TESIS DOCTORAL

Entorno para la gestión semántica de información biomédica en investigación traslacional

Jose Antonio Miñarro Giménez
Ingeniero en Informática

Dirigida por:

Dr. Jesualdo Tomás Fernández Breis

Dr. Rodrigo Martínez Béjar

Departamento de Informática y Sistemas
Facultad de Informática
UNIVERSIDAD DE MURCIA
Marzo 2012

Agradecimientos

Debo agradecer de manera especial a mis directores, Jesualdo Tomás Fernández Breis y Rodrigo Martínez Béjar, la oportunidad y confianza que depositaron en mi, ha sido un honor y un privilegio estar bajo su dirección. Me gustaría destacar el trabajo de Jesualdo al que no podré corresponder de manera justa por los esfuerzos que me ha dedicado.

Quiero dar las gracias a Mikel Egaña Aranguren y Boris Villazón Terrazas por su contribución, sin duda su participación ha enriquecido este trabajo. También a Rafael Valencia García y Francisco García Sánchez por su orientación, ya que me han facilitado la realización de esta tesis.

Quiero agradecer también a todos mis compañeros del grupo de *Tecnologías para el Modelado, Procesamiento y Gestión de Conocimiento* por su ayuda y por los recuerdos que me llevo de ellos de los buenos momentos que hemos compartido.

Mi gratitud al Prof. Jürg Bähler y a mis compañeros, del Department of Genetics, Evolution and Environment, que me acogieron y me proporcionaron los medios necesarios para mi estancia de investigación con ellos.

Quiero dedicar esta tesis a toda mi familia. A mis padres, Pepe y Pepi, que me lo han dado todo. A mis hermanos Alfonso y María, que me han animado en los momentos más difíciles. A mis abuelos que siempre han sido un ejemplo de trabajo y esfuerzo. De forma muy especial, quiero agradecer a Cati su comprensión y apoyo tanto personal como profesional.

Por último, este trabajo también ha sido posible a la financiación de la Fundación Séneca por la beca 07836/BPS/07 y del Ministerio de Ciencia e Innovación a través de los proyectos TSI2007-66575-C02-02 y TIN2010-21388-C02-02.

Índice general

Nomenclatura

Acrónimos

ADN Ácido Desoxirribonucleico

API Application Program Interface

CERN Conseil Européen pour la Recherche Nucléaire

CGD Candida Genome Database

CHEBI Chemical Entities of Biological Interest

CKAN Comprehensive Knowledge Archive Network

DAML+OIL DARPA Agent Markup Language + Ontology Inference Layer

DAS Distributed Annotation Service

DDBJ DNA Data Bank of Japan

DL Description Logic

DM Direct Mapping

EBI European Bioinformatics Institute

ECO Evidence Codes Ontology

EMBL European Molecular Biology Laboratory

ENA European Nucleotide Archive

ePSI European Public Sector Information

FGI Fungal Genome Initiative

FP7 7th Framework Programme

GO Gene Ontology

HCLS IG Semantic Web Health Care and Life Sciences Interest Group

HPO Human Phenotype Ontology

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

INSDC International Nucleotide Sequence Database Collaboration

IRI International Resource Identifiers

JGI Joint Genome Institute

JSP Java Server Pages

KB Knowledge Base

KEGG Kyoto Encyclopedia of Genes and Genomes

KOG euKaryotic Orthologous Groups

LATC Linked Open Data Around-The-Clock

LD Linked Data

LGPL Lesser General Public License

LOD Linking Open Data

LODD Linking Opend Drug Data

LSSW Life Sciences Semantic Web

MGD Mouse Genome Database

NCBI National Center for Biotechnology Information

NCBO National Center for Biomedical Ontology

NLM National Library of Medicine

OBO Open Biological and Biomedical Ontologies

OGO Ontological Gene Orthology

OMIM Online Mendelian Inheritance in Man

OO Object Oriented

OWL Web Ontology Language

PATO Phenotypic Quality Ontology

PRO Protein Ontology

R2O Relational to Ontology

R2RML RDB to RDF Mapping Language

RACER Renamed Abox and Concept Expression Reasoner

RDB Relational Data Base

RDF Resource Description Framework

RDFS Resource Description Framework Schema

RDQL RDF Data Query Language

REST Representational State Transfer

RO Relationship Ontology

SGD Saccharomyces Genome Database

SGTC Standford Genome Technology Center

SHOE Simple HTML Ontology Extensions

SOAP Simple Object Access Protocol

SPARQL SPARQL Protocol and RDF Query Language

SPARUL SPARQL Update

SQL Structured Query Language

TAIR The Arabidopsis Information Resource

UMLS Unified Medical Language System

URI Uniform Resource Identifier

W3C World Wide Web Consortium

WSDL Web Services Description Language

WWW World Wide Web

XAO Xenopus Anatomy Ontology

XML Extensible Markup Language

ZFA Zebrafish Anatomy Ontology

Índice de figuras

Índice de tablas

Bloque I

Introducción, estado del arte y objetivos

Capítulo 1

Introducción

La investigación traslacional [1] es la disciplina que permite aplicar los resultados de investigaciones biomédicas básicas a nivel clínico. El obstáculo más relevante al que esta disciplina debe enfrentarse es la heterogeneidad de los repositorios de información asociados al dominio de la biología y la biomedicina. Las investigaciones traslacionales tienen el objetivo de poner a disposición de investigaciones las evidencias obtenidas en investigaciones básicas para ensayos clínicos.

Para facilitar la investigación traslacional es necesario relacionar la información biológica y clínica. Esto se puede conseguir mediante la integración de repositorios en los que la información biomédica y biológica está almacenada. El proceso de integración de información presenta una serie de retos que deben superarse. Estos retos están principalmente relacionados con la heterogeneidad y complejidad de la información almacenada en los repositorios, las grandes cantidades de información que se deben procesar, los distintos métodos de acceso disponibles e incluso la existencia de errores en los repositorios.

Para llevar a cabo la integración de información deben tenerse en cuenta las características de los dominios que se integran. Así, el dominio de la biología es un campo de la ciencia en el cual para generar nuevo conocimiento a menudo se parte de la información previamente obtenida en otros experimentos. De este modo la ciencia avanza mediante la comparación de la

información obtenida por nuevos experimentos y por la información previa [2]. Además, la información biológica presenta una abundancia de tipos y formatos diferentes. Los experimentos biológicos generan información inherentemente compleja [3] que va desde datos simples de secuencias de nucleótidos y proteínas, pasando por estructuras tridimensionales de macromoléculas e imágenes de alta resolución de células y tejidos, hasta los resultados de análisis de microarrays [4] y secuenciación genética [5].

Actualmente se está generando información biológica nueva de manera constante y en grandes cantidades. Esta situación ha provocado que haya proliferado la creación de nuevos repositorios de información. A día de redacción de este documento, el número de bases de datos biomédicas se estima en unas 1380 [6], y su número sigue aumentando cada año.

La bioinformática es un campo de investigación que tiene el objetivo de hacer frente la dificultad de gestionar grandes cantidades de información biomédica. La bioinformática puede ser definida como la disciplina científica en la que la biología y las tecnologías de la información se unen.

Debido a la complejidad, multitud, diversidad y rápida evolución de la información biológica, es imposible gestionar los repositorios de información biológica de manera tradicional, el cual requiere de un trabajo manual que supone una gran inversión en tiempo y en esfuerzo. Por lo tanto, es cada vez más necesario dotar de nuevas herramientas de gestión que faciliten esta tarea y pueda ser realizada de manera autónoma [7].

Debido a la evolución de la tecnología y a las nuevas técnicas de secuenciamiento de genomas, hoy en día los experimentos producen altas tasas de información biológica que es almacenada en repositorios distribuidos por toda la Web, como el repositorio de Uniprot [8] que proporciona información sobre secuencias y funciones de proteínas. Esta situación agrava el problema de la gestión de la información.

Una de las técnicas que más interés está suscitando para mejorar la gestión, integración y publicación de información en el dominio de la biología y la biomedicina es la Web semántica [9]. La Web semántica es la base para la siguiente evolución de la Web y que está siendo desarrollada por W3C [10]. La Web semántica aboga por el uso de datos en vez de documentos,

como en la Web actual. Además, dota de semántica a dichos datos. Así, los datos en dicha red pueden ser procesados por aplicaciones autónomas. Estas aplicaciones pueden ser capaces de realizar tareas de búsqueda e inferencia actualmente imposibles sin intervención humana debido a cómo está representada la información en los repositorios actuales. La Web semántica puede permitir cumplir con las necesidades de gestión de información de las ciencias biológicas donde la información es abundante, se actualiza con frecuencia, es compleja y se encuentra distribuida en la Web en repositorios independientes [11].

El uso de la Web semántica aplicada en el dominio de la biología sigue creciendo y ha esta disciplina se le denomina "Life Sciences Semantic Web" (LSSW) [12, 13]. La LSSW está basada en el uso de los estándares de la Web semántica, como RDF [14], SPARQL [15] u OWL [16]. Estos estándares permiten conceptualizar el conocimiento de los dominios de ciencias naturales y anotar la información presente en la literatura científica y en repositorios de información, para que estén en un formato adecuado para ser procesados de manera autónoma por aplicaciones [17].

Para representar dicho conocimiento en la Web semántica se utilizan las ontologías [18]. Así, uno de los mayores éxitos en la definición de ontologías biológicas ha sido el proyecto de Gene Ontology (GO) [19]. El éxito de esta iniciativa ha impulsado la creciente definición de ontologías biomédicas estando recopiladas en su mayoría en el proyecto Open Biomedical Ontologies (OBO [20]).

Esta tesis presenta un entorno para la gestión e integración semántica de información biológica y biomédica, con el objetivo de facilitar el procesamiento de la información en tareas de investigación traslacional. Como validación de esta tesis se ha desarrollado el Sistema OGO [21] (Ontological Gene Orthology). Este sistema ha sido utilizado para integrar, gestionar y explotar la información sobre grupos de genes y proteínas ortólogas, y sobre sus enfermedades genéticas relacionadas, presentes en diversos repositorios bioinformáticos disponibles en la Web. Este sistema está basado en las tecnologías de la Web semántica, las cuales son utilizadas para representar, almacenar, explotar y guiar el proceso de integración de la información y conocimiento

biológico y biomédico.

La información sobre ortología es usada principalmente en estudios filogenéticos y clasificación taxonómica de organismos [22]. Esta información también puede ser utilizada para generar nuevas hipótesis de investigación. Por ejemplo, un gen ortólogo a una secuencia dada, que se encuentra en otra especie, podría orientar a los investigadores para descubrir su función en dicha especie [23]. Por lo tanto, la información sobre ortólogos puede ser utilizada en investigaciones relacionadas con enfermedades genéticas ya que pueden ayudar a entender sus causas genéticas [24].

1.1. Organización de la tesis

En esta tesis se distinguen cinco bloques principales compuestos de catorce capítulos.

El bloque I está formado por siete capítulos que tratarán de ofrecer una visión general del estado del arte de aquellos aspectos de más relevancia para esta tesis y se definirán los objetivos que se persiguen en la misma. El capítulo 1 proporcionará una breve introducción a la tesis y a su organización. El capítulo 2 presentará un estudio de los diversos tipos de repositorios de información biológica y biomédica. También describirá los repositorios de información sobre genes y proteínas ortólogas y sobre enfermedades genéticas utilizados para el desarrollo de esta tesis. El capítulo 3 proporcionará una descripción de las principales tecnologías asociadas con la Web semántica. Por otro lado, presentará también la iniciativa de Linked Data para publicar datos en la Web. El capítulo 4 describirá la aplicación de las tecnologías de la Web semántica para resolver los retos presentes en el campo de la bioinformática. El capítulo 5 presentará los diferentes tipos de metodologías para la integración de repositorios bioinformáticos y los ejemplo más importantes asociados cada una. El capítulo 6 presentará un estudio de las metodologías existentes para la publicación de información biológica y biomédica en repositorios basados en modelos semánticos. Estas metodologías permiten la publicación de información y la creación de repositorios basados en RDF y OWL, como base para el desarrollo de la Web semántica. En el capítulo 7

de este bloque se describirán los objetivos de esta tesis y la metodología de investigación seguida.

El bloque II está formado por tres capítulos que describirán las propuestas realizadas y las herramientas implementadas para llevar a cabo el trabajo de integración semántica. En particular, el capítulo 8 presentará la metodología de integración semántica. Se presentrán cada uno de los elementos en que se divide para integrar información biológica y biomédica almacenada en repositorios bioinformáticos. El capítulo 9 proporcionará una descripción de los modelos para: (1) la consulta avanzada de la información almacenada en el repositorio semántico obtenido mediante la metodología definida en el capítulo 8; y (2) la publicación del repositorio semántico como conjuntos de datos enlazados en la nube Linked Open Data (LOD [25]). El capítulo 10 describirá las herramientas desarrolladas y que implementan el método de integración semántica definido en la tesis.

El bloque III está dividido en dos capítulos que describirán la validación de las metodologías y herramientas presentadas en el bloque II mediante su aplicación en la construcción del Sistema OGO. En concreto, en el capítulo 11 se describirán los detalles asociados con la aplicación de la metodología de integración semántica para la construcción del repositorio semántico OGO y la definición de la ontología global del Sistema OGO. El capítulo 12 describirá las interfaces de consulta desarrolladas específicamente para el Sistema OGO, la transformación del repositorio semántico OGO para publicar en LOD el conjunto de datos asociado y también la utilización de la interfaz de consultas avanzadas definida en el capítulo 9 sobre el repositorio semántico OGO.

El bloque IV está formado por el capítulo 13 en el que se presentarán la discusión, comentarios y trabajo futuro relacionados con el trabajo aquí presentado. Además, se enumerarán las publicaciones y contribuciones a congresos relacionados con esta tesis y que podrán servir para completar la comprensión del mismo.

Finalmente, el bloque V lo forma el capítulo 14 en el que se realizará un breve resumen en inglés del trabajo de tesis realizado.

Capítulo 2

Repositorios bioinformáticos

Los repositorios bioinformáticos tienen como objetivo principal poner a disposición de los investigadores los datos biomédicos existentes. Estos repositorios intentan ofrecer un sitio centralizado donde almacenar un tipo determinado de información. Extraer los datos biológicos o médicos necesarios de la información publicada en artículos requiere mucho tiempo. Además, no toda la información puede estar publicada en artículos. Por otro lado, debido a la gran cantidad de información existente, su localización, consulta y análisis presenta grandes dificultades.

Por lo tanto sería útil disponer de un sistema informático que sea capaz de procesar dicha información. De este modo, para el análisis de datos biológicos se hace imprescindible un método automático que reduzca o elimine el tiempo que los investigadores invierten en realizar dichas tareas, así como que faciliten el almacenamiento y gestión de la información generada.

Con la evolución de la capacidad de los sistemas informáticos y la reducción en los precios de los ordenadores, éstos son utilizados cada vez más como medio de almacenamiento por la mayoría de los científicos. Así, en un principio la información se almacenaba físicamente mediante cintas, posteriormente se utilizaron discos, y en las sucesivas evoluciones de la tecnología surgieron las redes de comunicación entre universidades y centros de investigación, precursoras de la actual red, Internet. Con el desarrollo de la Web (World Wide Web - WWW), desde principios de los años 90, éste ha sido el

método estándar de distribución y acceso a los repositorios de información biológica y biomédica.

La bioinformática surge como una disciplina científica que pretende hacer frente a los problemas, retos y oportunidades creados a partir de la necesidad de gestionar, consultar y explotar estos grandes repositorios. La bioinformática puede ser definida como el campo de la ciencia en el que la biología, las ciencias de la computación y las tecnologías de la información se unen en una única disciplina.

Los tipos de repositorios más extendidos en la actualidad son las bases de datos. Estas se basan en el modelo relacional propuesto por Codd en 1970. En él los conceptos se representan por medio de relaciones o tablas que contienen todos los atributos del concepto y a través de referencias se relacionan unos conceptos con otros. Los modelos relacionales se transforman en esquemas lógicos, que son menos expresivos y son dependientes del modelo de sistema gestor de base de datos utilizado.

La Biología genera gran cantidad de información. Como ejemplo, indicar que el repositorio de secuencias de nucleótidos ENA [26] dispone, en el momento de redacción de esta tesis, de más de 238 millones de secuencias que suponen más de 393 mil millones de bases almacenadas. Por lo tanto, la necesidad de almacenamiento y distribución de grandes conjuntos de datos ha aumentando en gran medida. También, el número de bases de datos con información del dominio de la biológica ha aumentado considerablemente. El informe anual *"The Nucleic Acids Research Database Issue and the online Molecular Biology Database Collection"*, estima desde 1993 el número de bases de datos bioinformáticas. Así, en la figura 2.1 se puede observar la evolución del número de bases de datos bioinformáticas desde 2003 con 386 hasta 2012 con 1380. Estas bases de datos deben cumplir con una serie de requisitos para ser tenidas en cuenta en el informe, como por ejemplo que no restrinjan su acceso y que estén disponibles online.

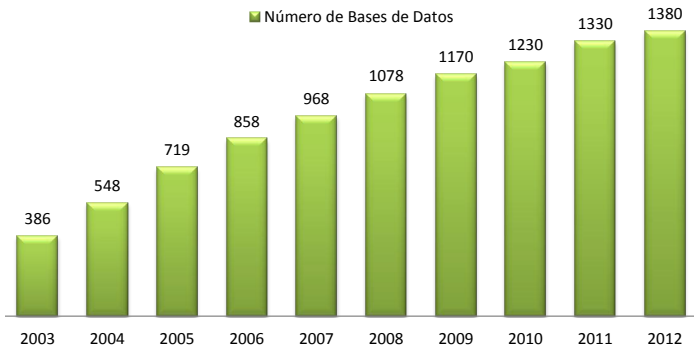

Figura 2.1: Evolución del número de bbdd en el dominio de la biología

Un problema a tener en cuenta en el procesamiento de la información de los repositorios bioinformáticos es la frecuencia y tipos de errores en los datos que contienen. Estos problemas dependen de la naturaleza de la información almacenada, y de si ésta ha sido añadida y comprobada manualmente por un grupo de personas o generada por métodos automáticos. Los errores pueden estar en los datos en sí o en cómo están anotados, por ejemplo una secuencia de nucleótidos puede presentar un error en un nucleótido, o puede estar mal indicada su localización en el genoma, o a qué producto corresponde.

La sección 2.1 describe una clasificación de grupos de bases de datos de información biológica. En ella se describen las características más importantes y se proporcionan varios ejemplos. La sección 2.2 presenta un conjunto de bases de datos especializadas en información sobre ortología y utilizadas para el desarrollo de la tesis. Además en esa sección se define las particularidades de ese tipo de información. Por último, la sección 2.3 describe las fuentes de información sobre enfermedades genéticas utilizadas en la tesis.

2.1. Tipos de repositorios bioinformáticos

Los repositorios bioinformáticos pueden clasificarse según diferentes criterios: (1) el tipo de información que contienen; (2) grado de curación de la información; (3) si están centrados en una especie o en un conjunto de especies; (4) o si los repositorios están libremente disponibles o bajo algún

tipo de licencia.

Según el tipo de información que contienen las bases de datos bioinformáticas, se puede señalar el grupo de las bases de datos de secuencias de nucleótidos, cuyos principales representantes son GenBank [27], ENA y DDBJ [28]. GenBank es una extensa base de datos que almacena gran cantidad de secuencias de ADN disponibles de manera pública sobre una gran cantidad de organismos. Las secuencias son obtenidas mediante contribuciones independientes de laboratorios y mediante grupos de contribuciones provenientes de proyectos de secuenciamiento a gran escala. ENA es la base de datos de secuencias de nucleótidos de EMBL [29]. Las funciones de ENA son las de añadir, organizar y publicar secuencias de nucleótidos disponibles en fuentes de información públicas. DDBJ forma parte del Instituto Nacional Japonés de Genética. Se encuentra disponible en diferentes formatos, incluidos FASTA y XML. Al igual que las anteriores bases de datos es parte del proyecto de colaboración internacional INSDC [30] con el objetivo de establecer las referencias entre los distintos recursos publicados.

Dentro de las bases de datos de secuencias de proteínas destacar el repositorio Uniprot que es el más utilizado y además está disponible públicamente. Uniprot está compuesto, entre otros, por el repositorio de SWISS-PROT y por el repositorio TrEMBL. Por un lado, el repositorio SWISS-PROT está manualmente anotado con el objetivo de evitar la redundancia y de producir anotaciones de gran calidad. Por el otro lado, el repositorio TrEMBL está anotado de manera automática y no está revisado.

Dentro de las bases de datos centradas en una especie concreta, citar a SGD [31], FlyBase [32] o MGD [33]. SGD es la base de datos del genoma de la especie *Saccharomyces cerevisiae*, que contiene información sobre el estudio de la biología molecular y genética de esta levadura. FlyBase está enfocada en la mosca de la fruta, *Drosophila melanogaster*, la cual es una de las especies eucariotas más estudiadas y además sirvió de modelo central en el proyecto del genoma humano [34]. MGD es la base de datos del genoma del ratón, que forma parte del laboratorio Jackson Laboratory [35], y es el principal recurso de información sobre el genoma del ratón, *Mus musculus*. Por otro lado están los metarepositorios que agrupan diversas bases de datos y unifi-

can su acceso y consulta. Dentro de esta categoría destacan tres miembros, NCBI [36], EBI [37] y KEGG [38]. NCBI es el centro nacional de información biotecnológica de los EE.UU. y proporciona acceso a información biomédica y genómica. El Instituto Europeo de Bioinformática es una organización académica sin ánimo de lucro que es parte del Laboratorio Europeo de Biología Molecular (EMBL). El objetivo de este metarepositorio es proporcionar de manera gratuita y libre, información y servicios bioinformáticos a la comunidad científica, para facilitar el progreso científico. El repositorio KEGG, denominado la Enciclopedia de Genes y Genomas de Kyoto, está desarrollado por los laboratorios Kanehisa [39], y está formado por un conjunto de bases de datos que integra información sobre genómica, química y funciones biológicas.

Muchos de estos repositorios (Uniprot, EBI, GenBank) distinguen dos claves que identifican los elementos de las bases de datos de dos maneras diferentes. Uno es el identificador que generalmente puede ser interpretado por científicos, como por ejemplo, las abreviaturas del nombre completo de genes o proteínas (PMVK - phosphomevalonate kinase). La otra clave es el código de acceso, que es la clave primaria utilizada en las bases de datos para identificar los elementos almacenados. Esta distinción de identificadores es utilizada para gestionar la evolución, cambios y eliminación de elementos en las bases de datos. Sin embargo, esta distinción también produce una abundancia de claves, y además según la política del repositorio con las claves, éstas pueden no permanecer coherentes durante el tiempo si se cambian. Iniciativas como el citado proyecto de colaboración internacional INSDC o el proyecto *identifiers.org* [40] tienen como objetivo establecer enlaces entre distintos repositorios bioinformáticos y producir identificadores únicos con los que identificar recursos biológicos de manera global.

2.2. Repositorios de información de ortólogos

Las últimas innovaciones y el rápido desarrollo de nuevas técnicas de secuenciamiento del genoma ha contribuido de manera fundamental en el enriquecimiento de la información existente sobre el éste [41]. Cada genoma

posee un conjunto único de genes que determinan un fenotipo y su interacción con el entorno. Debido al proceso de evolución, el número de especies se ha incrementado a lo largo de los años [42]. Estas especies tienen como origen una forma de vida simple que ha ido enfrentándose a diferentes entornos. Estas variaciones son el resultado del proceso de selección natural, donde los cambios están registrados en sus genomas y proporcionan las claves para entender sus diferencias con otros ancestros comunes. La inferencia de variaciones entre especies mediante el análisis de los conjuntos de genes es la base de la rama de investigación sobre la comparación de genomas [43].

Uno de los aspectos fundamentales de la comparación de genomas es la relación entre la presencia o no de ciertos genes con sus funciones biológicas. Para entender esta relación, primero es necesario reconstruir las relaciones entre genes de especies diferentes en términos evolutivos para después estudiar si cumplen con la misma función o no. El proceso evolutivo mediante la especiación, duplicación y transferencia horizontal de genes complica el estudio de los genomas, por lo que no es un trabajo trivial. Además, ciertos eventos como la eliminación, fusión y fisión de genes añaden aun más complejidad. Sin embargo todo proceso evolutivo puede ser representado en un árbol filogenético [44, 45]. Todo esto debe tenerse en cuenta para el estudio de la evolución biológica de las especies.

Mediante los eventos evolutivos mencionados en el párrafo anterior, las descripciones de la evolución de los genes, de conjuntos de genes y en última instancia del conjunto completo de genes de un organismo, pueden ser definidas con ciertos términos clave sobre la biología evolutiva como los homólogos, ortólogos, parálogos y xenólogos.

Homólogos: Cuando varias secuencias de nucleótidos de organismos de la misma o de diferentes especies son heredados de un antepasado común mediante una transferencia vertical, se dicen que son homólogas. Esta propiedad puede dividirse en otras dos, ortólogos y parálogos, según la transferencia vertical de genoma esté causada por un evento de especiación o duplicación respectivamente.

Ortólogos: Son las secuencias homólogas que han sido diferenciadas me-

diante un evento de especiación. Es decir, cuando una especie diverge por un proceso evolutivo en dos o más especies. Las copias divergentes de un gen en las especies resultantes son consideradas ortólogas.

Parólogos: Son las secuencias homólogas que han sido diferenciadas mediante un evento de duplicación, es decir, cuando un gen de un organismo se duplica, pasa a ocupar dos posiciones diferentes del mismo genoma, con posibles funciones biológicas diferentes. Las copias de los genes entonces son consideradas parálogas.

Xenólogos: Son las secuencias homólogas de nucleótidos que han sido adquiridas por una especie mediante un evento de transferencia horizontal desde otra especie. Es decir, cuando un gen de un organismo pasa a ocupar una posición en el genoma de otro. La copia del gen es considerada xenóloga respecto del gen original.

Aunque estos términos fueron definidos hace mucho tiempo por los evolucionistas, es ahora cuando se han convertido en sujeto de estudio y han provocado diversos malentendidos respecto al desarrollo de la evolución molecular y genómica [46, 47, 48, 49]. Los homólogos, en su definición más general, asocian una relación entre especies, sin aportar ninguna información sobre el escenario evolutivo. Así, los ortólogos, parólogos y xenólogos, definen diferentes subcategorias de homólogos. Los ortólogos están relacionados mediante una relación de especialización evolutiva, o de descendiente vertical en el árbol filogenético. Por otro lado, los parólogos, están relacionados mediante la duplicación y los xenólogos mediante una trasferencia horizontal o lateral. Aunque estos términos suelen definirse utilizando el concepto de gen, también puede aplicarse a cualquier región genómica, desde varios genes juntos a un único nucleótido.

