Universidad de Valladolid

E. T. S. De Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Prototipo de Consulta de Información Biológica en la Web de Datos

Autor:
Alberto Garzón García

Tutores:
Miguel Ángel Martínez Prieto
Javier David Fernández García

A mi familia, por todo el apoyo a lo largo de estos años,
en especial, a mi abuelo Domingo.

A mis amigos, por estar ahí durante todo este tiempo,
dando ánimos, y compartiendo muchos ratos de biblioteca.

Y a mis tutores, Javier y Miguel Ángel, por facilitarme el trabajo
en todo momento y por descubrirme el mundo del "Big Data".

Resumen

La tendencia actual de generar datos y compartir información en la Web ha dado lugar a la aparición del denominado “Big Data”. La disponibilidad de grandes colecciones de datos supone una materia prima de gran riqueza, pero las limitaciones físicas de los dispositivos de procesamiento, transferencia y almacenamiento, hacen que la gestión de los datos no sea sencilla. A ello se une la heterogeneidad de los datos, que suponen otra dificultad añadida a su tratamiento. Es por ello, que se necesitan nuevos estándares para dar formato a los datos, y nuevas herramientas que contribuyan a la gestión de los mismos.

Una gran parte de los datos provienen de las investigaciones científicas, entre ellas, del campo de la Biología. En este trabajo, se pretende analizar cómo interactúan los usuarios con la base de datos Bio2RDF y, en función de ello, modificar la herramienta HDT-it para facilitar las consultas sobre conjuntos de datos de Bio2RDF almacenados de forma local, en formato HDT.

Abstract

The current trend of generating data and sharing information on the Web has originated the so-called "Big Data" phenomenon. The availability of large collections of data is a raw material with great value , but the physical limitations of the devices of processing, transfering and storage, complicates the management of the data. Moreover, the heterogeneity of the data, is another challenge posed to their treatment. This is why new standards are needed to give format to the data, and new tools to help manage them.

Much of the data comes from scientific research, including the field of biology. In this work it is analyzed how users interact with the Bio2RDF database and, accordingly, modify HDT-it tool to facilitate queries on Bio2RDF datasets stored locally in HDT format.

Índice

Resumen 5

1: Introducción 11

1.1 Evolución de la Web 13

1.2 La Web Semántica 14

1.2.1 Búsqueda Simple en la Web 16

1.2.2 Grafos navegables 16

1.2.3 Clasificación de los datos según Berners-Lee 16

1.3 El Big Data y la Web de Datos 18

1.3.1 Roles y procesos en la Web de Datos 20

1.3.2 Aplicaciones reales de la Web de Datos. Bio2RDF 23

1.4 Objetivo del trabajo 24

1.5 Estructura de la memoria 24

2. RDF y SPARQL 27

2.1 Resource Description Framework (RDF) 29

2.1.1 El modelo de datos RDF 30

2.1.2 Vocabulario RDF 31

2.1.3 La sintaxis RDF y el lenguaje XML 31

2.1.4 El Esquema RDF o RDF Schema (RDFS) 33

2.2 SPARQL 36

2.2.1 Formato de datos 37

2.2.2 Patrones de triples 37

2.2.3 Resultados de SPARQL 39

2.2.4 Consultas por patrones de triples 40

2.2.5 Otras consultas 41

3. El proyecto Bio2RDF 43
3.1 Introducción a Bio2RDF 45
3.1.1 Procedencia y métricas de los datos de Bio2RDF 46
3.1.2 Métodos de integración en bioinformática 47
3.1.3 Integración utilizando una aproximación semántica 48
3.1.4 RDFización 49
3.2 Arquitectura de Bio2RDF 51

4. El formato HDT y la herramienta HDT-it 53
4.1 HDT y Diccionario Comprimido 55
4.2 Componentes de HDT 56
4.2.1 Header (Cabecera) 57
4.2.2 Diccionario 57
4.2.3 Triples 60
4.2.4 Un ejemplo sencillo 61
4.3 Resolución de SPARQL 63
4.4 Usos donde destaca HDT 63
4.5 La herramienta HDT-it 64
4.5.1 Motivación 64
4.5.2 Visualización de RDF 66

5. HDT-it y Bio2RDF: otra forma de consultar datos biológicos 69
5.1 Planteamiento 71
5.2 Estadísticas en las consultas a Bio2RDF 71
5.2.1 Uso de patrones simples 72
5.2.2 Tipo de consulta 72
5.2.3 Funciones 73
5.2.4 Usos de la función FILTER 74
5.2.5 Tipos de REGEX 75
5.2.6 Tipo de grafo 76
5.2.7 Número de patrones de triples 77

5.2.8 Camino más largo por consulta 78
5.2.9 Número de prefijos declarados por consulta 79
5.2.10 Tipos de Joins 80
5.2.11 Sobre las consultas 82
5.3 Modificaciones sobre HDT-it 84
5.3.1 Requisitos 85
5.3.2 Desarrollo 85
5.3.3 Pruebas 90
5.3.4 La nueva interfaz 91
5.3.5 Utilización de la aplicación 92

6. Conclusiones del trabajo 95
6.1 Conclusiones 97
6.2 Trabajo futuro 99

Bibliografía 101

Apéndices 107
A1. Glosario 109
A2. Contenido del CD 113

1. Introducción

1.1 Evolución de la Web

Desde su inicio, la Web se ha convertido en una potente herramienta de intercambio de información entre sus usuarios. A lo largo de su historia, el aspecto de las páginas web ha ido evolucionando, pasando de la Web de los documentos o Web 1.0, a la Web 2.0, o Web dinámica. Pero con motivo de los avances tecnológicos y las nuevas necesidades de los usuarios, se está cambiando la concepción de la Web y se está entrando en lo que hoy se conoce como la Web 3.0, o Web Semántica.

Inicialmente, las páginas web se componían de documentos en texto plano, relacionadas entre sí mediante enlaces entre los documentos. El contenido era estático y no permitía la interacción con el usuario. Con la aparición del formato HTML se consiguió mejorar el aspecto de los contenidos web, haciendo que la navegación por los mismos fuese más agradable, pero los contenidos seguían sin ser interactivos y no se actualizaban con frecuencia. Otro aspecto importante a tener en cuenta, es que el número de usuarios con acceso a Internet era muy reducido y, generalmente, con pocos recursos de procesado, por lo que no se producían grandes cantidades de información.

El cambio a la versión 2.0 se produce cuando los sitios web dejan de ser estáticos y comienzan a incluir contenidos multimedia, como imágenes animadas o videos, entre otros. Además, se permite que el usuario interactúe con la página web, a través de formularios, añadiendo comentarios o permitiendo la edición de los contenidos existentes. Por otro lado, el acceso a Internet se convierte algo al alcance de un mayor número de usuarios, que además cuentan con recursos de almacenamiento y procesamiento. Este aumento del número de usuarios da lugar a la aparición de los portales de edición colaborativa, o wikis, así como el surgimiento de las redes sociales, promueven una interactividad mayor entre los mismos.

La evolución de la Web 2.0 está dando lugar a lo que hoy en día se conoce como Web Semántica, Web de Datos o Web 3.0. Diariamente, los usuarios de Internet general grandes cantidades de datos e información, que ponen a disponibilidad del resto en la Web. Estos datos son de orígenes distintos y de diferentes formatos, lo cual da lugar a una mezcla heterogénea de la que es complicado extraer información. Los usuarios, cuentan con muchos recursos de procesamiento, pero en proporción a los datos disponibles, no es suficiente para lograr los mejores resultados. Por lo general, cuando un usuario necesita información concreta sobre algo, no tiene la capacidad ni el tiempo necesarios para realizar un análisis de todos los datos con los que cuenta. Es por ello, que la

tendencia que se quiere implantar en la Web sea la de dar un formato estándar a los datos, para facilitar su interrelación, y dotarlos de un significado semántico que facilite su integración con otros datos.

Por otra parte, a día de hoy, en varios países existen, o están en proceso, leyes de acceso a la información pública, que obligan a las Administraciones Públicas a publicar todos los datos que sean de interés público, excepto los que se consideren de alta sensibilidad. Esto, sumado a las publicaciones de datos resultantes de investigaciones de los grupos científicos, da lugar a un enriquecimiento de la información de la Web.

1.2 La Web Semántica

La Web Semántica surge de la idea de publicar los datos en Internet de manera que puedan ser procesados por una máquina de manera automática. Para ello, es necesario incorporar cierta semántica a los objetos de información albergados en la Web (páginas, servicios, fuentes de datos), de manera que se puedan identificar de manera unívoca, y que se pueda enlazar información relacionada. La filosofía es crear un espacio de información compartido y navegable.

La base de la Web Semántica es dotar a cada recurso con un identificador único, llamado URI (Uniform Resource Identifier). De esta forma, en lugar de relacionar documentos HTML, como es habitual en la Web tradicional de hipertexto, se relacionan datos, los cuales se escriben en un lenguaje específicamente desarrollado para capturar la semántica, denominado RDF (Resource Description Framework).

Uno de los objetivos de la Web Semántica es que los datos puedan ser reutilizados de forma espontánea gracias a las interconexiones entre ellos. Con el fin de facilitar el consumo automático, se utilizan marcadores o tags, que constituyen la metainformación sobre los datos. Se pretende crear documentos más autodescriptivos. Así, utilizando esta metainformación en una búsqueda usando el lenguaje natural, sería posible obtener en un único documento toda la información relacionada, suponiendo un ahorro de tiempo y esfuerzo a los usuarios. En lugar de devolver enlaces a distintas páginas, el buscador daría la información obtenida con extractos de distintas páginas.

A pesar de la existencia de estándares, como el lenguaje RDF, la adopción de la Web Semántica es un proceso costoso y que todavía está en desarrollo. Aunque la creación de nuevos sitios web orientados a la Web Semántica no es algo complicado, sí lo es reescribir los ya existentes.

Para fomentar el crecimiento de la Web Semántica, Tim Berners-Lee, precursor de la misma, propuso una versión menos rigurosa, denominada Datos Enlazados (Linked Data), y que se basa en las siguientes reglas:

1. Nombrar las cosas mediante URIs.
2. Uso de URIs HTTP para la búsqueda de esos nombres.
3. Ofrecer la información utilizando estándares como RDF o SPARQL.
4. Enlazar unos URIs con otros para relacionar más cosas.

Como se ha dicho anteriormente, los URIs son los identificadores de los recursos. Un recurso es cualquier objeto de información que se pueda identificar. Los URIs utilizan la sintaxis de las direcciones web URL. Cuando una URI es direccionable, esto es, además de identificar un recurso, conlleva una web asociada, se denomina URI HTTP.

Los URIs HTTP son nombres que se refieren a un objeto y definen una web de objetos de información. Es el responsable de publicar la información quien decide qué identifica un URI HTTP.

La información en la web se debería dar a través de URIs, algo que, en general, hacen las ontologías, pero no los grandes conjuntos de datos. Mediante la búsqueda de propiedades y clases en los datos, es posible extraer información y relaciones entre los términos de las ontologías RDF, RDFS y OWL.

RDF es un formato, amparado por el Consorcio de la Web, W3C, que se utiliza para representar la información en la Web Semántica. Aunque pueda ser leído por humanos, está concebido para que pueda ser procesado de forma automática por máquinas. Es un modelo de datos estándar que posibilita el intercambio y la reutilización de metadatos estructurados sin las ambigüedades producidas por la procedencia de los datos, desde distintas fuentes. [39]

Existen varios formatos de representación de RDF, siendo los más habituales RDF/XML, N3 y Turtle. Del mismo modo, la información en RDF podría consultarse mediantes distintos lenguajes o APIs web, siendo SPARQL el lenguaje habitual de consulta.

Para dotar de mayor valor a la información hay que relacionarla con más información existente en la Web. El propio valor de la Web aportará valor a la información. Así, cada objeto de información puede establecer enlaces, o links, con otros objetos de información identificados mediante URIs, conformando un grafo etiquetado de conexiones.

1.2.1 Búsqueda simple en la Web

Los URIs pueden contener el símbolo # seguido de un nombre. Eso significa que el individuo nombrado se define en el documento al que hace referencia el resto del identificador antes de #. Al hecho de truncar el URI antes del símbolo # para obtener el URI de un documento y acceder al mismo para obtener información sobre el individuo se llama deferencear el URI y da lugar a la Web Semántica Básica. Utilizando un URI en un navegador semántico se tiene que obtener alguna respuesta. Deferencear algo significa hacerlo accesible.

Sobre esto hay distintas variaciones. Una de ellas es no incluir el símbolo # en el URI, lo cual identifica conceptos abstractos acerca de los cuales existe un documento con información. Otra forma de enlazar datos es dar un documento donde se describen y un identificador en el mismo.

1.2.2 Grafos navegables

Se llama grafo navegable si para cualquier nodo, buscando por su URI, se devuelven todas las declaraciones en las que ese nodo es objeto y objeto y todos los bnodes adjuntos al mismo, a través de un arco. El subgrafo devuelto se conoce como molécula RDF o Grafo de Expansión Mínima (MSG).

Si los datos se almacenan en más de un documento se dan repeticiones que pueden ocasionar inconsistencias, lo cual es un problema para explorarlos. Es necesario que el conjunto de los datos navegables sea consistente, con enlaces bidireccionales. Estos datos se pueden generar de forma automática.

Debido al volumen de los datos, es necesario que exista un servicio de consulta SPARQL para conseguir eficiencia en las consultas remotas sobre el conjunto de datos. Si los datos están enlazados de forma eficiente, dado un URI, en un endpoint de SPARQL se tendría que encontrar su camino.

1.2.3 Clasificación de los datos según Berners-Lee

Berners-Lee [15] desarrolló un sistema de calificación de estrellas para la calidad de los datos enlazados. Partiendo de la definición de Datos Enlazados, se definen los Datos Abiertos Enlazados (LOD) como aquellos Datos Enlazados que pueden ser reutilizados de forma libre, gracias a una licencia abierta. El proyecto LOD (Linked Open Data), es una inciativa del W3C que pretende que todos los datos sean abiertos, excepto los más sensibles, y se creen estándares que favorezcan el intercambio de información y que ésta pueda ser aprovechada por terceros. Cada vez son más los conjuntos de datos que forman parte de la nube

del proyecto LOD, que en septiembre de 2011 tiene un aspecto como el que se muestra en la figura:

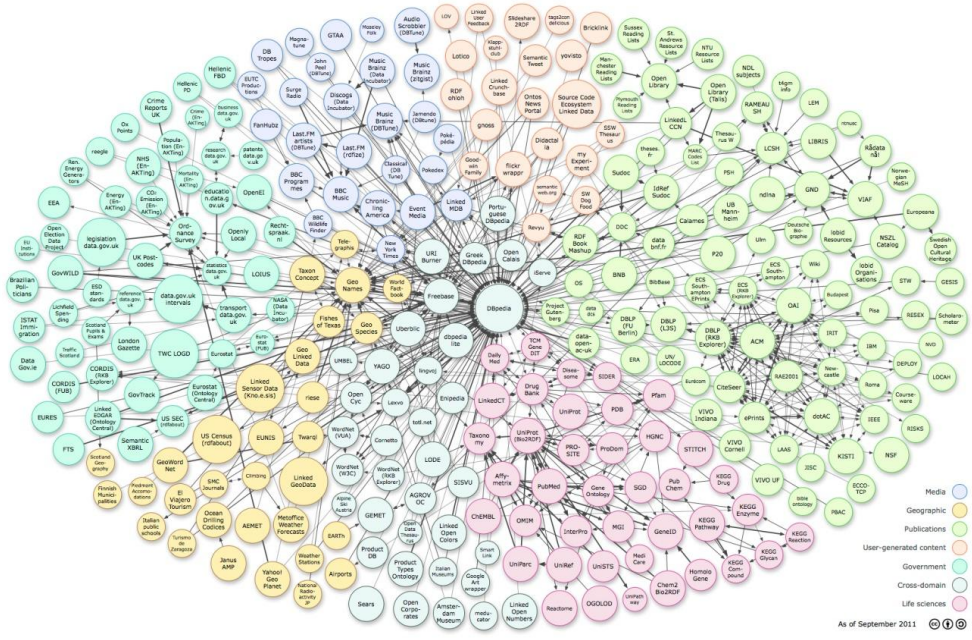


Figura 1: Nube LOD. Fuente: Library of Congress [29]

Si la información se hace pública en su totalidad, se obtiene una estrella, y cuanto más estrellas, más fácil será de utilizar por parte de la gente.

★ Si la información se publica en la web con licencia abierta de Datos Abiertos, sea cual sea el formato.

★★ Si los datos están estructurados y pueden ser interpretados por una máquina.

★★★ Si cumplen lo del punto anterior y además el formato es no propietario.

★★★★ Además de lo anterior, utilizar estándares abiertos de W3C, como RDF y SPARQL, cuando se identifiquen los recursos, de manera que puedan ser apuntados por otros.

★★★★★ Todo lo anterior y además establecer enlaces con datos de otros para crear un contexto.

1.3 El Big Data y la Web de Datos

A diario se genera gran cantidad de datos que necesitan ser almacenados y procesados para obtener información. Estos datos se almacenan en servidores web o bases de datos, pero el ritmo de crecimiento de los mismos hace que el almacenamiento sea cada vez más complicado. Es por ello, que se hace necesario comprimir los datos antes de almacenarlos, lo que plantea el siguiente problema: si se reduce mucho el tamaño, aumenta el tiempo de consulta. No obstante, muchos de los datos no son aprovechados en todo su potencial por carecer de mecanismos competitivos para su manejo.

Cuando la colección de datos supera la capacidad computacional que tradicionalmente tiene un sistema de bases de datos convencional, entonces se denomina Big Data. Esta gran cantidad de datos hace necesario que sea fácil de almacenarlos. Además, se necesita que haya herramientas automáticas para gestionarlos, teniendo en cuenta la variedad de los datos existentes así como la velocidad de transferencia, ya que pueden estar distribuidos.

El Big Data está caracterizado por las cuatro dimensiones que abarca, también conocidas como las 4 V's.

- Volumen: cada vez se generan más datos y el volumen está en constante crecimiento. Según la Ley de Moore, la potencia de un ordenador se duplica cada 18 meses, mientras que, en ese período, el volumen de los datos se triplica, lo que implica que los recursos de cómputo son insuficientes.
- Velocidad: en ocasiones los datos se pueden almacenar de forma distribuida, por lo que hay que optimizar el procesamiento y las comunicaciones, algo complicado, ya que la red no crece al mismo ritmo que los datos.
- Variedad: los datos que se almacenan pueden ser estructurados o no, y pueden provenir de diferentes fuentes de datos, por lo que hay gran heterogeneidad en los mismos. Debido a esto, el modelo relacional habitual no es el mejor y se hacen necesarias nuevas herramientas. El análisis conjunto de diversos tipos de datos da lugar a información.
- Veracidad: si no se puede garantizar la certeza de los datos, es difícil confiar en ellos.

El Big Data es, además, la posibilidad de generar y encontrar nuevo conocimiento con los datos existentes, algo que es cada vez más posible gracias al avance de las nuevas tecnologías. La importancia del Big Data va más allá de la compartición de información, ya que los datos también generan beneficios económicos. Para los gestores de negocios, obtener conocimiento de los datos les ayuda a tomar decisiones mejores y aumentar el rendimiento.

A pesar de todo, el Big Data también presenta problemas. Como se ha descrito anteriormente, el volumen y la variedad de los datos, junto con la velocidad de procesamiento, hacen que su gestión sea complicada. Además, hay que asegurar la seguridad de los datos y la privacidad de los mismos. Pero como también se ha dicho, las nuevas tecnologías permiten una gestión cada vez más eficiente del Big Data, a través de soluciones distribuidas y escalables, que permiten la flexibilidad en la representación de los datos, garantizando la privacidad y la seguridad.

Es necesario que el Big Data dé una identidad y un significado a la totalidad de los datos, lo que da lugar a la Web de Datos.

En las Bases de Datos Relacionales habituales, se considera el conocimiento presente en ellas como un todo, y lo que no aparece se toma como falso, por lo que funcionan bien cuando los escenarios están acotados y las operaciones son validadas de una forma sencilla. Los datos se almacenan por filas en tablas, asumiendo que la información es completa y poniendo NULL en los valores que no se conocen. Debido a la rigidez de las tablas, si cambia el modelo, tiene que cambiar su estructura.

Por el contrario, la Web de Datos se considera una base de datos universal y supone un Mundo Abierto, lo que implica que se asume que los datos son incompletos y que pueden estar detallados sin que seamos conscientes de ello. La información presente se puede completar desde distintos sitios, consiguiéndose así una información plural. Los datos se almacenan en grafos y si se desconoce algún valor, no se almacena.

Bajo la suposición del Mundo Abierto, el conocimiento nunca es completo, por lo que se consideran procesos tanto adquirirlo como usarlo y su ubicación no es fija. El conocimiento puede tener o no una estructura, o ser parcial, por lo que se hace necesario un modelo del mismo que sea independiente de la representación. De existir, la estructura es evolutiva y creciente con el paso del tiempo. Basado en conexiones, el conocimiento depende del contexto en el que se dé y tiene que presentar coherencia con su estructura lógica, la cual lo determina, junto con el uso que se dé del mismo por parte de los consumidores.

Aunque inicialmente un escenario no esté completo ni sea óptimo, se puede mejorar de forma incremental si incorporar nuevos datos es fácil. En ocasiones, la definición de un atributo hace posible una conexión entre fuentes de datos sin relación previa.

1.3.1 Roles y procesos en la Web de Datos

La finalidad del Big Data es la compartición de la información, por lo que los datos que se generan tienen que ser publicados, intercambiados y consumidos. Esto da lugar a la distinción de tres conjuntos, no necesariamente disjuntos, de roles:

- Generador de datos: es la entidad que origina los datos y puede ser de cualquier naturaleza.
- Publicador de datos: se encarga de hacer públicos los datos integrándolos en la Web. Tiene que encargarse de filtrar los datos más importantes y de dar metainformación sobre ellos. Tiene que tener gran capacidad de almacenamiento y una gran potencia computacional.
- Consumidor de datos: se asume que es el usuario final de los datos, cuyos recursos son limitados en almacenamiento y capacidad computacional.

El flujo de los datos da lugar a una serie de procesos encadenados que forman un círculo abierto, que podría convertirse en una espiral creciente, como se ve en la figura.

