame y MySQL.

Para poder ver la estructura de tablas tenemos que insertar un fichero RDF en el repositorio de MySQL, en este caso insertaremos toda la estructura en la base de datos *testDB*. En la barra de tareas superior de Sesame nos aparece la opción *Add File* que nos permite insertar el fichero RDF en el repositorio. En este caso se crea la estructura de tablas necesaria para guardar el modelo RDF. Entre otras se crea la tabla *resources* y literal con la siguiente estructura:

```
mysql> describe resources;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| id        | int(11)      | NO   | PRI |         |       |
| namespace | int(11)      | NO   | MUL |         |       |
| localname | varchar(255) | NO   |     |         |       |
+-----------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)


mysql> describe literals;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| id        | int(11)      | NO   | PRI |         |       |
| datatype  | int(11)      | NO   |     |         |       |
| labelHash | bigint(20)   | NO   | MUL |         |       |
| language  | varchar(16)  | YES  |     | NULL    |       |
| label     | text         | NO   |     |         |       |
+-----------+--------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

Figura 20. Tabla resources y literals de MySQL para almacenar RDF.

[46] Testuser es el usuario que se ha definido en MySQL, en el fichero System.conf tenemos los tres posibles usuarios que podemos utilizar, siempre y cuando otorgemos permisos a la base de datos creada en MySQL.
[47] testDB base de datos que creamos en MySQL para el usuario testuser (opensesame).

La tabla *literal* tiene los datos de todos los literales del fichero *rdf* que hemos utilizado para la prueba:

```
mysql> select *
    -> from literals;
+----+----------+---------------------+----------+----------------+
| id | datatype | labelHash           | language | label          |
+----+----------+---------------------+----------+----------------+
| -9 |        0 |  3377611726928704514 | NULL     | SE             |
| -8 |        0 |  4392663276230263460 | NULL     | Stockholm      |
| -7 |        0 | -1069917582795912141 | NULL     | 1970           |
| -6 |        0 |  4012375488195635440 | NULL     | ABBA           |
| -5 |        0 | -8474184111989701166 | NULL     | US             |
| -4 |        0 |  -916903428598998179 | NULL     | Los Angeles    |
| -3 |        0 |  5154088155679225422 | NULL     | 1990           |
| -2 |        0 |  3061240736834165418 | NULL     | 1980           |
| -1 |        0 |  7498525559910018183 | NULL     | Above the Law  |
+----+----------+---------------------+----------+----------------+
9 rows in set (0.00 sec)
```

Figura 21.Registros almacenados tras la carga de datos en testDB.

7.2. oracle 10g.

La versión 10g de oracle posibilita todas las capacidades del trabajo clásico en bases de datos relacionales pero además es una base de datos diseñada específicamente para soportar los requerimientos demandados por la Web y las nuevas aplicaciones basadas en Data Warehouse y Data Mining que permiten el tratamiento de información del Web en los negocios. Esta versión también incorpora las herramientas de funcionalidad y los requerimientos de escalabilidad necesarios para el trabajo con aplicaciones críticas.

La versión 10g cuenta entre sus principales características con la posibilidad de auto administrarse, organizarse en clusters y manejar la administración de su almacenamiento a través de la tecnología Grid Computing[48].

A través de muchos años oracle ha perfeccionado su software de base de datos hasta llegar a las versiones mas recientes que aparte de estar diseñadas para ambientes Web incluyen características únicas no proporcionadas por ningún otro proveedor, que le permiten a las empresas implementar soluciones informáticas mas seguras y con la mejor relación coste beneficio.

El amplio espectro de plataformas tecnológicas sobre las que se puede montar la base de datos oracle la convierte en la mas versátil y principalmente la mas escalable, ya que puede pasar de equipos INTEL con sistema operativo Linux o Windows a Servidores RISC multiprocesador con diversas versiones de Unix o VMS y aun mas su operación se amplia a clusters de varios servidores visibles como un solo equipo, esto le permite a las empresas desarrollar soluciones escalables sin preocupaciones de compatibilidad o inversión excesiva. Esta característica es una de las principales que se han tenido en cuenta para elegirlo en el desarrollo del caso práctico como DBMS de ficheros RDF.

La tecnología de objetos que proporciona es una capa de abstracción construida sobre su tecnología relacional, por lo que los datos se siguen almacenando en columnas y tablas.

[48] http://---

El hecho de que oracle sea object relacional hace que soporte estructuras de datos más complejas.

7.2.1. Propuesta de almacenamiento de datos en oracle 10g.

La propuesta de almacenamiento de modelos RDF en una base de datos oracle se realiza como una capa lógica (usando el modelo de datos Espacial de oracle). Oracle 10g soporta grafos directos o indirectos como parte del módulo Espacial NDM de oracle. Las tripletas RDF se almacenan como nodos los sujetos y objetos y se guardan como enlaces las propiedades. Cada tripleta crea un único objeto {sujeto, propiedad, objeto} en la base de datos.
Una tabla del sistema se crea para guardar información de todos los modelos definidos en la base de datos. Cuando se crea un nuevo modelo se añade una entrada en esta tabla. El usuario tiene que indicar el nombre del modelo y oracle le asigna un ID automáticamente. Si usamos el ID en vez de su nombre reduciremos espacio.