En esta sección se presentará diferentes repositorios que contienen información sobre genes y proteínas ortólogas. Estos repositorios han sido utilizados para el desarrollo de esta tesis y por lo tanto se detallarán sus características, así como el tipo de información que contienen. En concreto, los repositorios analizados son: KOG [50], OrthoMCL [51], HomoloGene [52] e Inparanoid [53].

Fichero	Descripción
kog	Grupos de ortólogos y sus categorías funcionales.
kyva=gb	Correspondencia entre las proteínas y los genes.
kyva	Secuencias de las proteínas utilizadas.
fun.txt	Descripción de las categorías funcionales.
lse	Información del linaje de los grupos ortólogos.
twog	Información sobre los grupos candidatos de dos especies.
pa	Resumen del resultado del análisis de las secuencias.

Tabla 2.1: Tabla con la descripción de los ficheros que componen el repositorio KOG.

2.2.1. KOG

KOG (euKaryotic Orthologous Groups)[1] está compuesto por una colección de grupos de ortólogos de entre siete especies eucariotas. En concreto, de las siete especies, tres son animales: *Caenorhabditis elegans*, *Drosophila melanogaster* y *Homo sapiens*; una planta: *Arabidpsis thaliana*; dos hongos: *Saccharomyces cerevisiae* y *Schizosaccharomyces pombe*; y un microorganismo intracelular parásito: *Encephalitozoon cuniculi*. En el momento de redacción de esta tesis el repositorio contiene información sobre 4852 grupos de ortólogos, unos 110655 genes y unas 59838 proteínas[2]. La información de este repositorio está disponible en ficheros de texto semi estructurados. La Tabla 2.1 muestra los ficheros disponibles y una pequeña descripción con la información que contienen.

El fichero *kog* contiene la información sobre los grupos de ortólogos (véase Figura 2.2). Cada grupo posee una clave única, una descripción, un conjunto de categorías funcionales identificadas cada una por un carácter y un conjunto de proteínas o genes junto con su especie. En la Figura 2.2 se observa el grupo *KOG0373* asociado a *[DT]* que se corresponde a las categorías funcionales *D* (control del ciclo celular, división celular o partición del cromosoma) y *T* (mecanismos de transducción de señales). Además, posee una descripción textual de la función relacionada con el grupo. Los ortólogos están definidos mediante un código de tres letras que identifica a la especie, ":" y el iden-

[1] `http://www.ncbi.nlm.nih.gov/COG/grace/shokog.cgi`
[2] `ftp://ftp.ncbi.nih.gov/pub/COG/KOG/`

tificador del ortólogo. Por ejemplo, "*ath: At1g50370*", se corresponde con el gen *At1g50370* de la especie *Arabidopsis thaliana*, o "*hsa: Hs4506029*", que se corresponde con el gen *Hs4506029* de la especie *Homo sapiens*

```
[DT] KOG0373 Serine/threonine specific protein phosphatase
     involved in cell cycle control, PP2A-related
  ath:  At1g50370
  ath:  At3g19980
  cel:  CE02156
  dme:  7290716
  hsa:  Hs4506029
  sce:  YDL047w
  spo:  SPCC1739.12
```

Figura 2.2: Grupo de ortólogos *KOG0373*.

2.2.2. OrthoMCL

OrthoMCL[3] es una base de datos de grupos de proteínas ortólogas. La primera versión de este repositorio fue publicada en 2005 y contenía 70388 grupos de proteínas ortólogas de 55 especies diferentes. La segunda versión del repositorio fue lanzada en 2008, y contenía información de 87 especies, así como mejoras en el proceso de actualización que permitían reducir el tiempo de actualización. La tercera versión contenía 128 especies, la cuarta 138 y, por último, la versión más reciente dispone de información de 150 especies. La última versión disponible de OrthoMCL, en el momento de redacción de esta tesis, contiene 124740 grupos de ortólogos, con 1192387 proteínas ortólogas, lo que supone un gran aumento de información respecto a la primera versión.

La información de este repositorio sobre los grupos de ortólogos está disponible en ficheros de texto semi estructurados. La Tabla 2.2 describe los ficheros disponibles[4] en este repositorio.

El fichero *groups_OrthoMCL-5.txt.gz* contiene la información sobre los grupos de ortólogos. Cada grupo posee una clave única y el conjunto de pares especie - proteína pertenecientes al grupo ortólogo. En la Figura 2.3 se observa que el grupo *OG5_159819* contiene siete proteínas ortólogas (cada

[3] `http://www.orthomcl.org/`

[4] `http://orthomcl.org/common/downloads/`

Fichero	Descripción
groups_OrthoMCL-5.txt.gz	Grupos de proteínas ortólogas.
aa_deflines_OrthoMCL-5.txt.gz	Descripción de las proteínas.
aa_seqs_OrthoMCL.fasta.gz	Secuencias de las proteínas utilizadas.
data_source_OrthoMCL.txt.gz	Fuentes de información donde se obtuvieron las secuencias.
iprscan_OrthoMCL-5.txt	Resumen del resultado del análisis de las secuencias.

Tabla 2.2: Tabla con la descripción de los ficheros que componen el repositorio OrthoMCL.

grupo de ortólogos está almacenado en una línea del fichero, aunque por claridad en la Figura 2.3 se ha presentado en varias líneas). La especie correspondiente a cada ortólogo está representada por cuatro caracteres. Por ejemplo, "*mmul*" representa a la especie *Macaca mulatta*, o "*hsap*" representa a la especie *Homo sapiens*.

```
OG5_159819:
 mmul|XP_001083774
 hsap|ENSP00000380250
 clup|ENSCAFP00000015668
 ecab|XP_001496411
 mmus|ENSMUSP00000039062
 ptro|ENSPTRP00000042409
 rnor|ENSRNOP00000029152
```

Figura 2.3: Grupo de ortólogos *OG5_159819*.

2.2.3. HomoloGene

HomoloGene[5] es una base de datos que almacena grupos de ortólogos y parálogos. Para encontrar los grupos de ortólogos, se utiliza el método de la sintenia [54], que se basa en la comparación de estructuras generales de genomas de diferentes especies, intentando encontrar bloques de secuencias similares dentro de un mismo cromosoma para especies diferentes. En el caso de los parálogos, la búsqueda se realiza con secuencias del cromosoma de la misma especie.

[5] `http://www.ncbi.nlm.nih.gov/homologene`

Fichero	Descripción
homologene.data	información de los grupos de homólogos.
all_proteins.data	Tabla con identificadores y descripción de las secuencias utilizadas.
proteins_for_clustering.data	Tabla con los identificadores de las secuencias agrupadas.
taxid_taxname	Tabla con los identificadores y nombres de las especies analizadas.
homologene.xml.gz	Documento XML con la información de los grupos de homólogos.

Tabla 2.3: Tabla con la descripción de los ficheros que componen el repositorio HomoloGene.

HomoloGene realiza la búsqueda de ortólogos utilizando el genoma de veinte especies diferentes. El número de ortólogos encontrados es de 244950, los cuales están agrupados en 43998 grupos[6] diferentes. Permite obtener la información utilizada para la construcción del repositorio, así como la información resultado en ficheros semi estructurados. La Tabla 2.3 muestra los ficheros disponibles y una breve descripción de cada uno de ellos.

El fichero *homologene.data* contiene la información sobre los grupos de ortólogos y parálogos. Cada línea del documento (véase Figura 2.4) describe un homólogo. En concreto, cada homólogo está descrito mediante el identificador del grupo al que pertenece, el identificador de su especie, el código de acceso del gen utilizado en el repositorio, su nombre descriptivo, el código de acceso de su proteína correspondiente y, por último, el nombre de dicha proteína. En la Figura 2.4 se observa un extracto del fichero *homologene.data* que contiene la información del grupo de homólogos número *7*. Este grupo contiene la información de 10 genes ortólogos, ya que no contiene más de un gen por especie, junto con la información de las proteínas que codifican y que fueron utilizadas para realizar la comparación. Por ejemplo, la línea "*7 9606 90 ACVR1 166235898 NP_001104537.1*" representa a un miembro del grupo identificado por el número *"7"*; pertenece a la especie *"9606"*, que es

[6] `ftp://ftp.ncbi.nih.gov/pub/HomoloGene/current/`

el identificador utilizado en la taxonomía del NCBI[7] para la especie *Homo sapiens*; el código de acceso *90* y el nombre *ACVR1* se corresponden con su gen *"activin A receptor, type I"*; y codifica a la proteína con código de acceso *"166235898"* y nombre *"NP_001104537.1"*.

```
6 4530 4346520 Os09g0252100 115478180 NP_001062685.1
6 5833 812066 PF14_0484 124809703 XP_001348658.1
7 9606 90 ACVR1 166235898 NP_001104537.1
7 9598 470565 ACVR1 114581323 XP_001145240.1
7 9615 478757 ACVR1 74004332 XP_856152.1
7 9913 338068 ACVR1 125991744 NP_788836.2
7 10090 11477 Acvr1 40254649 NP_031420.2
7 10116 79558 Acvr1 13324702 NP_077812.1
7 9031 395246 ACVR1 46048926 NP_989891.1
7 7955 30615 acvr1l 18858267 NP_571420.1
7 7227 35731 sax 24586379 NP_523652.2
7 7165 1269509 AgaP_AGAP007729 158285107 XP_308147.4
9 9606 6442 SGCA 4506911 NP_000014.1
9 9598 468403 SGCA 114669439 XP_523792.2
```

Figura 2.4: Extracto del fichero *homologene.data* con la información del grupo de ortólogos número siete.

2.2.4. Inparanoid

Inparanoid es una base de datos que almacena información sobre genes "inparalogs". La mayoría de las técnicas para descubrir ortólogos [55, 56, 57, 58, 59, 60, 61] tiene éxito cuando hay una copia de un gen en cada especie analizada. Mediante la distinción entre *"inparalogs"* y *"outparalogs"*, el algoritmo utilizado para la construcción del repositorio puede identificar relaciones entre ortólogos, de uno a muchos y de muchos a muchos.

Inparalogs: Secuencias genómicas de una especie que han sido duplicadas después de un evento de especiación. Así, pueden ser considerados como ortólogos de una o más secuencias en otra especie, ya que provienen de un ancestro común.

Outparalogs: Secuencias genómicas de una especie que han sido duplicadas antes de un evento de especiación y, por lo tanto, no pueden ser considerados ortólogos.

[7] `http://www.ncbi.nlm.nih.gov/taxonomy`

Inparanoid agrupa 1687023 secuencias de 100 especies diferentes obtenidas de los repositorios Ensembl [62], JGI[8] FGI[9], Flybase [63], NCBI [64], WormBase [65], Sanger[10], Génolevures Consortium [66], TIGR[11], VectorBase [67], PlasmoDB [68], CryptoDB [69], GiardiaDB [70], Panther [71], Rice Genome Annotation Project [72], Dictybase [73], CGD [74], University of Tokyo[12], SilkDB [75], SGD [76], SGTC[13] y TAIR [77].

Inparanoid compara las secuencias de las distintas especies de dos en dos. Por lo tanto, el repositorio proporciona el resultado de comparar las secuencias de una especie con las de otra en ficheros separados. Así, la combinación de las cien especies comparadas de dos en dos nos da como resultado unos 4950 ficheros[14] de grupos ortólogos. Estos ficheros indican en el nombre del fichero las especies comparadas. Por ejemplo, el fichero *InParanoid.A.aegypti-A.fumigatus* indica que las secuencias de las especies *Aedes aegypti* y *Aspergillus fumigatus* han sido procesadas para obtener los grupos de ortólogos. En la Figura 2.5, se muestra un grupo de ortólogos definido dentro del fichero *InParanoid.A.aegypti-A.fumigatus*. En él se encuentra el ortólogo *"AAEL012584-PA"* de la especie *Aedes aegypti* y el ortólogo *"AFUA_6G13690.t1"* de *Aspergillus fumigatus*. Además, se proporciona el índice de confianza entre los ortólogos, que en este ejemplo es del 100 %, lo que representa que existe una similitud idéntica en sus secuencias y, por lo tanto, la relación entre los ortólogos es muy grande.

```
Group of orthologs #23. Best score 1120 bits
Score difference with first non-orthologous sequence -
                          A.aegypti.fa:1120 A.fumigatus.fa:1120
 AAEL012584-PA        100.00% AFUA_6G13690.t1      100.00%
 Bootstrap support for AAEL012584-PA as seed ortholog is 100%.
 Bootstrap support for AFUA_6G13690.t1 as seed ortholog is 100%.
```

Figura 2.5: Extracto del fichero *InParanoid.A.aegypti-A.fumigatus* con la información del grupo de ortólogos.

[8] http://www.jgi.doe.gov/
[9] http://www.broadinstitute.org/annotation/fungi/fgi/
[10] http://www.sanger.ac.uk/
[11] http://www.tigr.org/
[12] http://merolae.biol.s.u-tokyo.ac.jp/
[13] http://med.stanford.edu/sgtc/
[14] http://inparanoid.sbc.su.se/download/7.0_current/table_stats/

2.3. Repositorios de información sobre enfermedades genéticas

2.3.1. OMIM

OMIM [78] (Online Mendelian Inheritance in Man) es un repositorio que en la versión más actual, al día de redacción de esta tesis, dispone de información sobre unas 20757 enfermedades genéticas y de los correspondientes genes relacionados[15]. En ella se establece la relación entre los genes humanos y los fenotipos genéticos que producen. OMIM está centrado en la relación fenotipo-genotipo. El origen de esta base de datos data de la década de los 60, cuando el Dr. Víctor McKusick comenzó un catálogo de enfermedades y rasgos genéticos. Sin embargo, la versión on-line del repositorio surgió en 1985 mediante la colaboración del NCBI y el William H. Welch Medical Library[16] de la Universidad Johns Hopkins.

En el repositorio se distinguen cinco perspectivas diferentes en las que están estructuradas las descripciones de las enfermedades. El primer tipo está centrado en la descripción de un gen. El segundo tipo en la descripción combinada del gen y el fenotipo. El tercer tipo está centrado en la descripción del fenotipo y su base molecular conocida. El cuarto es similar al tercero, salvo que proporciona un código o identificador del fenotipo. Por último, las descripciones de rasgos fenotípicos con posibles orígenes hereditarios.

Para obtener la información del repositorio OMIM es necesario solicitar una licencia[17]. La información se almacena en los ficheros descritos en la Tabla 2.4.

La Figura 2.6 muestra varias líneas del fichero *genemap*. Cada línea representa una relación entre un gen y una enfermedad genética. La información en cada línea está delimitada por "|", y está compuesta de 18 apartados descritos en la Tabla 2.5. Para cada entrada del repositorio, se indican los métodos por los que se obtuvieron las evidencias de las correspondencias del genotipo

[15] `http://www.omim.org/`
[16] `http://www.welch.jhu.edu/`
[17] `http://www.omim.org/downloads`

Fichero	Descripción
omim.txt.Z	Contiene la información sobre las enfermedades almacenadas en OMIM en lenguaje natural.
genemap	Resumen de los genes ordenados según su posición en el cromosoma y sus enfermedades relacionadas.
genemap.key	Descripción de los elementos que aparecen en el fichero *genemap*.
morbidmap	Resumen de la relación de enfermedades ordenadas alfabéticamente.
mim2gene.txt	Tabla con los identificadores de las enfermedades, los códigos de acceso de los genes del NCBI y los nombres de gen de HGNC [79].

Tabla 2.4: Tabla con la descripción de los ficheros que componen el repositorio OMIM.

con el fenotipo. Por ejemplo, "*1.90|8|25|04|1p36.2|PEX14|P|Peroxisome biogenesis factor 14||601791|R, REc|||Zellweger syndrome, 214100 (3)||||*" es la descripción de una correspondencia entre el gen *PEX14*, que tiene el estado de provisional, y la enfermedad genética del *Síndrome de Zellweger*. El código OMIM para acceder a esta descripción es *601791*. Por último, los métodos por los que se obtuvo esta correspondencia fueron: la irradiación de células, seguido de su rescate a través de su fusión con células no irradiadas (R); y la hibridación de cADN a un fragmento genómico.

```
1.89|5|9|95|1p36.2|NPPB,BNP|C|Natriuretic peptide precursor B||600295|
H,REa,A,REn||||||4(Nppb)|
1.90|8|25|04|1p36.2|PEX14|P|Peroxisome biogenesis factor 14||601791|
R,REc|||Zellweger syndrome,214100(3)||||
1.91|9|11|02|1p36.2|PIK3CD|P|Phosphatidylinositol 3-kinase,catalytic,
110kD,delta||602839|A,R|||||||
```

Figura 2.6: Extracto del fichero *genemap* con el resumen de la relación entre genes y enfermedades hereditarias de OMIM.

Apartado	Descripción
1	Clave primaria de la entrada.
2	Mes en el que se insertó la entrada.
3	Día en el que se insertó la entrada.
4	Año en el que se insertó la entrada.
5	Localización del gen en el cromosoma.
6	Nombre del gen.
7	Estado del gen, confirmado, provisional, inconsistente o sin evidencias.
8	Descripción de la entrada.
9	*Sin uso*
10	Identificador de la entrada en el repositorio OMIM.
11	Códigos de métodos para la correspondencia de los genes.
12	Comentarios.
13	*Sin uso*
14	Nombre de la enfermedad.
15	Nombre de la enfermedad, continuación.
16	Nombre de la enfermedad, continuación.
17	Correspondencias con genes de la especie *Mus musculus*.
18	Referencias.

Tabla 2.5: Tabla con la descripción de la estructura del fichero *genemap* del repositorio OMIM.

Capítulo 3

La Web semántica

La Web semántica fue definida por primera vez por Tim Berners-Lee [9] como una extensión sobre la Web actual, en la cual la información está dotada de un significado bien definido con el fin de que se permita trabajar con ella de manera cooperativa entre personas y ordenadores.

Para entender el proceso de evolución hacia la Web semántica, a continuación se abordan los orígenes de la Web, después se describirán las principales características de la Web actual y sus inconvenientes, y por último se analizará cómo la Web semántica pretende dar solución a dichos inconvenientes.

La Web tuvo su origen como un proyecto de gestión de documentación basado en hipertexto y llevado a cabo por Tim Berners-Lee y Robert Cailliau en el CERN [80]. Este sistema se basaba en el uso de identificadores únicos globales, llamados URI [81] (Uniform Resource Identifier), para la identificación de recursos en Internet y del desarrollo del lenguaje de hipertexto, llamado HTML [82]. Haciendo uso de este lenguaje se podían publicar recursos en la web y establecer enlaces unidireccionales entre ellos. La Web actual posee una gran capacidad para almacenar datos, consultar y visualizar contenidos, pero no es capaz de entender ni inferir nueva información con la información que contiene. Esta limitación reside en que la Web fue diseñada para la lectura humana y no para que su contenido fuera procesado por máquinas. La Web 2.0 no puede considerarse una evolución de la Web actual sino como un término que describe el cambio de tendencia en el sistema de distribu-

ción y producción de información. Tradicionalmente, éste ha estado basado en una relación cliente-servidor donde un agente producía contenidos y los usuarios consumían esos contenidos. Sin embargo, en los últimos años han surgido sistemas colaborativos donde los usuarios son clientes y productores de información, pero los inconvenientes descritos de la Web original siguen presentes porque la representación de información no ha cambiado. Por otro lado, actualmente la Web semántica pretende solucionar estos inconvenientes dotando de más significado a los recursos publicados en la Web mediante el uso de lenguajes semánticos, como RDF y OWL [83].

Así pues, la Web semántica pretende describir semánticamente la información publicada en la Web. Esta descripción no sólo contiene la definición de cómo está estructurada, sino que también permite asociar propiedades y conectar recursos de la Web a través de relaciones. Además, permite definir restricciones, axiomas y reglas que aporten mayor conocimiento sobre su dominio para su procesamiento en tareas de inferencia de manera automática. También, promueve la definición y reutilización de vocabularios comunes para facilitar el procesamiento de información. La compartición de un vocabulario común facilita la gestión de repositorios de información, ya que define formalmente y de manera lógica el conocimiento presente en ellos. Este conocimiento proporciona un contexto a la información, y además permite reutilizar y enlazar este conocimiento con otras descripciones formales enriqueciendo su contenido.

La Web semántica propone una arquitectura estándar que se divide en capas o niveles (véase Figura 3.1) para que usuarios y agentes software autónomos puedan procesar, razonar y hacer deducciones lógicas sobre el contenido de la web de datos.

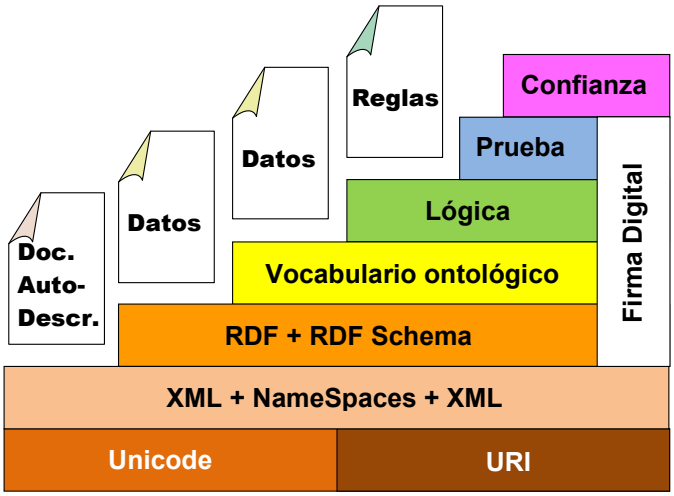

Figura 3.1: Arquitectura de la Web semántica

La capa *Unicode* se encarga de estandarizar un alfabeto que permita utilizar caracteres internacionales para poder codificar la información en cualquier idioma.

La capa *URI* proporciona un identificador uniforme a los recursos a publicar y éste permite además describir el espacio de nombre del recurso.

La capa *XML*, junto con el espacio de nombres (*NameSpace*) y el esquema XML (*XMLSchema*), sirve como base para la Web semántica. XML ofrece un formato común de documentos. El espacio de nombres sirve para cualificar elementos y atributos de nombres, y asociarlos con los espacios de nombres identificados en las URI. Los esquemas XML describen documentos XML que sirven como plantilla para la descripción de documentos en un formato común.

La capa *RDF* (Resource Description Framework) y *RDF Schema* se basan en la capa anterior para proporcionar un lenguaje universal con el que describir el conocimiento en la Web semántica mediante tripletas. Los esquemas RDF (RDFS) describen grafos dirigidos que están compuestos de nodos y relaciones, y que indican cómo se relaciona la información asociada a un dominio particular.

La capa *Vocabulario Ontológico* dota de mayor funcionalidad y expresivi-

dad a la capa RDF. Proporciona nuevos conceptos, relaciones y propiedades con los que conceptualizar un dominio concreto del conocimiento de manera más precisa.

La capa *Lógica* está basada en la posibilidad de definir reglas de inferencia. Estas reglas permiten a los agentes software procesar la información descrita y así ofrecer a los usuarios un procesamiento automático de la información a nivel semántico.

La capa *Pruebas* tiene como objetivo el intercambio de hechos y reglas de manera estándar para la interoperabilidad entre recursos de la Web semántica.

Por último, la capa *Confianza* evalúa las pruebas ofrecidas por la capa anterior y permite establecer si son confiables. Por lo tanto, esta capa tiene como misión evaluar la confiabilidad de los recursos, de manera que ésta sea verificable por agentes Web. La funcionalidad del componente *Firma Digital* debe permitir a las capas *Pruebas* y *Confianza* definir su ámbito de confianza de manera que los agentes software puedan verificar la seguridad de la información proporcionada en la Web.

Cabe destacar, dentro del desarrollo de la Web semántica, el proyecto OntoClean [84]. OntoClean es una metodología para la validación de la capacidad ontológica de relaciones taxonómicas. Así, esta metodología define formalmente los conceptos de identidad, esencia, y unidad, a través de metapropiedades, que caracterizan los elementos de las ontologías como las propiedades, clases y relaciones [85, 86, 87]. Esta metodología establece los criterios para evaluar la corrección de las definiciones de ontologías en la Web semántica.

OntoClean define el concepto de identidad a través del conjunto de propiedades de identificación que las distintas descripciones de un mismo objeto tienen en común. Estas propiedades de identificación son las características o relaciones que permiten identificar a una instancia dentro de las instancias de su clase. Por ejemplo, en el modelo relacional, las instancias vienen dadas por las claves primarias de cada tabla, mientras que en el dominio de la biología, un gen podría ser identificado mediante su secuencia de nucleótidos y su posición en el cromosoma.

El concepto de esencia viene definido por las condiciones necesarias, pero no suficientes, para considerar dos objetos idénticos. Así, este concepto está estrechamente relacionado con las propiedades de identificación. Por lo tanto, la manera más simple de establecer la diferencia de identidad entre objetos es mediante sus propiedades esenciales.

Otro concepto importante en la metodología OntoClean es el concepto de unidad. Este concepto viene asociado al problema de identificar qué es la parte de un objeto, qué no lo es y cuándo puede ser considerado el objeto como un todo o unidad. A las condiciones que tienen que cumplir las partes de un objeto para que éste sea considerado una unidad se las denominan criterios de unidad y se expresan a través de las relaciones de unidad. Además, cada parte de un objeto puede ser al mismo tiempo un objeto compuesto de otras partes. Por ejemplo, una célula eucariota es una unidad que está compuesta del núcleo celular, orgánulos y estructuras celulares, y el núcleo celular está, a su vez, compuesto de la membrana celular y el nucléolo.

En las siguientes secciones se presentarán las herramientas y proyectos asociados con el desarrollo de la Web semántica. Así, en la sección 3.1 se describirán diversas tecnologías desarrolladas para la Web semántica. La sección 3.2 aborda la iniciativa de Linked Data, que consiste en un conjunto de principios para la publicación e intercambio de datos en la Web y que tiene como objetivo el desarrollo de la Web semántica.

3.1. Tecnologías de la Web semántica

El elemento central en el desarrollo de la Web semántica es la utilización de lenguajes para describir la semántica de la información que se pretende publicar en la Web. Así, el lenguaje RDF (presentado en la sección 3.1.1) permite definir grafos RDF que representan cómo los conceptos están relacionados en el mundo real.