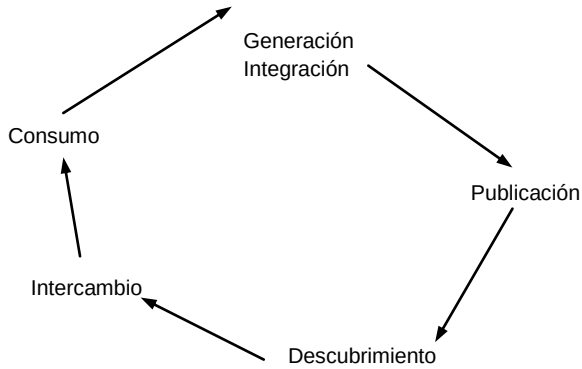


Figura 2: Procesos de la Web de Datos.

Generación

El proceso de generación de datos de cara a su publicación en la Web de Datos no es sencillo. Hay que especificar los datos, analizando sus fuentes, planteando su modelo lógico, aportando metainformación y diseñando los URIs que los van a representar. Después hay que modelarlos, utilizando para ello ontologías y vocabularios, a ser posible, ya existentes. Por último, hay que generar propiamente los datos, transformándolos a RDF, eliminando lo que no es relevante y enlazando con otras fuentes.

Integración

En ocasiones puede existir datos sobre lo mismo en distintas fuentes, y con distinto vocabulario. Se tiene que buscar un vocabulario común para fusionar las fuentes. Es un proceso que requiere control humano para encontrar similitudes entre los datos.

Publicación

Es un proceso que no está poco explotado a pesar de no haber un problema de escalabilidad definido. El entorno computacional de la publicación es muy potente, algo que podría aligerar el resto de procesos.

Descubrimiento

Cuando los usuarios de la Web de Datos buscan algo sobre un tema concreto, pueden encontrar nuevas cosas que no esperaban sobre el mismo, lo cual supone un descubrimiento que puede ser positivo en las investigaciones que se lleven a cabo.

Intercambio

El proceso de intercambio es complejo. En primer lugar hay que modelar los datos, debido a la gran variedad de tipos que existe, para lo que se recomienda usar RDF. Por otro lado, el volumen de los datos influye en la velocidad en una relación inversa, mayor volumen, menor velocidad, ya que el ancho de banda de la red no puede aumentar indefinidamente. La alternativa es reducir el volumen de los datos, pero esto también tiene sus inconvenientes. Se puede plantear enviar menos datos, pero en ocasiones esto no es posible.

Otra opción para reducir el volumen de los datos es comprimirlos. En principio parece una buena solución, puesto que el RFD serializado presenta muchas redundancias, y de hecho es muy utilizada, ya que se mejora el flujo del intercambio. Pero también tiene una parte negativa. Por un lado, se tienen que comprimir los datos antes de enviarlos, algo que se realiza una sola vez, pero cada consumidor tiene que descomprimirlos, algo para lo cual quizás no tenga recursos.

Consumo

El usuario final de los datos necesita descomprimirlos para procesarlos y consultarlos, algo dificultado por el volumen de los datos originales que se recuperan. Lo más probable es que el consumidor cuente con poco espacio de almacenamiento y una capacidad computacional limitada, con lo cual, bajo estas

consideraciones, el procesamiento de las consultas a los datos también se ve afectado.

Los sistemas de gestión de bases de datos específicos para el almacenamiento y consulta de triples se conocen como triple stores. Para resolver el problema de las consultas se ofrecen distintas soluciones de implementación de triple stores:

- Tabla de tres columnas: es una tabla con tres columnas, una para sujetos, otra para predicados y una última para objetos, que se indexa por una de ellas. Los mayores problemas de esta solución son que no escala bien en velocidad y que la resolución de los patrones en los que la variable es la dimensión indexada de la tabla es incorrecta.
- Posicionamiento Vertical: es una tabla de dos columnas, una para sujetos y otra para objetos, cuyo índice es una de ellas. Si se deja variable el predicado se produce una mala resolución. No escala bien en velocidad.
- Multi-índices: se utiliza un índice por cada posible ordenación de los triples, lo cual supone una multiplicación por 6 de los requisitos de espacio. No escala bien en espacio y se pierde mucho tiempo en operaciones I/O.
- Soluciones en memoria: la representación es específica para la memoria y ocupa mucho espacio. No son realmente una solución.
- Soluciones distribuidas: la representación se encuentra en distintos nodos, con intercambios poco eficientes entre los mismos, lo que, junto con la falta de escalabilidad de los nodos, supone un problema.

Como se comentó anteriormente, RDF se utiliza para representar datos y orientarlos al consumo automático. El modelo de RDF, simple pero potente, se adecua a la Web de Datos, siendo una herramienta útil para la implementación de un mundo abierto, ya que permite describir recursos y definir semánticas. Su estructura en grafo hace que sea flexible y, por tanto, es fácil integrar datos de distintas fuentes. Para representar este modelo se hace necesaria una sintaxis que permita almacenar sus instancias en archivos legibles por una máquina y que permita compartir estas instancias entre aplicaciones. La estructura formal que exige RDF es XML ya que tiene un modelo en árbol y soporta de forma consistente la representación de la semántica.

A pesar de que el diseño original de RDF era para el modelado de datos a pequeña escala, su uso se ha extendido para representar Big Data, algo que no puede consumir directamente un humano, aunque se presente en formato serializado. Un formato de serialización de RDF para Big Data debería orientarse a sus consumidores directos, es decir, estar en formato binario; y debería tener en cuenta las restricciones de volumen y velocidad, por lo que se tendría que comprimir. RDF se puede definir de forma serializada y comprimida.

Utilizando compresión se optimizan tanto el volumen como la velocidad y se puede mantener más información en memoria. Además, es posible operar sobre los datos comprimidos, aumentando el rendimiento.

La serialización comprimida de RDF, por tanto, permite mejorar las velocidades de intercambio y de consulta, así como reducir el espacio de almacenamiento, y sólo se genera una vez, por lo que no supone un problema que se necesite una potente configuración computacional, algo que no es necesario para el consumo, ya que se puede hacer sin descomprimir.

Una forma de modularizar RDF para utilizarlo en el flujo de datos es el formato HDT [38], el cual está orientado al consumo automático y facilita los procesos de la Web de Datos. El formato HDT facilita la compresión de los datos y además hace posible que éstos puedan ser entendidos por una máquina. Manteniendo los datos RDF comprimidos, el formato HDT soporta búsquedas y navegación, a la vez que ahorra espacio. La codificación HDT de un conjunto de datos se hace siguiendo un modelo de tres componentes: cabecera, diccionario y triples, como se verá más adelante. Una herramienta que permite la generación y el consumo de archivos RDF en formato HDT es HDT-it, que posibilita la búsqueda de patrones de triples básicos en ficheros HDT, además de mostrar la matriz de adyacencia en 3D del grafo RDF subyacente.

1.3.2 Aplicaciones reales de la Web de Datos. Bio2RDF

Gran parte de los datos, se generan de investigaciones científicas, se almacenan de forma distribuida y se consultan de forma constante. La Biología es una de las ciencias cuya información ha crecido desde hace años. No obstante, muchos de los datos no son aprovechados en todo su potencial por carecer de mecanismos competitivos para su manejo. Dos grandes colecciones biológicas son Uniprot y Bio2RDF, ambas orientadas a la Web de Datos.

Bio2RDF es una base de datos biológica que utiliza las tecnologías de la Web Semántica para proporcionar datos interrelacionados de las ciencias de la vida, apoyando el descubrimiento de conocimiento biológico. Bio2RDF genera e integra datos interpretables por máquinas a través de metodologías simples, utilizando técnicas sintácticas y semánticas. Estos datos pueden ser consultados utilizando SPARQL y se pueden obtener respuestas a preguntas complejas.

El objetivo de Bio2RDF es distribuir las grandes cantidades de datos procedentes de la bioinformática con el fin de facilitar el descubrimiento de conocimiento biológico, que es representado mediante lenguajes típicos de la

Web Semántica, como RDF/OWL, con los que se crean descripciones axiomáticas de las entidades biológicas.

Actualmente, Bio2RDF asegura la provisión de datos básicos en el campo de la biología, incorporándose en análisis bioinformáticos, consultas en bases de datos o aplicaciones para dispositivos móviles.

1.4 Objetivo del Trabajo

El objetivo del trabajo es facilitar la consulta de grandes cantidades de datos almacenados en archivos locales. Para ello se partirá de la herramienta HDT-it, ya existente. Esta herramienta permite realizar consultas sobre los datos comprimidos en formatos HDT. Se pretende con el trabajo realizado, cambiar la funcionalidad de esta herramienta para crear un prototipo de consultas más complejas.

Estas búsquedas más complejas se basan en el análisis de un log de consultas a los endpoints de Bio2RDF. De esta forma se han detectado los patrones de búsquedas más habituales en esta base de datos y se han utilizado como base para la actualización de la herramienta.

En concreto, se trata de introducir en la herramienta HDT-it la posibilidad de realizar búsquedas de dos patrones con una variable en común. Este tipo de consulta no sólo habitual de Bio2RDF, sino que se utiliza de forma común en las consultas tanto a bases de datos relacionales como a endpoints de bases de datos de la Web de Datos. No obstante, el prototipo propuesto no sólo funciona para Bio2RDF, sino que puede utilizarse para cualquier colección de datos que siga el modelo RDF/HDT.

Otros objetivos de este trabajo, es introducir conceptos de la Web 3.0, así como sus estándares RDF y SPARQL y, también del modelo HDT, propuesto para la compresión de los datos en RDF. Tanto el modelo HDT, como la herramienta HDT-it son fruto del trabajo de investigación que se realiza por parte de los miembros del grupo DataWeb Research de la Universidad de Valladolid [24], estando el formato HDT reconocido por el W3C desde marzo de 2011.[17]

1.5 Estructura de la memoria

Esta memoria se divide en 6 capítulos, en cada uno de los cuales se trata un tema distinto que se va enlazando con el resto.

En el capítulo 1 se introducen y se detallan los conceptos de Web Semántica, Web de Datos o Datos Abiertos Relacionados. También se hace una breve mención a otros aspectos relacionados con la Web 3.0, como el modelo RDF o el lenguaje SPARQL o el formato HDT. Se incluye también un ejemplo real de la Web de datos, como es el proyecto Bio2RDF, que sirve de base para el desarrollo de este trabajo, junto con la herramienta HDT-it. Los objetivos de este trabajo se recogen en este capítulo.

El capítulo 2 entra en detalle en la descripción del modelo RDF. En él se ven las principales características de este estándar de la Web Semántica. Enlazando con RDF, también se describe el lenguaje de consulta SPARQL, con ejemplos de consultas, para facilitar su comprensión.

El capítulo 3, describe el proyecto Bio2RDF. Este proyecto, es un ejemplo de aplicación práctica de la Web de Datos y los Datos Abiertos Enlazados.

El capítulo 4 trata sobre el formato HDT y la herramienta HDT-it. El formato HDT, reconocido por el W3C, supone un cambio en la compresión y navegación de los datos. En este capítulo se da una visión general del mismo, así como un ejemplo de utilización. También se describre la herramienta HDT-it, útil para consultar datos en formato HDT.

En el capítulo 5 se describe el trabajo realizado, propiamente dicho. Se incluyen estadísticas de uso de Bio2RDF que motivan la modificación de la herramienta HDT-it.

Por último, en el capítulo 6 se detallan las conclusiones del trabajo realizado y las posibilidades de trabajo futuras que pueden surgir a partir del trabajo actual.

También se incluye la bibliografía utilizada como base de la memoria, tanto artículos como referencias a sitios web, separadas en dichos apartados.

Finalmente, se incluyen dos apéndices. El primero se corresponde con el glosario, con algunos de las abreviaturas y conceptos utilizados en la redacción del texto. En el segundo, se detalla el contenido del CD adjunto al volumen de la memoria.

2. RDF y SPARQL

2.1 Resource Description Framework (RDF)

La información en la Web Semántica se representa utilizando un modelo de descripción conocido como RDF. Es un modelo de datos estándar, que aunque puede ser legible por humanos, está concebido con capacidad para ser procesado automáticamente por máquinas. La infraestructura de RDF hace posible intercambiar y reutilizar metadatos estructurados, sin que haya ambigüedades en sus semánticas, a pesar de proveer de distintas comunidades informativas. De esta forma se logra la interoperabilidad de distintas aplicaciones mejorándose el intercambio de información en la Web.

El modelo RDF se desarrolla amparado por el Consorcio de la Web, W3C, y provee a las comunidades de descripción de recursos de la capacidad para definir elementos de metadatos, sin acordar la semántica.

Entre las convenciones soportadas por RDF para facilitar la interoperabilidad están los mecanismos estándar de representación de semánticas. RDF también proporciona vocabularios que pueden ser interpretados por los humanos y por las máquinas. Los vocabularios son las propiedades o elementos de metadatos que se definen por las distintas comunidades, y pueden ser reutilizados o editados según las necesidades del dominio. La estandarización de la declaración de los vocabularios hace posible que la semántica se pueda compartir entre comunidades. RDF no necesita un registro general, ya que su diseño soporta la modularidad semántica.

Según la recomendación del W3C, RDF se puede definir como una combinación de varios elementos. Por un lado, un sistema que posibilita que las aplicaciones interactúen intercambiando información que se procesa automáticamente a través de la Web. Por otro lado, un mecanismo que hace que los procesos se puedan automatizar de manera que se puedan llevar a cabo con recursos Web. Además, una infraestructura permite codificar, intercambiar y reutilizar metadatos estructurados, además de permitir la fusión de diferentes sistemas de metadatos. Se puede concluir que RDF es una manera de expresión de las relaciones entre los objetos.

Si bien RDF puede ser serializado en distintos formatos, la Recomendación del W3C propone una sintaxis XML como medio común de intercambio y procesamiento automáticos de metadatos, con independencia del proveedor, lo que hace compatibles distintos sistemas de metadatos. La puesta en común de las necesidades de las distintas plataformas ha dado lugar al modelo RDF a

proporcionar una arquitectura robusta y flexible que soporta metadatos en la Web, independientemente del área en que se apliquen.

2.1.1 El modelo de datos de RDF

RDF proporciona un modelo de descripción de recursos. Los recursos son objetos identificables de forma única y se describen mediante una propiedad y un valor para la misma, lo que da lugar a un triple formado por sujeto, predicado y objeto, respectivamente, conocido como descripción. Con este modelo de triples se busca lograr un método que exprese semánticas de forma no ambigua, siendo su codificación legible por máquinas. La representación de los triples es un grafo dirigido, donde los objetos y las propiedades son los nodos y los arcos, respectivamente, y los valores están acotados. Debido a la uniformidad del modelo de RDF, un URI puede ser nodo y flecha, por lo que son posibles las autoreferencias.

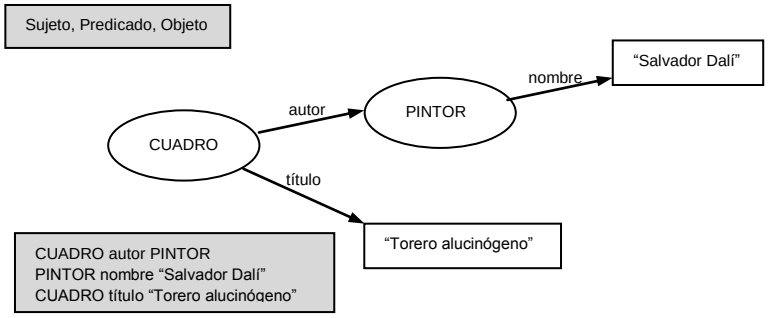


Figura 3: Grafo RDF

La totalidad de los términos utilizados en los triples da lugar al vocabulario, que se compone de URIs, blank nodes y literales, cuya definición se dará más adelante.

Los tipos de propiedades que se asocian a los recursos expresan la relación de los valores correspondientes a los recursos. Dichos valores pueden ser atómicos u otros recursos con sus respectivas propiedades. El modelo independiente de la sintaxis con el que cuenta RDF hace posible la representación de recursos y sus descripciones.

El modelo RDF se puede ver como un sistema de programación orientada a objetos. La diferencia está en que en RDF los objetos no tienen métodos, ya que son información, y la descripción de las propiedades se da en términos de clases de los recursos en los que se aplican. Además, RDF hace que se puedan definir nuevas propiedades con un dominio o un rango.

RDF proporciona un mecanismo de asociación de propiedades con recursos, pero es necesario que haya una declaración de un recurso que represente una cosa antes de decir algo sobre la misma. En algunos casos, un recurso puede tener el valor de distintas propiedades, algo que se soluciona con el uso de identificadores únicos. Esto permite que las propiedades se puedan asociar sin ambigüedades, algo que hace que la información pueda ser reutilizada de forma explícita.

Con RDF se pueden crear recursos en varios niveles, llegando tan bajo como lo requiera el dominio. Además, con el modelo RDF se pueden describir otras descripciones. El desarrollo correcto de RDF se basa en el uso de una sintaxis de representación neutral.

2.1.2 Vocabulario RDF

El vocabulario de RDF se compone de URIs, blank nodes y literales:

- URIs: identifican de forma unívoca un recurso de la web y se utilizan para integrar un dato en la Web de Datos. Dentro del conjunto de URIs, gran número de términos pueden tener un prefijo común.
  `http://dbpedia.org/resource/Salvador_Dalí`
- Bnodes: son nodos anónimos que en el grafo RDF se utilizan, generalmente, como nodos padre con el fin de agrupar datos.
- Literales: son los nodos hoja de los grafos RDF y tienen el rol de objeto. Pueden estar etiquetados con un lenguaje o un tipo de datos, aunque no es necesario.
  `"Surrealismo"@es`

Según el modelo de RDF, un sujeto no puede ser un literal y un predicado tiene que ser necesariamente una URI. Como objeto puede actuar cualquiera de los tipos.

2.1.3 La sintaxis RDF y el lenguaje XML

El modelo de definición de recursos de RDF es simple pero potente y necesita una sintaxis que lo represente. Esta sintaxis hace posible un almacenamiento eficiente de la información, así como su procesamiento automático. También permite que la información pueda intercambiarse entre aplicaciones. La sintaxis que recomienda el consorcio W3C para RDF es XML, ya que su estructura soporta que las semánticas se representen de forma consistente. Para describir los elementos sintácticos se utiliza la notación EBNF, como XML, garantizándose así que los lenguajes son compatibles entre sí.

El lenguaje XML da normas estructurales pero no del contenido y, mediante sus reglas, hace posible la creación de lenguajes de etiquetado. Una DTD (Document Type Definition) almacena información sobre el contenido de cada conjunto de datos. El desarrollo de etiquetas de RDF se hace utilizando una DTD de XML.

Los beneficios que obtiene RDF de la flexibilidad de XML se hacen patentes en la generación de conjuntos de etiquetas, la orientación multiplataforma o a través de una semántica que hace posible la definición de cualquier tipo de recurso.

Los elementos que componen un documento se pueden representar en forma de grafo y se puede representar completamente con la adicción o eliminación de etiquetas, gracias a XML. La coherencia que RDF aporta a las búsquedas facilita la consulta.

El intercambio de paquetes de metadatos que se definen en comunidades distintas se ve facilitado por las limitaciones que RDF impone en la estructura, que favorece que se codifique coherentemente y el intercambio de metadatos estandarizados.

Para codificar los objetos y sus características, el modelo RDF cuenta con dos tipos de sintaxis. Una es la sintaxis que ofrece un conjunto reducido de expresiones para realizar la descripción, llamada serializada. La otra, que sirve como complemento de la primera, gracias a un conjunto de entidades, es la sintaxis abreviada, que hace posible que los subconjuntos se puedan describir completamente. Las dos sintaxis pueden estar presentes en un mismo documento RDF.

Sintaxis serializada

La sintaxis serializada da lugar a una organización en la estructura de la descripción utilizando elementos básicos y un número reducido de valores. Esta sintaxis fija los parámetros con los que trabaja el elemento rdf:Description, el cual provee el URI y el estado del recurso que va a ser utilizado en RDF.

La sintaxis serializada hace posible que un recurso se puede poner en RDF de manera simple y estructurada. Si se busca un detalle más amplio del recurso, entonces hay que utilizar la sintaxis abreviada. Cuando el recurso es localizable e identificable, se pone el atributo *about*. Si el recurso no cuenta con URI o no existe, el atributo será ID y en un mismo documento no puede repetirse.

Sintaxis abreviada

Para poder interpretar DTDs de XML como modelos RDF está la sintaxis abreviada. Las formas de abreviar posibles son tres:

- Las propiedades que no se repiten en un elemento description se expresan como atributos XML del mismo.
- Los objetos pasan a ser atributos cuando un objeto es un recurso y sus propiedades son cadenas de caracteres.
- Los nombres de elemento utilizan el valor de una propiedad type dentro de Description.

Espacios de nombres

Dos de los objetivos que persigue XML son la universalidad y la orientación multiplataforma, lo que hace posible la generación de lenguajes propios que se pueden compartir entre desarrolladores. Pero esta capacidad de definición de semánticas por parte de las distintas comunidades puede dar lugar a ambigüedades. El significado de una propiedad puede variar enormemente en función de las necesidades de la misma.

Los espacios de nombres, o namespaces, hacen que un elemento pueda ser entendido en un entorno dado. RDF utiliza el espacio de nombres de XML, que identifica de forma no ambigua las semánticas.

La estructura presente en los documentos XML o los ficheros RDF es en grafo. Una aplicación en tres capas es necesaria para la interpretación de un documento XML. En la capa uno está la semántica y se soporta el contenido. La capa dos contiene la sintaxis que da estructura a la información. La tercera capa define los datos concretos con el léxico resultante de la unión de las dos primeras capas. Es precisamente en las capas 1 y 2 donde aparece RDF, con el fin de dotar de coherencia a la descripción.

2.1.4 El Esquema RDF o RDF Schema (RDFS)

Las propiedades de RDF hacen que se puedan ver como pares atributo-valor de los recursos, y las relaciones entre los mismos. Sin embargo, ni las propiedades ni las relaciones entre las mismas y otros recursos pueden ser descritas en RDF, para ello existe RDF Schema [40], que es el lenguaje de descripción de RDF, y que define las clases y las propiedades que a su vez pueden ser utilizadas como medio de descripción de clases, propiedades y otros recursos. Por tanto, RDF Schema es una extensión semántica de RDF que

permite describir grupos de recursos entre los que existe relación y las propias relaciones. Los recursos sirven como medio para caracterizar a otros recursos, como pueden ser el dominio y el rango de una propiedad. La validez de las propiedades en una descripción concreta de RDF se define en los esquemas de RDF. A su vez, RDF se utiliza para describir los vocabularios de RDF Schema, cuya estructura se basa en el modelo de datos de RDF. Los espacios de nombres son útiles para la identificación de esquemas RDF.