Para insertar namespaces en la base de datos, se genera un ID automáticamente para él, El ID de este namespaces se utiliza como clave foránea para referirse a ese namespace.

Las sentencias RDF se guardan en la **tabla RDF_VALUE$** cada sentencia se guarda con formato de tripleta: sujeto, propiedad y objeto. Esa tabla tiene los campos siguientes:
VALUE_ID (number), VALUE_NAME (UNITYPE), VALUE_TYPE (VARCHAR2(10), LITERAL_TYPE (XMLTYPE). Para cada sentencia se guardarán tres registros, uno para cada parte de la tripleta. El campo VALUE_TYPE está tipificado con los valores siguientes:
UR: indicará que el campo guardado es una URI.
PL: indicará que se trata de un literal plano.
TL: indicará que se trata de un literal de tipo.
TC_BAG indica que se trata de una colección rdf:Bag.
TC_ALT indica que se trata de una colección de tipo rdf:Alt
TC_SEQ indica que se trata de una colección de tipo rdf:Seq
BN: Indica que se trata de un nodo vacío.

Tabla RDF_BLANK_NODES. En esta tabla se almacenan las tripletas que tienen los objetos o sujetos desconocidos. Un nodo vacío se utiliza para representar nodos desconocidos. La tabla tiene los siguientes campos: NODE_ID (NUMBER), NODE_VALUE (VARCHAR2), ORIG_NAME (VARCHAR2) y MODEL_ID (NUMBER).

Tabla RDF_LINK$. Las propiedades RDF se mapean como enlaces (links) Antes de insertarse una tripleta en la tabla se comprobará que no exista, en caso de no existir se añaden tres nuevas filas (una para cada elemento de la sentencia “statement “). Las columnas de esa tabla son: LINK_ID (NUMBER), VALUE_ID (NUMBER), START_NODE_ID (NUMBER), END_NODE_ID (NUMBER), LINK_TYPE (

VARCHAR2(10)), ACTIVE (VARCHAR2(1)), LINK_LEVEL (NUMBER), PARENT_LINK_ID(NUMBER), MODEL_ID(NUMBER).
El campo VALUE_ID es el mismo que el de la tabla RDF_VALUE$.

Reificación RDF. En el modelo de datos RDF de oracle, se utiliza un enlace jerárquico para representar que una sentencia se realiza sobre otra (reificación). El campo LINK_TYPE de la tabla RDF_LINK$ puede tener dos valores:
SS: que representa una sentencia simple
RS: que representa una sentencia que es una reificación.
En el caso de que tengamos el valor RS el enlace representa el hecho de que la sentencia actual hace una reificación sobre otra sentencia de la base de datos (no es la sentencia reificada).

Contenedores RDF. En el modelo oracle un contenedor se manipula igual que otra tripleta, con un paso más: se crea un primer nodo con VALUE_NAME : blankNodeNode_id y un VALUE_TYPE = TC_CollectionType, Triples { _:blankNodeNode_id, rdf:blankNodeNode_id_coll#, collection_value } Estas tripletas se almacenan para cada miembro de la colección. Como con otras tripletas, el valor puede ser rechazado si ya existe en la tabla. El enlace property para una colección de miembros tiene su PARENT_LINK_ID.

En el siguiente gráfico podemos ver la estructura oracle 10g que se ha mencionado en este apartado:

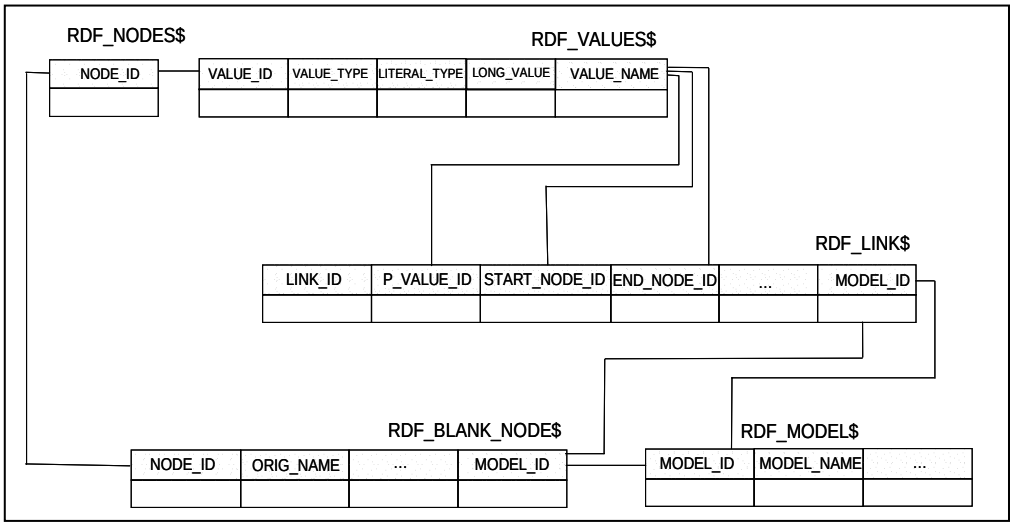


Figura 22. Estructura de datos RDF en oracle 10g.

8. Desarrollo práctico. Buscador Web Music

La parte práctica de este proyecto intentará unir los conceptos que se han visto a lo largo de la memoria para desarrollar un pequeño *buscador web* sobre elementos RDF.

La ontología que se utiliza pertenece al mundo de la música, concretamente se permitirán consultas a entidades *artistas* y *canciones*[49] . De esta forma podemos obtener la relación de los artistas de nacionalidad americana o el nombre de las canciones de un artista determinado.

Trataremos de buscar la máxima simplicidad posible que sea compatible con los requisitos exigidos. El patrón de diseño utilizado que hemos considerado es el google.

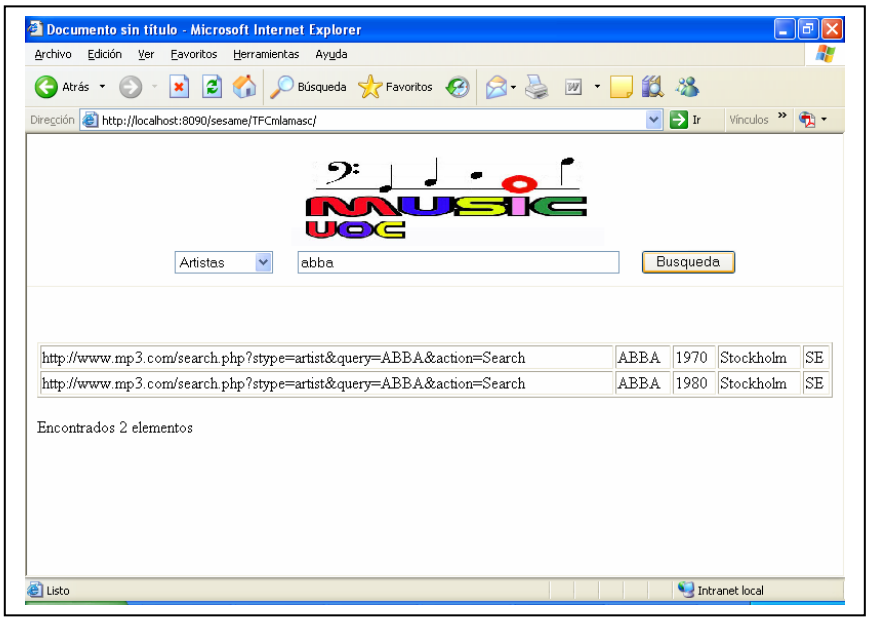


Figura 23. Diseño buscador Web Music

8.1. Consideraciones básicas de diseño.

Previamente a hacer el diseño detallado del sistema, conviene tomar una serie de decisiones de alto nivel respecto a la arquitectura del sistema, tecnologías que emplearemos según las necesidades que hay para obtener el resultado previsto.