Sin embargo, para dotar de mayor semántica a la Web que la proporcionada por RDF, se desarrollaron los lenguajes ontológicos, que son descritos en la sección 3.1.2. En este trabajo, el foco se ha puesto en OWL, que es el estándar adoptado por el W3C [88]. OWL tiene como base la lógica descrip-

tiva [89] y proporciona el vocabulario necesario para representar y compartir conocimiento.

Por otro lado, para poder gestionar el conocimiento definido en las ontologías y la información anotada con ellas se utilizan las interfaces de programación de ontologías, como OWL-API [90] y Jena [91]. En la sección 3.1.3, se ofrece una descripción de cada uno de estas interfaces de programación.

En la sección 3.1.4 se presentan diversos razonadores desarrollados para su aplicación sobre definiciones de ontologías en OWL. Estos razonadores pueden ser utilizados a través de las interfaces de programación de ontologías, descritas en la sección 3.1.3, para poder realizar funciones de inferencia y comprobaciones de la coherencia del modelo ontológico.

La sección 3.1.5 describe el lenguaje de consulta SPARQL, definido para acceder a la información almacenada en una ontología. Por último, la sección 3.1.6 describe el lenguaje de actualización SPARUL [92], que utiliza la misma sintaxis que SPARQL pero que es utilizado para crear, borrar y modificar tanto el grafo como el contenido de las ontologías.

3.1.1. Lenguaje RDF

El elemento básico del lenguaje RDF es la tripleta. Una tripleta está formada por dos nodos (sujeto y objeto) unidos por un arco (predicado), donde los nodos representan recursos, y los arcos relaciones o propiedades. Para identificar globalmente los recursos y las propiedades se utilizan las URI. Las URI se componen de un espacio de nombres y el identificador que debe ser único dentro de su espacio de nombres. Los sujetos de las tripletas deben ser nodos identificados por una URI. Sin embargo, los objetos de las tripletas pueden ser tanto nodos como valores de propiedades. Los elementos de un grafo RDF como son los nodos, propiedades y relaciones tienen asociado una URI para poder identificarlos.

De esta manera, encadenando los nodos objeto de las tripletas con los nodos sujeto de otras tripletas se construyen los grafos RDF. Los grafos RDF permiten describir la semántica de un dominio. Además, RDFS provee de un vocabulario definido para RDF, y que permite dotar de una semántica bien

definida a RDF, proporcionando metadatos. RDFS aporta relaciones bien definidas con las que definir jerarquías de clases y propiedades, o especificar las propiedades y relaciones entre conceptos. Por ejemplo, la Figura 3.2 representa un grafo RDF donde se puede observar las diferencias entre el grafo RDF y el RDFS. En el ejemplo, se puede observar que el modelo del grafo RDF se compone de la clase *"Homo sapiens"*, que se define como sub clase de *"Especie"* mediante la relación *"rdfs:subClassOf"* definida en RDFS. Por otro lado, la clase *"Gen"* tiene definida la relación *"Pertenece"*, definida en el grafo RDF, con la clase *"Especie"*, y que indica que cada gen pertenece a una especie concreta. Las instancias del modelo son *"Homo sapiens sapiens"*, para la clase *"Homo sapiens"*, y *"Brca1"*, para la clase *"Gen"*. Estas instancias están definidas utilizando la relación *"rdfs:type"*, que indica la clase asociada a cada instancia. Por último, las instancias están enlazadas a través de la relación pertenece, y que no tiene una semántica definida previamente, como en las definidas en RDFS.

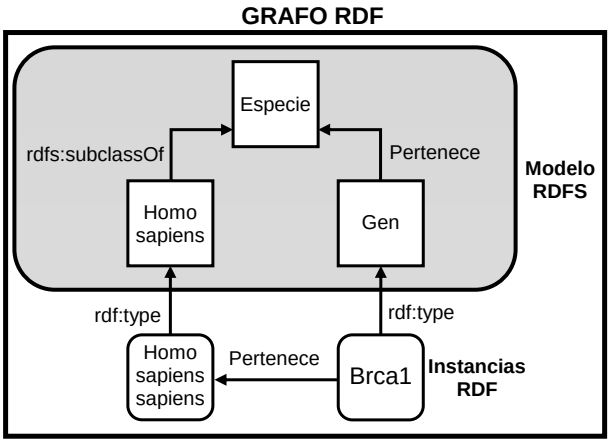

Figura 3.2: Ejemplo de grafo RDF y RDFS

3.1.2. Ontologías

Las ontologías representan una visión consensuada, compartible y reutilizable del conocimiento de un dominio. En la literatura, existen numerosas definiciones de ontología [93, 94, 95]. Una ontología es esencialmente un modelo

conceptual de información formal y estructurada. Una ontología está formada por conceptos, atributos, relaciones, individuos y restricciones. Los conceptos son las entidades más relevantes de un dominio y son equivalentes a las clases en Orientación a Objetos (OO). Los atributos, que también se corresponden con los atributos de la OO, pueden ser específicos, si son definidos para un concepto en particular o heredados, si los atributos son heredados a través de sus relaciones con otros conceptos del dominio. Las relaciones son los nexos de unión entre conceptos que estructuran un dominio. Éstas pueden tener asociadas un dominio, que identificará el conjunto posible de conceptos origen de la relación, y un rango, que representará al conjunto de conceptos destino de la relación. Los individuos son las instancias de los conceptos definidos en un modelo ontológico. Éstos estarán asociados al menos a un concepto de la ontología, pueden tener asignados valores en los atributos de su concepto y además estar relacionados con otros individuos. Por último, las restricciones son reglas semánticas definidas sobre el modelo ontológico y que deben respetar los elementos de éste para ser considerado consistente.

En la práctica, uno de los motivos básicos por los que la ontología ha cobrado tanta importancia y se ha extendido su uso, es por las ventajas que posee [96]: (1) reusabilidad, una misma ontología se puede reutilizar en diversas aplicaciones de forma individual o combinada con otras; y (2) compartición, el conocimiento que incluye permite que sea compartido por una determinada comunidad. Sin embargo, la definición de una ontología no es una tarea sencilla y requiere de bastante experiencia conseguir que ésta refleje adecuadamente el conocimiento de un dominio. A la hora de afrontar la integración de datos, las ontologías pueden ayudar a la comprensión del dominio mediante la introducción de jerarquías de conceptos, la definición de relaciones entre ellos, la descripción de sus propiedades, y la definición de reglas lógicas para la formalización de las condiciones y restricciones del dominio.

Los lenguajes de ontologías surgieron como un medio para proporcionar la capacidad de representación del conocimiento. El primer lenguaje para la definición de ontologías para la Web fue SHOE [97], que fue creado por Jim Hendler en 1997. Tras la aparición de SHOE, otras tecnologías surgieron

Concepto RDF	Concepto OWL	Descripción
rdfs:Class	owl:Class	La clase de las clases OWL.
rdf:Property	owl:ObjectProperty	La clase de las relaciones OWL.
rdf:Property	owl:DatatypeProperty	La clase de las propiedades OWL.
rdf:Property	owl:AnnotationProperty	La clase de las anotaciones OWL.

Tabla 3.1: Tabla que relaciona los elementos RDF con los de OWL.

con una finalidad similar [98], de entre las cuales las más representativas son RDF, DAML+OIL [99], y por último OWL.

OWL

OWL (Ontology Web Language) es un lenguaje de ontologías basado en RDF. Éste mejora la funcionalidad de otros lenguajes semánticos anteriores como DAML+OIL. El primer borrador de la especificación de este lenguaje (OWL 1.0) apareció en julio de 2002 y se presentó de manera formal por el W3C en febrero de 2004. En octubre de 2009 apareció la versión más actual en el momento de redacción de esta tesis, la 2.0.

OWL extiende la semántica definida en RDF y RDFS. De este modo, puede utilizar la misma estructura de tripletas para representar la información, pero añadiendo nuevo vocabulario semántico al definido en RDF y RDFS. En OWL se pueden utilizar todos los elementos definidos en RDF y RDFS. Sin embargo, se ha definido un vocabulario semántico específico para representar las ontologías y sus elementos en OWL. La Tabla 3.2 muestra los elementos RDF que con más frecuencia aparecen en OWL. La Tabla 3.1 muestra los conceptos de RDF que han sido extendidos en OWL para ampliar su semántica. Por último, la Tabla 3.3 muestra una serie de conceptos nuevos, definidos en OWL y que no estaban presentes en RDF, necesarios para enriquecer la semántica de las ontologías OWL.

OWL define el concepto de ontología (*owl:Ontology*), que es el primer elemento que se declara en un documento RDF/OWL. La declaración de una ontología puede tener asociados diferentes propiedades o relaciones, por ejemplo la propiedad *owl:imports* permite definir que se importa la definición de otra ontología en ella. Además, OWL predefine dos clases que estarán

Concepto RDF	Descripción
rdf:type	Relación que indica el tipo de un recurso.
rdfs:subClassOf	Relación que establece el orden de sub-clase de una clase.
rdfs:subPropertyOf	Relación que establece el orden de sub-propiedad de una propiedad.
rdfs:label	Propiedad que establece un nombre descriptivo de un recurso.
rdfs:comment	Propiedad que establece una descripción en lenguaje natural de un recurso.
rdfs:Literal	La clase de los valores literales.

Tabla 3.2: Tabla que muestra los elementos RDF reutilizados en OWL.

Tipos	Ejemplos	Descripción
Clases de relaciones semánticas	AllDisjointClasses, AllDifferent, AllDisjointProperties	Clases que describen semánticamente las relaciones entre individuos, clases o propiedades en las ontologías.
Nuevas Clases semánticas	Ontology, NamedIndividual, Thing	Nuevas clases con semántica propia en las ontologías.
Clases de propiedades semánticas	FunctionalProperty, AsymmetricProperty, TransitiveProperty	Clases que describen semánticamente las propiedades de las ontologías.
Nuevas propiedades semánticas	inverseOf, equivalentClass, sameAs	Propiedades con una semántica definida en las ontologias.

Tabla 3.3: Tabla que muestra los tipos de conceptos, con semántica bien definida, más representativos de las ontologías OWL.

presentes en todas las ontologías de manera implícita: (1) *owl:Thing*, que es la clase raíz de la ontología, por lo que todas las clases son sub-clases de ella; y (2) *owl:Nothing*, que es sub-clase de todas las clases en la ontología y no puede ser súper-clase de ninguna. Cabe destacar, además de los conceptos definidos en la Tabla 3.1, la definición *owl:NamedIndividual*, que es la clase de todos los individuos, y *owl:Restriction*, que es la clase de todas las restricciones definidas en las ontologías OWL.

Por otro lado, las diferentes versiones de OWL han definido diferentes sub-lenguajes expresivos con los que definir las ontologías. Cada sub-lenguaje posee ciertas características que los harán apropiados según su aplicación. Así, en la primera versión de OWL se definieron tres tipos de sub-lenguajes con una expresividad creciente. En concreto, OWL 1.0 definió: OWL Lite, OWL DL y OWL Full, que se explican a continuación.

- OWL Lite: es el sublenguaje más simple. Comparado con RDFS, añade restricciones de rango local, restricciones existenciales, restricciones de cardinalidad simple y varios tipos de propiedades (inversa, transitiva y simétrica). Está destinado a usuarios que necesiten sobre todo una jerarquía de clasificación y restricciones sencillas, como en el caso de taxonomías y tesauros.

- OWL DL: proporciona la máxima expresividad, garantizando la completitud computacional y la decidibilidad en un tiempo finito para tareas de inferencia. OWL DL incluye todas las construcciones de OWL pero respetando ciertas limitaciones como la condición de separación en el tipo de recurso. Su nombre proviene de la correspondencia que mantiene con la lógica descriptiva.

- OWL Full: está destinado a usuarios que desean una expresividad máxima y la libertad sintáctica ofrecida por RDF, aunque sin garantías de cómputo. Permite que una ontología aumente el significado del vocabulario predefinido en RDFS u OWL.

La Figura 3.3 muestra la relación existente entre los sublenguajes. Los lenguajes más simples están contenidos en los más complejos, y se puede

observar cómo la expresividad aumenta.

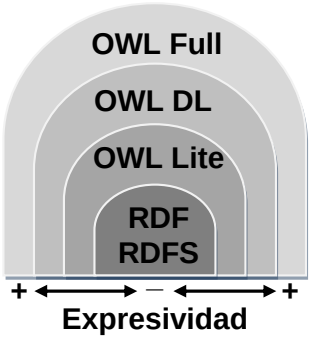

Figura 3.3: Diferentes sub-lenguajes definidos en OWL 1.0

En la versión 2.0 de OWL se sigue una estructura similar a la de su versión 1.0. La compatibilidad de las ontologías definidas en la versión 1.0 con la versión 2.0 es prácticamente completa. OWL 2.0 añade nuevas funcionalidades a las existentes en la versión 1.0, como la definición de claves en las clases, cadenas de propiedades, tipos de datos y rangos de datos más complejos, restricciones de cardinalidad cualificadas, propiedades asimétricas, reflexivas y disjuntas, y mejora de las características de las anotaciones. Por otro lado, cambia la estructura de los sub-lenguajes de la versión 1.0 al concepto de perfil. En la versión OWL 2.0 se definen tres perfiles, siendo cada uno más restrictivo que OWL DL (véase Figura 3.4). Cada perfil intenta explotar aspectos diferentes del poder expresivo de OWL para utilizarlos según las necesidades de los usuarios. Los perfiles son *OWL 2 EL*, *OWL 2 QL* y *OWL 2 RL*.

- OWL 2 EL: perfil cercano a la lógica descriptiva EL++, que asegura un tiempo polinomial para la resolución de problemas de razonamiento. Es particularmente útil en aplicaciones que emplean ontologías que contienen un gran número de propiedades y/o clases. Es sencillo de implementar y permite una gran escalabilidad para expresiones complejas, aunque la expresividad que proporciona es bastante limitada. Por ejemplo, no permite el uso de cuantificadores universales.

- OWL 2 QL: destinado a aplicaciones que utilizan un gran número de instancias de datos y en los que la consulta es la tarea de razonamiento más importante. Es una variante de OWL-Lite, muy habitual en tareas de integración en bases de datos. Resulta muy sencillo extender los habituales lenguajes relacionales, incorporando consultas con los axiomas definidos por el subconjunto. Finalmente, facilita el mapeo entre UML y Diagramas Entidad Relación, con lo que la representación de esquemas de datos es bastante inmediata. En cuanto a expresividad, sigue siendo bastante limitada. Por ejemplo, no permite cuantificadores existenciales ni propiedades encadenadas.

- OWL 2 RL: indicado para aplicaciones que requieren un razonamiento escalable sin sacrificar demasiada expresividad. Está orientado fundamentalmente a ser utilizado en otras tecnologías basadas en reglas, facilitando el razonamiento.

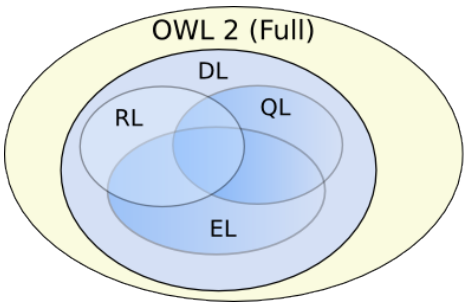

Figura 3.4: Diagrama de Venn de los perfiles de OWL 2.0

La funcionalidad *"Punning"* [100] introducida en la versión 2 de OWL permite definir recursos en las ontologías como clases e instancias al mismo tiempo. Así, se permite definir una clase y una instancia asociada a esa misma clase, pero compartiendo la misma URI. En OWL, diferentes recursos poseen diferentes URI, por lo que la funcionalidad "Punning" puede ser considerada como una excepción a esta norma. Sin embargo, estos recursos que comparten la misma URI son el mismo recurso, que según el contexto pueden ser tratados como clases o como instancias. Por lo tanto, a nivel de clases

estos recursos pueden tener declarados sub-clases o propiedades y a nivel de instancia pueden estar asignados a otras instancias del modelo ontológico. Esta funcionalidad es útil, ya que permite reutilizar a nivel de instancia la semántica de una conceptualización de un dominio realizado a nivel de clase.

Para representar e intercambiar la información de ontologías en OWL 2, es necesario definir una sintaxis para almacenar las ontologías en documentos OWL. Así, OWL 2 define varias sintaxis pero siendo la sintaxis basada en RDF/XML [101] la más importante y obligatoria de implementar. Esta sintaxis tiene como objetivo servir de medio de intercambio de documentos OWL 2. Todas las herramientas que implementen OWL 2, según el estándar, deben de poder leer y escribir este tipo de documentos. La sintaxis OWL/XML [102] tiene por objetivo cumplir con el estándar XML y que puedan reutilizarse herramientas XML para procesar este tipo de ontologías. OWL 2 también define una sintaxis funcional [103] para representar de forma más fácil de leer por personas la estructura formal de las ontologías. La sintaxis Manchester [104] es utilizada para facilitar a los usuarios leer y escribir ontologías que cumplan con las restricciones de la lógica descriptiva. Por último, la sintaxis Turtle [105] tiene como objetivo facilitar la lectura y escritura de tripletas RDF, por lo que su uso puede ser también aplicado en la definición de una ontología.

3.1.3. Interfaces de programación de ontologías

OWL-API

OWL-API es una interfaz de programación (API) escrita en Java para crear, manipular y serializar ontologías OWL. La primera versión surgió en 2002 como parte del proyecto WonderWeb [106] y fue diseñada para utilizar OWL 1.0. La última versión de OWL-API es la 3.0, y está enfocada en cubrir las características de la versión 2.0 de OWL. OWL-API es un proyecto de código abierto que ha sido publicado bajo las licencias LGPL [107] y Apache [108].

OWL-API está formado por los siguientes módulos:

- Una interfaz de programación para OWL 2.0.

- Un módulo para analizar y escribir documentos en RDF/XML.
- Un módulo para analizar y escribir documentos en OWL/XML.
- Un módulo para analizar y escribir documentos usando la sintaxis funcional de OWL.
- Un módulo para analizar y escribir documentos en Turtle.
- Un módulo para analizar documentos en el formato OBO [109].
- Un conjunto de interfaces con las que utilizar razonadores como FaCT++ [110], HermiT [111], Pellet [112] o RacerPro [113].

OWL-API proporciona las estructuras de datos necesarias para modelizar ontologías OWL. Además, permite procesar y analizar diferentes formatos de documentos de ontologías para convertirlos en una estructura interna de datos con la que poder trabajar. Esta estructura de datos es almacenada en memoria y puede ser manipulada posteriormente. También permite que la representación de la ontología en memoria sea serializada en otro formato distinto al del origen para almacenar la información en un fichero. Mediante las interfaces de razonamiento, la descripción de una ontología y los módulos externos de razonamiento, OWL-API es capaz de explotar las funcionalidades de inferencia que el lenguaje semántico formal OWL proporciona. OWL-API está diseñado para soportar el perfil OWL-DL, el cual se basa en la lógica de primer orden para permitir tareas de inferencia en términos de completitud computacional y decidibilidad en un tiempo finito.

Jena

Jena es una plataforma Java para la construcción de aplicaciones para la Web semántica. Proporciona las interfaces de programación necesarias para trabajar con los lenguajes RDF, OWL y SPARQL. Además, incluye un motor de inferencia basado en reglas que permite razonar con las ontologías definidas. Jena es un proyecto de código abierto desarrollado por Brian McBride en los laboratorios de Web semántica de HP [114].

La plataforma Jena incluye los siguientes componentes:

- Una interfaz de programación para manejar grafos RDF.
- Una interfaz de programación para manejar ontologías OWL.
- Un módulo para leer y escribir documentos RDF/XML, N3 y N-tripletas.
- Un módulo para almacenar el modelo en memoria y en un repositorio persistente.
- Un módulo para consultar el modelo utilizando el lenguaje SPARQL.

Esta plataforma permite analizar, crear, manipular y consultar modelos basados en RDF. Con Jena se pueden representar modelos, recursos, propiedades, literales, tripletas o cualquier otro concepto de RDF. Jena permite la incorporación de un razonador que sea capaz de inferir relaciones entre sus elementos que previamente no se hayan hecho explícitas. La jerarquía de niveles entre el modelo ontológico, el razonador y el grafo RDF del modelo se puede observar en la Figura 3.5.

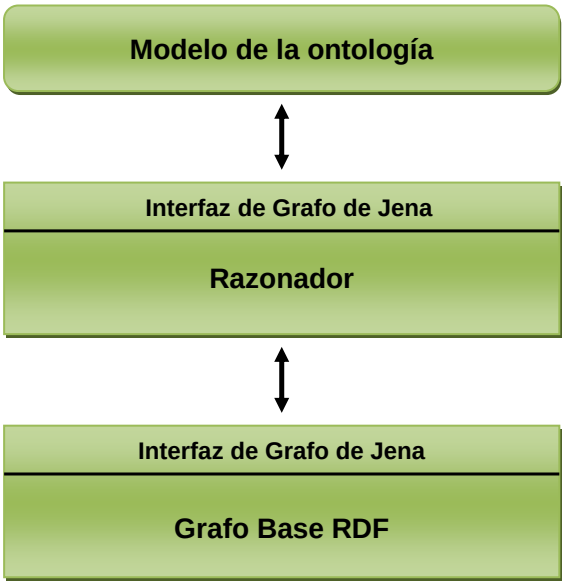

Figura 3.5: Niveles de la plataforma Jena entre el modelo, razonador y grafo RDF

Jena proporciona un constructor, llamado *ModelFactory*, para la creación de los modelos, y unas especificaciones de modelos predefinidos, disponibles mediante la clase *OntModelSpec*, para indicar el tipo de modelo ontológico a crear. Debido a la independencia del lenguaje que aporta Jena, su estructura de clases e interfaces resulta bastante compleja. En ella, las ontologías vienen representadas por la interfaz *OntModel*, y cada uno de los elementos de la ontología, por la interfaz *OntResource*. Un *OntResource* puede ser especializado en una clase, una propiedad, una relación, una anotación o un individuo. Los valores de las propiedades son considerados literales y pueden tener asociado un tipo de datos e incluso un idioma. Además, Jena permite almacenar el modelo de manera temporal en memoria, mediante repositorios RDF con los componentes SDB [115] y TDB [116], y también de forma persistente sobre diversas bases de datos.

Para crear repositorios RDF, Jena dispone de dos componentes en su plataforma para el almacenamiento de tripletas RDF y para su consulta mediante SPARQL. Estos componentes son SDB y TDB. SDB es un repositorio de tripletas RDF que utiliza una base de datos relacional para almacenar la información. La información es almacenada para optimizar su consulta con SPARQL. Un almacén SDB requiere de un fichero de configuración para establecer los parámetros del repositorio y los de la conexión a la base de datos. Por otro lado, TDB no utiliza una base de datos para almacenar la información de las tripletas, sino que define un sistema propio para mejorar la gestión de la información y sus metadatos. Ambos sistemas disponen de una interfaz para realizar consultas desde la consola del sistema y a través de la interfaz de programación de Jena.

Para crear un repositorio persistente en una base de datos utilizando Jena, se necesita asociar un modelo de la ontología con un esquema de una base de datos. Una vez establecida esa asociación, los cambios realizados sobre el modelo se consolidan automáticamente en la base de datos. Para establecer la conexión entre el repositorio y Jena desde un punto de vista técnico, es necesario definir un modelo base y un perfil ontológico. Por un lado, el modelo base es el encargado de asociar el repositorio con un modelo ontológico, y por otro lado, el perfil ontológico permite definir la funcionalidad que se busca

que tenga el modelo ontológico. Por ejemplo, el perfil *"OntModelSpec.OWL_DL_MEM"* indica que el modelo se corresponde con la descripción de una ontología que sigue la especificación *OWL DL* y es almacenado en memoria, pero no realiza ningún razonamiento sobre el modelo. Sin embargo, el perfil *PelletReasonerFactory.THE_SPEC* indica que se va a utilizar el razonador Pellet para realizar las tareas de inferencia sobre el modelo ontológico.

El repositorio persistente en bases de datos de Jena definirá, en el esquema correspondiente, tablas compuestas por cuatro columnas: *Subj*, *Prop*, *Obj* y *GrapID*. Cada fila contendrá la información de una tripleta RDF (Sujeto, predicado y objeto) junto con el índice del grafo RDF al que corresponde. Así, diferentes modelos ontológicos disponen de índices diferentes. Los valores de la columna *Subj* se corresponden con los sujetos de las tripletas RDF, e indican las URI de los recursos de la ontología. La columna *Prop* está relacionada con los predicados de las tripletas RDF y describen las URI de las propiedades, relaciones o anotaciones asociadas con cada sujeto de las tripletas. Por último, la columna *Obj* contiene la información asociada con el objeto de las tripletas y puede ser un valor literal o la URI de otro recurso ontológico.

Para valores literales que excedan el tamaño máximo de columna en el repositorio permanente, se dispone de otra tabla en el esquema para almacenarlos como objetos binarios grandes (BLOB). Jena crea una tabla diferente por cada ontología con un espacio de nombres diferente. Para diferenciar un modelo de otro, Jena proporciona la cuarta columna, *GraphID*. Esta columna tendrá un valor diferente para las tripletas de diferentes modelos y diferentes espacios de nombres, pudiendo compartir tabla en el repositorio persistente.

Para almacenar la información en el repositorio persistente, Jena ofrece dos tipos de procedimientos, uno para cargar un gran volumen de información y otro en el que se controlarán los detalles del proceso de inserción individual. El primer proceso está basado en el método *"read"* de la clase *"OntModel"* de Jena. Este método permite, que permite cargar la información almacenada en un fichero de información en el repositorio semántico con una sola instrucción, está diseñado para insertar información evitando que el usuario deba describir manualmente la información del modelo de la ontología. Jena

soporta distintos formatos de documentos, como XML/RDF, OWL y N3. Al leer el fichero de información, ésta se almacena automáticamente y de manera persistente en el repositorio, estando así disponible para la siguiente vez que se cargue el modelo ontológico.