A través de la deferenceación del esquema URI se puede acceder a una descripción de un RDF Schema procesable por humanos y por máquinas. El procesamiento automático de un esquema permite que una aplicación pueda extraer conocimiento de la semántica de las propiedades que aparecen en el esquema. Cuando se comprende la semántica de todas las propiedades de una descripción se habla de la comprensión de un esquema de RDF. Esto hace posible el análisis de la descripción en la propiedad y los valores correspondientes, a pesar de no tener la comprensión de un esquema en concreto, y transportarla de forma intacta.

La finalidad de RDF Schema es formalizar los vocabularios de metadatos de forma que sean interpretables tanto por máquinas como por humanos y que sean intercambiables entre comunidades.

Tipos de datos

El modelo RDF dice cómo pueden ser usados los tipos de datos, pero no da un catálogo con ellos. Los tipos de datos son identificados e interpretados mediante su URI y se caracterizan de forma externa al modelo. Debido a esto, RDF puede representar información de distintas fuentes sin tener que convertir los tipos entre específicos y nativos de RDF.

Para que sea válido, un tipo de datos de RDF tiene que contar con:

- Un espacio de valores que contenga los literales representables por un tipo.
- Un espacio léxico que determine qué caracteres pueden usar los literales de un tipo.
- Una correspondencia que dé el valor de una cadena de caracteres en concordancia con el espacio léxico.

Si ante cualquier cadena de caracteres un tipo de datos puede contestar si ésta forma parte del espacio léxico, y en caso afirmativo, dar su valor correspondiente, se considera que es válido.

En la práctica, los tipos de datos que describe XML Schema están implementados, mayormente, por software para RDF, lo cual hace que se soporten de forma generalizada. Esto hace que dos sistemas diferentes mejoren su interoperabilidad. La responsabilidad de determinar si un literal tipado es válido o no depende del software que lo procesa, no de RDF.

URIs

La Web de Datos asume como principio que un URI es recuperable vía HTTP, pero RDF no lo contempla, ya que el modelo puede utilizarse fuera la misma.

Una de las ventajas que ofrece el uso de URIs como predicados es que se pueden diferenciar las propiedades utilizadas por los agentes. Además, hace posible tratar a los predicados como recursos, lo que facilita la caracterización en un esquema RDF, permitiendo que se desarrollen vocabularios compartidos, algo que hace más fácil la interoperabilidad usando los predicados para describir un contexto concreto.

A través de la reutilización del conocimiento que se ha adquirido y que se ha acordado dentro de un área, los vocabularios compartidos hace colecciones puedan interoperar mejor. Antes de modelar en RDF hay que comprobar si ya existen vocabularios en esa área, para así poder reutilizarlos.

URIs vs URLs

El fin de un URI es la identificación global y unívoca de un recurso dado y accesible desde una red. De forma general, los URIs que se utilizan son, en su mayoría, URLs.

Los URIs son universales e independientes del recurso al que referencian, gracias a su sintaxis básica. Se pueden separar en tres partes:

- Protocolo: que da la descripción del mecanismo a través del cual se accede al recurso.
- Hostname: que da el nombre principal de la dirección. Puede ir acompañado de la ruta de acceso a un recurso.
- Consulta adicional: que es información complementaria que permite trabajar con el recurso directamente.

Los URIs son la unión de los URL y los URN. Por un lado, el URL determina la ubicación física, mientras que por otro, el URN da el nombre único, siendo el URI lo que describe todo. La especificación de los URIs está en el RFC2396. [43]

2.2 SPARQL

Tanto el lenguaje de consulta de RDF como el propio protocolo de acceso a datos se conocen con el acrónimo SPARQL [47], aunque generalmente se refiere a lo primero. Con este fin, su sintaxis es similar a un lenguaje SQL y se orienta a la consulta de grafos RDF buscando patrones coincidentes.

SPARQL se define en base al modelo de datos de RDF y se orienta a la Web Semántica, dando soporte a búsquedas en grandes cantidades de información variada. Pero también es un elemento clave en la Web 2.0, ya que con el respaldo de un modelo de datos flexible, puede ser un mecanismo de consulta que sirva para todas las aplicaciones.

Las consultas pueden buscar concordancias para patrones necesarios u opcionales, mediante filtros en los valores. También pueden ser sobre patrones conjuntos o disjuntos y se puede hacer pruebas con valores o consultas restringidas en el grafo fuente de RDF. Una consulta en RDF puede devolver tanto un conjunto de resultados como un grafo RDF y puede implicar a varias fuentes de datos, independientemente de si los datos están en RDF de forma natural o si son vistos como tal gracias a un middleware. Uno de los inconvenientes que presenta SPARQL es que es de sólo lectura, por lo que no puede hacer modificaciones en los datos, únicamente puede extraer información y comparar datos. Es por ello, una herramienta que localiza información específica de forma rápida.

Actualmente, la especificación de SPARQL ofrece una estabilidad suficiente para que sus capacidades puedan ser exploradas, tanto de forma práctica como teórica, gracias a la diversidad de mecanismo de consulta.

La funcionalidad de SPARQL se divide en tres partes que se complementan:

- SPARQL Query Lenguage: es el lenguaje de consulta y da la sintaxis de las sentencias.
- SPARQL Protocol: da el formato de las respuestas siguiendo un patrón XML.
- SPARQL Query XML Result Format: es el medio de transmisión de consultas y respuestas entre los clientes y los procesadores. Los protocolos HTTP y SOAP para las consultas remotas de bases de datos RDF se definen usando WSDL 2.0.

La funcionalidad de SPARQL está suministrada por diversas herramientas y APIs, muchas de las cuales contemplan las últimas especificaciones. Al igual que

RDF, SPARQL dispone de una recomendación W3C, la última de las cuales (21 de marzo de 2013) especifica la versión SPARQL 1.1.

2.2.1 Formato de datos

La orientación a datos del lenguaje SPARQL hace que únicamente se consulte información que contienen los modelos, sin inferir en el lenguaje. Cuando una aplicación realiza una consulta, SPARQL toma la descripción y como resultado da la información solicitada como un conjunto de uniones o como un grafo RDF.

Los datos consultados por SPARQL son grafos RDF, que son conjuntos de triples. Como se ha comentado, los triples "se escriben" de forma serializada en un formato concreto. Las distintas maneras de las que se puede codificar un grafo hacen que la serialización dificulte la visión que se tiene de los tiples. Por ello, a efectos de la consulta SPARQL, en donde lo importante son los propios triples y no su serialización, se acuerda un modelo de nombrado dentro de SPARQL, similar al formato de serialización Turtle.

La unidad que se consulta por SPARQL es un conjunto de datos RDF, consistente en un grafo por defecto y un número de grafos nombrados.

2.2.2 Patrones de triples

Un patrón de triple de SPARQL se compone de sujeto, predicado y objeto, al igual que un triple de RDF. De hecho, un triple RDF es un patrón de triple de SPARQL.

Los patrones de triple se escriben como sujeto, predicado y objeto y se terminan con un punto. Dentro del patrón, los URIs se encierran entre paréntesis angulares, <>, y se pueden compactar utilizando la cláusula PREFIX para la declaración de la parte común, suministrándose la parte propia en la consulta tras el identificador del prefijo. Los literales se escriben como cadenas entre comillas, simples o dobles. Las propiedades se pueden representar de dos formas: identificándolas por su URI o usando una sintaxis con estilo qname, donde el nombre de la propiedad se precede del espacio de nombres.

Consulta:

```
SELECT ?cuadro
WHERE {
        ?cuadro autor   “Salvador Dalí” .
}
```

Resultado:

?cuadro
“Torero alucinógeno”

Figura 4: Ejemplo de consulta SPARQL

Los patrones de triples complejos pueden ser abreviados. Una serialización breve de RDF, conocida como Turtle, se utiliza para la sintaxis y las abreviaturas. Es una alternativa a RDF/XML que representa RDF con éxito.

Con la finalidad de que el patrón de triple pueda ser utilizado como consulta, puede contener variables. Mientras que en un triple esto no puede darse, en un patrón, una o varias variables pueden sustituir los valores del sujeto, el predicado y el objeto. El uso de variables indica cuáles son los datos de interés que devolverá la consulta.

En el momento que una variable concuerda con un valor cualquiera, el patrón relacionará aquellos recursos RDF de los cuales se nombre una propiedad. Cada una de las variables será ligada al valor que tenga en el conjunto RDF para cuantos triples coincidan.

SPARQL tiene en consideración todas las ligaduras que puedan darse. En el caso de que un recurso, para una propiedad determinada tenga más de una instancia, se encontrará una ligadura para cada una, algo a tener en cuenta cuando se encuentran más datos de los esperados. Encontrar patrones en el grafo es posible gracias a la concordancia de grafos.

Debido a la semiestructuración de los datos de RDF, SPARQL puede consultarlos, pero no puede fallar si no existen. Es por ello que la consulta extiende la información encontrada en una solución utilizando una parte opcional, pero siempre devuelve la parte no opcional.

Combinando patrones de triples se tienen patrones más complejos, denominados patrones de grafo.

2.2.3 Resultados de SPARQL

Los resultados de SPARQL tienen cuatro formas posibles.

- SELECT: identifica en el conjunto del resultado las variables que se nombran. Si se quieren nombrar todas las variables, se puede poner el símbolo asterisco, *. Los resultados se devuelven en una tabla, que puede contener bnodes, cuyo nombre puede variar, aunque sí se muestre su información.
- CONSTRUCT: da como resultado un grafo RDF que tiene como base la plantilla de la consulta. Si se quiere ligar variables en la plantilla del grafo, se utiliza la cláusula WHERE. Para cada solución aportada por WHERE, y dada una plantilla, se calcula el fragmento del grafo. El resultado es la combinación en un solo grafo RDF de cada fragmento aportado por cada solución. Si en la plantilla se nombra un bnode, se crea de nuevo en la plantilla para cada solución.

```
CONSTRUCT { <http://www.miejemplo.org/Yo> miVocab:quieroVerExpo ?pintor
WHERE {
        ?pintor estilo “Surrealismo” .
        ?pintor lugarNacimiento “España” .
}
```

Resultado:

```
<http://www.miejemplo.org/Yo> miVocab:quieroVerExpo “Salvador Dalí”
```

- DESCRIBE: devuelve un grafo RDF cuya forma está dada por la configuración del procesador de consultas, no por la aplicación. El procesador provee un fragmento útil de RDF para cada URI que encuentre o que aparezca en la consulta de forma explícita.

```
DESCRIBE dbpedia:Salvador_Dalí
```

Resultado:

```
Toda la información que haya en DBPedia sobre “Salvador Dalí”
```

- ASK: es una consulta lógica que devuelve un valor verdadero si hay coincidencia con el patrón, o falso de no ser así.

```
ASK { ?pintor estilo “Surrealismo” }
```

Resultado:

```
yes
```

Modificadores de la solución

El conjunto de las soluciones que se producen por la coincidencia de patrones se puede modificar de distintas maneras:

- Proyección: únicamente mantiene las variables que se han seleccionado.
- OFFSET: limita las soluciones estableciendo un índice de inicio a partir del cual se muestran los resultados. Si el valor del desplazamiento es 0, no tiene efecto.
- LIMIT: limita las soluciones estableciendo el número de soluciones que se mostrarán. Si el valor indicado es menor que el número de soluciones, no tiene efecto.
- ORDER BY: ordena las soluciones según la expresión dada, incluyendo las funciones de cliente.
- DISTINCT: elimina las soluciones duplicadas dando como resultado una única fila para cada combinación de variables y valores. Debido al orden de prioridad de los modificadores, DISTINCT se aplica antes que OFFSET y LIMIT, pero después de la proyección de las variables requeridas.
- REDUCED: permite eliminar soluciones duplicadas. El conjunto de soluciones devuelto tendrá una cardinalidad entre 1 y la cardinalidad del conjunto de soluciones original.

Si se quiere obtener un fragmento definido del conjunto de resultados se recomienda usar OFFSET y LIMIT en conjunción con ORDER BY.

2.2.4 Consultas por patrones de triples

El patrón de triple es la unidad más simple de consulta en SPARQL. Puesto que cualquiera de los componentes puede ser sustituido por una variable, hay ocho posibilidades de patrones de triples:

- Patrón (S P O): consulta un triple en concreto. Su utilización es poca, y siempre es con las formas ASK y DESCRIBE.
- Patrón (S P ?O): devuelve todos los valores ?O que con la propiedad P se asocian a S.
- Patrón (S ?P O): recupera todas las propiedades ?P que enlazan un sujeto S con un valor O.
- Patrón (S ?P ?O): extrae toda la información disponible sobre un sujeto ?S.
- Patrón (?S P O): devuelve todos los sujetos ?S para los cuales la propiedad ?P tiene valor ?O.

- Patrón (?S P ?O): dada la propiedad P se devuelven todos los partes sujeto-valor (S, O) asociados a la misma.
- Patrón (?S ?P O): dado el valor O, recupera todos los pares sujeto-propiedad (S, P) asociados al mismo.
- Patrón (?S ?P ?O): cuyo resultado son todos los triples presentes en la colección. La consulta no impone ningún límite. En la práctica se utiliza poco.

2.2.5 Otras consultas

Mediante los patrones básicos, el patrón de la consulta tiene que coincidir totalmente para que sea solución. Sin embargo, no se puede dar por hecho que todas las estructuras de RDF son completas y regulares. La parte opcional, introducida con OPTIONAL, hace posible añadir la información disponible a la solución, de forma que si no hay coincidencia en la parte opcional ni se crean ligaduras ni se elimina la solución.

Las alternativas selectivas en el conjunto de datos se gestionan con UNION. De este modo, si hay varios patrones posibles que coincidan se pueden combinar. Por cada alternativa posible, puede haber más de un patrón de triple.

Otra forma de combinar dos patrones de triples es mediante un join. Una consulta de tipo join es la unión de dos patrones de triples donde existe una variable común. El join de SPARQL es similar al join de SQL, que en el ámbito relacional combina tuplas de dos tablas. Hay 6 combinaciones posibles, teniendo en cuenta la conmutatividad y la posición que ocupa la variable común:

- Sujeto-Sujeto: combina todos los sujetos que satisfacen las restricciones de los dos patrones de triple.

```
SELECT *
WHERE {
        ?pintor estilo “Surrealismo” .
        ?pintor lugarNacimiento ?n .
}
```

- Objeto-Objeto: combina todos los valores que comparten entre ambos patrones.

```
SELECT *
WHERE {
        “Salvador Dalí” lugarNacimiento ?lugar .
        ?pintor lugarExposición ?lugar .
}
```

- Sujeto-Objeto: combina todos los elementos que hacen posible un camino con ambos patrones.

```
SELECT *
WHERE {
        ?pintor estilo “Surrealismo” .
        ?artista influidoPor ?pintor .
}
```

- Predicado-Sujeto, Predicado-Objeto, Predicado-Predicado: son menos habituales, aunque también son posibles. En ocasiones, URIs que desempeñan el rol de predicado en una colección pueden ser sujeto u objeto en el esquema que los describe. Su principal uso es en aplicaciones de razonamiento.

3. El proyecto Bio2RDF

3.1 Introducción a Bio2RDF

Bio2RDF [18] es una base de datos biológica apoyada en las tecnologías de la Web Semántica, que permite el almacenamiento de datos sin necesidad de un repositorio central, facilitando tanto la consulta como la conexión de distintas bases de datos de Internet. Este proyecto open-source construye y proporciona la mayor red de datos interrelacionados de las Ciencias de la Vida facilitando el descubrimiento biológico.

Los datos que genera e integra Bio2RDF pueden ser interpretados por máquinas a través de metodologías simples, utilizando técnicas sintácticas y semánticas. Bio2RDF unifica el URL de acceso a distintas fuentes, de las que dispone los datos en RDF, y haciendo uso de URIs. Se pueden crear datos compatibles con RDF desde fuentes con distintos formatos, procedentes de varios proveedores de datos, con las convenciones simples que define Bio2RDF. Los datos pueden ser consultados con SPARQL y pueden contestarse preguntas complejas. Bio2RDF hace públicas distintas bases de datos enlazando documentos RDF con una ontología común y con URIs normalizados, dando lugar a mashups de datos bioinformáticos.

En enero de 2013 se publicó la segunda versión, que entre sus características cuenta con:

La segunda versión, de enero de 2013, cuenta con las siguientes características:

- Más de 1 billón de triples procedentes de 19 conjuntos de datos actualizados procedentes de directorios biológicos y químicos. Estos se forman con cerca de 58000000 sujetos únicos, aproximadamente 1000 predicados únicos y más de 298000000 sujetos únicos.
- Scripts actualizados y con licencia del MIT para ser utilizados con cualquier fin, ser modificados o redistribuidos.
- IRIs normalizados con un registro de un conjunto de datos común.
- Origen de los conjuntos de datos, con información de la versión y forma de generación de los mismos.
- Estadísticas sobre las conectividades interna y externa del conjunto de datos.
- Endpoints públicos de SPARQL 1.1 habilitados con CORS que permite filtrar las búsquedas y consultas de SPARQL federadas.
- Ficheros RDF para descargar y almacenes de triples de Virtuoso con indexación por texto completo.

3.1.1 Procedencia y métricas de los datos de Bio2RDF

Los metadatos sobre cada uno de los conjuntos de datos que dan lugar a Bio2RDF son accesibles gracias a un grafo con la procedencia del conjunto. En el grafo se detallan aspectos como la fecha en que los datos se convirtieron, el enlace al script que genera los datos de Bio2RDF, la fecha en la que se creó la base de datos o la licencia que ésta concede. También se incluye una descripción detallada del esquema que sigue el modelado de la procedencia, así como la forma en que se usa.

Los grafos de procedencia se almacenan en cada endpoint de SPARQL y el patrón seguido por su URI es http://bio2rdf.org/bio2rdf-[dataset]-provenance, donde [dataset] se sustituye por el nombre corto del conjunto de los datos de origen. El modelo hace uso del Vocabulario de Conjunto de Datos Interrelacionados (VoID), propuesto por el W3C, el Vocabulario de Procedencia (PROV) y el vocabulario Dublin Core.

Cada objeto del conjunto de datos de procedencia queda identificado por un IRI único formado por la conjunción del conjunto de datos y su fecha de creación. Con una consulta en un endpoint SPARQL se pueden recuperar todos los registros de procedencia de fechas diferentes, gracias a la unicidad del IRI.

Los metadatos originados por cada uno de los conjuntos de datos de Bio2RDF se estructuran en un grafo de metadatos con un recurso void:Dataset que, a través de un predicado void:inDataset, enlaza con un recurso de registro único en los datos enlazados que se han generado.

Gracias al grafo de procedencia, los usuarios pueden hacer consultas sobre datos concretos a través de un endpoint de Bio2RDF, consultando los registros de un dato dado.

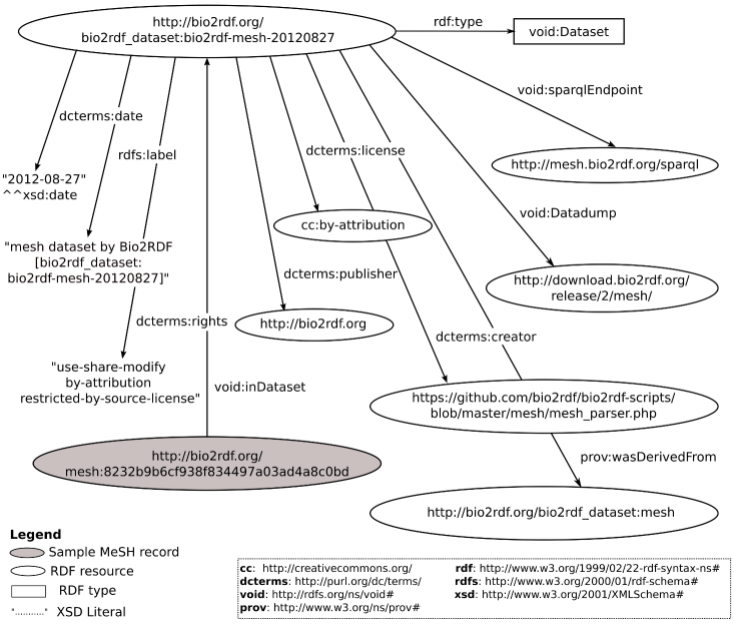


Figura 5: Ejemplo de grafo de procedencia. Extraído de [21]

Por otra parte, en un conjunto de métricas que se almacenan en un grafo nombrado, se resume el contenido de cada uno de los conjuntos de datos que forman Bio2RDF. El IRI de dicho grafo es de la forma http://bio2rdf.org/bio2rdf-[dataset]-statistics, donde [dataset] tiene el mismo significado que para el caso del IRI del grafo de procedencia. Para cada métrica hay un IRI asociado.

3.1.2 Métodos de integración en bioinformática

Desde hace tiempo se han estado buscando alternativas para la integración de datos bioinformáticos. En 1995, Davidson propuso para tal fin una serie de pasos con el fin de que todos los datos se encuentren en una base de datos federada, siguiendo un modelo común, con correspondencias entre los datos equivalentes, y que se transformen a la demanda. En Bio2RDF todos los datos se publican en RDF, por lo que no se contempla lo anterior.

El Semeda intentó en 2003 integrar bases de datos heterogéneas, pero Köhler identificó cuatro problemas:

- Una misma cosa se puede nombrar de formas distintas en función de la base de datos en la que esté.
- Los nombres de los atributos no informan sobre los atributos.

- Es necesario conocer el contenido de una base de datos antes de consultarla, algo contrario a lo que busca la aproximación a la Web Semántica.
- Sólo los atributos con mayor importancia son asociados, ya que no hay enlaces mecánicos.

Estos problemas impusieron la normalización de identificadores, meta del proyecto LSID.

Por otra parte, Stein, también en 2003, remarcó las tres aproximaciones que utilizan más asiduamente los integradores de datos: la integración de los datos utilizando la capacidad de la Web para enlazar, creación de portales que agreguen la información para integrar la vista, e incluir todos los datos en una base de datos única. La ontología propuesta se llamó nodos y nudillos, y su base es el almacenamiento de los enlaces, pero no de los datos, de forma similar a lo que hace Bio2RDF.

3.1.3 Integración utilizando una aproximación semántica

El proyecto Bio2RDF utilizó el editor de ontologías Protegé para la creación de su ontología, y está disponible en formato OWL.