En la fase de análisis del capítulo [Pág. 65] anterior se probó el repositorio Sesame[50] que tenemos en la página de sus desarrolladores. En ese estudio comprobamos que la

[49] *Algunos ejemplos se han ejecutado con los ficheros RDF que están en* anexo III en Pág.12]
[50] http://www.openrdf.com/download.jsp

version *Sesame 2.0-alpha-1*no se puede instalar[51] correctamente con la documentación que tenemos en su página y en el entorno Windows XP del que disponemos. Esto hace que tengamos que considerar una versión anterior *Sesame 1.2.3.* que descarta la utilización de SPARQL como lenguaje de consultas RDF. Se hace necesario recurrir al lenguaje SeRQL para poder realizar las búsquedas de información dentro de los ficheros RDF que se utilizarán.

Necesitaremos instalar Tomcat[52] [ANEXOS en Pág.82] que es la implementación de referencia para las Java Server Pages (JSP) y las especificaciones Java Servlet. Es el servidor Java disponible que más se ajusta a los estándares. Esto hace necesario la instalación de java.

La plataforma final necesitará de la instalación de: Java, Tomcat y Sesame. Las indicaciones para realizar estas indicaciones se encuentran en los anexos del final de esta memoria.

8.2. Integración de los datos.

La siguiente consideración de diseño nos obliga a decidir que configuración de Sesame emplearemos, como ya se ha citado Sesame puede configurarse para funcionar como repositorio que almacena sus datos en memoria o bien puede ser empleado con un gestor de base de datos como MySQL.
Por simplicidad se diseñará la aplicación con un repositorio de datos almacenado en memoria. Para esto utilizaremos el API de Sesame que está debidamente documentado en la versión 1.2.3.

En la siguiente figura se muestra el diseño del fichero RDF que vamos a utilizar para el ejemplo práctico. Este diseño se ha utilizado en ejemplos anteriores para mostrar las características de RDF, los lenguajes de consulta RDF (SeRQL, SPARQL) etc.

Partiremos entonces de los datos guardados en memoria. Esto significa que los datos se guardarán en un repositorio cada vez que ejecutemos la aplicación. Los dos ficheros de datos que se utilizarán son: *artist2.rdf* y *tracks2.rdf*[53] ambos se guardarán en un único repositorio.
El tiempo de carga de datos dependerá de los registros que tenga los ficheros *.rdf*. En este caso aunque la carga de datos toma su tiempo las consultas se resuelven rápidamente.

[51] En la documentación de usuario de la versión 2 alpha no hay ninguna indicación de la instalación de Sesame, es por lo que se intentando instalar con la documentación que hay sobre la instalación en la version 1.2.3.
[52] http://apache.osuosl.org/tomcat/

[53] Estos ficheros son los que nos han facilitado para el desarrollo de esta parte pràctica. Los ficheros artists.rdf y tracks.rdf que se encuentran en los anexos son una versión muy reducida de estos.

En la siguiente figura podemos ver una parte de los dos esquemas que se utilizarán para realizar las búsquedas a través de consultas SeRQL.

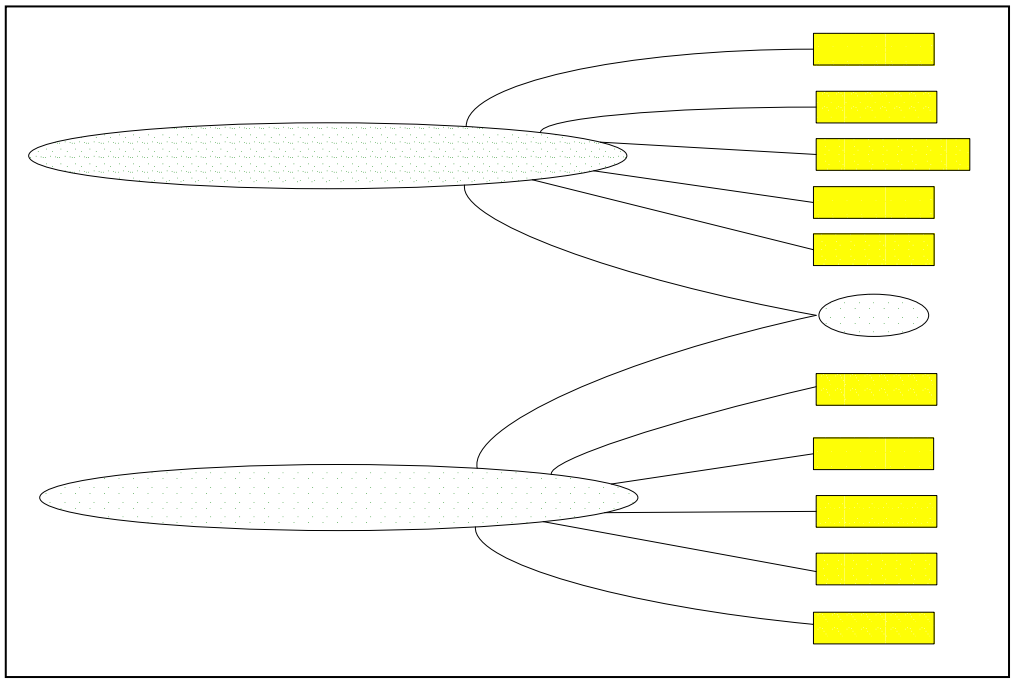


Figura 24. Grafo RDF de una muestra del fichero artists.rdf.

La figura representa una selección de **12 sentencias** del fichero RDF. Se han seleccionado los registros relacionados al grupo "*ABBA*" y al grupo "*Above de Law* ".
En la siguiente figura tenemos una muestra **9 sentencias** de 3 canciones de la discografía de " *ABBA*":

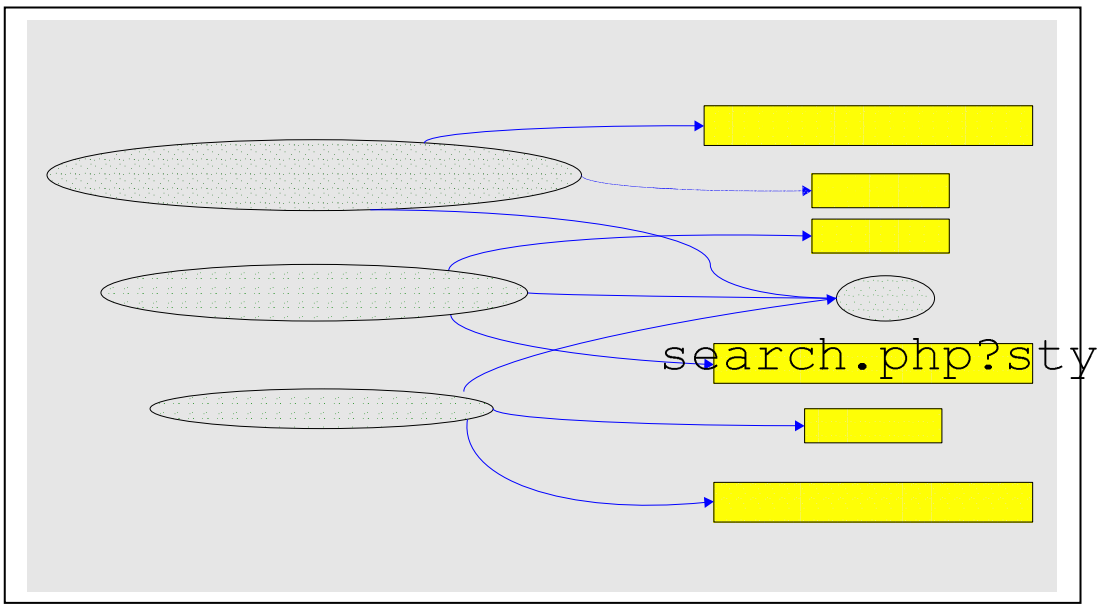


Figura 25. Grafo RDF de una muestra del fichero tracks.rdf.

8.3. API Sesame.

Para poder interactuar con los documentos RDF utilizaremos el API de Sesame que nos permite crear el repositorio en memoria, guardar los ficheros RDF y consultarlos. Para esto utilizaremos las siguientes librerias[54]. Las interfaces principales para manejar el repsitorio de Sesame se encuentran en el paquete **`org.openrdf.sesame.repository`**

8.3.1. Crear un repositorio y añadir datos.

El primer paso para el desarrollo de esta parte es crear un repositorio. El repositorio de Sesame se crea con las siguientes instrucciones:

```
LocalService service = SesameServer.getLocalService();
LocalRepository myRepository = service.createRepository("myRep42",
false);
java.net.URL myRDFData1 = new
java.net.URL("http://localhost:8080/sesame/TFC35/WEB-
INF/tracks2.rdf");
```

Estas órdenes permiten crear un repositorio local en memoria que guardará los datos de los ficheros RDF. Este tipo de repositorio es volátil, es decir se pierde al cerrar el programa.

Para poder crear este objeto necesitamos del paquete :

`"org.openrdf.sesame.server.SesameServer"`

Si necesitas crear un repositorio con un fichero que se almacene en local consulta la documentación de Sesame.

8.3.2. Realizar querys al repositorio de datos.

El siguiente paso es el que nos permite realizar consultas en el repositorio creado, para esto tenemos que ejecutar una Query en uno de los lenguajes RDF permitidos. Con esta versión de Sesame podemos utilizar tanto SeRQL-S, SeRQL-C y RDQL. El siguiente ejemplo permite ejecutar una consulta en la que veremos todas las sentencias del modelo que hemos cargado en memoria. La consulta se ha realizado con SeRQL:

[54] La explicación de estas librerías es parte del capítulo VII de la documentación de usuario de Sesame versión 1.2.3.

```
String query = "SELECT * FROM {x} p {y}";
QueryResultsTable resultsTable =
myRepository.performTableQuery(QueryLanguage.SERQL, query);

int rowCount = resultsTable.getRowCount();
int columnCount = resultsTable.getColumnCount();

for (int row = 0; row < rowCount; row++) {
    for (int column = 0; column < columnCount; column++) {
        Value value = resultsTable.getValue(row, column);

        if (value != null) {
            System.out.print(value.toString());
        }
        else {
            System.out.print("null");
        }