En caso de inserciones de instancias de manera individual, se deberán utilizar los métodos que proporciona Jena para la manipulación de modelos ontológicos. Por ejemplo, para crear una clase en la ontología se utilizará el método *"createClass"*, si no se especifica ninguna URI la clase es considerada anónima. Una vez creada la clase se le prodrán añadir sus propiedades y relaciones. De igual manera, para crear un nuevo individuo en el repositorio, se utiliza el método *"createIndividual"*, que obligatoriamente debe tener un recurso de la ontología como parámetro, y se corresponde con su clase de la ontología. Estos métodos permiten definir detalladamente la información del modelo de la ontología. Para borrar la información del modelo de la ontología, Jena proporciona varios métodos que pueden ser invocados en la clase *"OntModel"*. El método *"remove"* permite el borrado de tripletas concretas del modelo ontológico asociado, y que deben ser especificadas como parámetro del método. Para borrar un conjunto indeterminado de tripletas del repositorio, se utiliza el método *"removeAll"*. Este método permite borrar toda la información de un repositorio o especificar el valor de algún componente de las tripletas a borrar. Así, mediante la combinación de ambos métodos, es posible gestionar la eliminación de información del repositorio.

Otro apartado importante es la consulta de la información del repositorio persistente. Jena ofrece tanto la posibilidad de definir consultas semánticas en SPARQL como utilizar métodos de la interfaz de programación para obtener la información. Así, las consultas SPARQL tienen como finalidad la búsqueda de tripletas RDF almacenadas en el repositorio, sin la posibilidad de explotar su semántica. Sin embargo, utilizando el modelo ontológico y un perfil que permita generar el modelo inferido, es posible explotar dicha información. Así, por ejemplo se podrán obtener las instancias directas de una clase o todas las instancias asociadas a una clase, las directas y las inferidas, aunque es posible un enfoque mixto donde se puedan obtener un conjunto de datos de las consultas SPARQL y, con los resultados obtenidos, realizar operaciones más

avanzadas directamente en el modelo de la ontología. Estas características pueden ser útiles para la gestión de la información y la consulta específica de información del repositorio.

3.1.4. Razonadores

Existen diferentes niveles de expresividad para la descripción de modelos ontológicos. La capacidad expresiva de estos modelos está estrechamente relacionada con la capacidad de razonamiento y el tiempo de cómputo que requieren. Cuanto más complejo sea el modelo, más conocimiento se puede inferir y más recursos necesitan. Debido a esta dependencia, las ontologías limitan el tipo de constructores que se pueden utilizar para describir los modelos. OWL define el sublenguaje OWL-DL, que como se ha indicado se basa en la lógica descriptiva y que proporciona la máxima expresividad garantizando la completitud computacional y la decidibilidad de la inferencia en un tiempo finito. La versión 2 de OWL define tres perfiles diferentes, OWL EL, OWL QL y OWL RL, que restringen el sublenguaje OWL-DL para mejorar la eficiencia de los razonadores según la finalidad con que se defina la ontología.

Un razonador OWL-DL es una pieza clave para la gestión de sistemas de información basados en ontologías, ya que permite generar un modelo inferido para extraer el conocimiento implícito que contenga y verificar las propiedades de corrección y satisfacibilidad del modelo. A continuación, se describen las principales caracteristicas de los razonadores: Pellet, FaCT++, HermiT y RacerPro. Este conjunto de razonadores soportan OWL-DL y, por lo tanto, cumplen los requisitos computacionales y expresivos necesarios para su uso en sistemas basados en ontologías.

Pellet

Pellet es un razonador que implementa un motor de inferencia para OWL 2, ofreciendo diferentes servicios para inferir con ontologías OWL y soportando, de manera completa, todas las capacidades de inferencia proporcionadas por los perfiles de OWL 2. De este modo, permite optimizar la respuesta a

consultas simbólicas y deductivas, y las inferencias incrementales.

Este razonador proporciona servicios estándar de inferencia, como verificar la consistencia del modelo, determinar si un concepto puede tener instancias o si eso puede provocar inconsistencias, generar la jerarquía completa de clases a través de la relación de subclase, o la obtención del tipo directo de instancias tras la creación de la jerarquía de clases. Además, desde la implementación de OWL 2 por el razonador, éste es capaz de verificar las restricciones de cardinalidad cualificadas, utilizar axiomas de subpropiedad complejas, restricciones reflexivas locales, propiedades disjuntas y de declaración de negación, compartición de vocabulario entre individuos, clases y propiedades, y soportar rangos de datos definidos por los usuarios.

Pellet dispone de varias interfaces para utilizar el razonador, ofreciendo un conjunto de funciones para ejecutar desde la línea de comandos. También dispone de una interfaz de programación para desarrollar aplicaciones independientes que exploten las funcionalidades del razonador. Ofrece a las interfaces de programación de ontologías acceso al motor de inferencia del razonador para generar el modelo inferido y así poder navegar por él. El editor de ontologías Protégé [117] dispone de un módulo para integrar el razonador y utilizarlo durante la definición de una ontología.

Además, Pellet permite razonar sobre el modelo, aun cuando cambia su base de conocimiento. Actualmente, soporta la clasificación y la verificación de la consistencia incremental. Esto permite actualizar el modelo inferido mientras se modifica el modelo de la ontología.

En la actualidad, se está desarrollando una nueva interfaz de programación en Java, llamada Ortiz, que será publicada junto con la versión 3.0 del razonador. Esta nueva interfaz permitirá programar en Java aplicaciones que utilicen razonamiento con ontologías OWL2 QL, EL, DL y RL.

FaCT++

FaCT++ es la versión mejorada del razonador de ontologías OWL-DL FaCT. FaCT++ utiliza los mismos algoritmos de FaCT pero implementa una arquitectura interna diferente. Además, FaCT++ ha mejorado su eficiencia

siendo desarrollado en C++, en vez de en Common Lisp, e implementando nuevas optimizaciones y características.

FaCT++ utiliza los algoritmos de optimización tableaux [118] para la clasificación, la satisfacibilidad, verificación de consistencia y la instanciación de conceptos. Sin embargo, soporta parcialmente OWL 2. No soporta la definición de claves sobre los conceptos. Solamente soporta ciertos tipos de datos, como *Literal, string, anyURI, boolean*, etc.

Por último, indicar que FaCT++ dispone de un módulo para su integración en el editor de ontologías Protégé, puede ser utilizado en una aplicación Java mediante la interfaz de programación de ontologías OWL-API y, además, dispone de interfaces para su uso en C++ y en Lisp.

HermiT

HermiT es un razonador para ontologías descritas con OWL. Dada una ontología OWL, HermiT puede decidir si una ontología es consistente o no, o también identificar relaciones implícitas entre clases.

HermiT implementa el algoritmo Hypertableau [119] que proporciona una capacidad de inferencia más eficiente que los algoritmos tradicionales. Este algoritmo permite reducir el tiempo de clasificación, así como manejar ontologías más complejas que otros razonadores existentes. Además, HermiT soporta de manera completa las funcionalidades de la versión 2 de OWL.

Este razonador es un proyecto de código abierto y está publicado bajo la licencia LGPL. Utiliza la interfaz de programación OWL-API para manejar las ontologías. HermiT dispone de un módulo para su uso en el editor Protégé, así como una interfaz para la línea de comandos y otra para su uso en aplicaciones Java que utilicen interfaces de programación de ontologías compatibles.

RacerPro

RacerPro (Renamed Abox and Concept Expression Reasoner) es un motor de inferencia desarrollado para la Web semántica. RacerPro implementa una optimización del algoritmo tableaux para la decidibilidad del problema

de la consistencia del ABox, o conjunto de anotaciones de una ontología, expresado utilizando *SHIQ(D)-* [120]. La descripción del modelo de la ontología (TBox) y las instancias del modelo (ABox) pueden ser traducidas a términos en lógica descriptiva con los que realizar funciones de inferencia.

RacerPro soporta diferentes formatos y lenguajes ontológicos. Por ejemplo, soporta RDF, OWL Lite, OWL DL y un lenguaje nativo desarrollado para el razonador. En el momento de redacción de esta tesis, sólo se soporta a nivel sintáctico el lenguaje ontológico OWL 2.0, mientras que no a nivel funcional, es decir, la funcionalidad de los nuevos axiomas no está implementada aunque sí se procesan los elementos de la gramática OWL 2.0.

RacerPro proporciona su funcionalidad a aplicaciones externas mediante interfaces sockets sobre TCP. Así, se ofrecen interfaces de programación que adaptan el acceso a RacerPro mediante:

1. LRacer, una interfaz de programación en Lisp. Permite utilizar la funcionalidad del razonador RacerPro mediante funciones o macros en Lisp.

2. OWL-API, implementa la sintaxis del OWL-API para acceder a la funcionalidad del razonador mediante sockets TCP.

3. OWLlink [121], interfaz que proporciona un método estándar para ofrecer funcionalidades de razonamiento a ontologías OWL2.0.

4. JRacer, una interfaz de programación en Java que evita la sobrecarga de utilizar lenguajes intermedios para acceso a la funcionalidad del razonador.

5. AllegroGraph [122], permite utilizar el repositorio de tripletas RDF con la funcionalidad del razonador RacerPro mediante consultas con lenguajes semánticos.

3.1.5. Lenguajes de consulta semánticos: SPARQL

En esta sección, se describirá el lenguaje de consulta SPARQL. SPARQL es un lenguaje de consulta semántico diseñado para acceder a repositorios que

utilicen formato RDF. Este lenguaje alcanzó la categoría de recomendación del W3C en 2008.

Existen varios lenguajes similares a SPARQL, como SquishQL [123], RDQL [124] o TriQL [125], que tienen en común que consideran la información en RDF como tripletas, sin tener en cuenta la semántica del modelo.

SPARQL ha sido diseñado para un uso escalable en la Web, por lo que permite hacer consultas sobre orígenes de datos distribuidos, independientemente del formato. Es más sencillo crear una consulta sencilla a través de diferentes almacenes de datos que crear varias consultas, además de tener un coste menor y de ofrecer unos resultados mejores.

En concreto, para SPARQL existen diferentes especificaciones que describen las diferentes partes de su funcionalidad, que consisten en un lenguaje de consulta, un formato para las respuestas y un medio para el transporte de consultas y respuestas:

- SPARQL Query Language: componente principal de SPARQL. Describe la sintaxis para la composición de sentencias y su concordancia.
- SPARQL Protocol for RDF [126]: describe el acceso remoto de datos y la transmisión de consultas de los clientes a los procesadores. Utiliza WSDL [127] para definir protocolos remotos para la consulta de bases de datos basadas en RDF.
- SPARQL Query Results XML Format [128]: formato utilizado para la devolución de los resultados de las búsquedas (consultas SELECT o ASK), a partir de un esquema XML.

El lenguaje SPARQL está formado por tres componentes importantes: URI, literales y variables. Las URI, se representan entre "<" y ">", los literales entrecomillados y las variables se deben prefijar con el símbolo "?". La principal operación para la obtención de información es SELECT, la cual devuelve todo, o un conjunto de las variables que coinciden con el patrón de búsqueda. Una consulta se puede dividir en las siguientes cuatro partes (véase figura 3.6):

- Espacio de nombres: mediante la palabra reservada PREFIX, se introducen los espacios de nombres que se necesitan en la consulta, a través de la asociación de una URI a una etiqueta. (*PREFIX book_: <http://example.org/book/> y*
  PREFIX ttl_: <http://purl.org/dc/elements/1.1/>).

- Operación y valores a devolver: la operación para realizar consultas es SELECT. Para obtener todos los valores devueltos, se puede utilizar «*». En otro caso, se debe de indicar el nombre de las variables. La palabra clave FROM identifica el grafo RDF sobre el que se ejecutará la consulta, siendo necesario indicar que una consulta puede incluir varios FROM. La palabra clave WHERE indica el patrón sobre el que se filtrarán las tripletas RDF. (*SELECT ?title WHERE ...*).

- Consulta: entre los corchetes se introducen las tripletas (sujeto, predicado objeto) que forman la consulta. (*book_:book1 ttl_:title ?title*).

- Opciones: se pueden añadir opciones para ordenar los resultados a través de la palabra clave ORDER, para evitar repeticiones a través de DISTINCT, etc. (*ORDER BY ?title*).

```
PREFIX book_: <http://example.org/book/>
PREFIX ttl_: <http://purl.org/dc/elements/1.1/>

SELECT ?title
WHERE { book_:book1 ttl_:title ?title .}
ORDER BY ?title
```

Figura 3.6: Consulta SPARQL

SPARQL define cuatro tipos diferentes de consulta. La primera es *SELECT*, que devuelve todos los valores que concuerdan con el patrón de consulta definido. El segundo es *CONSTRUCT*, que devuelve un grafo RDF donde las variables del patrón de consulta han sido sustituidas por los resultados de la consulta y combinados todos en el mismo grafo. El tercer elemento es *ASK*, que indica si la consulta puede devolver algún valor o no. Y, por último, el tipo *DESCRIBE*, que es utilizado para la depuración de consultas,

porque describe los recursos encontrados en la consulta en vez de devolver los valores.

Las consultas SPARQL aceptan otros elementos que permiten una definición más precisa de los patrones de consulta. Estos elementos son *OPTIONAL*, *UNION*, *LIMIT* y *OFFSET*. *OPTIONAL* es utilizado dentro del apartado *WHERE* de la consulta, e indica que las tripletas RDF declaradas bajo su ámbito son opcionales. Esto es útil cuando se declaran propiedades en los conceptos con cardinalidad 0 y, por lo tanto, su existencia no es obligatoria para todas las ocurrencias de los conceptos. Por otro lado, *UNION* permite reunir el resultado de dos patrones de consulta diferentes. Esta flexibilidad permite obtener el resultado de diferentes alternativas dentro de una misma consulta. Por último, los elementos *LIMIT* y *OFFSET*, son utilizados para limitar el número de resultados y la posición de éstos respectivamente. Con estos dos elementos, es posible recorrer los resultados de una consulta de manera eficiente cuando el conjunto de resultados es muy grande.

Además, SPARQL ofrece soluciones para evaluar valores literales dentro de los patrones de consulta definidos. Para ello, se utiliza el elemento *FILTER*, que permite restringir el tipo de datos utilizado por los valores, el idioma en que están definidos, utilizar operadores booleanos o condiciones aritméticas, verificar que el valor sea vacío, o evaluar expresiones regulares.

3.1.6. Lenguaje de actualización de RDF: SPARUL

En esta sección, se comentará el lenguaje SPARUL (SPARQL/Update) y las funcionalidades que ofrece. Este lenguaje ha sido desarrollado por la empresa Hewlett-Packard para manipular grafos RDF. SPARUL amplía las funcionalidades del lenguaje SPARQL. Así, mientras que SPARQL está diseñado para consultar información en RDF, SPARUL está diseñado para actualizar el contenido y la descripción de grafos RDF. Al reutilizar la sintaxis de SPARQL, se reduce la curva de aprendizaje para desarrolladores y también disminuye el coste de implementación.

SPARQL/Update permite realizar tareas de actualización de la información y también de gestión de los grafos RDF. Las tareas son las siguientes:

- Insertar nuevas tripletas en un grafo RDF.
- Eliminar tripletas en un grafo RDF.
- Realizar un conjunto de operaciones de actualización en una sola acción.
- Crear un nuevo grafo RDF en un repositorio de grafos.
- Borrar un grafo RDF de un repositorio de grafos.

Para insertar datos en un grafo RDF, SPARUL proporciona varias operaciones. La primera es *LOAD*, que permite insertar todas las tripletas de un grafo RDF en otro. La Figura 3.7 muestra un ejemplo de la operación LOAD, donde *<http://grafo1>* se corresponde con la URI del grafo que se quiere importar, y *<http://grafo>* indica la URI del grafo donde se importan las tripletas. Si no se especifica el grafo donde se importan las tripletas, se utilizará el grafo por defecto del repositorio.

```
LOAD <http://grafo1> INTO <http://grafo>
```

Figura 3.7: Ejemplo de operación LOAD de SPARQL/Update

Para insertar tripletas individuales directamente a un grafo, se utiliza la operación *INSERT DATA*, en la cual se debe especificar el grafo donde se van a almacenar y una lista con las tripletas RDF a insertar. La Figura 3.8 muestra un ejemplo de inserción donde se indica directamente la tripleta a insertar.

```
INSERT DATA INTO <http://grafo>
{
 <http://grafo/resource/Pais/España>
 <http://grafo/ontology/capital>
 <http://grafo/resource/Ciudad/Madrid> .
}
```

Figura 3.8: Ejemplo de operación INSERT DATA de SPARQL/Update

La operación *INSERT* (véase la Figura 3.9) permite definir una plantilla que describe un conjunto de tripletas RDF donde algunos parámetros son

variables que siguen el mismo estilo que las definidas en SPARQL. Estas variables estarán referenciadas en el patrón de consulta de tripletas RDF definido dentro del apartado *WHERE* y que tiene el mismo significado que en las consultas SPARQL. Con los valores de las variables, se forman las tripletas que serán añadidas en el grafo especificado.

```
INSERT [INTO <URIgrafo>] {plantilla} [WHERE {patrón}]
```

Figura 3.9: Operación INSERT de SPARQL/Update

Para borrar las tripletas de un grafo RDF, SPARQL/Update proporciona el comando *CLEAR*, que elimina todas las tripletas del grafo RDF especificado. La Figura 3.10 muestra la sintaxis de esta operación.

```
CLEAR [GRAPH <URIgrafo>]
```

Figura 3.10: Operación CLEAR de SPARQL/Update

Al igual que con las operaciones de inserción *INSERT* e *INSERT DATA*, para eliminar tripletas se disponen de las operaciones *DELETE* y *DELETE DATA*, donde la primera necesita que se definan la plantilla y el patrón de consulta para identificar las tripletas a borrar, mientras que la segunda operación de eliminación sólo necesita que se enumeren las tripletas a borrar.

El lenguaje SPARQL/Update define una operación para modificar tripletas RDF que puede ser considerada como una operación de borrado, seguida de una inserción, pero en una sola consulta. En este caso, la operación se llama *MODIFY* y en ella se indican el grafo RDF sobre el que se realizará la operación, la plantilla de la operación de eliminación, la plantilla de la operación de inserción y el patrón de la consulta para describir las tripletas a modificar. La Figura 3.11) muestra un ejemplo de la operación *MODIFY* en la que todas las tripletas que indican el valor de la propiedad *Nombre* de las instancias del concepto *País* son cambiadas por la propiedad *Identificador*.

```
MODIFY <http://grafo>
DELETE {?sujeto <http://grafo/ontology/Nombre> ?valor}
INSERT {?sujeto <http://ogolod/ontology/Identificador> ?valor}
WHERE {
?sujeto
<http://grafo/ontology/Nombre>
?valor .

?sujeto
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://grafo/ontology/País> .
}
```

Figura 3.11: Ejemplo de operación MODIFY de SPARQL/Update

La sintaxis de las operaciones para crear, *CREATE*, y eliminar, *DROP*, grafos RDF de un repositorio de grafos se muestran en la Figura 3.12. Sólo necesita que se defina la URI del grafo RDF en la operación. Para evitar los mensajes de error por intentar crear un grafo que ya existe o por intentar eliminar un grafo que no existe, se puede utilizar el atributo *SILENT*.

```
CREATE [SILENT] GRAPH <URIgrafo>
DROP [SILENT] GRAPH <URIgrafo>
```

Figura 3.12: Operaciones para crear y eliminar grafos RDF utilizando SPARQL/Update

Los lenguajes SPARQL y SPARQL/Update proporcionan toda la funcionalidad necesaria para manipular los grafos RDF y consultar la información que contienen. Mediante estos dos estándares, se facilita la gestión de repositorios de grafos RDF, al igual que SQL [129] permite gestionar los esquemas en las bases de datos relacionales.

3.2. Linked Data

El término Linked Data [130, 131] representa un conjunto definido de buenas prácticas para publicar y enlazar datos estructurados en la Web. Mediante la adopción de estas prácticas por proveedores de datos, se pretende conseguir alcanzar un espacio único de datos global que contenga cada vez

más información, llamado Web de Datos. Linked Data utiliza RDF para representar la información en el espacio global y la Web para enlazar la información entre diferentes fuentes de información. La característica más importante de Linked Data es que la información esté publicada en la Web mediante datos individuales. Estos datos deben estar representados para que pueden ser procesados por aplicaciones y que el conocimiento asociado a los datos está definido explícitamente. Además, dichos datos estarán enlazados con otros conjuntos de datos externos y, al mismo tiempo, los datos pueden ser referenciados por otros conjuntos de datos diferentes.

Los principios de la Web de Datos están siendo adoptados por diversos organismos públicos en el ámbito nacional e internacional. Por ejemplo, Microsoft lidera *Open Government Data Initiative* [132], la cual ofrece soporte para publicar datos de servicio público en la Web de Datos a agencias gubernamentales. En Europa, siguiendo los objetivos clave de la Agenda Digital Europea [133], se está desarrollando la plataforma ePSI (European Public Sector Information), que pretende poner a disposición de los ciudadanos catálogos de información del sector público, así como otros catálogos creados por el sector privado. Además, la Unión Europea a través del Séptimo Programa Marco (FP7 [134]) está financiando proyectos, como LATC [135], LOD2 [136], o PlanetData [137], para el desarrollo de herramientas y metodologías para la implantación de la Web de Datos. Otras iniciativas como DATA.GOV [138] de EE.UU., Opening up government [139] del Reino Unido, o Portail du Gouvernement [140] de Francia, ofrecen información gubernamental de manera pública y gratuita. En el ámbito español, algunos gobiernos autonómicos como el de Asturias, Cataluña o Euskadi, entidades locales como los ayuntamientos de Zaragoza y Gijón, y agencias estatales como el Instituto Geográfico Nacional, están comenzando a publicar sus datos en la Web de Datos de manera libre y gratuita.

Los principios básicos que se deben seguir para publicar datos en la Web fueron propuestos por Berners-Lee en [141] y fueron adoptados por la iniciativa Linked Data para publicar y conectar los datos usando la infraestructura de la Web. Los cuatro principios definidos son:

1. Utilizar URI para identificar los recursos.
2. Utilizar HTTP URI para que los usuarios puedan localizar los identificadores de recursos.
3. Proporcionar información útil asociada a la URI de los recursos.
4. Enriquecer la descripción de los recursos enlazando con otros recursos.

Siguiendo estos principios, Linked Data proporciona datos de manera independiente de su formato o su presentación. Los conjuntos de datos son auto descriptivos, ya que los datos identifican los términos del vocabulario y éstos están, a su vez, accesibles a través de su URI. El uso del protocolo HTTP para acceder a los datos proporciona un mecanismo simplificado de acceso a datos comparado con otros sistemas propietarios. Los conjuntos de datos publicados pueden ser descubiertos de manera dinámica, siguiendo los enlaces proporcionados.

Uno de los proyectos más extendidos en este área es LOD, que surgió como un proyecto para ofrecer datos de manera abierta en todo el mundo. El objetivo principal es proporcionar a la Web, datos mediante la publicación de conjuntos de datos en RDF de manera libre y al mismo tiempo estableciendo enlaces entre los datos publicados en diferentes repositorios.

La Figura 3.13 muestra los repositorios de datos RDF publicados y los enlaces definidos entre ellos, dentro del ámbito del proyecto LOD. En concreto, a septiembre de 2011, doscientos tres repositorios han sido publicados, lo que supone unos veinticinco mil millones de tripletas RDF y cerca de trescientos noventa y cinco millones de enlaces declarados entre las tripletas. Además, para formar parte de este proyecto, los repositorios deben estar en CKAN [142], a saber, un registro de acceso libre que reúne los metadatos de los repositorios publicados bajo Linked Data. Además, de cumplir con los principios de Linked Data, los repositorios deben disponer de URI que puedan ser resueltas mediante un explorador Web, que la información está almacenada directamente en formato RDF, que el contenido sea original y que el repositorio está enlazado a otros repositorios de la Web de Datos.

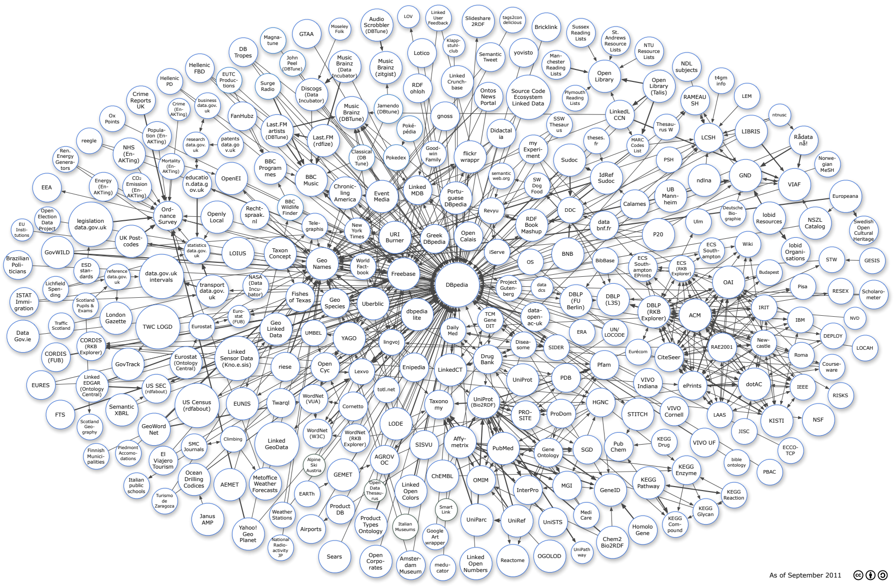

Figura 3.13: Conjuntos de datos RDF e interconexiones del proyecto LOD (Diagrama de la nube LOD, por Richard Cyganiak y Anja Jentzsch. Septiembre 2011)

Capítulo 4

La Web semántica biomédica

En biomedicina, la información almacenada en los repositorios es compleja, heterogénea, dispersa y muy cambiante. A estas desventajas, hay que añadir que las descripciones de la información que contienen están definidas en lenguaje natural y, por lo tanto, susceptibles de diferentes interpretaciones. Por lo tanto, es necesario que se defina el conocimiento específico del dominio biomédico para poder interpretar los valores presentes en los repositorios de manera precisa. De este modo, se utilizan las ontologías biomédicas para poner a disposición de los usuarios el conocimiento necesario para entender la información de los repositorios. Una ontología es la conceptualización precisa del conocimiento sobre un dominio.

Los investigadores necesitan consultar información relacionada con el área en la que realizan sus experimentos. En biomedicina, se dispone de gran cantidad de información que es utilizada para generar más conocimiento. Esta información está dispersa en cientos de bases de datos y es frecuente que durante una investigación se haga uso de varios repositorios. Para ello se debe saber exactamente el tipo de información que hay disponible y las restricciones de los datos en cada repositorio.