Un proyecto pionero en la construcción de una base de datos integrada en formato RDF fue YeastHub. Este proyecto sirvió como guía a Bio2RDF. YeastHub utiliza Sesame como almacén de triples y normaliza los documentos de URIs. A diferencia de YeastHub, Bio2RDF es de código abierto y es extensible, además de permitir el acceso a documentos de diferentes organismos.

A pesar de que Bio2RDF permite el acceso Web a distintas fuentes de datos, no permite una navegación facetada. Esto puede hacerse utilizando herramientas del tipo Simile Exhibit de forma directa con los datos de Bio2RDF.

La arquitectura del sistema y la descripción del conocimiento son dos aspectos en torno a los cuales se define la aproximación de integración de los datos. Bio2RDF describe el conocimiento utilizando ontologías y se encuentra en los sistemas Peer de gestión de datos.

Basándose en la experiencia y utilizando una aproximación a la Web Semántica, Bio2RDF construye una forma eficiente de integración de datos bioinformáticos. La finalidad de Bio2RDF no está en responder a consultas

concretas, y utiliza código abierto para que los resultados se difundan con más facilidad.

El almacén de triples Sesame, las tecnologías de generación de páginas web Elmo RDF, JSP y JSTL y la librería URLrewrite son el conjunto de fuentes de código abierto que sirven de pilares a Bio2RDF. Puesto que una de las principales finalidades de Bio2RDF es poner en RDF los documentos que existen en las bases de datos, su software es flexible y permite desarrollar programas RDFizantes adaptados a las necesidades particulares de las fuentes de datos.

Métodos y herramientas

Existen diversos mecanismos que conducen a la integración utilizando la aproximación a la Web Semántica. A continuación se describen algunos.

Diseño de ontologías

Como paso previo a la conversión a RDF de los documentos que hay en la web, hay que generar una descripción OWL para cada HTML. Para ello, es necesario, por un lado, saber cuáles son los predicados, que se corresponden con las etiquetas de los campos. Por otro lado, identificar las relaciones que existen entre las entidades de la página HTML, es decir, los hipervínculos, que dan origen al URI del recurso, que generalmente se definirá en otro espacio de nombres.

Partiendo de la ontología obtenida, se crea el programa RDFizante necesario en JSP que mapea los predicados originales con los de RDF y que normaliza los URIs en base a la sintaxis de Bio2RDF.

3.1.4 RDFización

Todos los documentos de Bio2RDF están normalizados con formato RDF, ya sean de bases de datos locales o de documentos HTML, accesibles vía http mediante una toolbox JSP creada para ello. La transformación a RDF se puede hacer utilizando XPath, expresiones regulares o consultas SQL.

Puesto que los programas RDFizantes son públicos pueden ser personalizados, lo que da lugar a diversidad de documentos RDF.

Los rdfizadores permiten consultas directas y también tienen una interfaz tipo REST con filtro urlrewrite. Esto permite URIs estables, puesto que los programas pueden ser cambiados sin tener que cambiar el método de consulta. La

RDFización de los los datos posibilita un análisis de los mismos utilizando SeRQL o consultas de tipo REST.

Los registros de las bases de datos se identifican de forma única. Bio2RDF distingue entre los identificadores originales y los que provienen de la creación del grafo. El patrón de Bio2RDF facilita la localización de los registros originales, ya que cada registro se identifica individualmente dentro del espacio de nombres de su proveedor: http://bio2rdf.org/namespace:identifier.

La codificación de los URLs tiene que ser ser la misma que la de los URIs, para lograr que los identificadores sean URL-seguros.

Los conjuntos de datos cuentan con dos categorías de espacios de nombres.

- namespace_vocabulary: se utiliza para serializar los datos que se añaden a un registro ya identificado pero que necesita incluir nuevos predicados o tipos. El patrón es el siguiente: http://bio2rdf.org/namespace_vocabulary:someUniqueString, donde 'namespace' se cambia por el nombre corto del conjunto de datos.
- namespace_resource: es el espacio de nombres utilizado para la creación de identificadores que no existían originalmente cuando se RDFiza el conjunto de datos.

Normalización de URIs

Para lograr un mashup hay que conectar los triples, lo cual se consigue normalizando los URIs que expresan las referencias externas. Es necesario que todos los URIs tengan el mismo patrón para garantizar todas las conexiones de Bio2RDF sobre un mismo concepto, independiente de su proveedor.

Para asegurar que los conceptos del grafo RDF de Bio2RDF son únicos, hay que utilizar la sintaxis proporcionada por Bio2RDF. Los grafos identificados por el proveedor se enlazan utilizando el predicado owl:sameAs.

Para lograr normalizar los URIs, Bio2RDF propone:

- Utilizar interfaces de tipo REST. Con esto se consiguen URIs estables sin necesidad de negociación de contenido, aplicación web o redirecciones del servidor.
- Utilizar minúsculas hasta los dos puntos (:) en el URI. La sensibilidad a mayúsculas y minúsculas da lugar a URIs distintas.
- Los URIs tienen que dirigir a un documento en RDF.

Cada URI pertenece a un único documento donde un objeto como un término ontológico. Un URI normalizado tiene una sintaxis del tipo: http://bio2rdf.org/<namespace>:<identifier>.

3.2 Arquitectura de Bio2RDF

En la arquitectura de Bio2RDF se distinguen cuatro partes: las bases de datos externas, los servidores Bio2RDF.org, la aplicación myBio2RDF.org y las GUIs. Las fuentes de datos están en diferentes formatos y existen dos formas diferentes para procesarlas. Por un lado están las fuentes que descargan sus datos en una base de datos MySQL en el servidor Bio2RDF.org, y cuyos documentos se obtienen directamente del servidor a alta velocidad. Por otro lado, los documentos que se solicitan al sitio web se rdfizan directamente de la fuente original, en la parte de myBio2RDF, y se pasan a la aplicación. En ocasiones, algunos datos se almacenan en caché para ofrecer mayor disponibilidad, ya que pocos proveedores cuentan con documentos en RDF, y velocidad, por las restricciones de acceso a los documentos de algunas fuentes.

En la parte de la aplicación myBio2RDF se ejecutan en un servidor Tomcat dos servlets: Elmo y Sesame. Elmo rastrea RDF y se originó para seguir el predicado rdfs:seeAlso que aparece en los archivos FOAF. Se aplica la capacidad de rastreo de Elmo para instanciar triples en un repositorio local de Sesame, cuya interfaz permite el acceso a la base de conocimiento con SeRQL. Tanto Sesame como Elmo han sido modificados para las necesidades específicas de Bio2RDF.

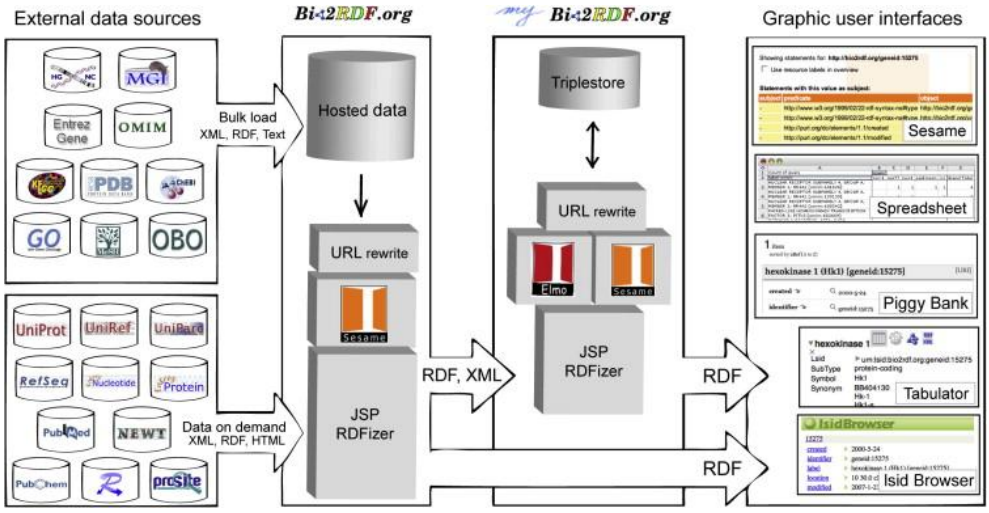


Figura 6: Arquitectura de Bio2RDF Extraída de [4]

4. El formato HDT y la herramienta HDT-it

4.1 HDT y Diccionario Comprimido

Para intercambiar información en RDF, el uso de diccionarios es una solución simple, pero efectiva, para la mejora de la escalabilidad de los procesos de gestión de colecciones de RDF a gran escala, pero no son la mejor solución. Con el fin de facilitar el flujo de datos, surge la propuesta HDT [6], la cual facilita la compresión de los datos y posibilita que sean entendidos por una máquina. Únicamente representando los datos sin comprimir, HDT reduce su tamaño unas 15 veces.

La estructura compacta de HDT mantiene grandes conjuntos de datos en RDF comprimidos a la vez que soporta operaciones de búsqueda y navegación. El formato de serialización binario para RDF que propone HDT permite un ahorro de espacio, lo cual lo hace perfecto para almacenar y compartir en la Web conjuntos de datos RDF.

La codificación en HDT de un conjunto de datos se modela con tres componentes lógicos que dirigen las particularidades de RDF:

- Cabecera (Header): ofrece metainformación en RDF plano sobre los datos de la colección. Se orienta a la publicación y es opcional. Es el punto a través del cual el consumidor accede a la información, permitiéndole obtener una idea del contenido sin necesidad de recuperar todo el conjunto de datos.
- Diccionario (Dictionary): organiza los datos. Contiene todos los términos que se utilizan en el conjunto de datos, asignando a cada uno un identificador único (ID), lo cual facilita la compresión. Cada triple se puede representar como tuplas de tres IDs, evitando la representación repetida de términos largos. Reemplazar cada término por su ID necesita menos bits para codificar. Se orienta a la consulta y al intercambio.
- Triples (Triples): representan la estructura del grafo. Gracias al diccionario, el grafo RDF original se convierte en un grafo de IDs, lo cual optimiza su codificación y facilita la compresión. Se orienta a la consulta y al intercambio.

Una vez que los componentes codificados en HDT de un conjunto de datos se han cargado en la memoria de un sistema computacional, pueden ser accedidos directamente. No obstante, en función de las necesidades de las aplicaciones, se puede ajustar el volumen de los conjuntos. El conjunto de datos que se codifica en HDT se puede complementar con estructuras de datos más simples. Para ser consultados, los datos en HDT tienen que ser cargados en memoria, por lo que, si están en disco, hay que pasarlos previamente a memoria (mmap).

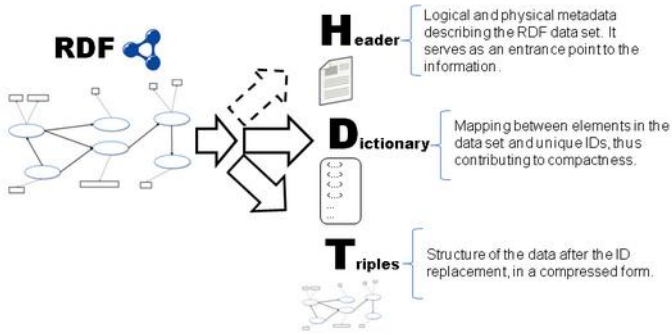


Figura 7: Componentes de HDT.

Los objetivos de HDT son:

- Compactibilidad, lo cual ahorra en almacenamiento, ancho de banda y tiempo, gracias a la ocupación de menos espacio.
- Favorecer la publicación y el intercambio.
- Permitir operaciones básicas mediante el acceso indexado a la demanda. De esta forma, el fichero puede ser accedido por partes sin necesidad de procesarlo completamente.
- Lograr mejores compresiones de RDF mejores que las conseguidas por los compresores universales.

El formato de HDT es abierto y está en proceso de estandarización y sus bibliotecas son una fuente abierta. La indexación del fichero HDT permite que se pueda acceder al mismo en poco tiempo. Debido a la compresión de HDT, se pueden mantener gran parte, o la totalidad, de la base de datos en la memoria principal, eliminando el cuello de botella que se forma en el acceso al disco. Esto proporciona un alto rendimiento en las consultas.

El acceso directo a una codificación en HDT se puede realizar cuando sus componentes se encuentran en cualquier parte de la jerarquía de memoria de un sistema de computación. En función del volumen de los datos o de la velocidad de recuperación que se necesite, se puede ajustar el lugar de la carga en memoria. Cuando toda la codificación HDT está en la memoria, añadiendo simples estructuras de datos se puede enriquecer la representación. Cuando la representación se encuentra en los niveles más altos de la memoria y es consultable de forma completa, se conoce como HDT-FoQ: HDT Focused on Querying (HDT Centrado en Consulta).

4.2 Componentes de HDT

Como ya se ha explicado, HDT está formado por tres componentes lógicos, que se describen a continuación.

4.2.1 Header (Cabecera)

En la cabecera se sitúan metadatos acerca de un conjunto de datos RDF. Se trata de un componente con gran flexibilidad donde se pueden dar las características que se quiera:

- Publicación de metadatos: son metadatos como el lugar de publicación, fechas de creación y modificación, versión utilizada del conjunto de datos, codificación, lenguaje o espacio de nombres, por ejemplo. Es habitual utilizar el Vocabulario Dublin Core para estas descripciones.
- Metadatos estadísticos: dan estadísticas como pueden ser número distinto de triples, sujetos, predicados u objetos. También puede proveer histogramas. Esta información es de utilidad para el software de visualización o los mecanismos que evalúan las consultas.
- Metadatos de formato: informan sobre la codificación del Diccionario y los Triples. De esta forma puede haber diferentes implementaciones de los mismos datos. Con estos datos, el consumidor es capaz de comprobar cómo se puede acceder a la estructura de datos de un conjunto codificado en HDT.
- Metadatos adicionales: donde se puede incluir información sobre el conjunto de datos o aplicación en concreto.

La cabecera está expresada en RDF plano y se recomienda utilizar VoID como vocabulario. VoID es un vocabulario con RDF Schema que ofrece términos y patrones para las descripciones de conjuntos de datos RDF, sirviendo de unión entre los publicadores y los consumidores.

La extensión de vocabulario que utiliza el espacio de nombres se conoce como *hdt*. El conjunto formado por todos los elementos lo da el vocabulario HDT.

En la cabecera, como mínimo tiene que haber un recurso del tipo hdt:Dataset, que se describe por los cuatro elementos de alto nivel, correspondientes a los tipos de metadatos descritos. Para gestionar el conjunto de datos se debe representar un formato de descripción de metadatos y, además, son necesarias las definiciones de hdt:dictionary y hdt:triples, o subpropiedades de ellos.

4.2.2 Diccionario

Como se explicó anteriormente, el conjunto de los términos utilizados en los triples forma el vocabulario. La finalidad del uso de diccionarios es la compresión de los datos asignando a cada elemento del mismo un ID único. La mayoría de los almacenes de RDF contienen diccionarios de índices, no obstante, ninguno implementa el diccionario de forma óptima, ya que usan dos estructuras independientes sin aportar una solución efectiva.

En un diccionario HDT, las cadenas representadas se identifican con un valor entero dado por una función biyectiva que permite su indexación, y dando lugar a dos operaciones de gran importancia:

- *locate(p):* cuya salida es el ID correspondiente a la cadena *p* si ésta existe en el diccionario, o -1 en caso contrario.
- *extract(i):* cuya salida es la cadena que ocupa la posición i-ésima en el diccionario, o NULL si no existe en el mismo.

El diccionario se divide en secciones dependiendo de qué términos realicen los roles de sujeto, predicado u objeto. No obstante, en datos semánticos es bastante común que un URI aparezca como sujeto en un triple y como objeto en otro. Para evitar repetir estos términos dos veces en las secciones de sujetos y de objetos, podemos extraerlos en una cuarta sección llamada Sujeto-Objeto compartidos.

La organización del diccionario comprimido tiene esto en cuenta. Por un lado, los términos del diccionario se organizan, dentro de una partición, considerando el rol que desempeñan en el conjunto de datos:

- Sujetos y objetos comunes (SO): donde se encuentran todos los términos que desempeñan simultáneamente los roles de sujeto y objeto. Con esto se consigue que estos términos aparezcan una única vez y se ahorre un espacio considerable. Estos términos se mapean en el rango [1, |SO|].
- Sujetos (S): donde están todos los términos que sólo desempeñan el rol de sujeto. Se mapean en el rango [|SO|+1, |SO|+|S|].
- Objetos (O): donde se localizan los términos que únicamente tienen el rol de objeto. Se mapean en el rango [|SO|+1, |SO|+|O].
- Predicados (P): mapea los predicados en [1, |P|].

Los términos se mapean en tres rangos de IDs, correspondientes a sujetos, objetos y predicados. Al contrario de lo que se pueda pensar, no hay ambigüedad para la función extract cuando un ID pertenece a más de un rango, puesto que el rol del término se conoce de antemano y se puede dar junto con su ID.

A su vez, cada partición se subdivide en función de las clases que puede tener (URIs, Bnodes o literales). En cada subdivisión, cada término se identifica en su subdiccionario. Dentro de las particiones SO y S se encuentran las subdivisiones para URIs y Bnodes. La partición O posee, además, una partición para literales, que a su vez tiene tres subdivisiones, siendo una para literales generales, otra para literales de idioma y, la última, para literales tipados. Las dos últimas subdivisiones están divididas para clasificar, respectivamente, los lenguajes y los tipos diferentes. La propia jerarquía permite eliminar la representación de las etiquetas, lo cual supone un ahorro de espacio. Por último, la partición P

únicamente contiene URIs. En todas las particiones los términos se almacenan en orden lexicográfico.

Esta organización hace necesaria una estructura de mapeo, llamada *ptrs*, para integrar todas las particiones de un diccionario. En cada registro del primer nivel se guardan un puntero a su clase correspondiente en el diccionario y el número de términos representados previamente. Se hace necesario un segundo nivel para representar las tres subparticiones en la parte de literales en la sección de objetos (O). Esta estructura es fija y cuenta con 8 celdas para el primer nivel y 3 para el segundo. Además, la estructura *ptrs*, almacena dos índices simples que apuntan al comienzo de cada lenguaje y tipo de datos utilizados en el diccionario. Se implementan mediante dos arrays ordenados por el lenguaje o el tipo de datos al que apuntan. Una representación del diccionario se puede ver en la siguiente figura:

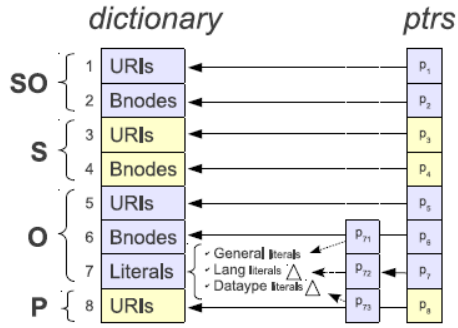


Figura 8: Organización de D_{comp} (Diccionario + *ptrs*)

Un objetivo del diccionario RDF es la optimización del espacio y del tiempo de respuesta de locate y extract. Las regularidades que posee el vocabulario hacen posible reducir el espacio. Por otra parte, el tiempo de consulta es dependiente de la eficiencia de locate y extract, las cuales tienen en cuenta qué rol desempeñan en los patrones de triples los términos y las variables.

Para representar el diccionario, HDT permite diferentes técnicas. Independientemente de las distintas formas que pueda tener el catálogo de términos, siempre deben estar implementadas las operaciones locate y extract.

La forma de codificación más simple es Front-Coding. Su técnica, basada en compresión diferencial, es simple y efectiva. En primer lugar, se ordena lexicográficamente cada sección, y se dividen en bloques de b términos. Dependiendo del valor de *b* son posibles distintos ratios espacio/tiempo. En cada bloque, el primer término se guarda de forma explícita y el resto se codifica con respecto a sus precedentes: primero se codifica el prefijo común y después se añade el sufijo. En cada sección los IDs se asignan de forma correlativa de 1 a n.

Para aquellos términos que puedan ser Sujeto y Objeto, se usan los IDs x, tales que x<m, siendo m el número de términos de la sección SO.

4.2.3 Triples

La codificación de los triples permite compactar los datos explotando la redundancia del grafo una vez organizada la información. Las operaciones de recuperación básicas se pueden llevar a cabo de una forma eficiente gracias a que la codificación se puede mapear de forma sencilla en una estructura de datos que lo permita.

La recuperación de RDF se lleva a cabo con patrones de triples, que son la forma más simple de consulta SPARQL. Aquellos triples que coincidan con una plantilla dada (s, p, o), donde cualquiera de los elementos pueda ser variable, se tienen que recuperar de forma directa de los triples codificados.

La codificación de triples propuesta por HDT se conoce como Bitmap Triples (BT). Esta técnica necesita que una ordenación previa de los triples siguiendo un determinado orden. BT soporta cualquier ordenación de triples, pero es recomendable el uso del orden intuitivo sujeto-predicado-objeto (SPO). Lo que hace BT es reemplazar los IDs de los triples conforme al diccionario, creando listas de adyacencia de predicados y objetos, para las cuales se utiliza, respectivamente, SP y SO, dos vectores de enteros. La delimitación se hace utilizando BP y BO, dos secuencias de bits que almacenan listas de cardinalidades. El grafo se separa en tantos árboles como sujetos diferentes haya en el conjunto de datos, y se ordenan por el ID del sujeto, donde el i-ésimo árbol corresponde al i-ésimo sujeto. Esto permite suprimir la representación de los sujetos, ahorrando espacio. Los árboles tienen 3 niveles. En la raíz se encuentran los IDs de los sujetos, en el segundo nivel están los IDs de los predicados y en las hojas están los IDs de los objetos. Los niveles de predicado y objeto también están ordenados y se codifican aislados.

El vector SP enlaza los predicados según la ordenación del árbol y la secuencia BP utiliza un bit por cada elemento de SP, donde el primer predicado de un determinado árbol se pone con un 1, siendo los predicados restantes codificados con bits 0. Los vectores SO y BO siguen una codificación análoga. Tanto SP como SO utilizan, respectivamente, log(|P|) y log(|O|) bits por elemento.

La estructura de triples que se obtiene del mapeo directo de la codificación hace que la recuperación basada en sujeto sea rápida, pero no ocurre así cuando se accede por predicado u objeto. El orden de BT se basa en sujeto, aunque se han creado índices adicionales al tiempo de carga para hacer posible la recuperación por predicado y objeto.

Las estructuras adicionales aseguran la recuperación por predicado y por objeto. El vector SP se carga como un árbol balanceado que reorganiza la secuencia de enteros en un rango [1; n] permitiendo la ejecución en tiempo logarítmico de algunas operaciones de acceso. Con el índice adicional O-índice se mejora el componente de triples, ya que resuelve el acceso por objeto. Su función es reunir las posiciones en las que aparece cada objeto en el SO original.