        System.out.print("\t");
    }

    System.out.println();
}
```

Para poder realizar consultas necesitamos el paquete

"org.openrdf.sesame.query.QueryResultsTable"

8.4. Funcionamiento aplicación buscador Music Web.

8.4.1 Requerimientos para probar aplicación.

Para realizar la prueba de la aplicación se necesita tener instalado JVM, una de las últimas versiones de Tomcat y Sesame 1.2.3.

Dentro de la carpeta de instalación de Tomcat tiene que estar la instalación de Sesame en el directorio ../Tomcat/Webapps/Sesame. En la carpeta de Sesame copiamos el fichero de la práctica para probarlo.

La carpeta del proyecto tiene una carpeta *web-inf* en la que están los ficheros RDF[55] que se están utilizando para probar la aplicación:

- El fichero artists2.rdf contiene registros de los artistas " ABBA " y "Above de Law"
- El fichero tracks2.rdf contiene registros de las canciones del grupo " ABBA ".

[55] La aplicación funciona igualmente con los ficheros originales que contienen de 45851 sentencias.

Levantamos el servicio de Monitor Tomcat y abrimos el explorador en la dirección local

http://localhost:8080[56]/sesame/TFCmlamasc/

Con esto accedemos al entorno web del buscador.

8.4.2. Interacción con buscador Web Music.

La aplicación solo tiene una lista de valores, una caja de texto y el botón de búsqueda.

Para realizar una búsqueda tenemos:

- Seleccionar en el campo lista: Artistas o Discografías
- Introducir texto de búsqueda en el cuadro de texto.
- Pinchar en el botón de Búsqueda para obtener los resultados.

Una búsqueda de la discografía de ABBA nos traerá 4 canciones MP3.

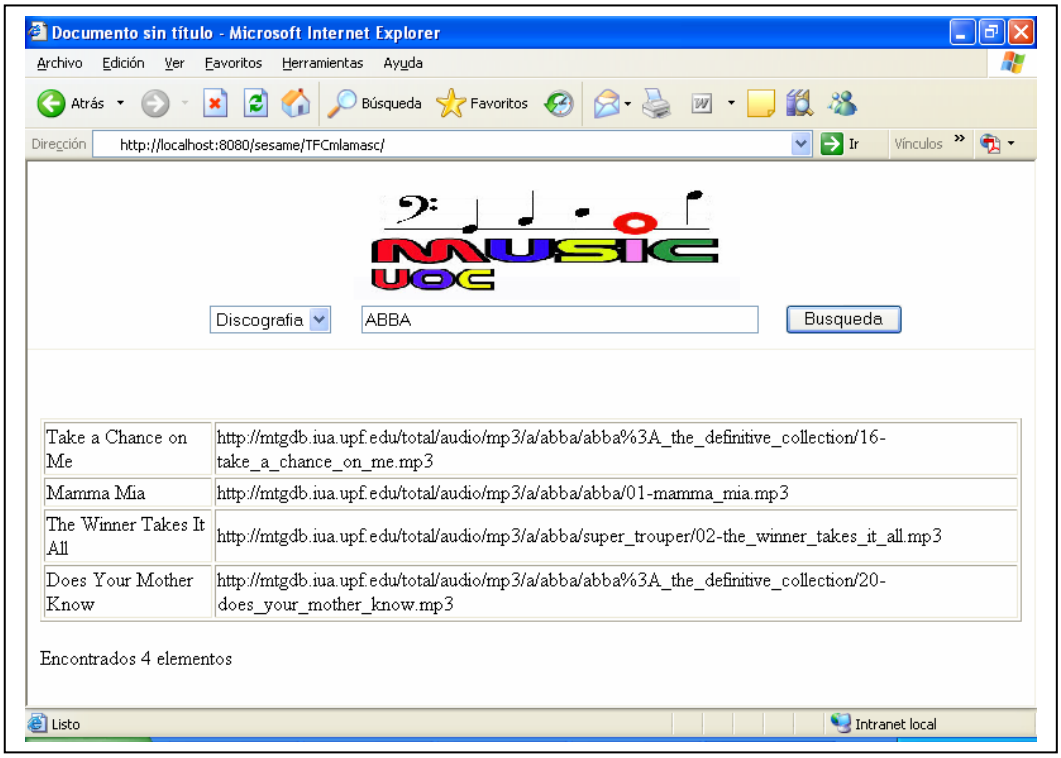


Figura 26. Búsqueda de Discografía de ABBA en Web Music.

[56] Puerto configurado por defecto en la instalación de Tomcat. Este paremetro puede modificarse en el fichero web.xml.

9. Conclusiones.

Todo lo que engloba a la Web Semántica no deja de ser algo más que los primeros pasos a lo que puede llegar a ser en un espacio corto de tiempo. El hecho de intentar que los ordenadores entiendan lo que hay en la Web, es intentar aplicar una serie de propiedades que solo las personas tienen. Se están logrando avances en este campo y muchos de ellos importantes, pero aún queda mucho camino por recorrer.

Los metadatos son una parte importante de la infraestructura de la información para ayudar a crear orden en la Web, infundiendo descripción, clasificación y organización que ayuden a crear almacenes útiles de información. Para conseguir estructurar adecuadamente la información es necesario cierta convergencia de formatos de codificación y una semántica acordada.

RDF puede ser una arquitectura de metadatos en la Web, es una capa adicional a XML(capa sintáctica) que proporciona semántica. La semántica entendible por la máquina nos permitirá obtener información de forma automatizada, gestión de recursos e intercambio de los mismos. Obtendremos mayor precisión en las búsquedas de información.

RDFS viene a ser un modelo conceptual apropiado para la descripción de los recursos.

OWL permitirá crear ontologías tan necesarias para que los ordenadores puedan entender el significado de los datos que están tratando.

Todos estos lenguajes han necesitado de nuevas herramientas de desarrollo que permitan explotar los datos; Jena y Sesame dos frameworks creados para guardar ontologías (en OWL o RDFS) y sus instancias (los datos) además de permitirnos utilizar algunos de los lenguajes de consulta RDF analizados.

Estos lenguajes de consulta RDF nos recuerdan en muchas de sus cláusulas al SQL utilizado en los SGBDs tradicionales. El resultado de la comparativa realizada demostró claramente que SeRQL y SPARQL son los lenguajes que más se acomodan a las necesidades de los modelos semánticos. Algunos de estos lenguajes carecían de operadores como el LIKE o el ORDER BY tan básicos para el manejo de cadenas y para filtrar resultados.

Jena y Sesame ser configurados para almacenar datos en memoria principal, en ficheros directamente o bien utilizando un gestor de base de datos relacional como MySQL 5, PostgreSQL y oracle 10g.

La persistencia de datos en Sesame y MySQL o oracle 10g. se consigue creando una base de datos relacional con varias tablas relacionadas por diferentes claves que manejan el concepto de tripleta. De esta forma se dispone por ejemplo de tablas resources para guardar los recursos, literals para guardar los literales o URIs.. Cualquiera de estas soluciones aventaja la solución que nos ofrece Jena, que genera una

base de datos sin relaciones de claves foráneas en la que resulta difícil garantizar la confiabilidad de la información.

La ejecución de consultas en cualquiera de estos modelos ha sido rápida, el tiempo de respuesta es mínimo.

En la parte práctica hemos visto el funcionamiento de herramientas que abarcan diferentes niveles de la Web Semántica, citemos por ejemplo IsaViz que nos permitió ver el modelo de grafos del caso práctico que se aportaba para este trabajo.

Protegé nos permitió ver la ontología y trabajar con el fichero para ver las clases y subclases que tenía la ontología musical que se definió para este propósito.

Jena nos permitió ejecutar consultas SPARQL desde un entorno poco amigable[57] como es ARQ, cualquier error en una query te podía llevar a revisar dos ficheros de datos. El lenguaje SPARQL a persar de ser una estandarización de W3C no es fácil de aprender.

Con Sesame y su entorno Web se estudiaron los lenguajes RDQL y SeRQL comprobando lo rápidos que resultan para consultas en ficheros RDF de considerable tamaño.

MySQL 5.0 y Sesame 1.2.3. se han utilizado para ver la estructura de tablas que se crea automáticamente desde Sesame para guardar los datos del grafo RDF.

[57] MSDOS.

10.Glosario de Términos.

Agentes. Una entidad de software que funciona continua y autonomamente en un medio particular a menudo habitado por otros agentes y procesos,sin requerir de guía constante o intervención humana.
DAML+OIL: El DAML (DARPA Agent Markup Language) es un lenguaje para modelar ontologías creado como una extensión de RDF. El DAML provee un rico juego de construcciones con el que crear ontologías y aumentar la información para que sea legible y comprensible por las máquinas. En diciembre de 2000 el DAML pasó a decirse DAML+OIL, debido a la revisión de las especificaciones del lenguaje.

NameSpaces. Los espacios de nombres fueron introducidos por XML para resolver conflictos de nombres entre elementos en un documento XML cuando los elementos se derivan de diferentes fuentes, permitiendo el uso de múltiples vocabularios en un mismo documento. Un NameSpace es un vocabulario definido dentro de un URI (Identificador de recursos Universal).

Ontología. Permite definir vocabularios que las máquinas puedan entender y que sean especificados con la suficiente precisión como para permitir diferenciar términos y reverenciarlos de forma precisa. Una ontología es un vocabulario para describir las cosas que existen. Se utilizan para estudiar la existencia de todo tipo de entidades, abstractas y concretas, que componen el universo.