La cantidad de información biomédica disponible en la actualidad provoca que una sola persona no pueda abarcar todo el conocimiento del dominio. Más aún, los análisis de microarrays o las nuevas tecnologías de secuenciamiento están ofreciendo la posibilidad de secuenciamiento de genomas com-

pletos, lo cual puede dificultar las investigaciones si no se utilizan mejores herramientas para gestionar la información. Estas herramientas han de ser capaces de poder manejar la información y así reducir el tiempo que los investigadores expertos emplean en tareas más mecánicas. Para alcanzar esta meta, se deberá disponer de un método para representar la información y el conocimiento de manera que pueda ser procesado automáticamente por aplicaciones y con la mínima intervención humana.

Las ontologías, parte fundamental de la Web semántica, pueden jugar un papel muy importante en los sistemas de gestión de conocimiento biológico y biomédico [143], ya que son capaces de describir de manera formal el conocimiento de un dominio y ponerlo a disposición de humanos y aplicaciones para su uso de manera autónoma. Las ontologías proporcionan un método para representar conceptos y relaciones de manera muy flexible. Así, éstas permiten aportar el conocimiento asociado con la información producida para que pueda ser procesada de manera automática por aplicaciones.

Dentro de este ámbito, cabe destacar el *Semantic Web Healthcare and Life Sciences Interest Group* (HCLS [144]) del W3C, que tiene como objetivos desarrollar, defender y apoyar el uso de las tecnologías de la Web semántica en los dominios de la atención médica, las ciencias biológicas, investigación médica e investigación traslacional. Tiene relación con otros grupos del W3C, como los grupos de trabajo de SPARQL [145] o RDB2RDF [146]. El trabajo que realiza el HCLS está relacionado con: (1) la creación de conjuntos de datos enlazados (Linked Data) y guías para la creación de datos por otros grupos; (2) la definición de vocabularios compartidos; (3) la implementación de ejemplos prácticos; (4) la ayuda a otros grupos en la creación de datos y herramientas en su ámbito; e (5) informar al sector empresarial de la importancia y madurez de las herramientas existentes. Un ejemplo de este tipo de proyectos es *Linking Open Data Drug* (LODD) [147], que tiene el objetivo de enlazar información relacionada con ensayos clínicos sobre medicamentos y sobre su impacto en la expresión de los genes. Este proyecto pretende relacionar la información dispersa en la nube *Linked Data* para poder resolver las consultas de interés de grupos de investigación o empresas relacionadas con este dominio.

Para la consulta de la mayoría de datos biomédicos publicados en la Web semántica, se proporcionan interfaces basadas en lenguajes semánticos como DL o SPARQL. Estos lenguajes permiten explotar la semántica definida del dominio, siendo una herramienta muy útil ya que permiten incorporar restricciones del modelo en las consultas. Sin embargo, su utilización requiere de unos conocimientos o aprendizaje previos, como en el caso del lenguaje SQL. Así, los potenciales usuarios de estos sistemas, como los investigadores biomédicos, a menudo no poseen la suficiente habilidad como para utilizarlos. Por lo tanto, se necesitan mecanismos más simples que permitan definir consultas en lenguajes semánticos. Estos sistemas no sólo deben permitir a los usuarios con un conocimiento limitado de estos lenguajes de consulta obtener la información que requieran de la Web semántica, sino que también deben facilitar el descubrimiento de los modelos ontológicos subyacentes de los repositorios semánticos disponibles para consulta.

Este capítulo está dividido en dos secciones: la sección 4.1, en la que se describirá el uso de las ontologías en el dominio de la biología y la biomedicina; y la sección 4.2, en la que se describirá la aplicación de los principios de Linked Data para representar y publicar conjuntos de datos de información biológica y biomédica en la Web.

4.1. Bio-ontologías

Las bio-ontologías, según la información biológica que contengan, pueden estar divididas en tres tipos diferentes: (1) orientadas a un dominio biológico concreto; (2) orientadas a una función biológica; y (3) genéricas, las cuales representan conceptos genéricos que pueden ser reutilizados en diferentes dominios y funciones biológicas. La mayoría de las ontologías utilizan una combinación de más de un tipo, según el propósito por el que son definidas.

Una de las principales características de las ontologías es la reusabilidad [148], ya que, una vez definidas, su conocimiento puede ser reutilizado por otros sistemas. A diferencia de otros sistemas de representación de información, como los esquemas de bases de datos, las ontologías definen el conocimiento del dominio y la información de manera homogénea, permi-

tiendo descubrir cómo la información está representada. La reusabilidad de las ontologías puede estar limitada, dependiendo del propósito para el que fue definida e incluso es posible que sólo parte de la ontología pueda ser reutilizada.

Uno de los primeros usos de las bio-ontologías es su uso como terminologías o vocabulario controlado de términos en un dominio concreto. Así, en 1986, La Biblioteca Nacional de la Medicina (NLM [149]) de EE.UU. comenzó con el proyecto UMLS [150], sistema para un lenguaje médico unificado (Unified Medical Language System), el cual facilita el desarrollo de sistemas informáticos que puedan procesar el lenguaje biomédico. UMLS proporciona un conjunto de repositorios de conocimiento. Estos repositorios contienen al *Methathesaurus*, el cual proporciona un vocabulario común sobre conceptos biomédicos y sobre las ciencias de la salud, sus sinónimos y las relaciones entre ellos. Tiene una estructura en forma de catálogo o tesauro en la que los términos están agrupados según su significado. También la *Red Semántica* (Semantic Network) forma parte de UMLS, y representa las relaciones semánticas entre los grupos de conceptos definidos en el *Methathesaurus*. El último elemento que proporciona UMLS es el *SPECIALIST Lexicon*, el cual proporciona información léxica sobre vocabulario común en inglés, términos biomédicos y términos utilizados en el Methathesaurus, para utilizarlo en sistemas de procesamiento de lenguaje natural.

Uno de los aspectos más importantes a tener en cuenta en el proceso de definición de los conceptos relacionados con la información biológica y biomédica es la heterogeneidad de la información y de su representación. Los tipos de heterogeneidad con los que se deben tratar son:

- Heterogeneidad sintáctica: cuando la información biomédica no está expresada con el mismo lenguaje, como ocurre entre XML, ficheros tabulados o RDF.

- Heterogeneidad terminológica: esto es debido a los diferentes nombres o referencias utilizados para identificar el mismo concepto en diferentes repositorios como en UniGene y EntrezGene.

- Heterogeneidad conceptual o semántica: debido a las diferencias en el modelado del mismo dominio. Los tipos de problemas que se pueden encontrar son:
  - Diferencias en la cobertura: ocurre cuando dos modelos describen regiones diferentes o solapadas del mundo al mismo nivel de detalle y desde la misma perspectiva, como las presentes entre bases de datos especializadas en una especie y en las que se especializan en varias, como FlyBase y Entrez [151].
  - Diferencias de granulidad: ocurre cuando dos modelos describen la misma región del mundo desde la misma perspectiva pero con diferente nivel de detalle.
  - Diferencia en perspectiva: ocurre cuando dos modelos describen la misma región del mundo, al mismo nivel de detalle, pero desde una perspectiva diferente, como ocurre con la información biomédica que se especializa en una función biológica o una propiedad relevante.
- Heterogeneidad pragmática: termino asociado con cómo la gente interpreta las entidades según el contexto.

Con el progresivo aumento del número y los tipos de ontologías biomédicas, es necesario un marco común que establezca un conjunto de principios que se sigan para su desarrollo de forma que puedan ser reutilizables y que sean interoperables en el dominio biomédico. Así, OBO Foundry [152] surge con el objetivo de liderar el desarrollo de los principios de buenas prácticas para la definición de ontologías biológicas y biomédicas. En la siguiente subsección, se abordará el proyecto OBO Foundry y se comentarán las diversas ontologías de interés en el presente trabajo.

4.1.1. OBO Foundry

OBO Foundry es un proyecto colaborativo en el que están involucrados investigadores que trabajan en el desarrollo de ontologías para el dominio de

la biología y la biomedicina. En concreto, las ontologías OBO tienen el objetivo de definir un conjunto de principios para la descripción de conocimiento ortogonal de referencia para el dominio de la biología y la biomedicina.

OBO Foundry define sus ontologías siguiendo los siguientes principios:

1. Las ontologías deben ser abiertas y estar libremente disponibles para ser usadas sin ninguna restricción, salvo: (1) su origen debe quedar reconocido; y (2) no pueden ser alteradas y distribuidas bajo el mismo nombre o identificadores.

2. Las ontologías deben estar expresadas en un lenguaje ontológico común y aceptado.

3. Las ontologías deben disponer de un espacio de nombre único dentro de OBO.

4. Las ontologías deben de definir algún método para distinguirlas entre diferentes versiones.

5. Las ontologías deben de tener un contenido claramente especificado y delimitado.

6. Las ontologías deben usar las relaciones que han sido definidas de manera no ambigua siguiendo el patrón de definiciones de relaciones de OBO [153].

7. Las ontologías deben estar bien documentadas.

8. Las ontologías deben tener asociados una pluralidad de usuarios independientes.

9. Las ontologías de OBO Foundry serán desarrolladas mediante trabajo colaborativo.

En OBO Foundry se distinguen entre ontologías miembro y ontologías candidatas. Las ontologías miembro tienen el potencial de abarcar un amplio

rango del dominio de las ciencias naturales de manera modular. Además, estas ontologías han pasado un proceso de evaluación para verificar que cumplen con los principios arriba enunciados. Las ontologías miembros, en el momento de redacción de esta tesis, son:

- CHEBI [154] (Chemical Entities of Biological Interest): proporciona un vocabulario de entidades moleculares centrado en compuestos químicos elementales.
- GO (Gene Ontology): ontología que pretende estandarizar la representación de los genes y los productos génicos independientemente de su especie y repositorio. Está dividida en tres sub ontologías que describen: (1) el proceso biológico (Biological Process); (2) la función molecular (Molecular Function); y (3) el componente celular (Cellular Component).
- PATO [155] (Phenotypic Quality Ontology): ontología de características del fenotipo, que se utilizan para describir los fenotipos.
- PRO [156] (Protein Ontology): ontología para anotar proteínas y compuestos de proteínas.
- XAO [157] (Xenopus Anatomy Ontology): ontología de la anatomía y desarrollo de la especie Xenopus laevis.
- ZFA [158] (Zebrafish Anatomy Ontology): ontología de la anatomía y desarrollo de la especie Danio rerio.

Las ontologías pertenecientes a OBO Foundry están expresadas en el formato OBO. El formato OBO fue definido con los siguientes objetivos:

1. Poder definir las ontologías de manera que sean fácilmente entendible por los usuarios.
2. Poder procesar las ontologías de manera sencilla por aplicaciones software.

3. Que el lenguaje pueda ser extendido.

4. Que el lenguaje evite la redundancia en la definición de las ontologías.

Para procesar y gestionar documentos en formato OBO, se han desarrollado diferentes editores, entre los que se pueden destacar las herramientas OBO-Edit [159] y OBO-Explorer [160]. OBO-Edit es un editor de ontologías de código libre escrito en Java. Este editor surgió originariamente con el nombre de GO-Edit, y tenía como objetivo facilitar la edición de las ontologías de *Gene Ontology*. Las características principales del editor son: (1) una interfaz para edición, fácil de usar; (2) implementación de un razonador simple pero eficiente; (3) y la disponibilidad de un motor de consulta. Está optimizado para el procesamiento y representación de ontologías en el formato OBO. Sin embargo, el editor es capaz de procesar otros lenguajes ontológicos como OWL. Se ha implementado un plug-in para Protégé con el fin de transformar las ontologías en formato OBO al formato OWL utilizando la herramienta OBO-Edit [161]. Además, esta herramienta implementa un razonador capaz de hacer explícitos los hechos implícitos en la definición de la ontología. El razonador está en desarrollo y en el momento de redacción de esta tesis, no toda la semántica definida en las bio-ontologías OBO es soportada. Así, es capaz de procesar las propiedades transitivas, pero no las simétricas. Por último, destacar que está patrocinado por el *Gene Ontology Consortium*.

Por otro lado, el OBO-Explorer es también un editor de ontologías en formato OBO. Forma parte del proyecto COBrA-CT [162], el cual tiene como objetivo solucionar los problemas existentes al dar formato, editar y gestionar diferentes versiones de bio-ontologías OBO en formato OWL. Así, el editor OBO-Explorer permite transformar las ontologías en formato OBO a formato OWL. Este editor está implementado como un plug-in de Protégé. De esta manera, permite generar una representación en OWL de bio-ontologías OBO.

Debido al gran interés de la comunidad de la Web semántica en el lenguaje OWL, se han desarrollado otras herramientas, como la librería OboInOwl del paquete onto-perl [163], con el fin de procesar las bio-ontologías con el formato OBO y transformarlas al formato OWL. Por ejemplo, las figuras 4.1 y 4.2 muestran las diferencias en la representación de conceptos entre el formato

OBO y el formato OWL para la descripción del concepto *GO:0003674* de la GO.

```
[Term]
id: GO:0003674
name: molecular_function
namespace: molecular_function
alt_id: GO:0005554
subset: goslim_candida
subset: goslim_pir
subset: goslim_plant
subset: goslim_yeast
subset: gosubset_prok
synonym: "molecular function" EXACT []
synonym: "molecular function unknown" NARROW []
```

Figura 4.1: Concepto *GO:0003674* en formato OBO

```
<owl:Class rdf:about="http://purl.org/obo/owl/GO#GO_0003674">
  <rdfs:label xml:lang="en">molecular_function</rdfs:label>
  <oboInOwl:inSubset rdf:resource="http://purl.org/obo/owl/obo#goslim_candida"/>
  <oboInOwl:inSubset rdf:resource="http://purl.org/obo/owl/obo#goslim_pir"/>
  <oboInOwl:inSubset rdf:resource="http://purl.org/obo/owl/obo#goslim_plant"/>
  <oboInOwl:inSubset rdf:resource="http://purl.org/obo/owl/obo#goslim_yeast"/>
  <oboInOwl:inSubset rdf:resource="http://purl.org/obo/owl/obo#gosubset_prok"/>
  <oboInOwl:hasExactSynonym>
    <oboInOwl:Synonym>
      <rdfs:label xml:lang="en">molecular function</rdfs:label>
    </oboInOwl:Synonym>
  </oboInOwl:hasExactSynonym>
  <oboInOwl:hasNarrowSynonym>
    <oboInOwl:Synonym>
      <rdfs:label xml:lang="en">molecular function unknown</rdfs:label>
    </oboInOwl:Synonym>
  </oboInOwl:hasNarrowSynonym>
  <oboInOwl:hasOBONamespace>molecular_function</oboInOwl:hasOBONamespace>
  <oboInOwl:hasAlternativeId>GO:0005554</oboInOwl:hasAlternativeId>
</owl:Class>
```

Figura 4.2: Concepto *GO:0003674* en formato OWL

A continuación, se describirán las ontologías pertenecientes a OBO Foundry, ya sean miembros o candidatos, que han sido utilizadas en esta tesis. Las ontologías son Gene Ontology, Evidence Codes Ontology [164], OBO Relationship Types [165], NCBI Organismal Classification [166] y Human Phenotype Ontology [167].

4.1.2. Gene Ontology

La Gene Ontology (GO) define un conjunto de bio-ontologías que proporcionan a la comunidad científica información sobre genes a partir del uso de un vocabulario controlado y específico del dominio. La GO ha sido definida mediante un proyecto colaborativo para definir las anotaciones de productos asociados a genes con las que describir las funciones y localizaciones en la célula de éstos en el momento en que realizan su función [168]. Se entiende por una anotación en GO a la asociación, basada en la evidencia, entre el producto de un gen y un término de GO. El proyecto GO surgió en 1998 como un proyecto abierto y colaborativo entre tres grupos biológicos que mantenían los repositorios: (1) *FlyBase*, especializado en el organismo *Drosophila*; (2) *Saccharomyces Genome Database*, centrado en la especie *Saccharomyces*; y (3) *Mouse Genome Informatics*, especializado en *Mus musculus*. Actualmente, otras bases de datos se han anotado con GO. Así, ésta ha crecido para poder representar adecuadamente y de manera general a estos repositorios, por lo que se la puede considerar una ontología independiente de la especie a la que se aplique.

Aunque GO no fue una de primeras ontologías biomédicas, sí es una de las que más éxito ha tenido [11]. La GO permite describir los productos de los genes en términos de su proceso biológico (*Biological Process*), componente celular (*Cellular Component*) y función molecular (*Molecular Function*). De esta manera, GO está dividida en tres ontologías disjuntas que describen cada uno de los aspectos anteriores.

Biological Process describe las operaciones y eventos moleculares que tienen un principio y un fin establecidos. *Cellular Component* describe las partes de la célula y su entorno extracelular. Por último, *Molecular Function* describe las actividades elementales de un producto génico a nivel molecular. Estas ontologías están organizadas como grafos dirigidos directos, donde un nodo puede tener más de un padre.

Los términos de la ontología *Cellular Component*, describen localizaciones a nivel de estructuras subcelulares y compuestos macromoleculares, como por ejemplo la membrana interna del núcleo celular. Así, un producto de un gen

Indicador	Valor
Conceptos asociados a Procesos Biológicos	21394
Conceptos asociados a Funciones Moleculares	9062
Conceptos asociados a Componentes Celulares	2896
Organismos con anotaciones	367887
Cantidad de productos génicos anotados	11855555
Cantidad de productos génicos anotados manualmente	437164

Tabla 4.1: Resumen del estado de desarrollo de la Gene Ontology.

puede ser localizado o es un sub-componente de un componente celular concreto. La relación "parte-de" es la que relaciona los términos de la ontología entre sí. El término raíz de esta ontología es el *GO:0005575*.

La ontología *Biological Process* contiene una serie de eventos asociados a uno o más compuestos ordenados de funciones moleculares. Ejemplos de estos procesos biológicos pueden ser proceso fisiológico celular o transducción de señal. Para distinguir un proceso biológico de una función molecular, el proceso debe de tener más de una etapa diferente que describa claramente un comienzo y un final. El término raíz de esta ontología es *GO:0008150*.

La ontología *Molecular Function* describe actividades que ocurren a nivel molecular. Los términos de la ontología describen actividades, no las entidades que realizan esas actividades, ni indican dónde o cuándo tienen lugar. Ejemplos de este tipo de actividades son la actividad catalítica o la actividad adenilato ciclasa. El término raíz de esta ontología es *GO:0003674*.

El desarrollo de la GO está enfocado sobre tres aspectos principales: (1) el mantenimiento y desarrollo de las tres ontologías; (2) la anotación de los productos de los genes, que implica la definición de enlaces entre los conceptos de GO y los genes y sus productos en otros repositorios externos; y (3) el desarrollo de herramientas que faciliten la creación, mantenimiento y uso de las ontologías. La Tabla 4.1 muestra el estado actual de desarrollo de la GO.

Gracias al uso de los conceptos de la GO para anotar diferentes repositorios de información, las búsquedas a través de los términos definidos es más uniforme. Además, como los conceptos están jerarquizados, las consultas pueden definirse con diferente nivel de detalle. Sin embargo, la descripción

de cada concepto está definida en lenguaje natural.

4.1.3. Evidence Codes Ontology

La ontología de códigos de evidencia (ECO) surgió como vocabulario común formalmente representado que satisficiera las necesidades de la GO para anotar con sus términos los productos génicos. ECO es una ontología que aporta un rico vocabulario para representar evidencias experimentales y de otros tipos. Es candidata de OBO Foundry. La ontología está siendo usada actualmente en diversos proyectos, como el anotador [169] o el índice de recursos [170] NCBO de BioPortal [171].

GO inicialmente utilizó códigos de tres letras para identificar códigos de evidencia. Por otro lado, ECO extiende la definición de códigos de evidencia de GO. Los términos de ECO se identifican mediante el patrón *ECO:0000000*, donde los dígitos numéricos cambian para cada término de la ontología. Para obtener la relación entre los códigos de evidencia definidos en ECO y los de GO, en la ontología ECO se incluyen como sinónimos a sus identificadores, los códigos de tres letras definidos por GO. Por ejemplo, el código *TAS* de la GO indica la evidencia de que se puede rastrear el autor de la anotación y se corresponde al término *ECO:0000033* de ECO.

La Figura 4.3 muestra al concepto *ECO:0000002*, que representa la evidencia experimental basada en una medida directa de algún aspecto de una característica biológica, y que está representado en OWL a partir de su transformación del formato OBO. Se pueden observar las referencias externas indicadas a otros repositorios mediante la etiqueta *<oboInOwl:hasDbXref>*. De esta manera, en la definición del concepto define como un sinónimo exacto al término *http://purl.org/obo/owl/GO#GO_IDA*, que se corresponde con el código de evidencia *IDA* de la GO.

```
<owl:Class rdf:about="http://purl.org/obo/owl/ECO#ECO_0000002">
 <rdfs:label xml:lang="en">direct assay result</rdfs:label>
 <oboInOwl:hasDefinition>
  <oboInOwl:Definition>
   <rdfs:label xml:lang="en">
     Experimental evidence that is based on direct measurement
      of some aspect of a biological feature.
   </rdfs:label>
   <oboInOwl:hasDbXref>
    <oboInOwl:DbXref>
     <rdfs:label>ECO:MCC</rdfs:label>
     <oboInOwl:hasURI rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
   http://purl.org/obo/owl/ECO#ECO_MCC
 </oboInOwl:hasURI>
    </oboInOwl:DbXref>
   </oboInOwl:hasDbXref>
   <oboInOwl:hasDbXref>
    <oboInOwl:DbXref>
     <rdfs:label>GO:IDA</rdfs:label>
     <oboInOwl:hasURI rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
   http://purl.org/obo/owl/GO#GO_IDA
 </oboInOwl:hasURI>
    </oboInOwl:DbXref>
   </oboInOwl:hasDbXref>
  </oboInOwl:Definition>
 </oboInOwl:hasDefinition>
 <oboInOwl:hasExactSynonym>
  <oboInOwl:Synonym>
   <rdfs:label xml:lang="en">IDA</rdfs:label>
   <oboInOwl:hasDbXref>
    <oboInOwl:DbXref>
     <rdfs:label>GO:IDA</rdfs:label>
     <oboInOwl:hasURI rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
   http://purl.org/obo/owl/GO#GO_IDA
 </oboInOwl:hasURI>
    </oboInOwl:DbXref>
   </oboInOwl:hasDbXref>
  </oboInOwl:Synonym>
 </oboInOwl:hasExactSynonym>
 <oboInOwl:hasExactSynonym>
  <oboInOwl:Synonym>
   <rdfs:label xml:lang="en">inferred from direct assay</rdfs:label>
   <oboInOwl:hasDbXref>
    <oboInOwl:DbXref>
     <rdfs:label>GO:IDA</rdfs:label>
     <oboInOwl:hasURI rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
   http://purl.org/obo/owl/GO#GO_IDA
 </oboInOwl:hasURI>
    </oboInOwl:DbXref>
   </oboInOwl:hasDbXref>
  </oboInOwl:Synonym>
 </oboInOwl:hasExactSynonym>
 <oboInOwl:hasOBONamespace>evidence_code2.obo</oboInOwl:hasOBONamespace>
 <rdfs:subClassOf rdf:resource="http://purl.org/obo/owl/ECO#ECO_0000006"/>
</owl:Class>
```

Figura 4.3: Concepto *ECO:000002* en formato OWL.

Se han llevado a cabo diversas mejoras en ECO durante su proceso de

revisión. Se ha reducido la inconsistencia entre los términos que contiene para describir sólo tipos de evidencias, no inferencias ni experimentos. Así, se han renombrado algunos términos y cambiado sus definiciones, en especial los que contenían el texto "*inferred from*". Otros términos han sido definidos como obsoletos, como *ECO:0000037*, y, por lo tanto, su uso está desaconsejado. Sin embargo, también se han añadido términos nuevos para completar los tipos de evidencia y mejorar el nivel de detalle que describen.

4.1.4. OBO Relationship Types

La ontología de los tipos de relaciones OBO (RO) tiene como objetivo definir un conjunto de relaciones independientes del dominio con las que relacionar los conceptos de las ontologías OBO entre sí. Además, estas relaciones son definiciones generales y pueden ser extendidas con otras definiciones de relaciones más adecuadas para el dominio de la ontología OBO que las utilicen. Por ejemplo, la relación *es-un* es semánticamente equivalente a la relación en GO y en la ontología de tipos de células [172]. El nombre de las relaciones que se definen y las propiedades asociadas se describen la Tabla 4.2. Las propiedades dan cuenta de la semántica de las relaciones, que son necesarias para interpretar las anotaciones realizadas con ellas.

La relación *OBO_REL:is_a* tiene el mismo significado que la relación entre una clase y sus subclases en OWL. La relación *OBO_REL:part_of* indica que una instancia o concepto de la ontología es parte de otra instancia o concepto diferente. La relación *OBO_REL:integral_part_of* se define como "una instancia I es parte integral de I' sí y solo sí I es parte de I' e I' tiene como parte a I". La relación *OBO_REL:proper_part_of* es igual a la relación *OBO_REL:part_of* pero, además, se exige que el sujeto y objeto de la relación sean distintos. La relación *OBO_REL:located_in* asocia una instancia con una localización concreta. La relación *OBO_REL:contained_in* indica que una instancia está contenida en otra instancia y, por lo tanto, también está localizada en ella. La relación *OBO_REL:adjacent_to* expresa una proximidad espacial entre dos instancias. La relación *OBO_REL:transformation_of* relaciona dos clases en la que las instancias de la primera clase mantienen su

Nombre	Propiedades
OBO_REL:is_a	transitiva, reflexiva y anti-simétrica.
OBO_REL:part_of	transitiva, reflexiva y anti-simétrica.
OBO_REL:integral_part_of	transitiva, reflexiva y anti-simétrica.
OBO_REL:proper_part_of	transitiva.
OBO_REL:located_in	transitiva y reflexiva.
OBO_REL:contained_in	
OBO_REL:adjacent_to	
OBO_REL:transformation_of	transitiva.
OBO_REL:derives_from	transitiva.
OBO_REL:preceded_by	transitiva.
OBO_REL:has_participant	
OBO_REL:has_agent	
OBO_REL:instance_of	

Tabla 4.2: Tabla con los tipos de relaciones y sus propiedades definidas en la ontología RO.

identidad hasta que, por algún tipo de transformación, cambian a la otra clase. La relación *OBO_REL:derives_from* indica que una instancia ha derivado de otra instancia mediante una transformación previa. La relación *OBO_REL:preceded_by* hace referencia a que un proceso precede a otro. La relación *OBO_REL:has_participant* asocia una instancia con un proceso en el que participa. La relación *OBO_REL:has_agent* tiene el mismo significado que la relación *OBO_REL:has_participant*, pero la instancia tiene un papel activo en el proceso asociado. Por último, la relación *OBO_REL:instance_of* asocia una instancia con la clase a la que pertenece.