Esta infraestructura hace posible que se puedan resolver en los niveles más altos de la jerarquía de memoria y con un espacio comprimido algunos patrones de triples básico. Aunque la mayor parte de las consultas son básicas, SPARQL tiene el núcleo definido en base a Basic Graph Pattern (BGP), cuya semántica permite la construcción de conjunciones, disyunciones y partes opcionales donde puede haber varios patrones de triples. Para conseguir una cobertura total de SPARQL, HDT-FoQ tiene que proporcionar mecanismos que resuelvan consultas avanzadas.

4.2.4 Un ejemplo sencillo

A continuación se muestra un ejemplo del formato HDT. Se parte de un conjunto de triples que se pueden estructurar como un grafo.

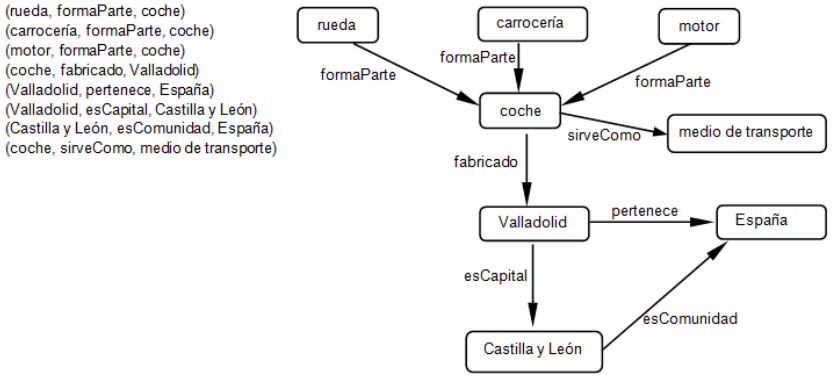


Figura 9: Grafo de triples

A continuación se crea el diccionario asignando un ID a cada término, en función del rol que desempeñe, y se codifican los triples sustituyendo los términos por sus IDs correspondientes, como se ve en la siguiente figura:

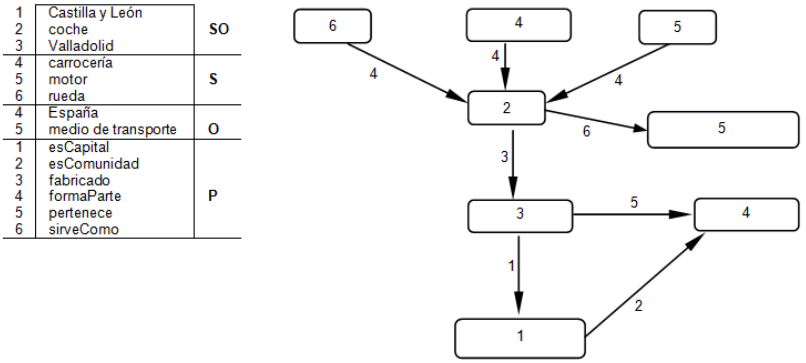


Figura 10: Diccionario y codificación de triples

Posteriormente, se crean las listas de adyacencia del grafo. En total, habrá tantas listas como sujetos tenga el grafo, estando representada cada lista como un árbol de tres niveles, donde el sujeto es la raíz, los predicados son los nodos intermedios y los objetos son las hojas. La navegación de las listas desde el sujeto a los objetos es como “dereferencear” el sujeto.

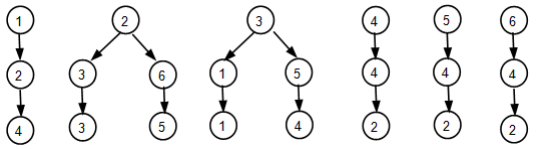
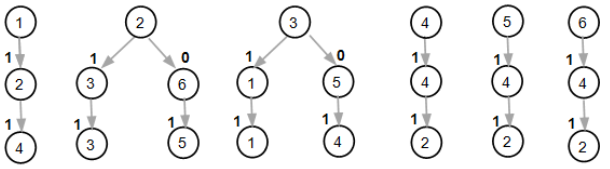


Figura 10: Listas de adyacencia del grafo.

Finalmente se obtiene una representación sencilla, con dos secuencias de bits y dos secuencias de enteros. La obtención de las secuencias de bits se hace conforme a la explicación en el apartado de Triples. Quedaría como se ve en la imagen:



BP	1	1	0	1	0	1	1	1	Secuencia de bits
SP	2	3	6	1	5	4	4	4	Secuencia de enteros
BO	1	1	1	1	1	1	1	1	Secuencia de bits
SO	4	3	5	1	4	2	2	2	Secuencia de enteros

Figura11: Representación final de los triples

4.3 Resolución de SPARQL

Los diccionarios, como se explicó anteriormente, tienen como fin comprimir los datos. Este hecho es un problema a la hora de consultarlo pues, si se quiere optimizar el espacio en SPARQL, es necesario que haya índices de diccionario junto con índices de evaluación e históricos. [12]

Existe una estrecha relación entre los diccionarios RDF y la resolución de SPARQL, puesto que los patrones de triples se consideran consultas atómicas usadas para construir otras más complejas. El funcionamiento de SPARQL es el siguiente: primero se localiza el ID correspondiente a cada uno de los términos que aparecen en los patrones de triples, después se realiza la consulta transformada y las variables que aparecen se ligan a los IDs resultantes. El resultado final se obtiene cuando se extraen los términos asociados a dichos IDs. En este proceso se utiliza *extract* para cada variable de la consulta, mientras que *locate* solo se utiliza para las constantes, lo que hace que *extract* tenga un mayor uso y necesite ser optimizado.

4.4 Usos donde destaca HDT

Los principales casos donde HDT toma relevancia son:

- Compartición de datos en RDF en la Web gracias que el conjunto de datos es compacto y puede ser utilizado directamente.
- Tras descargar el HDT se lanza el software de HDT SPARQL Endpoint y se crea un reflejo de un endpoint.
- Se pueden realizar consultas federadas. Las bases de datos a las que más se accede se pueden mantener en caché local, sin grandes costes de tiempo y espacio.
- La realización de estudios estadísticos, minería de datos o clustering son más eficientes gracias al emplazamiento local de los datos.
- Desarrollo de aplicaciones semánticas que permiten un uso eficiente de recursos con poca capacidad de procesamiento y de almacenamiento, como dispositivos móviles o embebidos.

El equipo del Proyecto RDF/HDT ofrece distintas herramientas para tratar HDT:

- Bibliotecas HDT en C++ y Java, además de herramientas que permiten la creación y la búsqueda de archivos HDT desde la línea de comandos.
- Jena Wrapper como proveedor de un Modelo Jena de un fichero HDT concreto, lo cual hace que se pueda usar Jena ARQ para hacer consultas

SPARQL o establecer un endpoint Fuseki SPARQL donde la especificación de los ficheros HDT se da como grafos nombrados.
- Una GUI llamada HDT-it! que posibilita la creación y la navegación de ficheros HDT, dando una vista en 3D de la matriz de adyacencia correspondiente al grafo RDF.
- Un servicio Web Online para la conversión de ficheros de RDF a HDT.

4.5 La herramienta HDT-it

Debido a la gran acogida del formato RDF, cada vez hay disponibles más conjuntos de datos públicos en dicho formato. Estos datos generalmente se consultan desde endpoints de SPARQL, pero puede que se necesite consultarlos de forma local, para no colapsar el servidor o la red.

Los ficheros que se descargan los usuarios, habitualmente están en formato NTRIPLES o RDF/XML y se comprimen con mecanismos genéricos como GZIP o BZIP2. Para procesar el fichero y poder consultarlo o navegarlo, es necesario descomprimirlo, parsearlo y cargarlo en un almacén de RDF con los índices apropiados. Utilizando el formato HDT, no es necesario realizar las acciones anteriores, ya que los datos pueden ser directamente utilizados en forma comprimida. Con la herramienta HDT-it [2], se pueden generar ficheros HDT desde serializaciones RDF y consultar los datos directamente de la representación comprimida. También permite obtener una matriz de adyacencia del grafo RDF, lo cual sirve para comprender la estructura general del conjunto.

Actualmente, la herramienta está disponible para los siguientes sistemas operativos:

- Windows (32 y 64 bits)
- Mac OS X
- Linux (i386 y x86_64)

4.5.1 Motivación

Con HDT se consigue representar de forma compacta datos en RDF, lo cual facilita la gestión de los mismos. Como ya se vio anteriormente, HDT distingue tres componentes: cabecera, diccionario y triples.

Utilizando el formato HDT, el tiempo de descarga de los datos se ve reducido, y el fichero descargado puede utilizarse directamente con la herramienta HDT-it, sin nigún tipo de procesamiento previo.

En la interfaz de la herramienta se pueden distinguir dos partes: a la izquierda aparecen tres campos editables, correspondientes a sujeto, objeto y predicado; y debajo de ellos, aparece un grupo de pestañas. En la parte derecha, aparece otro grupo de pestañas. Cuando los datos se cargan en la herramienta, se muestra en la parte izquierda de la misma toda la información disponible de los datos en la pestaña de “Info”. Entre la información mostrada aparecen el tamaño de los conjuntos original y reducido. Entre las pestañas de la derecha está la pestaña “Metadata”, en la que se puede navegar por los metadatos del conjunto en formato RDF plano.

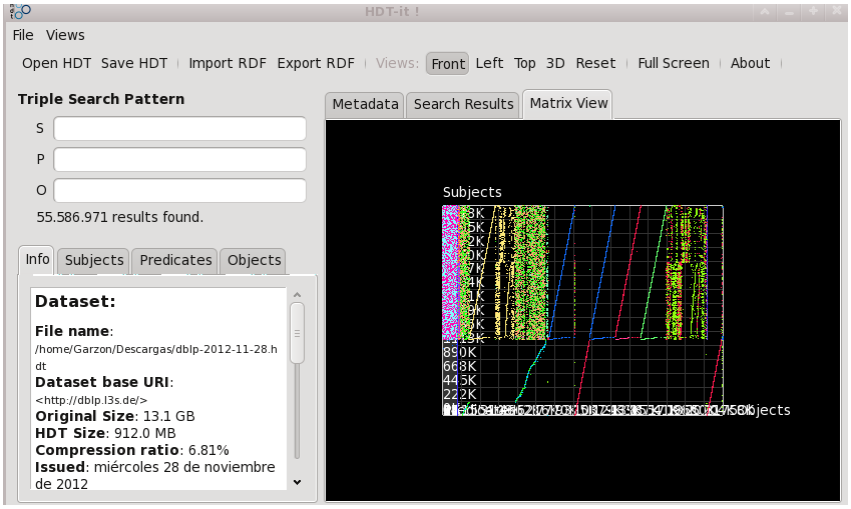


Figura 11: Interfaz de HDT-it

Para buscar un patrón cualquiera, el usuario tiene que rellenar los cuadros de texto de la izquierda. Según se van editando, aparecen sugerencias, lo cual se debe al uso de prefijos en el diccionario, que facilitan las búsquedas. En el grupo de pestañas de la derecha se encuentra la correspondiente a “Search Results”, en la cual se pueden ver, en una lista plana, los triples obtenidos de la consulta. En la pestaña “Matrix View”, se puede ver la matriz de adyacencia de los triples. Los resultados pueden ser exportados en los formatos RDF o HDT.

En el grupo de pestañas de la izquierda, se puede navegar entre las listas de “Subjects”, “Objects” y “Predicates”, donde se pueden ver, respectivamente, los distintos sujetos, objetos y predicados. Haciendo doble clic sobre un elemento cualquiera de estas listas, éste se añade de forma automática al patrón de búsqueda, apareciendo en el cuadro de texto correspondiente, lo cual hace que se ejecute la nueva consulta con los detalles de ese elemento concreto.

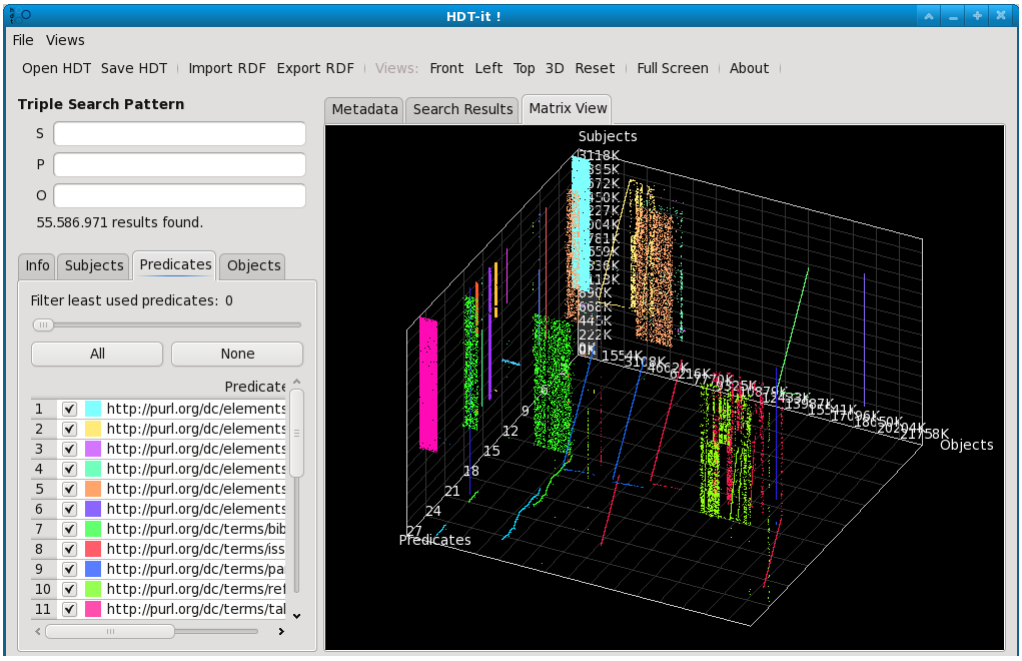


Figura 12: Visualización de los predicados

Para poder realizar una búsqueda con “regex”, es necesario que el diccionario sea de tipo literal. En ese caso, al abrir un fichero HDT se mostrará a la derecha de la pestaña “Matrix View” una pestaña con “Substring Search”, donde aparece un cuadro de texto para incluir la expresión regular a buscar y la lista de resultados se muestra en la misma pestaña, en forma de tabla.

4.5.2 Visualización de RDF

La herramienta HDT-it también permite visualizar la matriz de adyacencia de triples en 3D, seleccionando la pestaña “Matrix View”. Cada triple o identificador, se representa como una coordenada en el espacio y se renderiza en un gráfico de dispersión. Con esta visualización, se puede observar la forma en que se distribuyen los datos RDF, así como su complejidad.

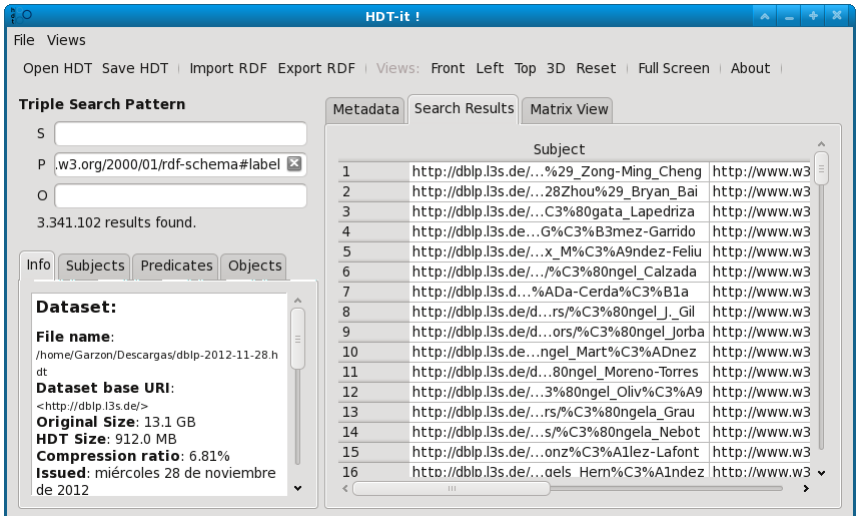


Figura 13: Visualización de la lista de resultados

La matriz se puede obtener directamente del fichero HDT y se puede interactuar con ella, haciendo rotaciones, zoom o desplazamientos. A cada predicado se le asigna un color distinto, lo cual facilita su identificación. Poniendo el cursor sobre un punto cualquiera de la representación, se muestra información sobre el triple concreto al que hace referencia. De esta forma, se pueden identificar los tipos de datos que hay en cada área del conjunto. Haciendo clic en un punto, el predicado actual se selecciona y el resto pasa a color gris, con lo que un usuario puede navegarlo de forma separada.

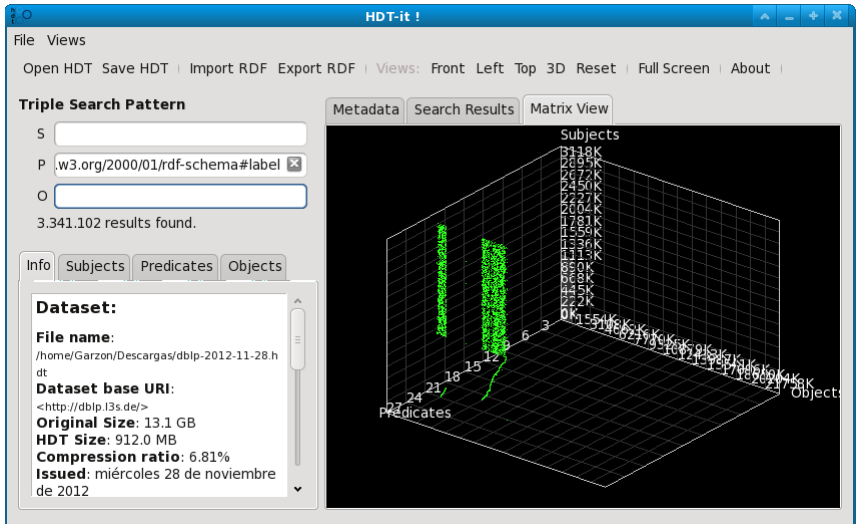


Figura 14: Visualización de la matriz de resultados

Una demostración de uso de HDT-it se puede ver en “Demo *of the HDT-it! application to browse big RDF datasets*”. [42]

5. HDT-it y Bio2RDF: otra forma de consultar datos biológicos

5.1 Planteamiento

En este capítulo se describe el trabajo realizado, propiamente dicho. El objetivo principal es modificar la herramienta HDT-it para lograr una mayor utilidad para consultas más complejas en datos biológicos.

La planificación del trabajo queda resumida en la siguiente tabla:

Fecha inicio	Fecha fin	Actividad	Descripción actividad
20/02/2012	20/02/2013	Asignación TFG	Primera reunión con los tutores para obtener una visión general del TFG.
05/03/2012	08/03/2012	Curso de introducción a la Web de Datos	Curso de 10 horas impartido por los tutores para introducir los conceptos generales relacionados con el TFG.
09/03/2012	12/08/2012	Actividades académicas	Paro en la realización del TFG para realizar otras actividades del curso y prácticas en empresa.
13/08/2012	30/11/2012	Recopilación de documentación y análisis	Búsqueda de información relacionada con los temas del TFG. Comprensión de los conceptos clave. Reuniones con los tutores para aclarar dudas.
01/12/2012	08/01/2013	Trabajo y vacaciones	Paro en la realización del TFG por trabajo y vacaciones de Navidad.
09/01/2013	28/02/2013	Análisis de log	Análisis del log de consultas de Bio2RDF y realización de estadísticas.
01/03/2013	02/03/2013	Obtención de requisitos	Obtener los requisitos para modificar la aplicación, en función de los resultados del análisis del log
03/03/2013	31/07/2013	Modificación de la aplicación	Toma de contacto con el editor QtCreator. Análisis del código existente. Modificaciones y pruebas del código.
01/08/2013	31/08/2013	Redacción de la memoria	Redacción de la memoria del TFG.

Las partes de impresión y encuadernación de la memoria, y de presentación final del trabajo, no se incluyen, puesto que son posteriores a la fecha de redacción.

Para realizar la modificación que se propone de la herramienta HDT-it, lo primero ha sido que se ha realizado ha sido un análisis sobre un log de consultas a Bio2RDF. Tras obtener datos estadísticos sobre los patrones de búsqueda e interpretarlos, se ha buscado la forma de trasladar esas conclusiones a la aplicación, para facilitar las búsquedas.

5.2 Estadísticas en las consultas a Bio2RDF

Para el estudio actual se cuenta con un log de consultas en SPARQL a la base de datos de BIO2RDF. En total, en el log utilizado aparecen 1558744

consultas, de las cuales 1399547 son duplicadas, lo que da lugar a 159197 consultas efectivas. Se han realizado estadísticas sobre las consultas a Bio2RDF para poder cuantificar diversos aspectos, como tipo de patrones, número de patrones por consulta o tipos de consulta, entre otros, y conseguir con ello acotar el tipo de consulta más común.

Las estadísticas se han realizado siguiendo el procedimiento explicado en [1], y utilizando tanto el script cedido por sus autores como las funcionalidades que ofrece Microsoft Excel.

5.2.1 Uso de patrones de triples

Cuando se hace una consulta, se puede preguntar por el sujeto, el predicado, el objeto o las distintas combinaciones de los 3 elementos. Eso da lugar a distintos patrones en la estructura de las consultas. SPARQL proporciona medios para comparar grafos especificando el patrón, donde se puede fijar un valor (C), o dejarlo como variable (V), en el triple. En la siguiente tabla se muestra el porcentaje de uso de cada uno de los patrones posibles en las consultas a BIO2RDF:

Uso de patrones de triples	
Type	Bio2RDF
C C C:	27,995
C V V:	27,514
V C V:	14,325
V C C:	13,196
C C V:	9,619
V V V:	6,767
V V C:	0,524
C V C:	0,06

Como se puede observar, más de una cuarta parte de las consultas fijan el valor para los tres elementos del triple, y un porcentaje muy similar sólo fija el valor del sujeto. El segundo patrón tiene más sentido, puesto que se consulta sobre todo lo relacionado con algo concreto, mientras que el primer patrón es una consulta mucho más específica y que no aporta información, ya que todos los valores se conocen de antemano.