Las ontologías son colecciones de entidades que definen las relaciones entre conceptos y especifica reglas lógicas para realizar razonamientos sobre estas relaciones. Las máquinas entenderán el significado de la semántica asociada a una página Web siguiendo links que especificarán las ontologías. Las ontologías también contienen la información sobre equivalencias semánticas.

OWL es el acrónimo del inglés **Web Ontology Language**, un lenguaje de marcado para publicar y compartir datos usando ontologías en Internet desarrolaldo por W3C. Se puede considerar una extensión de RDFS. OWL es una extensión del vocabulario de RDF (*Resource Description Framework*) y es una derivación de los lenguajes de ontologías Web DAML+OIL. Junto al entorno RDF y otros componentes, estas herramientas hacen posible el proyecto de Web semántica. Actualmente, OWL tiene tres variantes: **OWL Lite**, **OWL DL**, y **OWL Full**. Estas variantes incorporan diferentes funcionalidades, y en general, OWL Lite es más sencillo que OWL DL y OWL DL es más sencillo que OWL Full. OWL Lite está construido de tal forma que toda sentencia pueda ser resuelta en tiempo finito, la versión mas completa de OWDL puede contener 'bucles' infinitos.

Una **URI** es simplemente un identificador Web: como las cadenas comenzando por "http:" o "ftp:" que pueden encontrarse usualmente en la World Wide Web. Cualquiera puede crear una URI, y su propiedadad es claramente delegada, asi forman una tecnologia bae ideal sobre la cual puede contruire una Web global. De hecho, la World Wide Web es algo asi: cualquier cosa que tiene una URI se considera estar "en la Web".

RDF ***Resource Description Framework*****, Marco de descripción de recursos. 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).** Se trata de un modelo para definir relaciones semánticas entre distintas URIs. RDF está basado en la sintaxis XML, y permite describir semánticamente una URI asociándole un conjunto de propiedades y valores. Los modelos RDF se construyen como grafos dirigidos especificando "triples" (URI, propiedad, valor). Los metadatos especificados con RDF son "comprensibles por las máquinas, y por tanto, procesables de forma automática.

Tesauros documentales son un tipo de lenguaje combinatorio que consta de listas de términos que representan un ámbito científico y técnico determinado y que posee una serie de relaciones semánticas entre los términos que lo conforman

W3C. World Wide Web Consortium, abreviadamente W3C, es una organización que produce estándares para la World Wide Web (o Telaraña Mundial)

11. Bibliografía.

http://www.w3.org Consorcio World Wide Web (W3C)

http://dat.etsit.upm.es/~abarbero/curso/xml/xmltutorial.html Tutorial sobre XML.

http://www.iua.upf.es/~jblat/material/doctorat/students/jccbis/Ontologias.htm Concepto de ontología.

http://www.openrdf.org/doc/rql-tutorial.html Manual de RQL

http://www.openrdf.org/doc/sesame/users/ch06.html Manual sobre SeRQL.

http://www.w3.org/RDF/ Manual sobre RDF.

http://www.w3.org/TR/rdf-sparql-query/ Lenguaje SparQL.

http://www.malditainternet.com/node/96/print Página sobre RDF.

http://kowari.org/ Página oficial de kowari.

http://www.joseki.org/protocol.html Joseki: RDF WebApi

http://www.hipertexto.info/documentos/introduc.htm Tesis sobre Web Semántica.

http://tripletest.sourceforge.net/2005-06-08/index.html Comparativa de w3c con Kowari Sesame y Jena

http://dublincore.org/documents/dces/ Dublín core

http://www.ub.es/bid/13perez2.htm IsaViz última versión diciembre 2005.

http://www.openrdf.org/forum/mvnforum/viewthread?thread=769 Foro de sesame

http://www.mysql.com/ Mysql

http://www.monografias.com/trabajos24/oracle/oracle.shtml Oracle versus mysql

http://www.ub.es/bid/13perez2.htm

12. ANEXOS

12.1. INSTALACIÓN DE SESAME.

Cualquier instalación que necesitemos de Sesame podemos bajarla de la página de descargas[58].

La primera instalación de Sesame que podemos hacer es **como librería de Java**, para esto solo tenemos que copiar la librería sesame.jar dentro del directorio lib/ de la versión de java que tengamos.

Otra forma de instalarlo es **como servidor**. Sesame permite varias opciones de almacenamiento de datos RDF: podemos almacenar datos en memoria principal (opcionalmente podríamos dejarlos en un fichero dump), podemos almacenarlos directamente en ficheros o podemos guardarlos en una base de datos relacional. Para esta última opción la instalación de Sesame requiere además de los drivers para DBMS que utilicemos (PostgreSQL, MySQL u oracle..).

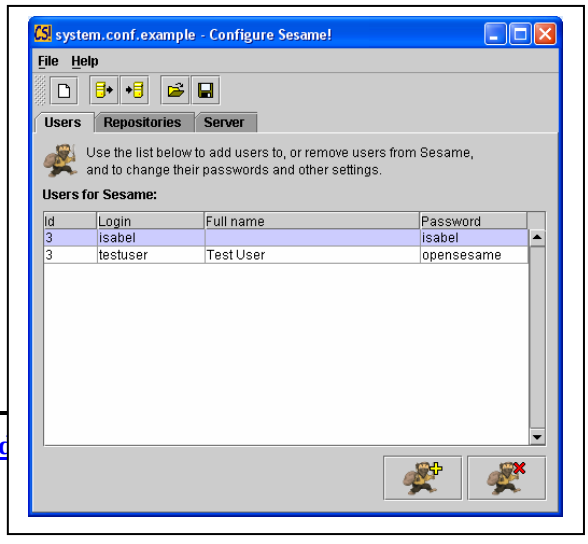
En este estudio empezamos probando la instalación de Sesame con Oracle 10g (podría probarse también la versión 10g).
La instalación para Oracle requiere descomprimir el fichero sesame.war (antes se renombra como sesame.zip) al directorio sesame dentro de /lib de java.

Deberemos modificar el fichero web.xml para finalizar la instalación, será necesario consultar la documentación de Sesame para poder indicar los parámetros adecuados.

12.1.1. Primeros pasos con Sesame y un gestor de bases de datos relacional.

El primer paso tras la instalación de Sesame como servidor será la definición de repositorios y nuevos usuarios de la aplicación.

Para esto tenemos que arrancar el fichero configSesame.bat que nos abre la siguiente aplicación:



[58] http://www.openrdf.org/c

12.2. INSTALACIÓN DE JENA CON ORACLE 10g.

La instalación de Jena 2.3. se puede bajar de la página http:// . La instalación se encuentra en el fichero readme.html. Los pasos a seguir son:

- Descomprimimos el fichero de instalación en el directorio lib donde tenemos instalado el JDK.
- Incluimos el directorio jena en la variable CLASSPATH. (ej. C:\jdk\lib\jena).
- Ejecutamos el fichero test.bat para comprobar si la instalación de jena se ha realizado correctamente.

12.2.1. Módulo ARQ y SPARQL.

Jena ofrece también una utilidad llamada ARQ[59] que permite ejecutar sentencias SPARQL sobre ficheros .rdf (--) desde la línea de comandos.
El fichero de instalación se encuentra en la referencia indicada para ARQ. Para instalarlo solo tenemos que descomprimir un fichero y crear una nueva variable de entorno ARQROOT.
Para probar las sentencias anteriores utilizamos el siguiente formato de comando desde la línea de comandos:

C:\ARQ> sparql --data=artistas.rdf --query=q1.rq

12.2.2. Instalación de Oracle 10g.

Probamos una primera instalación con oracle 9i para poder definir un repositorio de datos.

1. *Importante:*

   Edita el fichero `com.hp.hpl.jena.db.impl.Driver_Oracle.java`. Eliminar los comenatarios que aparecen en las siguientes lineas:

   ```
   import oracle.jdbc.OracleResultSet;
   import oracle.sql.BLOB;
   ```

2. Cargar el driver JDBC. Esto nos permite conectarnos desde Jena a la base de datos.
3. Crear una conexión a la base de datos.
4. Crear o abrir la conexión a la base de datos