4.1.5. NCBI Organismal Classification

Esta ontología representa una jerarquía de conceptos, los cuales están relacionados con la taxonomía de las especies del Centro Nacional para la Información Biotecnológica (NCBI). El objetivo de esta base es proporcionar una referencia para la relación de filogenia entre las distintas especies de organismos. La única relación presente en la ontología es la relación entre una clase y su subclase en la ontología. Por ejemplo, el linaje de la especie

Homo sapiens es la siguiente: *cellular organisms*, *Eukaryota*, *Opisthokonta*, *Metazoa*, *Eumetazoa*, *Bilateria*, *Coelomata*, *Deuterostomia*, *Chordata*, *Craniata*, *Vertebrata*, *Gnathostomata*, *Teleostomi*, *Euteleostomi*, *Sarcopterygii*, *Tetrapoda*, *Amniota*, *Mammalia*, *Theria*, *Eutheria*, *Euarchontoglires*, *Primates*, *Haplorrhini*, *Simiiformes*, *Catarrhini*, *Hominoidea*, *Hominidae*, *Homininae* y *Homo*.

La Figura 4.4 muestra un ejemplo de una clase de la ontología de la clasificación de organismo de NCBI, asociada a la especie *Homo sapiens*. Dentro de las propiedades de las clases se puede destacar *has_rank*, la cual clasifica los conceptos. En dicha figura se indica que se refiere a una especie, pero otros valores posibles pueden ser, familia, orden, suborden, clase, etc. Además, aporta sinónimos para facilitar la búsqueda en texto de lenguaje natural.

```
<owl:Class rdf:about="http://purl.org/obo/owl/NCBITaxon#NCBITaxon_9606">
 <rdfs:label xml:lang="en">Homo sapiens</rdfs:label>
 <oboInOwl:hasExactSynonym>
  <oboInOwl:Synonym>
   <rdfs:label xml:lang="en">man</rdfs:label>
  </oboInOwl:Synonym>
 </oboInOwl:hasExactSynonym>
 <oboInOwl:hasExactSynonym>
  <oboInOwl:Synonym>
   <rdfs:label xml:lang="en">human</rdfs:label>
  </oboInOwl:Synonym>
 </oboInOwl:hasExactSynonym>
 <oboInOwl:hasDbXref>
  <oboInOwl:DbXref>
   <rdfs:label>GC_ID:1</rdfs:label>
   <oboInOwl:hasURI rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
      http://purl.org/obo/owl/GC_ID#GC_ID_1
   </oboInOwl:hasURI>
  </oboInOwl:DbXref>
 </oboInOwl:hasDbXref>
 <rdfs:subClassOf rdf:resource="http://purl.org/obo/owl/NCBITaxon#NCBITaxon_9605"/>
 <has_rank xmlns="" rdf:resource="http://purl.org/obo/owl/NCBITaxon#species"/>
</owl:Class>
```

Figura 4.4: Concepto *NCBITaxon_9606* asociado al *Homo sapiens* en formato OWL.

4.1.6. Human Phenotype Ontology

La ontología del fenotipo humano (HPO) ofrece un extenso vocabulario, alrededor de unos 9500 términos que describen anomalías fenotípicas individuales. Esta ontología permite clasificar las distintas malformaciones que pueden ocurrir en humanos. Los términos se organizan en una jerarquía de conceptos, donde la relación de sub-clase indica una relación del tipo *es_un*, que además es transitiva. Un ejemplo de los conceptos definidos en HPO pueden ser, el concepto *HP:0000002*, "Anomalía de la Altura del Cuerpo", que es sub-clase del término *HP:0001507*, el cual indica una anomalía en el crecimiento.

La jerarquía de conceptos de HPO está dividida en una serie de sub-ontologías: (1) ontología de Anomalías de órganos, que describen las anomalías clínicas del cuerpo humano; (2) ontología de la Herencia, que ofrece patrones de herencia del fenotipo; y (3) ontología del Comienzo y Curso Clínicos, que describe el tiempo en el que se desarrollan los síntomas y la progresión de éstos en un curso clínico.

HPO ha sido desarrollada a partir del repositorio OMIM, que contiene un catálogo de enfermedades genéticas hereditarias humanas. Prácticamente todas las enfermedades descritas en este repositorio han sido incluidas en la ontología HPO. Así, HPO ofrece una herramienta que facilita el procesamiento de la información que describe por otras aplicaciones, además de proporcionar una jerarquía que permite establecer diferentes niveles de detalle, a la hora de anotar anomalías del fenotipo humano.

La Figura 4.5 muestra el concepto *HP:0000002* de la ontología HPO, en formato OWL, aunque la ontología está definida en formato OBO. Los conceptos de la ontología contienen una etiqueta con el nombre de la anomalía en lenguaje natural y además proporciona enlaces a conceptos definidos en la ontología UMLS, de términos médicos.

```
<owl:Class rdf:about="http://purl.org/obo/owl/HP#HP_0000002">
 <rdfs:label xml:lang="en">Abnormality of body height</rdfs:label>
 <oboInOwl:hasOBONamespace>medical_genetics</oboInOwl:hasOBONamespace>
 <oboInOwl:hasDbXref>
  <oboInOwl:DbXref>
   <rdfs:label>UMLS:C0000768</rdfs:label>
   <oboInOwl:hasURI rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
      http://purl.org/obo/owl/UMLS#UMLS_C0000768
   </oboInOwl:hasURI>
  </oboInOwl:DbXref>
 </oboInOwl:hasDbXref>
 <oboInOwl:hasDbXref>
  <oboInOwl:DbXref>
   <rdfs:label>UMLS:C0005890</rdfs:label>
   <oboInOwl:hasURI rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
      http://purl.org/obo/owl/UMLS#UMLS_C0005890
   </oboInOwl:hasURI>
  </oboInOwl:DbXref>
 </oboInOwl:hasDbXref>
 <oboInOwl:hasDbXref>
  <oboInOwl:DbXref>
   <rdfs:label>UMLS:C1704258</rdfs:label>
   <oboInOwl:hasURI rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
      http://purl.org/obo/owl/UMLS#UMLS_C1704258
   </oboInOwl:hasURI>
  </oboInOwl:DbXref>
 </oboInOwl:hasDbXref>
 <rdfs:subClassOf rdf:resource="http://purl.org/obo/owl/HP#HP_0001507"/>
</owl:Class>
```

Figura 4.5: Concepto *HP_00000002* en formato OWL, que representa a una anomalía en la altura del cuerpo.

4.2. Linked Data en la biología y biomedicina

Las tecnologías de la Web semántica están siendo aplicadas intensivamente en muchos campos de la ciencia, siendo los dominios de la biología y la biomedicina donde su uso más se ha extendido, como en el repositorio YeastHub [173]. Actualmente, hay publicados diversos repositorios de datos bioinformáticos en la red de Linked Data: Uniprot[1], Protein Data Bank[2], DrugBank[3], Gene Ontology Annotations[4], etc. Estos repositorios publican los datos contenidos en sus bases de datos siguiendo los principios de Linked Data. Por lo tanto, estos repositorios pueden ser enlazados por otros

[1] http://www.uniprot.org/uniprot/
[2] http://semanticscience.org/projects/pdb2rdf/
[3] http://www4.wiwiss.fu-berlin.de/drugbank/
[4] http://www.bioontology.org/wiki/index.php/OBD:SPARQL-GO

repositorios y así enriquecer la red de Linked Data.

Como ejemplos de dos repositorios relacionados con la publicación de conjuntos de datos que siguen los principios de Linked Data en el dominio de la biología y biomedicina, se puede citar Bio2RDF [174] y Linked Life Data. Estos repositorios tienen como objetivo principal reunir información de repositorios tradicionales, publicarlos en la nube de *Linked Data* y, por último, establecer referencias entre ellos cuando sea posible.

En concreto, Bio2RDF es un proyecto que utiliza las tecnologías de la Web semántica para proporcionar conjuntos de datos de los dominios de la biología y la biomedicina para ayudar al descubrimiento de nuevo conocimiento. Bio2RDF utiliza técnicas de integración sintáctica y semántica para implementar una metodología que genere e integre información que pueda ser procesada e interpretada por aplicaciones autónomas. El objetivo de esta integración es conseguir realizar consultas basadas en lenguajes semánticos como SPARQL para realizar consultas más avanzadas que en los sistemas tradicionales. En el momento de redacción de esta tesis, Bio2RDF ha publicado información de unos 42 repositorios pertenecientes al NCBI, KEGG, Gene Ontology, PDB, Mesh, BioPAX, etc.

Linked Life Data es un proyecto para la integración de información semántica basada en RDF. Este proyecto está co-financiado por la Unión Europea, a través del proyecto LarKC [175]. La información que integra ha sido obtenida de 27 bases de datos del dominio de la biología y biomedicina. Linked Life Data publica el contenido que integran siguiendo los principios de Linked Data. La información que integran es generada y mantenida por diferentes organizaciones. Por lo tanto, la información se encuentra distribuida entre los diferentes repositorios y su integración produce altas tasas de redundancia. La información está disponible en diferentes formatos sintácticos y estructurales, con una semántica particular para cada uno que dificulta su integración. Además, disponen de sistemas de consulta con una funcionalidad muy limitada que dificultan su explotación.

Este repositorio contiene más de cinco mil millones de tripletas RDF que definen más de mil millones de entidades[5]. Dispone de una interfaz de consul-

[5] `http://linkedlifedata.com/sources`

ta basada en SPARQL para consultar el repositorio, y además cada entidad dispone de su URL para ser consulta a través de la Web. La información que integran sigue un modelo común de datos que facilita la relación entre entidades y su consulta. La información utilizada en este repositorio posee restricciones de copyright.

Capítulo 5

Metodologías de integración de información biomédica

Debido al gran número de repositorios bioinformáticos disponibles, y a la gran variedad y complejidad de la información biológica y biomédica, la comunidad investigadora requiere que dichos repositorios están almacenados y representados de manera que se facilite su consulta. Por otro lado, los bioinformáticos son conscientes de estos problemas y llevan proponiendo diferentes metodologías para la integración de dicha información desde el origen de la disciplina.

La integración de información se basa en la utilización de datos compartidos entre diferentes repositorios para identificar las correspondencias entre ellos [176]. Estos datos compartidos son muy variados: valores de datos, nombres, identificadores, esquemas de propiedades, términos ontológicos, palabras clave, etc. Pueden utilizar un único dato compartido o varios. Principalmente, se pueden diferenciar las metodologías de integración en base a los mecanismos que implementan para establecer las correspondencias. Ejemplos de tipos de metodologías de integración son: la definición de enlaces entre registros de bases de datos; el establecimiento de índices cruzados entre repositorios; desarrollo de protocolos para el intercambio de datos; la integración de conjuntos de datos sobre un esquema común a ellos; la definición de las correspondencias de nombres y valores de datos entre diversos conjuntos de

datos; la reunión de toda la información asociada con una misma instancia; y la interconexión de diferentes programas informáticos para la obtención de datos con el fin definir un sistema que los integre.

Los repositorios de información biomédica pueden presentar determinadas características que pueden dificultar su integración con las metodologías de integración que se describe en este capítulo. Las desventajas más comunes son: (1) las interfaces de consulta están diseñadas para ser utilizadas por personas, no por aplicaciones, por lo que la extracción de información es más difícil; (2) la información se suministra sin una semántica que permita su interpretación por usuarios no expertos; (3) las instancias están identificadas mediante el uso de códigos de acceso locales a cada repositorio y, por lo tanto, no compartirán identificadores con los que se podrían establecer correspondencias; y (4), a menudo, los tipos de información que contienen no están definidos explícitamente en el repositorio, sino que con mucha frecuencia se describen en lenguaje natural. Estos problemas dificultan la búsqueda y procesamiento de su información de manera automática, así como su enlace e integración con otros repositorios.

Las siguientes sub-secciones presentan los tipos de enfoques más importantes en la definición de las metodologías de integración de información biológica y biomédica. La Figura 5.1 muestra una clasificación de los tipos de metodologías que van a ser definidas según el nivel de integración de información que soportan. En la Figura 5.1, se exponen dos conjuntos de metodologías, uno horizontal, que representa los tipos de metodologías según el nivel de integración de información que alcanzan, y otro transversal, que representa los tipos de métodos que pueden utilizarse para implementar los tipos de integración del conjunto horizontal. Por otro lado, la aplicación real de estas metodologías puede llevarse a cabo mediante enfoques mixtos que combinen varios métodos en un mismo proyecto.

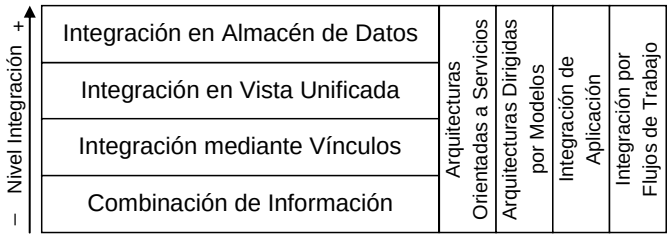

Figura 5.1: Diagrama con la clasificación de los tipos de metodologías de integración.

5.1. Integración en almacenes de datos

A diferencia de la integración basada en enlaces, la integración basada en la creación de almacenes de datos requiere de un mayor esfuerzo en el proceso de integración de la información. En esta metodología de integración, los conjuntos de datos a integrar deben ser recopilados, analizados, transformados y almacenados en un modelo de datos compartido. Este método ofrece como resultado una integración real de varias fuentes de datos en un único repositorio homogéneo.

Los repositorios creados a partir de esta metodología permiten consultar la información integrada mediante una sola consulta sin que los usuarios tengan que procesar la información manualmente. Ejemplos de estos repositorios pueden ser ATLAS [177] y Columba [178]. Como ejemplos de herramientas que implementan este tipo de metodología, se pueden destacar BioMART [179] y BioWarehouse [180].

Estos sistemas deben establecer un modelo común para representar la información integrada. Este modelo común a menudo es difícil de modificar o gestionar en el tiempo. Además, deben mantener una referencia al repositorio original de datos para su mantenimiento. A pesar de almacenar estas referencias, su costo de mantenimiento y sincronización con sus fuentes de datos es muy alto.

Tampoco son muy flexibles con respecto a los cambios en sus fuentes de información, lo que lleva a definir adaptadores para cada repositorio poco

reutilizables y susceptibles a errores. Así, aunque este tipo de repositorios son muy utilizados, resultan de difícil mantenimiento en una disciplina en constante cambio como es el caso de la biología y la biomedicina.

En la Figura 5.2, se muestra un ejemplo del método de integración basado en almacenes de datos. En esta figura, se representa al sistema "D" como un sistema que integra en un repositorio único la información extraída de los repositorios "A", "B" y "C". La información integrada es almacenada en el repositorio "D" y es ofrecida por el sistema "D" para consulta. La integración mediante esta metodología es completa, y la información puede ser ofrecida a los usuarios de manera integrada.

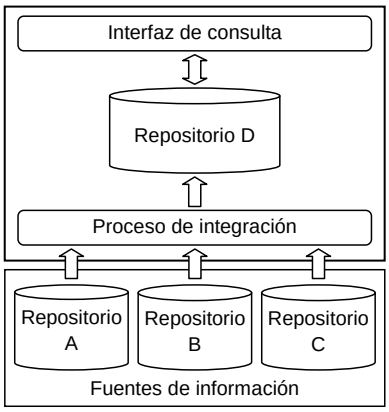

Figura 5.2: Ejemplo de caso de uso de un sistema que utiliza la integración basada en almacén de datos para integrar la información de tres sistemas independientes.

5.2. Integración mediante vistas unificadas

En esta sección, se describirá el enfoque de integración de información basado en una vista integrada sobre varios repositorios. En este caso, la información no se integra en un repositorio común, sino que permanece sólo en los repositorios que hacen de fuente de información. Sin embargo mediante la construcción de una interfaz común, crean la ilusión de que están accediendo a un único repositorio. Este sistema puede ser considerado como un almacén virtual de información.

Este tipo de sistemas requiere de un proceso de integración complejo, al igual que en los métodos de integración basados en almacén integrado. Por otro lado, estos tipos de sistemas requieren de programas intermedios entre los repositorios fuentes y el sistema para la obtención y correspondencia de la información en el modelo virtual en que se basa la vista integrada. Así, para cada repositorio fuente, se debe implementar una aplicación que extraiga sus datos y los haga corresponder con los elementos obtenidos de los otros repositorios fuentes. De esta manera, el coste de actualización de la información de la vista integrada es mínimo. Este tipo de enfoques de integración ha sido utilizado ampliamente en el ámbito de las ciencias biológicas y biomédicas. Por ejemplo, se pueden citar los sistemas TAMBIS [181] y BIOZON [182], que proporcionan una vista unificada e integrada de información dispersa en diferentes repositorios. Otro ejemplo es YOGY [183], un sistema que unifica en una sola vista el resultado de la consulta sobre diversos repositorios de información sobre ortólogos. En este caso no se integra la información, sino que se muestran los resultados de los distintos repositorios utilizando una única vista.

El modelo en que se basa la vista integrada es similar al desarrollado para los sistemas de integración basados en creación de almacenes de datos, ya que la información que representan es similar. Por lo tanto, la flexibilidad de los modelos definidos hacia los cambios o evolución de los repositorios fuente es muy limitada. Además, las correspondencias definidas y las aplicaciones desarrolladas para obtener e integrar la información también poseen las mismas desventajas, ya que están muy especializados. Las fuentes de información pueden no estar disponibles o cambiar con frecuencia, por lo que la vista integrada debe ser robusta contra estas situaciones. Todas estas características hacen que estos sistemas sean complejos y que su tiempo de respuesta sea lento.

La Figura 5.3 describe gráficamente la integración mediante vistas unificadas. En este ejemplo, se muestra el sistema “D” que integra la información de los repositorios “A”, “B” y “C”. A diferencia del sistema de integración mediante almacenes de datos, aquí la información integrada no es almacenada, sino que el proceso de integración se realiza en el momento de la consulta

del usuario en el sistema "D". Además, la información a integrar en cada caso, sólo estará relacionada con la información solicitada por los usuarios. Por lo tanto, la información está actualizada en todo momento, pero el sistema "D" tiene un tiempo de respuesta mayor y es más susceptible a fallos ya que tiene que obtener la información en cada consulta de los repositorios fuente.

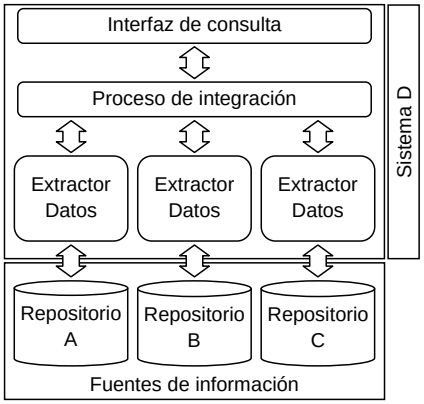

Figura 5.3: Ejemplo de caso de uso de un sistema que utiliza la integración basada en vistas unificadas para integrar de tres fuentes de información independientes.

5.3. Integración mediante vínculos

Esta metodología se basa en la definición de referencias cruzadas entre un registro de un repositorio con su homólogo en otro repositorio. Así, los usuarios pueden utilizar esa referencia para consultar la información de otra fuente de datos. Esta referencia puede estar definida, por ejemplo, a nivel de página Web donde los usuarios pueden consultar la información de diferentes repositorios mediante hipervínculos entre diferentes páginas Web.

Un ejemplo de este tipo de sistemas es el proyecto *identifiers.org*, el cual proporciona vínculos a páginas Web de distintas fuentes de datos que comparten información. Los enlaces son comprobados para asegurar su disponibilidad, pero la integración de la información no es completa, ya que este proceso debe ser realizado manualmente por las personas que lo consulten.

Otro ejemplo es el repositorio NCBI, que establece enlaces entre los diferentes repositorios biológicos y biomédicos que agrupa. Además, contiene el repositorio GenBank, que forma parte del proyecto colaborativo internacional INSDC, para la compartición de referencias entre el Banco de Datos de ADN de Japón (DDBJ), el Laboratorio Europeo de Biología Molecular (EMBL) y GenBank.

La Figura 5.4 muestra un ejemplo de caso de uso donde un sistema "A" define un hipervínculo en su página HTML, de la interfaz de acceso, para referenciar a un registro del sistema "B" y viceversa. En este ejemplo, se puede observar que no hay una integración completa de la información entre sistemas y que los usuarios continúan necesitando realizar un trabajo manual de búsqueda e integración.

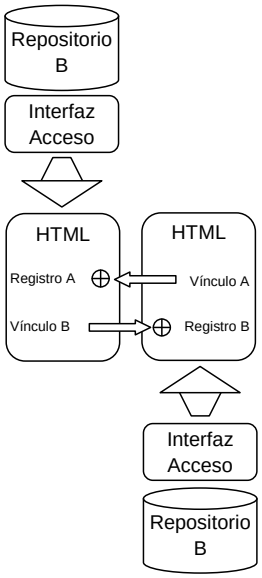

Figura 5.4: Ejemplo de caso de uso de dos sistemas que utilizan integración basada en vínculos para definir referencias cruzadas entre ellos.

5.4. Combinación de información

Este tipo de metodologías tienen como fin combinar diversas fuentes de información y funcionalidades para ofrecer nuevos contenidos o servicios por

agregación [184]. Los tipos de fuentes de información más utilizados por estos métodos son los servicios Web REST [185] y los servicios de distribución de información tipo RSS [186], que pueden ser fácilmente combinados para generar nuevos recursos en la Web.

Un ejemplo de uso de este tipo de sistemas sería obtener información sobre fenotipos humanos y aplicar dicha información a un modelo del cuerpo humano tridimensional donde localizar dichas expresiones fenotípicas. El resultado sería un servicio donde los usuarios pudieran consultar en un modelo visual más intuitivo las expresiones fenotípicas humanas.

Mediante la combinación de contenido y servicios existentes, se pueden ofrecer nuevas aplicaciones y funcionalidades. Estos métodos pueden ser considerados más como agregación que como integración de información, pero tienen el potencial de generar aplicaciones prácticas para la comunidad científica. Las ventajas de estos métodos son su transparencia, facilidad de implementación y menor complejidad que el resto de métodos de integración descritos.

Los sistemas que implementan estos métodos suelen ser producidos para casos concretos de integración debido a su diseño basado en el uso de interfaces de programación existentes y técnicas de desarrollo ágil de software. Así, pueden desarrollarse soluciones especializadas para problemas particulares. Por el contrario, otras soluciones más complejas y grandes a menudo no logran cumplir con las necesidades de los usuarios para las que fueron diseñadas.

Por lo tanto, el ámbito de la metodología de combinación de información se ajusta a los desarrollos simples y rápidos que pueden cumplir con las necesidades de los usuarios de manera más sencilla. Estas características hacen que sean unos métodos que pueden tener diversas aplicaciones en el dominio de la bioinformática. El servicio de anotaciones distribuidas *DAS* (*Distributed Annotation Service*) [187] puede considerarse como una primitiva aproximación a las metodologías por combinación de información, donde el cliente DASTY [188] utiliza los servicios ofrecidos por Uniprot en DAS y los combina con información sobre anotaciones del repositorio InterPro para una visualización más intuitiva de las mismas.

La combinación de información y servicios siguiendo esta metodología no puede considerarse una verdadera integración, sino agregación. Por lo tanto, la nueva información o servicios desarrollados mediante estos métodos no suelen diseñar un modelo para representación de la información y su uso no suele prolongarse en el tiempo. Otros sistemas de integración tienen como objetivo una actividad más permanente, en el sentido de que la integración de la información pueda ser consumida por otras aplicaciones.

5.5. Integración mediante arquitecturas orientadas a servicios

Este tipo de metodologías de integración se basan en la utilización de servicios Web, como SOAP [189] o REST, para proporcionar un método de extracción de información heterogénea. Los sistemas que implementan este método obtienen la información de los repositorios a través de sus respectivas interfaces basadas en servicios Web, para posteriormente combinar la información y presentarla al usuario de manera homogénea. Los servicios Web se basan en estándares bien definidos, por lo que facilitan su implementación tanto por parte de los servidores como de los clientes de dichos servicios. Sin embargo, es necesario que los sistemas clientes de los servicios Web definán un esquema o modelo común en el que integrar la información obtenida.

La implementación de servicios Web en bioinformática está teniendo un gran éxito [190]. Esto es debido a que ofrecen interfaces de programación con las que obtener de manera sencilla datos y, así, poder utilizar dicha funcionalidad por parte de otros sistemas bioinformáticos. El sistema BioCatalogue [191] proporciona un método para identificar y publicar los servicios Web disponibles en el ámbito de la biología y biomedicina. Este sistema contiene un conjunto de servicios Web desarrollados para el dominio de la biología y biomedicina junto con su descripción para poder ser localizados y reutilizados por otros sistemas bioinformáticos.

Sin embargo, el gran número de servicios disponibles, la diversidad de esquemas que utilizan, su mantenimiento por múltiples grupos muy disper-

sos y una pobre documentación disponible, pueden hacer difícil su uso por parte de otros grupos. Además, el uso de servicios SOAP puede provocar un mayor consumo de recursos y complejidad innecesarios en los sistemas de integración.

En la Figura 5.5 se muestra un ejemplo de uso de una arquitectura basada en servicios Web con la que se extrae la información de los repositorios "A", "B" y "C", para ser integrada en un sistema cliente.

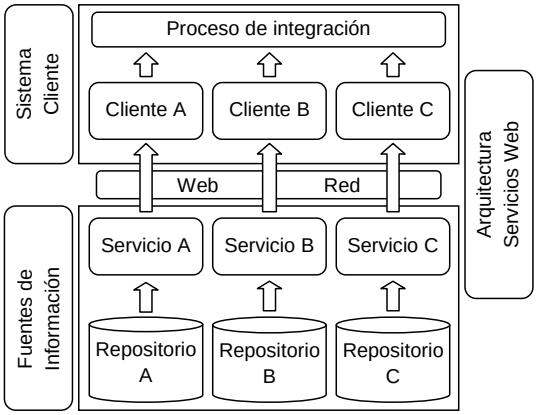

Figura 5.5: Ejemplo de caso de uso de un sistema que utiliza la integración basada en arquitectura orientada a servicios para obtener los datos de repositorios de información heterogéneos.