5.2.2 Tipo de consulta

Como ya se vio en el capítulo 2, el lenguaje de consulta SPARQL permite cuatro tipos distintos de consulta:

- SELECT: que devuelve los valores asociados a las variables por las que se consulta.
- CONSTRUCT: devuelve un grafo RDF en el que las variables de la consulta se sustituyen por la solución.
- ASK: devuelve un valor lógico acerca de si la consulta tiene o no solución, sin más información.
- DESCRIBE: devuelve un grafo RDF con información de cada uno de los recursos que se identifican en cada consulta

El tipo de consultas más habituales es SELECT, seguidas de las de tipo CONSTRUCT. La suma de las consultas de tipo ASK y DESCRIBE, apenas es un 0,3% del total de las consultas.

Tipo de consulta	
Consulta	Bio2RDF
SELECT:	67,917
CONSTRUCT:	31,762
ASK:	0,311
DESCRIBE:	0,01

Que la mayor parte de las consultas sea de tipo SELECT implica que lo que se desea, generalmente, es conocer los valores asociados a las variables, es decir, información concreta sobre elementos específicos, más que cuestiones generales sobre algo o la creación de nuevos grafos.

5.2.3 Funciones

Cuando se va a realizar una consulta, se pueden utilizar funciones para conseguir los resultados deseados. Por ejemplo, se pueden filtrar los resultados en función de alguna expresión dada, o se puede consultar sobre la unión o la disyunción de varios triples. Como se ve en la siguiente tabla, lo habitual en las consultas a BIO2RDF es que no haya ninguna función.

Funciones	
Patrón	Bio2RDF
:	78,117
AND:	13,061
FILTER:	7,397
AND FILTER:	0,823
OPTIONAL:	0,285
OPTIONAL UNION:	0,129
UNION:	0,059
AND FILTER UNION:	0,034
OPTIONAL FILTER:	0,029
AND OPTIONAL FILTER:	0,023
AND OPTIONAL:	0,018
AND UNION:	0,014
FILTER UNION:	0,007
OPTIONAL FILTER UNION:	0,004

5.2.4 Usos de la función FILTER

SPARQL permite restringir los resultados de las búsquedas a soluciones que contengan filtros cuyas expresiones se evalúen como verdaderas. Utilizando FILTER se pueden eliminar soluciones que no se correspondan con la expresión dada. A continuación se muestra el porcentaje de uso de las distintas funciones utilizadas en las expresiones de la función FILTER en las consultas a BIO2RDF:

Usos de la función FILTER	
Función	Bio2RDF
not:	24,106
isBlank:	21,137
eq:	15,591
or:	10,779
isURI:	10,674
regex:	7,361
and:	3,38
str:	3,353
isIRI:	1,594
ne:	1,196
gt:	0,401
lang:	0,124
langMatches:	0,111
lt:	0,092
http://www.w3.org/2001/XMLSchema#string:	0,027
bound:	0,014
isLiteral:	0,014
http://www.w3.org/2005/xpath-functions#starts-with:	0,014
bif:contains:	0,008
http://www.w3.org/2001/XMLSchema#integer:	0,005
contains:	0,005
http://www.w3.org/TR/xpath-functions/#contains:	0,005
sameTerm:	0,003
datatype:	0,003
http://www.w3.org/TR/xpath-functions/#func-contains:	0,003

5.2.5 Tipos de REGEX

Las expresiones dadas en FILTER pueden buscar una expresión regular para alguno de los términos de la consulta, para ello se utilizan distintos tipos de funciones, las más utilizadas en las consultas a BIO2RDF se muestran en la tabla siguiente:

Tipos de REGEX	
Tipo	Bio2RDF
substring:	57,941
regex:	40,257
prefix:	1,434
exact:	0,294
empty:	0,074

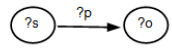
5.2.6 Tipo de grafo

Los patrones de los grafos de las consultas tienen normalmente forma de estrella y, generalmente son pequeños. También pueden tener cadenas. Para caracterizar los patrones, se serializa el grado de salidas de cada nodo, en orden decreciente. Las estrellas cuentan con un nodo central con un gran número de salidas hacia nodos hoja, sin salidas. Como se ve en la tabla siguiente, el patrón más utilizado es el más simple, con un único triple.

Tipo de grafo	
Grafo	Bio2RDF
1 0:	85,187
1 1 1 0 :	6,545
2 0 0:	4,078
1 1 0:	2,688
3 0 0 0 :	0,311
2 1 0 0 :	0,201
2 1 1 0 0 :	0,151
3 1 0 0 0 :	0,067
2 2 0 0 0 :	0,059
7 0 0 0 0 0 0 0 :	0,056
3 1 1 0 0 0 :	0,053
4 0 0 0 0 :	0,047
5 0 0 0 0 0 :	0,04
1 1 0 0 :	0,039
3 2 0 0 0 0 :	0,031
2 2 0 0 0 0 :	0,026
3 3 1 1 1 1 1 0 0 0 0 0 :	0,026
6 0 0 0 0 0 0 :	0,021
8 0 0 0 0 0 0 0 0 :	0,018
3 1 0 0 :	0,017
13 0 0 0 0 0 0 0 0 0 0 0 0 0 :	0,016
1 1 1 0 0 :	0,014
2 1 1 1 0 0 :	0,014
3 2 1 0 0 0 0 :	0,014
3 1 1 1 0 0 0 :	0,013
2 1 0 0 0 :	0,012
14 0 0 0 0 0 0 0 0 0 0 0 0 0 0 :	0,011
Otros patrones (suma)	0,231

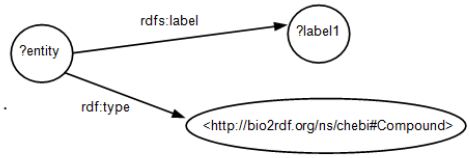
A continuación, se muestran algunos ejemplos de tipos de grafos de consultas a Bio2RDF. Como se puede ver en ellos, el tipo de grafo se obtiene, como se indicó anteriormente, contando el número de arcos que salen de cada nodo, y colocándolos en serie, de mayor a menor.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
select *
where {
          ?s         ?p         ?o .
          ?o bif:contains "Plasmodium" .
} OFFSET 4600 LIMIT 50
```



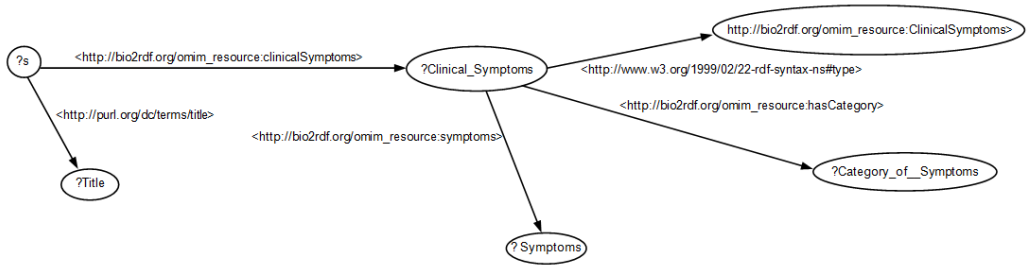
Ejemplo de consulta tipo 1 0

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?entity ?label1
WHERE {
          ?entity    rdfs:label           ?label1 .
          FILTER regex(str(?label1), "Emodin", "i")
          ?entity    rdf:type   <http://bio2rdf.org/ns/chebi#Compound> .
          FILTER isIRI(?entity)
} LIMIT 3
```



Ejemplo de consulta tipo 2 0 0

```
select ?Clinical_Symptoms ?Symptoms ?Category_of_Symptoms ?Title
where {
 ?s <http://bio2rdf.org/omim_resource:clinicalSymptoms> ?Clinical_Symptoms .
 ?Clinical_Symptoms <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://bio2rdf.org/omim_resource:ClinicalSymptoms> .
 ?Clinical_Symptoms <http://bio2rdf.org/omim_resource:symptoms> ?Symptoms .
 ?Clinical_Symptoms <http://bio2rdf.org/omim_resource:hasCategory> ?Category_of_Symptoms .
 ?s <http://purl.org/dc/terms/title> ?Title .
 ?Title bif:contains "'cancer'" .
} Limit 10
```



Ejemplo de consulta tipo 3 2 0 0 0 0

Figura 15: Ejemplos de patrones de consulta

5.2.7 Número de patrones de triples

Las consultas en SPARQL pueden hacer referencia a varios triples a la vez, relacionados o no entre sí, dando lugar a distintos tipos de grafo, como se explicó en el punto anterior. Como se ve en la siguiente tabla, alrededor del 90% de las consultas a BIO2RDF se refieren a un único triple, y solo un porcentaje muy bajo de las consultas engloban más de 3 triples.

Número de patrones de triples	
Triples	Bio2RDF
1	90,018
2	4,58
3	4,765
4	0,251
5	0,095
6	0,07
7	0,065
8	0,029
9	0,022
10	0,018
11	0,024
12	0,003
13	0,017
14	0,009
15	0,006
16	0,001

5.2.8 Camino más largo por consulta.

Relacionado con el número de triples o el tipo de grafo, está el camino más largo por consulta, que hace referencia a la longitud de la cadena formada por los triples. Como se observa en la tabla siguiente, la consulta más larga suele ser de longitud uno, algo que concuerda con el dato de que la mayor parte de las consultas únicamente tienen un triple.

Camino más largo por consulta	
Longitud	Bio2RDF
0	0,013
1	89,931
2	3,196
3	6,746
4	0,039
5	0,05
6	0,005
7	0,001

A continuación se muestra un ejemplo de consultas con caminos de diferentes longitudes.

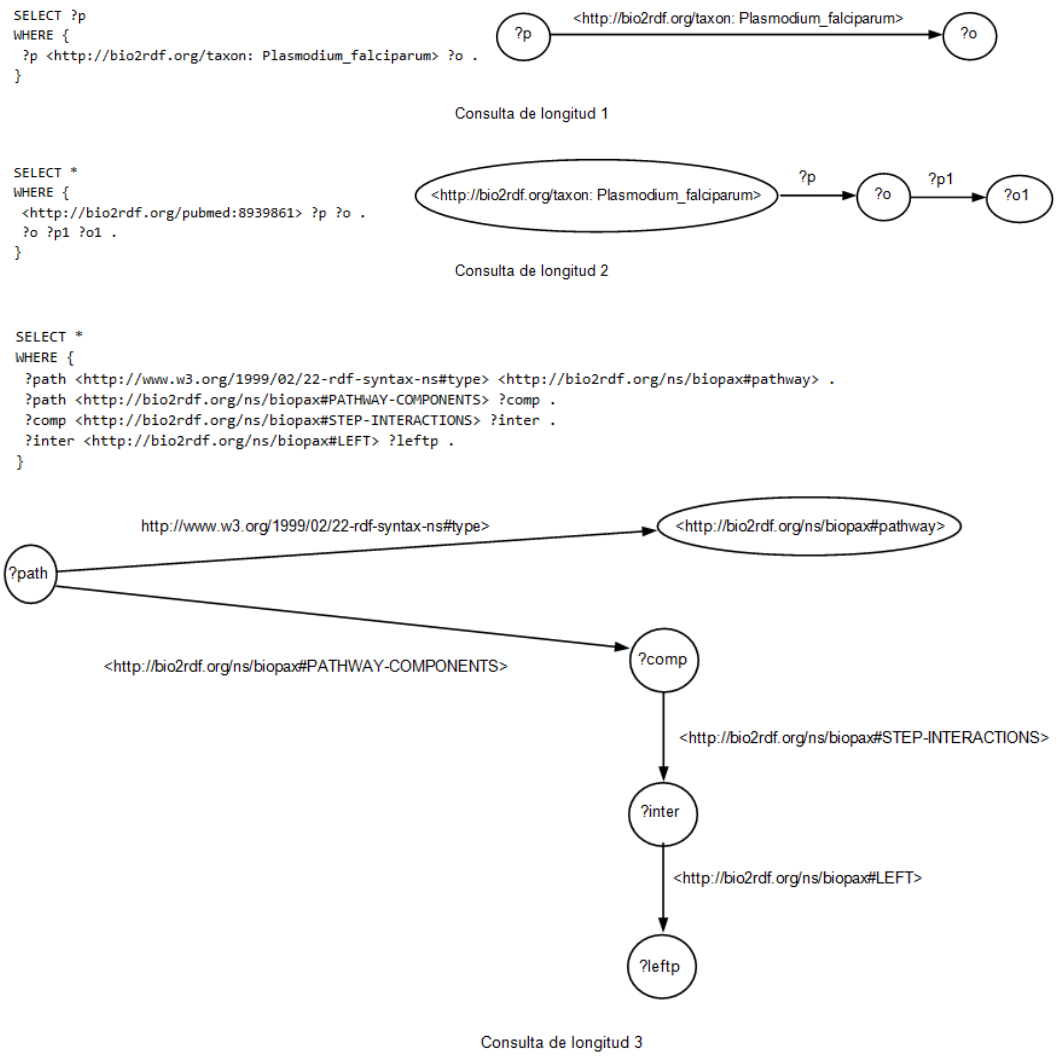


Figura 16: Ejemplo de consultas de distinta longitud

5.2.9 Número de prefijos declarados por consulta

El uso de prefijos en las consultas de SPARQL permite reducir la escritura de partes comunes en varios elementos de los triples que forman las mismas. Se declaran utilizando la palabra PREFIX, que asocia una etiqueta a una IRI. Dicha etiqueta seguida de una parte local, separadas por dos puntos (:), forman un nombre con prefijo, que se mapea con una IRI concatenando la IRI asociada y la parte local. Tanto la etiqueta como la parte local pueden estar vacías.

En la siguiente tabla se puede observar el porcentaje de prefijos declarados por consulta:

Prefijos por consulta	
Prefijos por consulta	Bio2RDF
0	92,047
1	1,163
2	6,411
3	0,068
4	0,039
5	0,044
6	0,029
7	0,031
8	0,112
9	0,013
10	0,005
11	0,003
12	0
13	0,005
14	0,01

Puesto que, como se vio anteriormente, las consultas más habituales son aquellas que sólo incluyen un triple, la mayor parte de las consultas no declaran prefijos. Observando las consultas, se aprecia que en algunas se declaran prefijos que no se utilizan.

A continuación se muestra una consulta con prefijos:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?entity ?label1
WHERE {
 ?entity rdfs:label ?label1 .
 FILTER regex(str(?label1), "13-cis-Retinoic Acid", "i")
 ?entity rdf:type <http://bio2rdf.org/ns/chebi#Compound> .
 FILTER isIRI(?entity)
} LIMIT 3
```

5.2.10 Tipos de Joins

SPARQL busca optimizar la planificación de las consultas de tipo join. Una consulta de tipo join es la unión de dos patrones de triples donde existe una variable común. Teniendo en cuenta la propiedad conmutativa y la posición de la variable común, hay 6 combinaciones posibles: Sujeto-Sujeto (SS), Predicado-Predicado (PP), Objeto-Objeto (OO), Sujeto-Predicado (SP), Sujeto-Objeto (SO) y Predicado-Objeto (PO).

En la tabla se muestra el porcentaje de cada uno de los tipos de join presentes en las consultas a BIO2RDF:

Tipos de join	
Join Type	Bio2RDF
SO	62,601
SS	31,501
SS SO	5,342
OO	0,205
SS OO SO	0,182

Como se ve, más de la mitad de las consultas son del tipo SO, es decir, se pregunta sobre algo que desempeña los roles de sujeto y de objeto de forma simultánea en el conjunto de datos. De esta forma se consigue gran cantidad de información sobre algo concreto.

A continuación se muestra el porcentaje del número de joins de cada tipo por consulta. Como se ve, la mayor parte de las consultas no tienen ningún tipo de join, lo cual se justifica con el hecho de que la mayor parte de las consultas sólo incluyen un triple.

SS Join por consulta	
SS Join	Bio2RDF
0	94,53
1	4,456
2	0,576
3	0,127
4	0,083
5	0,036
6	0,071
7	0,025
8	0,02
9	0,011
10	0,008
11	0,01
12	0,016
13	0,01
14	0
15	0
16	0,001

SO Join por consulta	
SO Join	Bio2RDF
0	89,975
1	3,212
2	6,685
3	0,047
4	0,029
5	0,005
6	0,027
7	0
8	0,001

OO Join por consulta	
OO Join	Bio2RDF
0	99,911
1	0,052
2	0,011
3	0,002
4	0,001
5	0
6	0
7	0
8	0
9	0,004

5.2.11 Sobre las consultas

Analizando un poco más a fondo las consultas, se pueden apreciar diversas cosas. En primer lugar, es relevante el número de consultas que se repiten, siendo las consultas únicas aproximadamente un 10% del total. Hay un elevado número de consultas que se repiten de manera consecutiva, lo cual favorece el uso de cachés, y hace que obtener los datos consultados continuamente sea un proceso más rápido. Algunos ejemplos de esto son los siguientes:

- `ASK WHERE { ?s ?p ?o .}`: Esta consulta se repite en 106101 ocasiones, lo que supone un 6,8% de las consultas originales, y, como se puede ver, no aporta ninguna información, ya que pregunta si hay algo que se pueda consultar, algo carente de sentido, ya que cuando se accede a una fuente de información, es esperable encontrar algo.

- ```
  SELECT *
  WHERE {
      ?chebiDrug <http://purl.org/dc/elements/1.1/title> "Cetuximab" .
      ?chebiDrug <http://bio2rdf.org/ns/bio2rdf#image> ?chebiImage
  }
  ```
  : Esta consulta se repite 40009 veces, suponiendo un 2,57% del total.

Hay también, patrones que se repiten habitualmente, en el que sólo cambian los valores de consulta. Un ejemplo es el que sigue:
  ```

```
SELECT *
WHERE {
<http://bio2rdf.org/dr:α><http://bio2rdf.org/ns/bio2rdf#xRef><http://bio2rdf.
org/cas: β>
}
```

Donde los símbolos α y β son valores que cambian en cada consulta. Este patrón se repite 173494 en el log original, lo que supone un 11,13% del original. Eliminando las consultas duplicadas, las veces que este patrón aparece son 51553, lo que supone un 23,29% del total.

En cuanto al contenido y la sintaxis, cabe destacar que hay varias conjuntos de consultas que se repiten con pequeñas variaciones sintácticas, como sustituir los espacios entre palabras por guión bajo o por punto, como por ejemplo: “palabra1 palabra 2”, “palabra1_palabra2” ó “palabra1.palabra2”. Otros casos similares son los ocasionados por la colocación de los paréntesis o llaves u otros elementos sintácticos, que en algunas consultas es errónea, y por ello se prueba con otra colocación. En otras ocasiones, se producen fallos sintácticos en la consulta, sobre todo en las llaves de apertura y cierre, pero no se prueba de nuevo, sino que se repite la misma consulta pero con otro argumento, por lo que vuelve a dar error. Esto se puede considerar como casos de prueba-error de la sintaxis de las consultas, que se va probando hasta conseguir un resultado satisfactorio por parte del usuario.

Por otro lado, es habitual que en las consultas en estrella se repita el sujeto, en lugar de omitirlo a partir del segundo triple, es decir:

```
?s  propiedad1  ?objeto1 .
?s  propiedad2  ?objeto2 .
```

es equivalente a:

```
?s   propiedad1  ?objeto1 ;
     propiedad2  ?objeto2 .
```

lo cual permite ahorrar tiempo de escritura en la consulta.

Para buscar en un texto expresiones dadas existen dos operadores: REGEX y bif:contains. El operador REGEX busca texto que coincida con un patrón dado. Por otro lado, bif:contains también busca palabras o frases, incluso aunque se usen caracteres no canónicos de Unicode, y puede ser utilizado en documentos XML o HTML. Para una misma consulta, el uso de un operador puede no obtener el mismo resultado que si se usase el otro. Bif:contains utiliza un índice de palabras para resolver las consultas en vez de uno de caracteres y eso hace que

los resultados de ambos no sean exactamente los mismos. Por ejemplo, si el patrón que se pasa para REGEX son dos palabras separadas por un espacio, buscará ese patrón exacto, mientras que si se utiliza bif:contains, buscará esas palabras en cualquier orden. REGEX puede buscar también por subcadenas.

Se ha observado que en las consultas a BIO2RDF, lo habitual es buscar por palabras o frases completas, no por subcadenas de palabras. Destaca que el uso de estas funciones se da en un 5,81% de las consultas, usándose FILTER en el 7,397%.

En algunos casos, las consultas utilizan la función regex para filtrar el sujeto. Esto no lo contempla el estándar de SPARQL, ya que regex sólo puede ser utilizado para literales y el sujeto no puede serlo, pero puesto que el log de consultas proviene de Virtuoso, su adaptación hace que sea posible. Lo que carece de sentido es aplicar un filtro para evitar que el sujeto sea un literal, algo que de por sí no es posible.

Hay varias consultas que presentan errores cuando se filtran por REGEX que se solucionan si se filtran por bif:contains y viceversa. El “argumento” de bif:contains no puede llevar signos de puntuación si no va entre comillas dobles y simples “’a.sí’”. En otros casos, puede llevar “comillas dobles” o ‘comillas simples’. La longitud de la cadena depende del endpoint que se utilice.

En total, hay unos 50 conjuntos de datos que forman parte de BIO2RDF. Casi todos los conjuntos de datos tienen un endpoint de SPARQL, provisto por Virtuoso, al que se accede con “nombredateset.Bio2RDF.org/sparql”. Una misma consulta puede tener distintos resultados dependiendo del endpoint utilizado. Además, cada endpoint puede admitir consultas que otros consideran erróneas. Tampoco es igual el tiempo de respuesta, una consulta válida o errónea para un endpoint puede exceder el tiempo de respuesta en otro.

5.3 Modificaciones sobre HDT-it

Con este trabajo, se pretende incluir en HDT-it la posibilidad de búsquedas que impliquen dos un join entre dos patrones. Como se vio anteriormente, en el caso del log de consultas a Bio2RDF analizado, se encuentra que de las consultas con join, aproximadamente un 15%, se tiene que un 62,6% es de tipo SO y un 31,5% es de tipo SS. En base a las estadísticas de las consultas a Bio2RDF, el segundo tipo de consulta más utilizado es el que incluye un join entre dos patrones. Pero esto no es sólo útil para este conjunto de datos. La posibilidad de buscar la conjunción de patrones por una variable común aporta un ahorro de tiempo en las búsquedas, y la interfaz gráfica facilita la escritura de la

consulta, ya que no es necesario conocer la sintaxis propia de SPARQL para realizar una consulta compleja, con lo cual también se evitan las consultas erróneas debido a los intentos de prueba y error de las consultas con fallos sintácticos.

5.3.1 Requisitos

Con las conclusiones obtenidas del análisis del log de consulta los principales requisitos que tiene que cumplir la nueva versión de HDT-it son:

- Incorporar la posibilidad de búsqueda con join.
- Mantener la funcionalidad de la versión anterior.