```
// Load the Driver
```

[59] http://jena.sourceforge.net/ARQ/download.html

```
String className = "oracle.jdbc.driver.DriverOracle" // path of
driver class
Class.forName (className); // load driver
String DB_URL = "jdbc:oracle:oci:@"; // URL of database server
String DB_USER = "scott"; // database user id
String DB_PASSWD = "tiger"; // database password
String DB = "Oracle"; // database type
// Create database connection
IDBConnection conn =
new DBConnection ( URL, DB_USER, DB_PASSWD, DB );
// Create a model in the database
ModelMaker maker = ModelFactory.createModelRDBMaker(conn);
ModelRDB m = (ModelRDB) maker.createModel ();
< your Jena code to work with the model goes here >
// Remove the model. NOTE: this deletes the model from the
database.
// Subsequent attempts to open this model will fail.
m.remove();// NOTE: remove deletes the model from the database
// Close the database connection
conn.close();
// Alternative to creating an unnamed model ...
// Create a named model
ModelRDB m = maker.createModel(
"myName");
... later, or in another Jena application ...
// Open a named model.
ModelRDB m = maker.openModel(
"myName");
```

12.3. Instalación de Tomcat 5.5.12. Windows XP.

Tomcat (también llamado Jakarta Tomcat o Apache 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.

La instalación de la aplicación es muy sencilla solo tenemos que ejecutar el instalador e indicar el puerto que va a utilizar Tomcat, por defecto utiliza el puerto 8080 pero se le puede indicar cualquier otro siempre que sea >1024.
Si tiene instalado oracle puedes tener ocupado el puerto 8080 por lo que tendrás que cambiar el puerto de tomcat. Para hacer esto cambia el parámetro *port =8080* por otro con valor > 1024.
La jerarquía de directorios de instalación de Tomcat incluye:

bin - arranque, cierre, y otros scripts y ejecutables
common - clases comunes que pueden utilizar Catalina y las aplicaciones web
conf - ficheros XML y los correspondientes DTDs para la configuración de Tomcat
logs - logs de Catalina y de las aplicaciones
server - clases utilizadas solamente por Catalina
shared - clases compartidas por todas las aplicaciones web

webapps - directorio que contiene las aplicaciones web
work - almacenamiento temporal de ficheros y directorios

El directorio ***webapps*** será el que utilizaremos para instalar los repositorios de la práctica. En este caso vamos a instalar Sesame 1.2.3. en la carpeta "../*webapps/sesame*"
I

Una vez instalado podemos probarlo levantando el Monitor Tomcat y abriendo en el navegador la web siguiente:

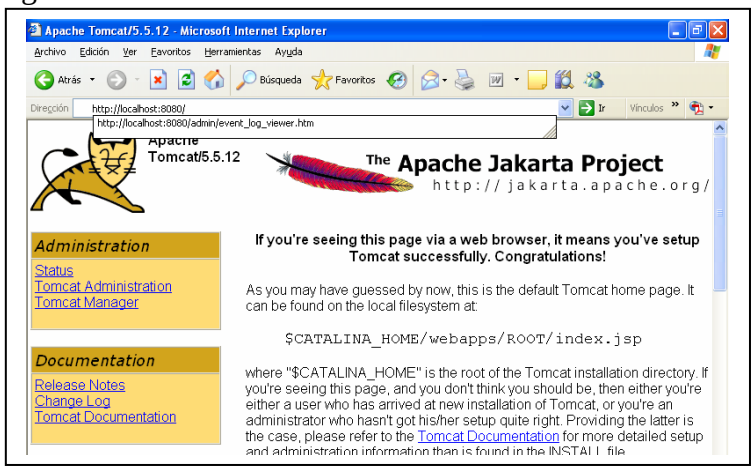


Figura 27. Página principal de Tomcat 5.5.

12.4. Ficheros RDF de los ejemplos de la memoria.

Artists.rdf : 12 sentencias RDF.

```
<!DOCTYPE rdf:RDF [
      <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
      <!ENTITY music 'http://www.semanticaudio.org/ontology/0.1#'>
]>
<rdf:RDF
      xmlns:rdf="&rdf;"
      xmlns:music="&music;"
>

  <rdf:Description
rdf:about="http://www.mp3.com/search.php?stype=artist&amp;query=Above+
the+Law&amp;action=Search">
    <rdf:type rdf:resource="&music;Artist"/>
    <music:name>Above the Law</music:name>
    <music:decade>1980</music:decade>
    <music:decade>1990</music:decade>
    <music:city>Los Angeles</music:city>
    <music:nationality>US</music:nationality>
  </rdf:Description>

  <rdf:Description
rdf:about="http://www.mp3.com/search.php?stype=artist&amp;query=ABBA&a
mp;action=Search">
    <rdf:type rdf:resource="&music;Artist"/>
    <music:name>ABBA</music:name>
    <music:decade>1970</music:decade>
    <music:decade>1980</music:decade>
    <music:city>Stockholm</music:city>
    <music:nationality>SE</music:nationality>
  </rdf:Description>

</rdf:RDF>
```

Tracks.rdf : 9 sentencias RDF.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE rdf:RDF [
      <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
      <!ENTITY music 'http://www.semanticaudio.org/ontology/0.1#'>
]>
<rdf:RDF
      xmlns:rdf="&rdf;"
      xmlns:music="&music;"
>


  <rdf:Description
rdf:about="http://mtgdb.iua.upf.edu/total/audio/mp3/a/abba/abba%3A_the
_definitive_collection/16-take_a_chance_on_me.mp3">
    <rdf:type rdf:resource="&music;Track"/>
    <music:title>Take a Chance on Me</music:title>

<music:played_by>http://www.mp3.com/search.php?stype=artist&amp;amp;qu
ery=ABBA&amp;amp;action=Search</music:played_by>
  </rdf:Description>

  <rdf:Description
rdf:about="http://mtgdb.iua.upf.edu/total/audio/mp3/a/abba/abba/01-
mamma_mia.mp3">
    <rdf:type rdf:resource="&music;Track"/>
    <music:title>Mamma Mia</music:title>

<music:played_by>http://www.mp3.com/search.php?stype=artist&amp;amp;qu
ery=ABBA&amp;amp;action=Search</music:played_by>
  </rdf:Description>

  <rdf:Description
rdf:about="http://mtgdb.iua.upf.edu/total/audio/mp3/a/abba/super_troup
er/02-the_winner_takes_it_all.mp3">
    <rdf:type rdf:resource="&music;Track"/>
    <music:title>The Winner Takes It All</music:title>

<music:played_by>http://www.mp3.com/search.php?stype=artist&amp;amp;qu
ery=ABBA&amp;amp;action=Search</music:played_by>
  </rdf:Description>


</rdf:RDF>
```