5.6. Integración mediante arquitecturas dirigidas por modelos

La metodología de integración mediante el uso de arquitecturas dirigidas por modelos puede ser considerada como una especialización del método de integración en vista unificada. Es decir, los servicios y aplicaciones que capturan y procesan la información para su integración se definen en una arquitectura que sigue el desarrollo dirigido por modelos. Por lo tanto, éstos requieren la definición de tipos de datos y vocabulario común para interactuar. Todas las herramientas y recursos siguen estos modelos comunes para poder ser integrados e intercambiar datos.

La ventaja principal de estos sistemas es que están desarrollados como una sola entidad en vez de definir adaptadores entre diferentes sistemas. Por lo general, sólo es posible de lograr entre sistemas fuertemente acoplados, donde el mayor esfuerzo de integración debe ser llevado a cabo por los sistemas que proporcionan la información.

Una aproximación a este tipo de metodologías es el sistema Gaggle [192], que define un conjunto de tipos de datos (nombres, matrices, redes, y colecciones) para unificar diferentes repositorios y sistemas. La integración de sistemas poco acoplados requieren la definición de un conjunto estable de estándares y formatos para el intercambio de información, lo cual es difícil en el dominio de la bioinformática, la cual se caracteriza por su diversidad y heterogeneidad de sistemas, ya que cada uno tiende a definir los suyos propios.

5.7. Integración de aplicaciones

Este tipo de sistemas se basan en el desarrollo de aplicaciones de integración especializadas en un dominio. Están diseñadas para un dominio de información concreto y, por lo tanto, son más difíciles de extender con información de otros dominios. La mayoría se basan en arquitecturas dirigidas por modelos.

Utopia [193] es un ejemplo de este tipo de sistemas especializados. Se trata de una colección de herramientas interactivas para el análisis de secuencias y estructuras de proteínas. Este sistema ofrece una intuitiva interfaz basada en un sofisticado modelo que evita a los usuarios la necesidad de manejar diferentes fuentes de información, formatos de archivos y servicios Web.

5.8. Integración por flujos de trabajo

Este tipo de métodos de integración describe una serie de procesos asociados que se llevan a cabo de manera secuencial. Los sistemas que implementan estos métodos deben coordinar el flujo de información entre los servicios de

extracción de información y las herramientas de análisis, transformación, correspondencia, carga, etc. relacionados.

Este tipo de enfoque puede ser adaptado fácilmente para utilizarse junto con otros métodos de integración, como en los casos de almacenes de datos o vistas integradas. Para ello, se deben definir los procesos necesarios para la integración de la información en un almacén único, en caso del método de almacenes de datos, o los procesos para ofrecer dinámicamente la información, en el caso del método de la vista unificada.

Este método es muy flexible y adaptable. Además, no requiere de ningún modelo común, el cual depende del nivel de integración deseado. Otra ventaja de este tipo de metodología es que permite generar información del estado del proceso de integración, informando del estado en cada fase del flujo de trabajo.

Sin embargo, la flexibilidad y adaptabilidad requieren un gran esfuerzo de diseño e implementación. Por lo tanto, cuanto mejor están diseñados sus procesos, mayor beneficio se obtendrá, pero más costoso será su desarrollo. Estos servicios pueden ser reutilizados para otros flujos de trabajos, reduciendo así el coste de desarrollo de otros sistemas.

Un ejemplo de este tipo de sistemas es Taverna [194]. Este sistema de código libre facilita el uso e integración de repositorios y herramientas bioinformáticas disponibles en la Web, en especial servicios Web. Permite a sus usuarios definir los flujos o secuencias de servicios con los que realizar un conjunto variado de procesos, como análisis de secuencias o anotación genómica. Este sistema puede integrar múltiples servicios para realizar un análisis avanzado de la información.

En la Figura 5.6 se representa al sistema “C”, que implementa un flujo de trabajo para la integración de las fuentes de información “A” y “B”. En el flujo de trabajo definido para la integración de información, se utilizan módulos para extraer la información de los repositorios, para analizar la información obtenida, para transformarla en una representación común con la que poder explotarla, para hacer corresponder la información con un modelo común y para almacenar la información para una posterior consulta.

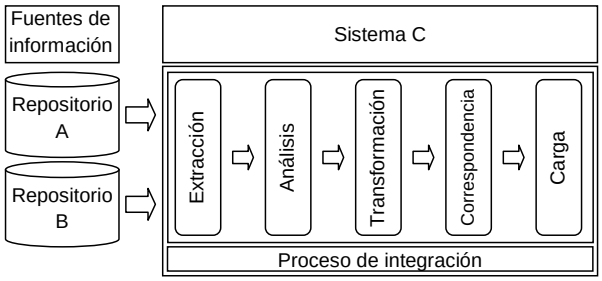

Figura 5.6: Ejemplo de caso de uso de un sistema que utiliza la integración basada en flujos de trabajo para integrar la información de los repositorios "A" y "B".

Capítulo 6

Metodologías de publicación semántica de información biomédica

En este capítulo, se presenta un estudio sobre las diferentes metodologías más importantes para la publicación semántica de información en modelos RDF/OWL. Estas metodologías se basan en la definición de correspondencias entre repositorios tradicionales y modelos semánticos RDF u OWL.

Una de las iniciativas más importantes para la puesta en común de investigaciones sobre las distintas metodologías de publicación tuvo lugar en octubre de 2007, cuando W3C organizó un taller sobre el acceso RDF a bases de datos relacionales. A partir de este taller, se creó el grupo de trabajo RDB2RDF Incubator Group para el estudio de este campo de investigación. Así, en febrero de 2009 se publicaron dos informes: el estudio sobre los últimos avances en el área [195], y la recomendación de la estandarización de un lenguaje para la definición de correspondencias de esquemas de bases de datos con RDF y OWL [196].

Las distintas metodologías de publicación se basadan en la definición de correspondencias para asociar los elementos de los esquemas de los repositorios de información con los elementos de los modelos semánticos [197, 198]. Así, estas metodologías definen lenguajes de correspondencias que intentan

representar las relaciones entre diferentes modelos de representación de información. Las metodologías de publicación de información en modelos semánticos se han clasificado en: (1) metodologías de publicación directa; (2) metodologías de publicación parametrizable; y (3) metodologías de publicación en vistas virtuales.

6.1. Metodologías de publicación directa

Estas metodologías utilizan los esquemas de repositorios para generar automáticamente el modelo ontológico asociado y las correspondencias entre ambos, de modo que pueden considerarse el tipo de enfoque más simple, ya que la publicación de la información del repositorio se lleva a cabo de manera automática. Sin embargo, la definición del modelo semántico u ontológico queda fuera de la supervisión de un experto, por lo que el modelo es muy dependiente de la estructura utilizada para representar el esquema del repositorio.

Ejemplos de sistemas que utilizan esta metodología son DM [199] y OntoGrate [200], los cuales permiten la definición automática de grafos RDF y sus correspondencias con esquemas relacionales de bases de datos para importar sus datos. A continuación, se describirán ambos sistemas.

6.1.1. DM - Direct Mapping

Este sistema está formado por el lenguaje desarrollado por el grupo de trabajo RDB2RDF Incubator Group de W3C. Tiene como objetivo la definición de correspondencias simples que permitan crear grafos RDF o grafos RDF virtuales a partir de bases de datos relacionales, y que puedan ser consultados usando SPARQL o mediante interfaces de manejo de grafos RDF.

A partir de los datos y esquema de una base de datos, este lenguaje genera automáticamente un grafo RDF llamado "grafo directo". Este grafo define diferentes IRI (International Resource Identifiers [201]), compuestos por el IRI base y el IRI relativo. Las claves ajenas de las bases de datos relacionales y los demás valores en las filas de las tablas están expresados en el grafo

directo. Así, la estructura y contenido del grafo directo están influenciados directamente por la estructura y contenidos de la base de datos.

Las correspondencias definidas en este lenguaje transforman la definición de los esquemas de base de datos a un grafo RDF. Por lo tanto, el nombre de una tabla se corresponde con la IRI relativa de tipo del esquema del grafo RDF. La clave principal de la tabla es utilizada para construir la IRI relativa de cada instancia, de modo que la IRI relativa se construye uniendo el nombre de la tabla con el valor de la clave principal. Las claves ajenas, que relacionan dos filas (de la misma o diferente tabla), se corresponderán con la relación existente entre dos instancias, siendo el valor de la clave ajena el identificador que se utiliza para construir la IRI relativa de la instancia destino. Las columnas de las tablas se corresponden con las propiedades de las instancias. La IRI relativa de cada propiedad se construirá juntando el nombre de la tabla con el nombre de la columna. Los valores de las columnas se corresponderán con valores literales de las propiedades o referencias a otras instancias, en caso de ser una clave ajena.

Se proporciona a continuación un ejemplo para explicar el tipo de metodología de correspondencia directa según lo descrito en el párrafo anterior. La tabla 6.1 muestra un ejemplo de tabla de base de datos. La tabla se llama "Persona" y contiene tres columnas: (1) "ID", que se corresponde con la clave primaria; (2)"Nombre", que es el nombre de cada persona; y (3) "Madre", que representa la clave ajena a otra fila de la misma tabla. La tabla contiene dos filas, que se corresponden con las personas "Pepe" y "María".

Persona		
CP		CA
ID	Nombre	Madre
1	Pepe	2
2	María	NULL

Tabla 6.1: Ejemplo de tabla de base de datos.

La tabla 6.2 muestra las tripletas RDF generadas a partir de la tabla 6.1 siguiendo las definiciones de correspondencias dadas por el lenguaje *Direct*

Mapping. Se puede observar la traducción directa entre el esquema de la tabla y las tripletas RDF.

@base <http://foo.example/DB/> @prefix xsd: <http://www.w3.org/2001/XMLSchema#>		
<Persona/ID=1>	<rdf:type>	<Persona>
<Persona/ID=1>	<Persona#ID>	1
<Persona/ID=1>	<Persona#Nombre>	"Jose"
<Persona/ID=1>	<Persona#Madre>	<Persona/ID=2>
<Persona/ID=2>	<rdf:type>	<Persona>
<Persona/ID=2>	<Persona#ID>	2
<Persona/ID=2>	<Persona#Nombre>	"Pepi"

Tabla 6.2: Ejemplo del lenguaje de correspondencias *Direct Mapping*.

Las IRI de las instancias están compuestas por la IRI base y la IRI relativa. Así, la IRI de la tabla es <http://foo.example/DB/Persona>; la IRI de una fila de la tabla es <http://foo.example/DB/Persona/ID=1>; y la IRI de una columna de la tabla es <http://foo.example/DB/Persona#Nombre>.

Este sistema requiere que el esquema de la base de datos esté bien definido. Así, los repositorios donde los tipos de referencias no estén definidos o se permitan excepciones a las reglas de integridad por motivos de optimización, provocan que las correspondencias no puedan definirse correctamente. Además, cuando el esquema contiene referencias tipo N:M o existen columnas con un significado equivalente en tablas diferentes, las correspondencias no producen grafos RDF de buena calidad. Por último, cada repositorio definirá un grafo RDF diferente que dependerá de la estructura de su esquema relacional.

6.1.2. OntoGrate

OntoGrate es una plataforma para la publicación en modelos ontológicos de información presente en bases de datos relacionales. La plataforma está compuesta por: (1) un módulo que transforma esquemas de bases de datos relacionales a ontologías de manera automática; (2) un módulo para alinear conceptos entre dos ontologías diferentes; (3) un módulo para el

descubrimiento automático de correspondencias entre propiedades, clases o instancias y las alineaciones del módulo anterior; (4) un motor de inferencia para resolver las consultas y trasladar los datos entre los distintos recursos de datos usando las correspondencias definidas; (5) una interfaz para definir consultas en algún lenguaje semántico de consultas, como SPARQL u OWL-QL [202]; y (6) un módulo para la encapsulación de la sintaxis de los distintos lenguajes que utiliza, como SQL, SPARQL o Web-PDDL [203].

Esta plataforma fue diseñada para unificar la semántica de esquemas de bases de datos con ontologías mediante la definición de correspondencias entre ellos. Para descubrir las correspondencias de manera automática, se desarrolló un sistema que aprovecha la minería de datos y la alineación de términos para llevarla a cabo. Además, la plataforma permite la traducción de consultas entre distintos repositorios relacionales a través de modelos ontológicos para mejorar la escalabilidad del sistema cuando contiene un gran conjunto de instancias.

Para llevar a cabo la publicación de repositorios a ontologías, la plataforma OntoGrate primero genera automáticamente un modelo ontológico del esquema de base de datos correspondiente. Esta traducción se realiza transformando las tablas en clases, las columnas como propiedades, las claves primarias como propiedades funcionales y las restricciones de integridad como axiomas en la ontología. Una vez conseguida la ontología resultado de la traducción, OntoGrate descubre automáticamente las correspondencias entre la ontología obtenida y los repositorios objetivo.

6.2. Metodologías de publicación parametrizable

Este tipo de metodologías suministra un lenguaje para la definición de las correspondencias entre un esquema relacional y un modelo ontológico existente. Así permiten un mayor control de los procesos de publicación de información en modelos semánticos, ya que la correspondencia con los esquemas de repositorios de información puede ser definida manualmente. Ejemplos de

este tipo de metodologías son los lenguajes R2RML [204], D2R [205] y R2O [206].

6.2.1. R2RML - RDB to RDF Mapping Language

R2RML es un lenguaje para definir correspondencias que está siendo desarrollado por el grupo de trabajo RDB2RDF de W3C, al igual que el lenguaje DM descrito anteriormente. Este lenguaje permite personalizar correspondencias entre bases de datos y grafos RDF. Las correspondencias que se definen relacionan los datos y esquemas de bases de datos relacionales con modelos de datos RDF, utilizando para ello la estructura y vocabulario definido por el usuario. Las correspondencias en el lenguaje R2RML se representan como grafos RDF y se codifican usando la sintaxis Turtle. Al grafo RDF generado se le llama "grafo de correspondencias" y el documento codificado usando Turtle que contiene el grafo de correspondencias es llamado "documento de correspondencias R2RML".

Este lenguaje es independiente de la metodología utilizada para realizar la publicación de información. Así, un sistema podría utilizarlo para almacenar las tripletas RDF obtenidas a partir de las bases de datos o para definir una interfaz de consulta virtual basada en SPARQL, mientras los datos son extraídos de las bases de datos originales.

Las correspondencias R2RML están formadas por una o más estructuras llamadas *TriplesMap* (véase Figura 6.1). Las *TriplesMap* describen las reglas para realizar la traducción del contenido de una fila de una tabla a un conjunto de tripletas RDF. Las tripletas RDF se almacenan por defecto en el grafo RDF base, pero las reglas pueden ser modificadas para que se almacenen en otros grafos RDF etiquetados. Cada *TriplesMap* está compuesta por un componente *SubjectMap*, opcionalmente, por uno o más componentes *PredicateObjectMap* y, opcionalmente también, por uno o más componentes *RefPredicateObjectMap*. El componente *SubjectMap* hace referencia a una definición de una tabla. Una definición de tabla puede tomar la forma de esquema de base de datos, de una vista de la base de datos o de una consulta SQL sobre la base de datos. Los componentes *PredicateObjectMap* definen los

pares (predicado, objeto) relacionados con cada fila de una tabla y que se corresponden con los pares *PredicateMap* y *ObjectMap*. *PredicateMap* contiene las reglas para obtener el identificador del predicado, mientras que *ObjectMap* define las reglas para obtener el valor del objeto. Los componentes *RefPredicateObjectMap* definen los pares (predicado, objeto) relacionados con las claves ajenas de una tabla y está compuesto por los pares *RefPredicateMap* y *RefObjectMap*. *RefPredicateMa*p contiene las reglas para obtener el identificador de la relación predicado del par y el *RefObjectMap* contiene las reglas para obtener el identificador de la instancia objeto del par.

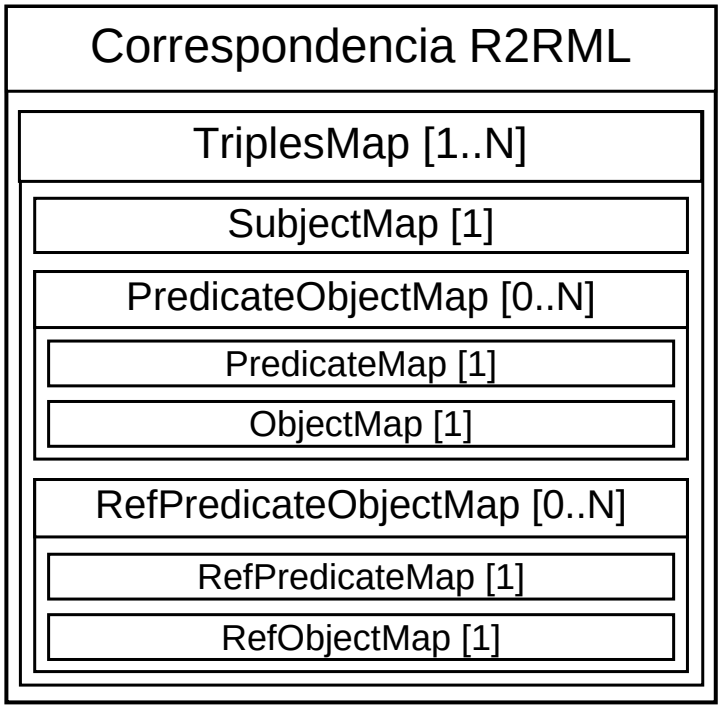

Figura 6.1: Arquitectura de las correspondencias de R2RML

Este lenguaje proporciona un alto nivel de personalización en las correspondencias. Permite el uso de *Nodos Anónimos* (Blank nodes) como sujeto en la creación de las tripletas RDF. También permite la definición de propiedades a partir del valor de una columna de la tabla en los componentes PredicateMap. Sin embargo, esto no es posible en el componente RefPredicateMap, el cual sólo permite la definición de un identificador fijo para todos

los valores de la clave ajena.

Además, puede ocurrir que un nodo anónimo definido para una tripleta RDF no pueda ser referenciado por otra tripleta RDF si estos han sido almacenados en grafos RDF diferentes, ya que, por defecto, su ámbito de uso está dentro de un único grafo.

6.2.2. D2R

D2R es una plataforma para la publicación de información almacenada en repositorios relacionales en repositorios que utilizan un modelo semántico para representar la información. Esta plataforma está compuesta por el lenguaje de correspondencias *D2R Map* y el módulo de integración *D2R Processor*.

El lenguaje *D2R Map* es un lenguaje declarativo que permite definir las correspondencias existentes entre un esquema de base de datos relacional y un modelo ontológico RDF u OWL. Se utiliza el lenguaje XML para codificar las correspondencias.

Este lenguaje está diseñado para definir correspondencias con la suficiente flexibilidad para no necesitar cambiar el esquema de las bases de datos existentes y así poder manejar modelos relacionales más complejos. Esta flexibilidad se logra mediante el uso de consultas SQL en las definición de reglas de correspondencia. El resultado de las distintas consultas SQL se agrupa y luego asocia a las instancias creadas. Este lenguaje permite definir correspondencias sobre relaciones binarias o de mayor grado, sobre propiedades con varios valores, y sobre esquemas complejos derivados de la normalización de tablas en la base de datos.

Una correspondencia D2R en el documento XML se define dentro de la etiqueta "<Map>", que, a su vez, está compuesta por seis elementos principales. En la Figura 6.2, se muestra la estructura de cada correspondencia con la cardinalidad de cada elemento. El primero es "<DBConnection>", que especifica el tipo de conexión y los parámetros de identificación del usuario de la base de datos. El segundo elemento es "<Namespace>", que mediante los atributos "prefix" y "namespace", describe el prefijo y su correspondiente

identificador del espacio de nombres. Los elementos tercero y cuarto se corresponden con "<Prepend>" y "<Postpend>", que permiten añadir fragmentos de código al principio o al final del fichero de salida, respectivamente. El quinto elemento es "<ProcessorMessage>" que, mediante los atributos "saveAs" y "outputformat", indican el nombre y el formato del fichero de salida. Por último, el elemento "<ClassMap>" describe una correspondencia. En este elemento se definen los siguientes atributos: "type", que indica la clase asociada a las instancias que se van a crear; "sql", la cual contiene la consulta SQL a ejecutar para obtener las propiedades de las instancias; "groupBy", que indica la columna sobre la que se va a agrupar el resultado de la consulta SQL; y "uriPattern", el cual establece la URI base de las instancias a crear. El elemento "ClassMap" está compuesto por otros dos tipos de elementos, "<DatatypePropertyBridge>" y "<ObjectPropertyBridge>". El primero se corresponde con la descripción de una correspondencia de una columna con una propiedad, y el segundo de una columna con una relación de la ontología.

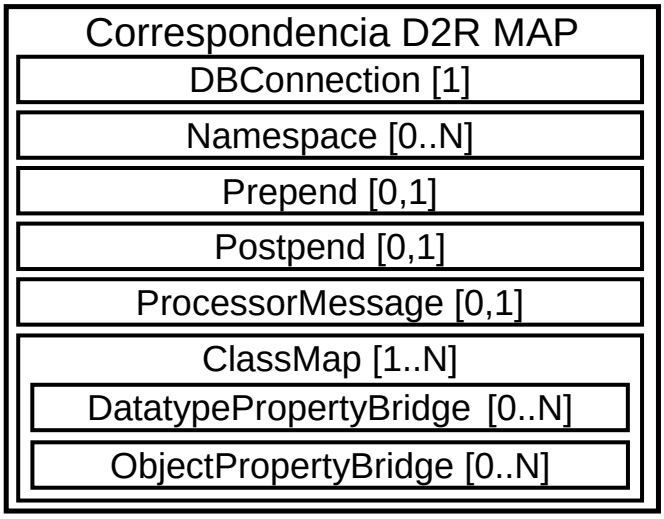

Figura 6.2: Arquitectura de las correspondencias D2R MAP

Por otro lado, el módulo de publicación de información D2R Processor utiliza las correspondencias para importar los datos desde los repositorios relacionales. La publicación se lleva a cabo mediante cuatro etapas:

1. Selección de los registros asociados a una clase o grupo de clases similares con la consulta SQL definida dentro de cada ClassMap.

2. Agrupación de los registros agrupados en base a la columna de la tabla especificada en el atributo groupBy de ClassMap.

3. Creación de las instancias asignándoles un identificador.

4. Adición a cada instancia de las propiedades y relaciones que le corresponden de la base de datos.

Por último, cabe indicar que el resultado de la publicación puede ser exportado como RDF, N3, N-Triples o modelos ontológicos generados con Jena API.

6.2.3. R2O

Este sistema define un lenguaje declarativo y extensible que permite definir las correspondencias entre esquemas de bases de datos y ontologías codificadas en RDF u OWL. R2O proporciona un conjunto de primitivas XML con una semántica bien definida que facilita la especificación de correspondencias entre modelos de base de datos y ontologías con una baja similitud.

Por lo tanto, este lenguaje tiene como objetivo ofrecer la herramientas necesarias para definir las correspondencias de manera flexible sin que se requiera una modificación de los modelos de base de datos o de la ontología existentes. Una vez definidas las correspondencias, éstas pueden ser procesadas automáticamente para la publicación de datos con la ontología existente. Las principales características de R2O son:

- Una correspondencia indica cómo crear una instancia a partir de los datos recogidos de una base de datos. Estas correspondencias pueden ser procesadas de manera automática para poblar una ontología.

- Este lenguaje puede ser implementado por otras herramientas que permitan la definición de correspondencias automáticas, como en el caso de Cupid [207, 208] o DB2OWL [209].

- La definición de las correspondencias puede ser verificada para detectar inconsistencias y ambigüedades.

- Las correspondencias también permiten verificar la integridad de los esquemas de base de datos según las restricciones del modelo ontológico.

- Las correspondencias pueden ser utilizadas para caracterizar las fuentes de datos y, así, permitir distribuir consultas dinámicamente sobre sistemas semánticos de información integrada.

En la Figura 6.3, se muestra la arquitectura de las correspondencias R2O. Los elementos principales son: (1) *import*, que se utiliza para especificar un conjunto de URI a importar al conjunto de instancias resultado de las correspondencias; (2) *ontology*, que define una URI base para construir las URI de las instancias que se van a crear; (3) *dbschema-desc*, que define el esquema de base de datos que se utilizará en la correspondencia; (4) *conceptmap-def*, que define una correspondencia entre la clave primaria de base de datos y un concepto de la ontología; (5) *attributemap-def*, que define una correspondencia de un atributo de la base de datos con una propiedad de una instancia en la ontología; (6) *relfromatt-def*, que define una correspondencia de un atributo de la base de datos con una relación en la ontología, y que implica la creación de una nueva instancia; y (7) *dbrelationmap-def*, que define una correspondencia de una clave ajena de la base de datos con una relación en la ontología entre dos instancias.

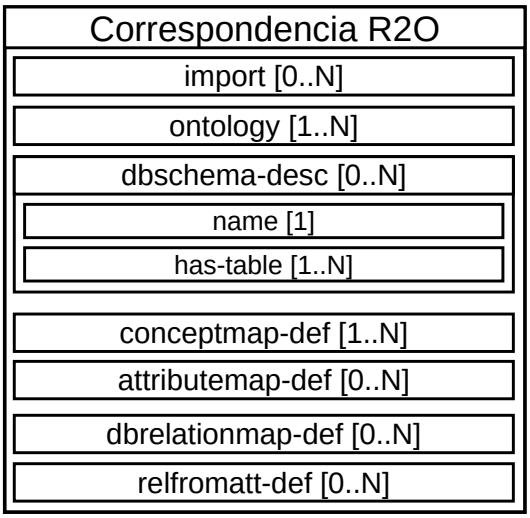

Figura 6.3: Arquitectura de las correspondencias R2O

Además, en los elementos (4), (5), (6) y (7), se incluyen opciones para transformar los datos recogidos de la base de datos y que van a ser asignados a una instancia. También, permiten hacer una selección o proyección en una tabla de la base de datos para especializar las correspondencias, mediante la definición de condiciones evaluadas como expresiones booleanas. Todas estas opciones ofrecen un alto grado de flexibilidad a la hora de definir correspondencias, lo que permite que este lenguaje se adapte a la mayoría de repositorios y ontologías. Sin embargo, a la hora de definir las correspondencias, se exige que se especifique la URI de los conceptos, propiedades y relaciones de la ontología involucrados, por lo que si la ontología posee jerarquías de recursos con muchos elementos, se han de definir muchas correspondencias, aunque luego no vayan a ser necesarias.