5.3.2 Desarrollo

Puesto que se parte de una aplicación ya desarrollada, no ha sido necesario realizar un proceso de desarrollo de software de la forma habitual. En función de las nuevas necesidades, se han analizado las clases ya existentes para incluir la nueva funcionalidad. El desarrollo completo está explicado dentro del proyecto *rdfhdt* [38] La modificación se ha llevado a cabo sobre la versión de HDT-it para distribuciones Linux de 64 bits.

Tras el estudio de la aplicación, se concluye que sigue un patrón de Modelo-Vista-Controlador. Entre las clases más importantes, en la parte de la vista se incluyen HDTit y MatrixViewWidget. En ellas se codifica la interfaz de la aplicación, lo que ve el usuario. La clase más importante del controlador es HDTController, donde se implementa la gestión de la vista y las principales relaciones con la lógica del modelo. En la parte del modelo, se incluyen las clases que definen HDT, como son las clases en las que se definen el propio HDT, los diccionarios o los triples, por ejemplo. También se incluyen en este paquete las clases que modelan la presentación del los resultados y de las estructuras.

A continuación, se muestra una simplificación del diagrama de clases de la aplicación, donde se incluyen las más relevantes, desde el punto de vista de la modificación realizada:

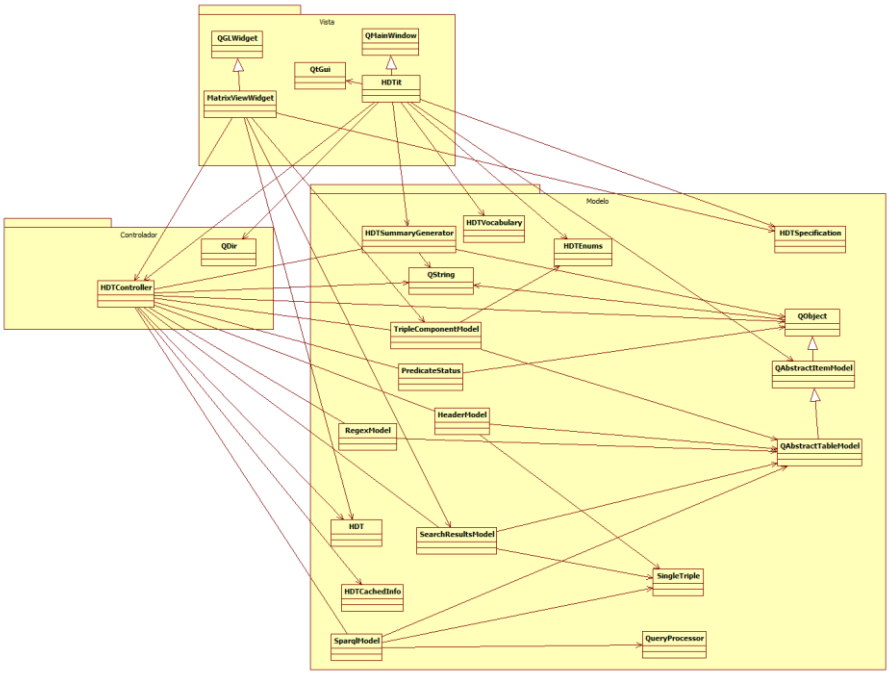


Figura 17: Diagrama de las principales clases

Las modificaciones llevadas a cabo en la aplicación, han supuesto la modificación de las clases HDTit, HDTController y SparqlModel, las cuales se muestran a continuación, incluyendo las nuevas características y las relaciones existentes entre ellas:

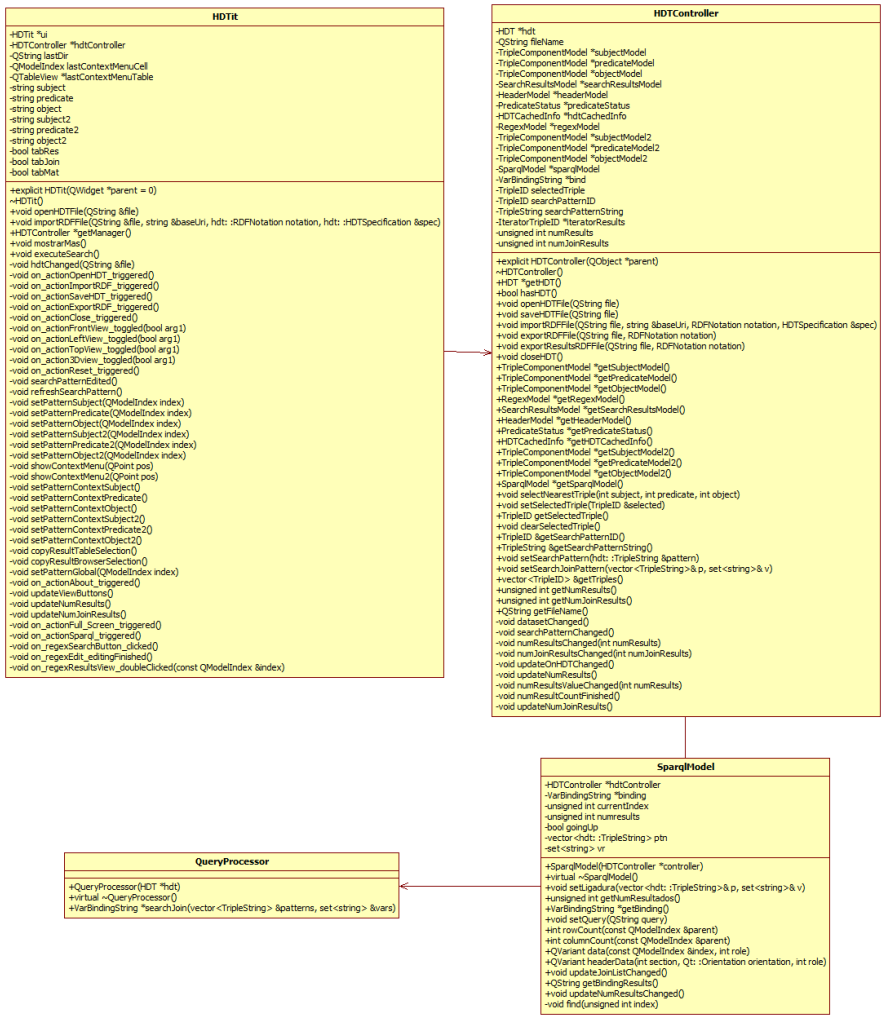


Figura 18: Detalle de las clases modificadas

En la clase HDT-it se han incluido los nuevos cuadros de texto para editar el segundo patrón de búsqueda, la nueva pestaña y los nuevos botones, así como las acciones derivadas de pulsar cada uno de ellos. También se ha incluido la nueva pestaña, para mostrar los resultados de la búsqueda con join. En resumen, en ella se incluyen todos los cambios que afectan a la interfaz. Estos cambios se describen en el apartado “La nueva interfaz”, que aparece un poco más adelante.

La clase HDTController se ha modificado para que se relacione con la clase SparqlModel y actúe como intermediaria entre ésta y la clase HDTit, De esta forma, se consigue que la clase HDTit reciba los eventos producidos por los resultados de la búsqueda con join y actualice la interfaz.

La clase SparqlModel es la que se relaciona con la clase QueryProcessor para obtener los resultados del join de los patrones, y es donde se lleva a cabo el formateado de los mismos para su presentación en la interfaz. En esta clase es donde se han realizado las principales modificaciones, para que muestre los resultados en una tabla y se actualice el número de resultados encontrados.

La relación entre las clases descritas, en función de los eventos que se registran cuando se hace una búsqueda por join, se reflejan, de forma simplificada, en el siguiente diagrama de secuencia:

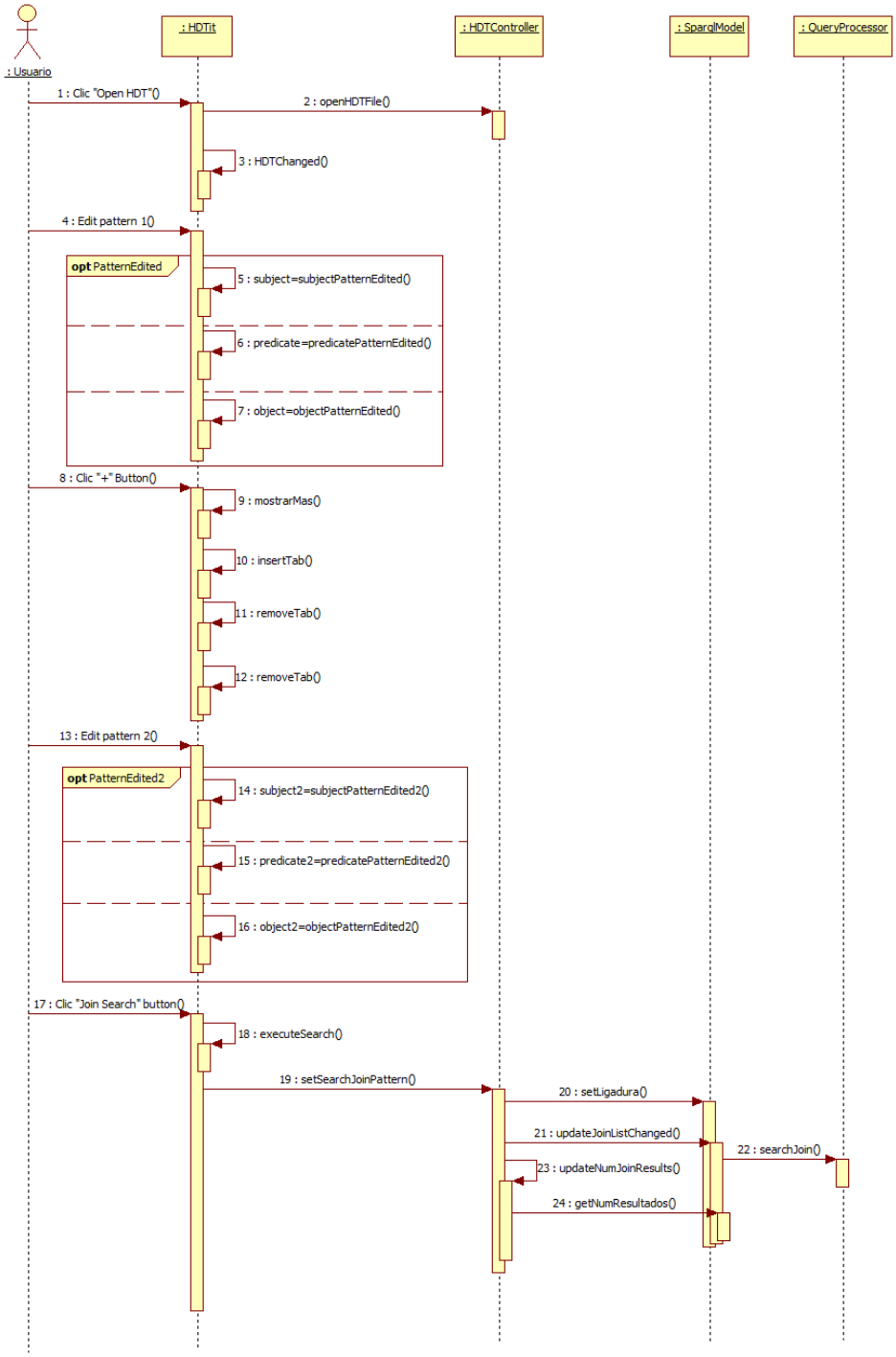


Figura 19: Diagrama de secuencia de una búsqueda con join.

5.3.3 Pruebas

Con el fin de asegurar un correcto funcionamiento del código, se han realizado distintas pruebas, con varias posibilidades de búsqueda, en función del tipo de join o de los valores fijados. Para comparar los valores de las variables y del número de resultados de las salidas esperadas con las obtenidas, los pasos realizados han sido extraer resultados de consultas sencillas y realizar un join de forma manual sobre el papel y así poder obtener los resultados esperados de la consulta en la nueva versión. Debido al elevado número de triples existentes en los conjuntos de datos en HDT, realizar todas las pruebas de combinaciones posibles sería una labor complicada, por lo que se han probado sólo algunas.

Para comprobar que el número de columnas y sus correspondientes nombres, coinciden en la salida esperada y la obtenida, se han realizado las siguientes pruebas de patrones de búsqueda, donde ?var es la variable de join, ____ son valores no especificados, por tanto son variables del patrón, y “Valor”, “Valor1” y “Valor2”, es el valor que se asigna a ese elemento del patrón:

Prueba			Columnas Salida esperada	Columnas Salida obtenida	Resultado
?var ?var	_____ _____	“Valor” _____	?var, ?predicate1, ?predicate2, ?object2	?var, ?predicate1, ?predicate2, ?object2	OK
?var ?var	_____ _____	“Valor” “Valor2”	?var, ?predicate1, ?predicate2	?var, ?predicate1, ?predicate2	OK
?var ?var	“Valor” _____	_____ _____	?var, ?object1, ?predicate2, ?object2	?var, ?object1, ?predicate2, ?object2	OK
?var ?var	“Valor1” _____	_____ “Valor2”	?var, ?object1, ?predicate2	?var, ?object1, ?predicate2	OK
?var ____	_____ _____	_____ ?var	?var, ?predicate1, ?object1, ?predicate2, ?object2	?var, ?predicate1, ?object1, ?predicate2, ?object2	OK
?var ____	_____ _____	“Valor” ?var	?var, ?predicate1, ?subject2, ?predicate2	?var, ?predicate1, ?subject2, ?predicate2	OK
____ ?var	?var _____	_____ _____	?var, ?subject1, ?object1, ?predicate2, ?object2	?var, ?subject1, ?object1, ?predicate2, ?object2	OK
____ ____	?var _____	_____ ?var	?var, ?subject1, ?object1, ?subject2, ?object2	?var, ?subject1, ?object1, ?subject2, ?object2	OK

5.3.4 La nueva interfaz

De forma visual, el principal cambio sobre la interfaz original de HDT-it es la inclusión de una nueva zona con tres cuadros de texto, correspondientes al sujeto, predicado y objeto del segundo patrón. Estos campos se muestran sólo si se hace clic sobre el botón "+" que aparece junto al primer patrón, y que se activa siempre que haya algo en alguno de los cuadros de texto del primer patrón. De forma simultánea, en las pestañas de la derecha también se produce un cambio. Desaparecen las pestañas de "Search Results" y "Matrix View" y aparece la pestaña correspondiente a "Join Results". En ella aparecen los resultados de la consulta de tipo join. Estos resultados se muestran en una tabla, con una fila para join encontrado y donde las columnas tienen el nombre de las variables.

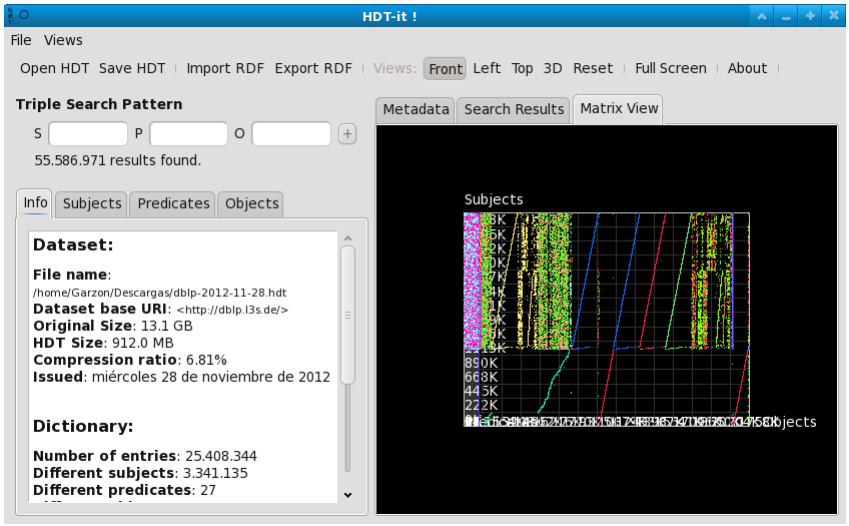


Figura 20: Visualización de la nueva interfaz

Otro cambio es la inclusión del botón "Search". En la versión original, la búsqueda se realizaba de forma instantánea con cada edición de uno de los componentes del triple. En la versión propuesta, esto sigue funcionando de igual modo excepto si se realiza una búsqueda por join. Cuando se pulsa el botón "+" para que se muestren los cuadros del segundo patrón, la búsqueda instantánea se deshabilita, ya que es necesario tener fijadas las variables para hacer el join. Además, con los cambios introducidos en la nueva versión, tampoco se pueden incluir en los cuadros de texto los elementos seleccionándolos al hacer doble clic en ellos en las listas de sujeto, predicado u objeto. Esta posibilidad se ha deshabilitado para evitar los errores que se puedan ocasionar debido a la colocación que el usuario espera de los elementos en los patrones y la obtenida

en la realidad. Es decir, al hacer doble clic sobre un sujeto de la lista, puede esperar que forme parte del primer patrón, del segundo o de ambos y el resultado obtenido podría ser que sólo se añada a uno de los patrones o a ambos. La decisión de a qué patrón se incorporarían los elementos es de diseño y tendría que estar implementada de antemano y, puesto que las necesidades de los usuarios son variables, introducir esta restricción puede hacer que el usuario no obtenga los resultados deseados. No obstante, cuando se editan los cuadros de texto, se siguen mostrando sugerencias, al igual que en la versión inicial.

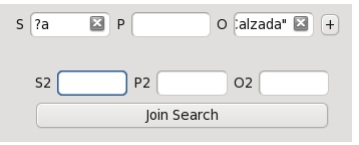


Figura 21: Botones “+” y “Join Search”

5.3.5 Utilización de la aplicación

El principal cambio en el uso de la aplicación se da cuando se va a realizar la búsqueda de un join entre dos patrones. Las búsquedas de patrones simples se realizan como en la versión original.

Para realizar una consulta por con join, hay que comenzar editando el primer patrón. Para fijar el valor de un campo, hay que introducirlo. Según se van editando los diferentes cuadros de texto, se muestran sugerencias de valores posibles para los mismos. Si un campo se quiere mantener como variable, basta con dejarlo en blanco. El campo que se va a utilizar como variable de join hay que editarlo incluyendo el nombre que se le quiere dar a la variable precedido por el símbolo “?”, por ejemplo, si el nombre que se quiere dar a la variable de join es “varjoin”, lo que se tiene que poner en el campo correspondiente es “?varjoin”.

Una vez editado al menos un campo del primer patrón, se habilita automáticamente el botón “+”, que despliega los campos editables del segundo patrón y el botón “Search”. También cambian las pestañas que se muestran en la parte derecha, donde desaparecen las correspondientes a la búsqueda simple y a la vista de matriz, y aparece la correspondiente a la búsqueda de join. Una vez mostrados los campos del segundo patrón, se editan de forma análoga a los del primero, teniendo en cuenta que la variable de join tenga el mismo nombre en ambos patrones.

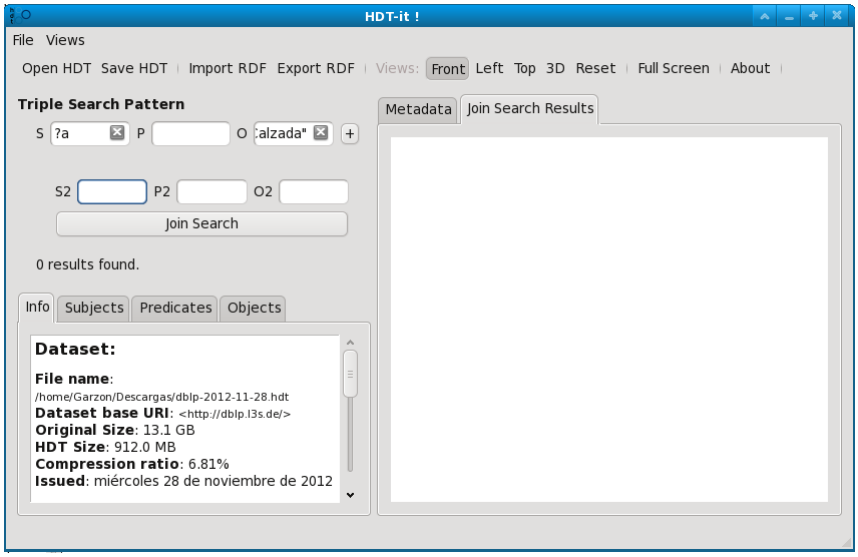


Figura 22: Edición del segundo patrón de búsqueda

Si no se incluye el símbolo “?” precediendo al nombre de la variable de join, la aplicación lo detectara como valor asignado a la variable, por lo que devolverá un resultado erróneo.

Cuando se han editado los patrones, se hace clic sobre el botón “Search” y se muestra en la parte de la derecha, en la pestaña “Search Join Results” los resultados de la consulta, en forma de tabla.

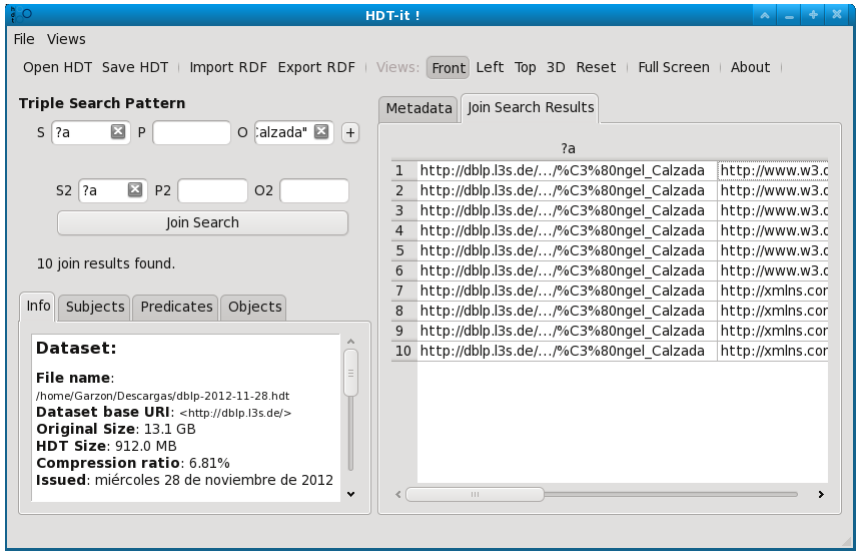


Figura 23: Visualización de los resultados de una búsqueda con join.

Si se vuelve a hacer clic sobre el botón “+”, la aplicación oculta los campos del segundo patrón y el botón “Search” y vuelven a aparecer las pestañas de “Search Results” y “Matrix View”, desapareciendo la correspondiente a los resultados de búsqueda de la consulta de join.

Por ejemplo, la siguiente consulta en lenguaje SPARQL:

```
SELECT *
WHERE {
    ?chebiDrug <http://purl.org/dc/elements/1.1/title> "Cetuximab"
    ?chebiDrug <http://bio2rdf.org/ns/bio2rdf#image> ?chebiImage
}
```

en HDT-it se traduciría a editar los campos de los patrones de búsqueda de la siguiente manera:

S	?chebiDrug	P	http://purl.org/dc/elements/1.1/title>	O	"Cetuximab"
S2	?chebiDrug	P2	<http://bio2rdf.org/ns/bio2rdf#image>	O2	

quedando de una forma similar a como se muestra en la siguiente imagen:

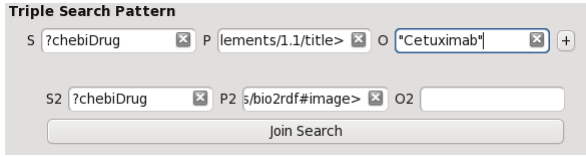


Figura 24: Ejemplo de edición de los patrones en HDT-it

6. Conclusiones del trabajo

6.1 Conclusiones

Como se dijo en el capítulo 1, la Web ha evolucionado bastante desde su comienzo, y no se sabe qué nos depara en un futuro, pero para poder evolucionar, hay que ser capaces de comprender y asimilar todo lo que la Web 3.0 pone a nuestro alcance a día de hoy.

Como se ha detallado con anterioridad, cada día se generan más datos en la Web. Todos estos datos pueden provenir de distintas fuentes y tener distintos formatos. El proceso de analizarlos e interrelacionarlos para extraer información de ellos es muy costoso debido a la heterogeneidad de los mismos. Esto causa el desaprovechamiento del potencial de los datos, en un mundo donde la tendencia es a compartir cada día más información, y donde las administraciones públicas están siendo obligadas por ley a la publicación de datos. Además, hay que tener en cuenta el potencial económico subyacente de la información que se puede extraer de los datos. Un análisis exhaustivo de los datos con el fin de obtener información concreta y veraz, puede ser costoso, pero de gran rentabilidad si el resultado puede ser objeto de negocio y puede generar beneficios, ya sea a intereses privados o públicos.

El uso de estándares, como RDF, para modelar los datos de la Web, hace posible que los datos se puedan relacionar entre sí con mayor facilidad, pero las labores de análisis siguen siendo complicadas debido al volumen de los datos y a las restricciones físicas de los sistemas de almacenamiento, procesamiento y redes.

Formatos como HDT, contribuyen a una consulta de los datos más eficaz, ya que permite comprimir grandes cantidades de datos con un gran índice de reducción. La herramienta HDT-it permite mantener ficheros con muchos datos en equipos locales con pocos recursos, pero pudiendo ser consultados sin necesidad de descomprimirlos. Además, no es necesario conocer la sintaxis propia de SPARQL para consultar los datos, algo que reduce los intentos de prueba y error ocasionados por errores sintácticos.

La modificación introducida en la herramienta aumenta las posibilidades de búsqueda, con lo que los resultados se pueden acotar más y conseguir información más específica en menor tiempo.

A nivel personal, la realización de este trabajo, me ha descubierto un campo de la informática y de las tecnologías de la información, desconocido para mí hasta el momento. Gracias al esfuerzo de mis tutores, he tenido la oportunidad

de asistir a un curso de introducción sobre los aspectos generales de la Web Semántica y el Big Data, y a diversas conferencias relacionadas con ello. En estas conferencias, he podido ver múltiples aplicaciones reales de lo que se puede lograr con los datos, viéndolos como una materia prima de información de diversa índole. También me han permitido escuchar a personas de gran importancia en este campo, como Ricardo Baeza-Yates, vicepresidente de Investigación para Europa y Latinoamérica, líder de los laboratorios de Yahoo! Research en Barcelona y Santiago de Chile, o Mar Cabra, periodista de datos y destacada por sus labores de investigación.

En una situación económica y política como la que se está viviendo actualmente, tanto a nivel mundial como a nivel nacional, cada vez más, los ciudadanos reclamamos mayor acceso a la información pública. Esto está llevando al planteamiento de leyes de acceso a la información, que ponen a disposición de los ciudadanos grandes cantidades de datos, lo que plantea la necesidad de herramientas que nos sirvan de ayuda para analizar esos datos y poder interpretarlos. De esta forma, conoceríamos el uso que se da al dinero público, datos de actividad política, relaciones entre proyectos y adjudicatarios y otros muchos datos de gran importancia. Con la realización de este trabajo, he comprendido que si los datos no cumplen determinados requisitos, no son útiles de por sí, por lo que aunque dispongamos de ellos, poco podemos hacer sin un gran trabajo previo de transformación a otros formatos.

A nivel social, cada vez compartimos más datos por Internet, y no sólo aquellos de los que somos conscientes, puesto que son muchos los dispositivos que, sin estar conectados a la red, emiten señales de localización. Aparte, también generamos información con otros actos cotidianos que realizamos, como usar el transporte público, por ejemplo. Todos estos datos generados a diario, sometidos a análisis estadísticos, pueden dar grandes beneficios económicos. Esto me ha hecho plantearme introducirme más en el mundo de los datos, para poder tener un mayor abanico de salidas laborales, ya que cada vez se solicitan más expertos en Big Data.

En el mundo de la investigación científica, cada vez hay herramientas más complejas que permiten obtener más y mejor información, pero sin algo que analice todos los datos obtenidos, no se consigue nada. Los análisis tradicionales, ya no son suficientes, ya que los laboratorios de investigación no están, generalmente, preparados para gestionar grandes volúmenes de datos, algo de lo que me he dado cuenta con este trabajo.

Es por todo lo recogido anteriormente, que este trabajo ha supuesto un reto para mí, ya que me suponía introducirme en un mundo del que no sabía nada y donde la mayor parte de los conceptos eran nuevos para mí.

6.2 Trabajo futuro

Como se ha explicado, en la versión propuesta sólo se incluye la posibilidad de hacer consultas de join sobre dos patrones. En futuras versiones, se podría aumentar el número de patrones en la búsqueda e incluir búsquedas más complejas, con filtros en las consultas, que permitan acotar más los resultados obtenidos. Las nuevas mejoras, facilitarían el análisis de grandes volúmenes de datos en tiempos reducidos y ocupando poco espacio. Esto contribuiría a un desarrollo de la labor investigadora en aquellos campos en los que se manejan una gran cantidad de datos no estructurados y de orígenes diferentes.

También se podría modificar la versión propuesta en este trabajo para su uso en distintos sistemas operativos.

Bilbiografía

A continuación se detallan los principales artículos, documentos o sitios web utilizados para la realización de esta memoria. **Los sitios web referenciados, en ambos apartados, han sido accedidos todos por última vez el 31 de agosto de 2013, para comprobar que los contenidos citados siguen apareciendo**.

Publicaciones y artículos

[1] Arias Gallego, Mario; Fernández, Javier D.; Martínez-Prieto, Miguel Ángel. *An Empirical Study of Real-World SPARQL Queries.* Proc. 1st International Workshop on Usage Analysis and the Web of Data (USEWOD), 2011. Disponible en: http://arxiv.org/abs/1103.5043

[2] Arias Gallego, Mario; Fernández, Javier D.; Martínez-Prieto, Miguel Ángel. *HDT-it: Storing, Sharing and Visualizing Huge RDF Datasets.* http://dataweb.infor.uva.es/wp-content/uploads/2011/10/iswc2011.pdf

[3] Arias Gallego, Mario; Fernández, Javier D.; Martínez-Prieto, Miguel Ángel. *RDF Visualization using a Three-Dimensional Adjacency Matrix*. Proc. 4th International Semantic Search Workshop (SemSearch), 2011. Disponible en: http://km.aifb.kit.edu/ws/semsearch11/8.pdf.

[4] Belleau, Francois. Nolin, Marc-Alexandre. Tourigny, Nicole. Rigault, Philippe. Morissette, Jean. *Bio2RDF: Towards a mashup to build bioinformatics knowledge systems.* 21 de Marzo de 2008. Journal of Biomedical Informatics. http://mashup.pubs.dbs.uni-leipzig.de/files/Belleau2008BioRDFTowardsamashuptobuildbioinformaticsknowledge.pdf

[5] Cabrera, Juan Ignacio. *La Web Semántica*. Revista PC actual. Nº 249.

[6] Fernández, Javier D.; Martínez-Prieto, Miguel Ángel; Gutiérrez, Claudio; Polleres, Axel. *Binary RDF Representation for Publication and Exchange (HDT)*. W3C Member Submission, 2011.

[7] Fernández, Javier D.; Martínez-Prieto, Miguel Ángel; Gutiérrez, Claudio; Polleres, Axel; Arias, Mario. *Binary RDF representation for publication and exchange*. Journal of Web Semantics, 19:22-41, 2013.

[8] Fernández Medina, David. *Estudio del modelo de representación XML/RDF.* 2003-2004. Universitat Oberta De Catalunya. http://openaccess.uoc.edu/webapps/o2/bitstream/10609/1024/1/20847tfc.pdf

[9] Köhler, Jacob. Philippi, Stephan. Lange, Matthias. *SEMEDA: ontology based semantic integration of biological databases.* 9 de Junio de 2003. BIOINFORMATICS. http://bioinformatics.oxfordjournals.org/content/19/18/2420.full.pdf+html?sid=1cdf8fa5-f5a8-42e4-beb5-bc8da80b63b9

[10] Martínez-Prieto, Miguel Ángel; Arias, Mario and Fernández, Javier D. *Exchange and Consumption of Huge RDF Data.* Proc. 9th Extended Semantic Web Conference (ESWC), pp. 437-452, 2012.

[11] Martínez-Prieto, Miguel. A; Fernández, Javier D.; Cánovas, Rodrigo. *ACM Sigapp Applied Computing Reviews* 12 (2), 64-77, 2012.

[12]Martínez-Prieto, Miguel Ángel; Fernández, Javier D.; Cánovas, Rodrigo. *Querying RDF Dictionaries in Compressed Space*. http://dataweb.infor.uva.es/wp-content/uploads/2012/06/acr2012.pdf

[13] Valencia Castillo, Edwin. *Recuperación y organización de la información a través de RDF usando SPARQL*. Universidad Pontificia de Salamanca – Campus de Madrid. ggomez.files.wordpress.com/2008/09/informe-sparql.doc

Sitios Web

[14] Apache Jena - SPARQL Tutorial. http://jena.apache.org/tutorials/sparql.html

[15] Berners-Lee, tim. Linked Data - Desing Issues. 27 de Julio de 2006. http://www.w3.org/DesignIssues/LinkedData.html

[16] Big Data at the Speed of Business. IBM. http://www-01.ibm.com/software/data/bigdata/

[17] Binary RDF Representation for Publication and Exchange (HDT) Submission. W3C Member Submission. http://www.w3.org/Submission/2011/03/

[18] Bio2RDF. http://bio2rdf.org/

[19] bio2rdf - Linked Data for the Life Sciences - Google Project Hosting. http://code.google.com/p/bio2rdf/

[20]Bio2RDF datasets and statistics. Fichero de Google Docs. https://docs.google.com/spreadsheet/ccc?key=0AnGgKfZdJasrdElfQzRWWWhKUFR0UnRpeG14NGZRS2c#gid=0

[21] bio2rdf/bio2rdf-scripts - HitHub. https://github.com/bio2rdf/bio2rdf-scripts/wiki

[22] Bioqueries. Universidad de Málaga. http://bioqueries.uma.es/query/enzymes-synonyms-url-kegg-web-direction-cofactor-and-url-kegg-based-metabolite

[23] Calderón Blanco, Marcos. *Web 3.0 Recuperación y Acceso a la Información*. Universidad Carlos III de Madrid. http://web30websemantica.comuf.com/websemantica.htm

[24]DataWeb - Compresión, Indexación y Aplicaciones sobre Grandes Colecciones de Datos. Departamento de Informática. Universidad de Valladolid. http://dataweb.infor.uva.es/

[25] Dbpedia español. http://es.dbpedia.org/

[26] Dodds, Leigh. Introducing *SPARQL: Querying the Semantic Web*. 16 de Noviembre de 2005. XML.com. http://www.xml.com/lpt/a/2005/11/16/introducing-sparql-querying-semantic-web-tutorial.html

[27]Dumontier, Michel et al. Bio2RDF: Linked Data for the Life Sciences Blog. http://bio2rdf.blogspot.com.es/

[28]Guía Breve de Web Semántica. W3C España. http://www.w3c.es/Divulgacion/GuiasBreves/WebSemantica

[29] Johnston, Leslie. *Data is the New Black*. 14 de octubre de 2011. Library of Congress. http://blogs.loc.gov/digitalpreservation/2011/10/data-is-the-new-black/

[30] Lamarca Lapuente, María Jesús. Hipertexto: *El nuevo concepto de documento en la cultura de la imagen. Hacia la Web Semántica*. http://www.hipertexto.info/documentos/web_semantica.htm

[31] McAfee, Andrew. Brynjolfsson, Erik. Big *Data: The Management Revolution. Harvard Business Review*. http://hbr.org/2012/10/big-data-the-management-revolution/ar/1

[32] McCarthy, Philip. Search *RDF data with SPARQL. SPARQL and the Jena Toolkit open up the semantic Web*. 10 de Mayo de 2005. developerWorks. http://www.ibm.com/developerworks/xml/library/j-sparql/

[33] Miller, Eric. *An Introduction to the Resource Description Framework*. Mayo 1998. D-Lib Magazine. ISBN 1082-9873. http://www.dlib.org/dlib/may98/miller/05miller.html

[34] OpenLink Virtuoso (Product Blog). http://virtuoso.openlinksw.com/blog/

[35] OpenLink Virtuoso Universal Server: Documentation. http://docs.openlinksw.com/virtuoso/

[36] Pérez Valdés, Damián. *Web Semántica y sus principales características*. 26 de Junio de 2007. Maestros del Web. http://www.maestrosdelweb.com/editorial/web-semantica-y-sus-principales-caracteristicas/

[37]Poole, David. Mackworth, Alan. Artificial *Intelligence. Foundations of computational agents*. http://artint.info/html/ArtInt_314.html

[38] RDF HDT. http://www.rdfhdt.org/

[39] RDF Primer. W3C Recommendation 10 February 2004. W3C. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/

[40] RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation 10 February 2004. W3C. http://www.w3.org/TR/rdf-schema/

[41] RDF. Freebase. http://www.freebase.com/m/0f2vj

[42]rdfhdt. Demo of the HDT-it! application to browse big RDF datasets. Youtube. http://www.youtube.com/watch?v=HMPkc725sMY

[43]RFC 2396. Uniform Resource Identifiers (URI): Generic Syntax. The Internet Society. 1998. http://www.ietf.org/rfc/rfc2396.txt

[44] Senso Ruiz, José A. *Resource Description Framework*. 2003. Universitat Pompeu Fabra. http://www.upf.edu/hipertextnet/numero-1/rdf.html

[45] SIMILE Project. http://simile.mit.edu/

[46] SourceForge.net: bio2rdf. http://sourceforge.net/apps/mediawiki/bio2rdf/index.php?title=Main_Page

[47] SPARQL 1.1 Overview. W3C Recommendation 21 March 2013. W3C. http://www.w3.org/TR/sparql11-overview/

[48] SPARQL by Example - Cambridge Semantics. http://www.cambridgesemantics.com/es/semantic-university/sparql-by-example

[49]SPARQL Protocol and Query Language: SPARQL Frequently Asked Questions. http://www.thefigtrees.net/lee/sw/sparql-faq

[50] Vialfa, Carlos. *Web semántica: las aplicaciones actuales*. Kiosquea.net. http://es.kioskea.net/faq/7082-web-semantica-las-aplicaciones-actuales

[51]Virtuoso Open-Source Wiki : Main.VirtEC2AMIBio2rdfInstall. http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VirtEC2AMIBio2rdfInstall

[52] Wikipedia. The Free Encyclopedia. http://www.wikipedia.org/

[53] World Wide Web Consortium (W3C). http://www.w3.org/

Apéndices

A1. Glosario

A continuación se definen algunos términos y siglas utilizados en la redacción de esta memoria:

- API: Application Programming Interface, es un conjunto de funciones o servicios disponibles para el programador, para crear nuevo código.

- CPU: Central Processing Unit, es la parte hardware de un computador que ejecuta las instrucciones de los procesos siguiendo las reglas aritméticas y lógicas y las operaciones de entrada y salida.

- DTD: Document Type Definition, es un documento con formato XML o SGLM donde se expecifican qué restricciones estrucuturales y sintaxis tiene que tener.

- Dublin Core (DC): es un modelo de metadatos que ofrece 15 definiciones semánticas descriptivas.

- EBNF: Extended Backus-Naur Form, es un metalenguaje utilizado para definir gramáticas libres de contexto.

- Endpoint: es una interfaz de acceso a un servicio web que permite el intercambio de mensajes.

- Exhibit: es un componente del Proyecto SIMILE que permite la creación de páginas web con soporte para ordenaciones, filtros y visualizaciones enriquecidas escribiendo código en HTML. No es necesario contar con una base de datos ni con tecnologías de aplicaciones web.

- GUI: Graphical User Interface, es un entorno gráfico que permite interactuar con el computador a través de imágenes o similares, en lugar de por línea de comandos o consola.

- HDT: Header-Dictionary-Triples, es un formato binario para la publicación y el intercambio de datos en RDF a gran escala.

- HTML: HyperText Markup Language, es un lenguaje de marcado que se utiliza para estructurar y dar formato a los documentos web.

- HTTP: Hypertex Transfer Protocol, es un protocolo de red para la publicación de páginas Web, basado en peticiones y respuestas entre cliente y servidor.

- I/O: Input/Output, hace referencia a la comunicación entre las máquinas y los usuarios. Son operaciones de entrada y salida, de intercambio de mensajes.

- ID: Identificator, es la secuencia de cifras que identifica un elemento dentro de un conjunto dado.

- IRI: Internationalized Resource Identifier, es un identificador de recursos de la Web, como un URI, pero admite un mayor conjunto de caracteres. Mientras que los URIs solo admiten caracteres ASCII, los URIs permiten caracteres Unicode.

- LGPL: Lesser General Public License, es un tipo de licencia de software que asegura que el software licenciado es libre para todos los usuarios.

- LSID: Life Science Identifiers, es un identificador único para localizar en la Web recursos del dominio de las Ciencias.

- Mashup: es una aplicación o página web que toma datos de distintas fuentes y los muestra de forma conjunta con distinto formato del original, para dar lugar a nuevos servicios.

- Ontología: definición de un esquema conceptual de un dominio.

- OWL: Web Ontology Language, es un leguaje que cuenta con marcas y etiquetas, cuya finalidad es la publicación y la compartición de datos en la Web mediante el uso de ontologías.

- Protégé: es un editor de ontologías y un marco de bases de conocimiento. Es de código libre y gratuito.

- PROV: Provenance Vocabulary, es un modelo de datos basicos para la procedencia para construir representaciones de entidades, personas y procesos involucrados en el proceso de producción de datos.

- RDF: Resource Description Framework, es un formato, amparado por el Consorcio de la Web, W3C, que se utiliza para representar la información en la Web Semántica.

- RDFS: RDF Schema, es el lenguaje de descripción de RDF, y que define las clases y las propiedades que a su vez pueden ser utilizadas como medio de descripción de clases, propiedades y otros recursos.

- Semeda: Semantic Meta Database, consiste en una interfaz de estados finales de bases de datos relacionales para almacenar ontologías, metadatos de la base de datos y definiciones semánticas de la base de datos. En una capa intermedia se utilizan Java Server Pages (JSPs) para generar de forma dinámica la interfaz de estados iniciales en HTML, que son los que ve el usuario.

- Sesame: proyecto open source que proporciona un servidor que permite el almacenamiento y la consulta de triples.

- SGML: Standard Generalized Markup Language, es un metalenguaje estándar para la definición de lenguajes de marcado personalizados para distintos tipos de documentos.

- SIMILE: Semantic Interoperability of Metadata and Information in unLike Environments, es un proyecto centrado en el desarrollo de herramientas de robustas de código abierto para la gestión, la visualización y la reutilización de recursos digitales.

- SOAP: Simple Object Access Protocol, es un protocolo basado en XML para el intercambio de información entre aplicaciones utilizando HTTP.

- SPARQL: SPARQL Protocol And RDF Query Languaje. Este acrónimo hace referencia tanto al lenguaje de consulta de RDF como al propio protocol de acceso a datos.

- SQL: Structured Query Languaje, es un lenguaje estándar de acceso a bases de datos.

- Unicode: es un estándar de codificación de caracteres.

- URI: Uniform Resource Identifier, es un identificador único para cada recurso de la Web. Se puede interpretar como la unión de un URL con un URN.

- URL: Uniform Resource Locator, es un identificador que representa un recurso disponible a través de Internet.

- URN: Uniform Resource Name, identifica un recurso en la Web dándole un nombre único, pero sin indicar su localización.

- VoID: Vocabulary of Intelinked Datasets, es un vocabulario en RDFS para expresar metadatos sobre conjuntos de datos RDF.

- W3C: World Wide Web Consortium, es la principal organización de estandarización de la Web.

- WSDL: Web Service Description Language, es un formato XML para describir servicios de red como un conjunto de endpoints que operan sobre mensajes que contienen tanto información orientada a documentos como a procedimientos.

- XML: Extensive Markup Languaje, es un lenguaje de marcado que permite almacenar documentos de forma que sean entendibles por máquinas y por humanos.

- XSLT: Extensible Stylesheet Language Transformations, es un estándar de hojas de estilo para XML.

A2. Contenido del CD

En este apartado se describe el contenido del CD adjunto a este tomo:

- memoria.pdf: contiene la versión en pdf de este documento.
- hdt-it[Original]: carpeta que contiene el código original de la aplicación HDT-it en la versión descargable para distribuciones Linux de 64bits.
- hdt-it[Modificado]: carpeta que contiene el código con la versión desarrollada en este trabajo.
- Bio2RDF: conjunto de datos de Bio2RDF en formato HDT.
- Diagramas: carpeta que incluye las imágenes de los diagramas de clases y de secuencia incluidos en este documento, con mayor resolución.
- Bibliografía: artículos en pdf utilizados como referencia para la escritura del documento de la memoria.