6.3. Metodologías para la publicación en vistas virtuales

Esta metodología está enfocada a la definición de correspondencias para la generación de vistas virtuales en RDF u OWL. Estas correspondencias

permiten acceder a la información presente en bases de datos relacionales a través de las relaciones establecidas entre el modelo semántico y el esquema del repositorio correspondiente. La información es obtenida del repositorio y publicada en las vistas virtuales dinámicamente, según las peticiones de los usuarios. Estas vistas pueden ser utilizadas para realizar búsquedas de información mediante lenguajes de consultas semánticos. Las consultas definidas de este modo serán trasformadas al lenguaje SQL para obtener los datos de los repositorios. Por lo tanto, no se necesita generar un nuevo repositorio con el modelo semántico, sino que la información sólo está almacenada en el repositorio original. Este enfoque reduce el coste del mantenimiento y actualización de la información publicada, pero introduce una mayor latencia en su acceso debido a la capa software necesaria para acceder al repositorio y publicar su información. Esto perjudica su eficiencia para aplicaciones como las de inferencia. Además, el sistema es dependiente de la disponibilidad de acceso y consulta mediante SQL de los repositorios originales.

Así, D2RQ [210], las vistas RDF de Virtuoso [211] y R2RML pueden ser considerados ejemplos de sistemas que utilizan este tipo de metodología. El sistema R2RML ya ha sido descrito en la sección 6.2.1, pero el resto se describen a continuación:

6.3.1. D2RQ

La plataforma D2RQ tiene como objetivo ofrecer un acceso basado en RDF sobre repositorios sin tener que replicar su contenido. La plataforma está formada tres elementos principales: (1) el *lenguaje D2RQ* de definición de correspondencias; (2) el *motor D2RQ*, para adaptar las interfaces de programación de Jena, y Sesame [212], para que permitir el acceso a la información utilizando consultas SQL; y (3) el *servidor D2R*, que tiene como finalidad publicar en la Web una interfaz para navegar por la información del repositorio mediante consultas SPARQL o Linked Data.

En concreto, el lenguaje D2RQ define correspondencias entre esquemas de bases de datos relacionales y ontologías. Las correspondencias se definen como un documento RDF. El espacio de nombres de las correspondencias D2RQ es

http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/0.1#. En la Figura 6.4, se muestran las partes de las que se compone un documento RDF donde se definen las correspondencias D2RQ. El documento consta de cuatro elementos bien diferenciados. El primero es el conjunto de *prefijos* que se usarán en la descripción de las correspondencias. El segundo es *d2rq:Database*, que describe los parámetros necesarios para realizar la conexión a la base de datos. El tercero es el elemento *d2rq:ClassMap*, el cual representa a una clase o un conjunto de clases similares de la ontología. Y el cuarto es *d2rq:PropertyBridge*, el cual especifica cómo se pueden obtener las propiedades de las instancias.

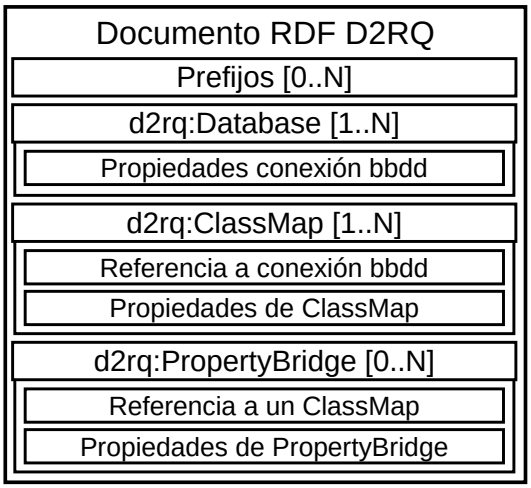

Figura 6.4: Elementos de un documento RDF de correspondencias D2RQ

Además de los elementos descritos en la Figura 6.4, el lenguaje D2RQ contiene otros elementos que extienden su funcionalidad. Estos elementos permiten traducir los datos cuando los trasladan de la base de datos a la vista RDF y viceversa, o añadir condiciones a los elementos ClassMap y PropertyBridge para restringir valores para crear las instancias.

6.3.2. Vistas RDF de Virtuoso

Las vistas RDF de Virtuoso son proporcionadas por el Servidor Universal OpenLink Virtuoso [213]. Esta característica permite al servidor transformar

```
@prefix rdf: < <http://www.w3.org/1999/02/22-rdf-syntax-ns#>> .
@prefix prd: < <http://localhost:8890/rdfv/schemas/product>> .
graph <http://localhost:8890/rdfv/testdata/products#>
subject prd:product_iri(PRODUCT.PRODUCT_ID)
predicate rdf:type
object prd:Product
```

Tabla 6.3: Ejemplo de patrón de correspondencia de una clase RDF con una clave primaria.

dinámicamente el contenido de bases de datos relacionales a RDF y ponerlo a disposición para su consulta a través de una interfaz de consultas SPARQL. De esta manera, Virtuoso permite definir consultas SPARQL sobre las vistas RDF y, al mismo tiempo, sobre repositorios RDF locales.

En el modo más simple de uso de las vistas RDF, las correspondencias transforman el resultado de una consulta SQL a tripletas RDF. Las correspondencias se definen usando un patrón de correspondencias de cuatro elementos. La definición de estos patrones supone la representación de un meta esquema RDF (consultar Tabla 6.3). Los cuatro elementos del patrón se corresponden con el grafo, el sujeto, el predicado y el objeto. Además, en los patrones se pueden definir consultas SQL con el fin de restringir los valores de las tablas a utilizar en las transformaciones.

Para llevar a cabo las transformaciones, se han de definir una serie de elementos (véase Tabla 6.4): primero la IRI de clase RDF para cada tabla de la base de datos; segundo la IRI de los sujetos de las tripletas RDF relacionados con una clave primaria en la tabla; y por último la IRI de los predicados RDF de las columnas que no son clave en la tabla.

Con los elementos descritos, Virtuoso es capaz de generar vistas RDF y así mostrar la información del repositorio como tripletas RDF que pueden ser consultadas mediante SPARQL. Además, como la transformación se realiza de manera dinámica, los cambios en la definición de las correspondencias tiene un efecto inmediato sobre las vistas RDF sin tener que modificar el esquema de la base de datos o el grafo RDF.

```
@prefix rdfs: < <http://www.w3.org/2000/01/rdf-schema#>> .
@prefix prd: < <http://localhost:8890/rdfv/schemas/product>> .
prdf:Producta rdfs:Class ;
  rdfs:label “Product” ;
  rdfs:comment “An OpenLink product” .
prdf:product_ida rdfs:Property ;
  rdfs:domain prdf:Product ;
  rdfs:range xsd:integer ;
  rdfs:label “product id” .
```

Tabla 6.4: Ejemplo de definición de una clase y una propiedad RDF.

Capítulo 7

Objetivos de la tesis

El desarrollo de las tecnologías de la información y las comunicaciones, y en particular la evolución de la Web, han permitido que esté al alcance de un mayor número de grupos de investigación biológicos y biomédicos publicar sus resultados y que dichos resultados estén accesibles de manera libre para toda la comunidad científica. Aunque han surgido proyectos a nivel nacional e internacional para reunir información biomédica y biológica en grandes repositorios y establecer referencias entre ellos, la situación actual de la Web permite que sólo se puedan compartir documentos para ser consumidos por humanos, no estando éstos en un formato adecuado para su explotación por aplicaciones de manera autónoma.

Así, con la implantación de la Web semántica, se pretende que la información no esté representada en documentos Web, sino que los datos estén anotados individualmente y relacionados entre sí. Este cambio de paradigma permite que la información no pueda ser sólo consumida por humanos, sino también por aplicaciones.

La Web semántica permite representar conjuntos de datos, anotarlos con información semántica del dominio y relacionar entre sí la información del mismo conjunto de dato y con otros conjuntos de datos externos. De esta manera, aplicaciones autónomas pueden descubrir, procesar y navegar entre la información de distintos conjuntos de datos. En los últimos años, han surgido iniciativas para crear los modelos necesarios para anotar la informa-

ción biológica mediante ontologías. Uno de los proyectos con más éxito en este aspecto es la Gene Ontology, una ontología para describir genes desde diferentes aspectos, como su función molecular, su componente celular o su proceso biológico.

Además, en la disciplina de la bioinformática, han surgido diversos sistemas que explotan las tecnologías de la Web semántica para la anotación de repositorios relacionales utilizando ontologías o mediante la generación automática de éstas a partir de esquemas de repositorios tradicionales. Estos sistemas tienen como objetivo transformar la información de los repositorios relacionales aportando una conceptualización con la que mejorar la semántica de sus datos. Otras iniciativas como la Linked Open Data, trata de establecer los requisitos para publicar conjuntos de datos RDF, y favoreciendo la interconexión entre repositorios para poder navegar entre ellos.

Este nuevo escenario para gestionar datos biológicos permite explotar la información de manera automática, implementando sistemas que puedan recuperar datos directamente de los repositorios y, al mismo tiempo, poder navegar entre ellos para buscar nueva información. Estas funcionalidades permiten que surjan aplicaciones que puedan establecer relaciones entre datos que no estén directamente asociados en un mismo repositorio. La investigación traslacional entre el dominio de la biología y el de la biomedicina persigue este objetivo, hacer explícitas relaciones implícitas entre ambos dominios.

El trabajo de esta tesis ha sido desarrollado para definir un entorno para la gestión semántica de información biológica y biomédica, y así facilitar la investigación traslacional. Se ha definido una metodología para la integración de diferentes repositorios de información biológica y biomédica. Esta metodología abarca la definición y creación del repositorio semántico, la integración de las fuentes de información biológica y biomédica, y la definición de una ontología global que guía el proceso de integración. Así, no sólo se integra la información de un repositorio con otro, evitando la redundancia de información, sino que se establecen las relaciones necesarias para poder navegar entre la información de los distintos dominios.

A continuación, se describirán los objetivos que han guiado el trabajo de esta tesis y la metodología de investigación empleada.

7.1. Objetivos

Esta tesis pretende facilitar la gestión y explotación de los repositorios de información biológica y biomédica, mediante la anotación semántica de la información, y a través de su integración con otros repositorios y bio-ontologías asociadas para crear un repositorio semántico. Este repositorio semántico permite explotar tanto su modelo semántico asociado como las anotaciones e instancias para facilitar la investigación traslacional. Así, los objetivos de esta tesis se pueden resumir en los siguientes:

- Diseño e implementación de una metodología para la integración en un repositorio semántico de bases de datos relacionales sobre información biológica y biomédica.
- Diseño e implementación de un método de integración para permitir que repositorios relacionales con un esquema distinto puedan ser integrados evitando la redundancia de la información y compartiendo una misma definición de ontología.
- Diseño e implementación de distintos métodos de consulta y explotación de la información para facilitar su uso por personas y aplicaciones, y favorecer la investigación traslacional.
- Validación de los resultados obtenidos, en el escenario de los genes ortólogos y las enfermedades genéticas hereditarias.

7.2. Metodología de investigación

La metodología propuesta para el proceso de integración se basa en el uso de tecnologías de la Web semántica para anotar la información almacenada de manera estructurada en bases de datos biológicas y biomédicas con el fin de obtener un repositorio semántico que permita su explotación tanto por agentes software autónomos como por personas. La definición y reutilización de bio-ontologías es la pieza central de la metodología. Como resultado de la metodología de integración, se obtenido un repositorio donde se evita

la ambigüedad de la información y su posible redundancia. Para lograr los objetivos de la tesis, se ha seguido la siguiente metodología de investigación:

- Análisis del estado del arte:
  - Estudio de los repositorio bioinformáticos de información biológica y biomédica. Esto implica estudiar los esquemas de representación de la información en los repositorios bioinformáticos, las características de los elementos almacenados y las relaciones entre los distintos repositorios. A partir del estudio de los distintos tipos de repositorios, selección de los más útiles para el objetivo de integración de repositorios bioinformáticos sobre grupos de genes ortólogos y enfermedades genéticas humanas.
  - Estudio de las tecnologías de la Web semántica. Se estudiaron las tecnologías y herramientas que ofrece la Web semántica para gestionar la representación de información mediante las ontologías y la funcionalidad que proporcionan los razonadores actuales. Además, se analizaron los lenguajes de consulta y actualización asociados a los repositorios RDF, así como, la iniciativa Linked Data para la publicación y compartición de conjuntos de datos enlazados a través de la Web.
  - Estudio de las aplicaciones de las tecnologías de la Web semántica al dominio de la biología y biomedicina. En concreto, el estudio se enfocó en las bio-ontologías existentes y el conocimiento del dominio que modelan. También se estudió la aplicación de la iniciativa de Linked Data para publicar conjuntos de datos RDF sobre información biológica y biomédica a través de la Web.
  - Estudio de las metodologías de integración aplicadas en el dominio de la bioinformática.
  - Estudio de las metodologías para la publicación semántica de información biológica y biomédica.
- Definición de una metodología para la integración y publicación semántica de información de repositorios bioinformáticos en base al uso de tec-

nologías de la Web semántica para facilitar dicha tarea. En concreto, el proceso de integración estará guiado a través de la definición de una ontología global que será la conceptualización de la semántica asociada a los repositorios a integrar. La metodología permitirá el uso de un método automático de integración de información que, a través de las correspondencias entre los esquemas de los repositorios bioinformáticos y la ontología global, genere el repositorio semántico.

- Definición de una ontología global que conceptualice el conocimiento de los dominios de los genes ortólogos y las enfermedades genéticas. Definición de las relaciones entre ambos dominios de información y el estudio del conocimiento reutilizable de las bio-ontologías existentes.

- Definición y diseño del repositorio semántico que integre la información de repositorios bioinformáticos. Análisis del resultado de la metodología de integración.

- Diseño y desarrollo de métodos de consulta de información a través del uso de parámetros y filtros de búsqueda especificados por el usuario para obtener información precisa del repositorio semántico.

- Métodos de explotación de datos almacenados en un repositorio semántico:

  - Diseño y desarrollo de métodos que exploten el conocimiento representado en un repositorio semántico para asistir a los usuarios en la definición de consultas avanzadas que permitan extraer datos de manera más precisa.
  - Diseño y desarrollo de métodos que permitan explotar la información y conocimiento del repositorio semántico por un agente software de manera autónoma.

- Publicación y asociación de la información almacenada en el repositorio semántico en la nube de LOD, así como desarrollar las interfaces para su consulta.

- Validación del trabajo de tesis en el escenario de la integración de información de los dominios de los genes ortólogos y de las enfermedades genéticas humanas.

Bloque II

Metodologías y herramientas para la gestión semántica de información biomédica en investigación y traslacional

Capítulo 8

Metodología de integración semántica de repositorios bioinformáticos

En este capítulo, se describirá la metodología para la integración semántica de información biológica y biomédica desarrollada en este trabajo. A través de dicha metodología, se puede conseguir que repositorios bioinformáticos sean integrados en un repositorio semántico, el cual sigue un esquema basado en un modelo ontológico. Como resultado, se obtendrá un sistema donde los dominios, biológicos y biomédicos, están conectados y, por lo tanto, se puedan facilitar las tareas de investigación traslacional.

La metodología mostrada en la Figura 8.1 utiliza como parte central una *Ontología Global* que conceptualiza el conocimiento del dominio de los repositorios a integrar. La ontología global es utilizada para guiar el *Proceso de Integración* con el que se obtiene como resultado, un *Repositorio Semántico*. Por su parte, el repositorio utiliza la ontología global definida como modelo para representar la información, de manera que ésta puede estar enlazada entre sí y/o con información de otras ontologías biomédicas. El repositorio semántico está diseñado para que su contenido pueda ser procesado de manera autónoma por aplicaciones de terceros.

En concreto, la metodología está compuesta por los siguientes módulos:

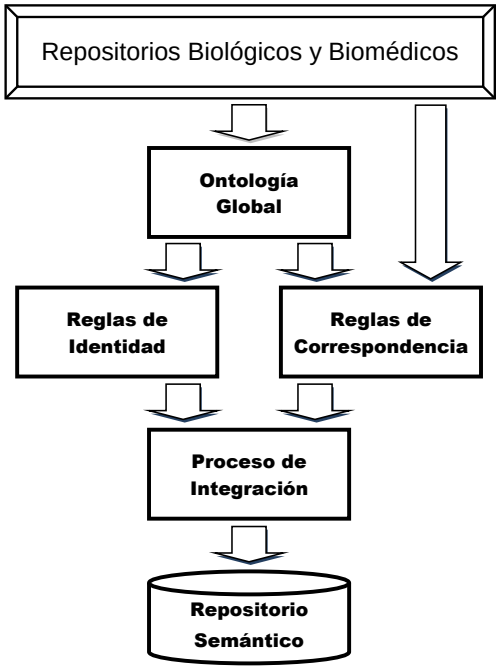

Figura 8.1: Metodología para la integración semántica de información biológica y biomédica.

Repositorios Biológicos y Biomédicos: repositorios de información de donde se extraerá la información a integrar.

Ontología Global: abarca el conocimiento del dominio biológico y biomédico de los repositorios fuente, así como la importación del conocimiento definido en otras bio-ontologías. Permite guiar el proceso de integración y es el modelo que representa cómo la información del repositorio semántico está relacionada.

Reglas de Correspondencia: indican la correspondencia entre los repositorios de información y la ontología global. El formato utilizado para representar las reglas permite su utilización independientemente del dominio de la información. Las reglas de correspondencia permiten automatizar el proceso de integración y el mantenimiento del repositorio semántico.

Reglas de Identidad: definen la identidad o equivalencia entre individuos

de una misma clase. Describen a través de las propiedades, anotaciones y relaciones que tienen asociadas un individuo, los requisitos para establecer sus propiedades de equivalencia con otros individuos del repositorio semántico. Estas reglas, que permiten utilizar el conocimiento del dominio para identificar de manera avanzada, que con un identificador, a los individuos del modelo. Se utilizan en la integración de las instancias para reducir la duplicidad de información en el repositorio semántico.

Proceso de Integración: establece un procedimiento automático para la integración de información de repositorios relacionales con un repositorio semántico, modelado por una ontología. Este módulo toma como entrada las reglas de correspondencia y las de identidad para integrar la información en un repositorios semántico.

Repositorio Semántico: almacena la información integrada de los repositorio de entrada y la representa siguiendo el modelo semántico definido en la ontología global. Este repositorio permite que las aplicaciones puedan procesar, de manera autónoma, la información y así facilitar el desarrollo de nuevas herramientas que exploten las ventajas de la representación formal del conocimiento del dominio.

Para llevar a cabo la integración de información biológica y biomédica entre diferentes repositorios, se debe tener en cuenta la redundancia. Los repositorios bioinformáticos contienen instancias repetidas que no comparten un identificador único. Además, la integración de datos con un formato distinto o la aparición de datos en lenguaje natural, como en los nombres de los genes y enfermedades genéticas, provoca que sea más difícil realizar la integración.

En las siguientes secciones de este capítulo, se describirán las características de la ontología global, la metodología de integración, la definición de las reglas de correspondencia y las reglas de identidad, y el repositorio semántico. Por último, en la sección 8.4, se incluye un ejemplo ilustrativo de la metodología de integración definida.

8.1. Definición de la ontología global

La ontología global establece las relaciones entre clases y propiedades del modelo ontológico que se van a utilizar para publicar la información integrada. Así, la ontología global representa el modelo semántico común de los repositorios fuente de información.

Para la definición de la ontología, se debe analizar la información y el conocimiento presentes en los repositorios fuentes para conceptualizar su dominio. Dicha conceptualización debe ser revisada formalmente para evitar incoherencias, así como evaluada por un experto del dominio, ya que cada repositorio puede contener diferente información que está representada de diversas maneras. Es recomendable que el modelo ontológico común no esté influenciado por la representación física del contenido de los repositorios, sino por su representación conceptual.

La definición de la ontología global puede implicar la conceptualización de diferentes dominios de conocimiento que produzcan diferentes modelos semánticos. Dicha situación puede provocar la definición de conceptos aislados en el grafo del modelo. Para facilitar el proceso de integración y para favorecer la investigación traslacional utilizando el repositorio semántico, se debe evitar dicha situación, ya que los individuos creados que tengan como tipo un concepto aislado no podrán ser relacionados con el resto. Por lo tanto, esta situación debe ser considerada como un indicador de una baja calidad en la conceptualización.

La ontología global guía el proceso de integración mediante la definición de correspondencias entre éste y los recursos de información a integrar. De esta manera, los distintos recursos tienen que tener definidos los conceptos, relaciones y propiedades asociados a la información a integrar en la ontología global. Esta será poblada instanciando los conceptos con la información de los repositorios y creando los individuos que formarán parte del repositorio semántico.

En la definición de la ontología global, se pueden importar definiciones de otras ontologías y definir relaciones entre ellas para reutilizar información ya definida y verificada por la comunidad científica y así mejorar la calidad de

la misma. Por ejemplo, en el dominio de la biología y biomedicina, se pueden reutilizar las bio-ontologías de OBO Foundry para facilitar la consulta y navegación de la información integrada en el repositorio semántico mediante los conceptos definidos en ellas.

8.2. Proceso de integración

Este proceso está diseñado para integrar información de repositorios existentes en un repositorio semántico. El repositorio semántico contiene la información descrita utilizando un modelo ontológico que abarca todos los repositorios integrados, y representado como la ontología global en esta metodología. Así, este proceso se basa en la definición de correspondencias que permitan establecer explícitamente las relaciones entre los esquemas de diferentes repositorios y los modelos ontológicos asociados.

La Figura 8.2 muestra gráficamente las relaciones entre los diversos componentes involucrados en el proceso de integración. Primero, el *Módulo de Reglas de Correspondencia* es el encargado de procesar la definición de las *Reglas de Correspondencia*, las cuales relacionan un esquema del repositorio fuente con un modelo ontológico. Segundo, el *Módulo de Reglas de Identidad* se encarga de procesar la definición de reglas de identidad con el fin de describir ,de una manera más avanzada, la identidad de los individuos presentes en el repositorio semántico, y no sólo mediante el uso de claves o identificadores. Esto es necesario para evitar la duplicidad o redundancia de información. Las reglas de identidad permiten solucionar los problemas de heterogeneidad de los recursos de información provocados por los diferentes identificadores utilizados en los distintos repositorios. Así, cada una de las *Reglas de Identidad* describen los requisitos de identidad de las instancias para un concepto de la ontología. Por último, el *Módulo de Integración* utiliza las definiciones de reglas de correspondencia para crear los individuos en la ontología y las reglas de identidad para descubrir si los individuos creados ya están representados en el *Repositorio Semántico*.

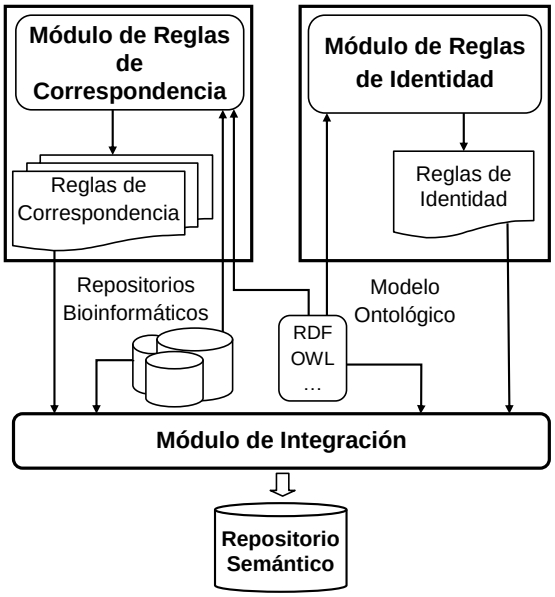

Figura 8.2: Proceso de integración.

8.2.1. Reglas de correspondencia

Las reglas de correspondencia tienen como objetivo relacionar los esquemas de los repositorios a integrar, como los modelos ontológicos para la instanciación de individuos, la asignación de sus propiedades y el establecimiento de las relaciones con otros individuos de la ontología.

Las reglas de correspondencia se codificadan utilizando documentos XML, donde cada documento contiene las reglas relacionadas con las correspondencias entre un esquema de un repositorio y una ontología concretas. En el Anexo 13.3.2, se describe en detalle la gramática definida para representar estas reglas. En concreto, la Tabla 1 muestra formalmente los elementos y gramática de las reglas de correspondencia.

Para la definición de las reglas de correspondencia, se distinguen entre tres tipos diferentes de reglas. El primero, identificado mediante la etiqueta "<type>DB2Class</type>", tiene como objetivo establecer las correspondencias entre las claves primarias de las tablas de los repositorios bioinformáticos y las clases definidas en el modelo ontológico. El segundo tipo de reglas se utiliza para representar las correspondencias entre columnas de una

tabla y las propiedades de un individuo de la ontología. Este tipo de reglas está identificado mediante la etiqueta "<type>DB2Prop</type>". El último tipo de reglas de correspondencia es el que establece las relaciones entre distintos individuos de la ontología mediante la información presente en los repositorios fuente. Se identifica este tipo de reglas por medio de la etiqueta "<type>DB2Rel</type>". Cada definición de un tipo de regla de correspondencia está delimitada entre las etiquetas XML "<map>" y "</map>" dentro del documento general.

A continuación, se describirán cada tipo de regla de correspondencia.

Reglas de correspondencia DB2Class

En la tabla 8.1 se muestra un ejemplo del primer tipo de regla de correspondencia, *DB2Class*. Este tipo de regla permite asociar una clase de la ontología con una tipo de identificador y tabla específicos del repositorio. Normalmente, en un esquema de base de datos una tabla puede considerarse que está asociada a una clase de la ontología, donde su clave primaria identifica las instancias. Así, la columna identificada en este tipo de reglas se corresponde con los identificadores de instancias en la tabla. Los distintos valores de la columna se corresponden con los posibles individuos en la ontología. Así, la clase o concepto de la ontología se identifica en la regla mediante su URI, y se indica como valor en la etiqueta "<class>". La correspondiente tabla y columnas de la base de datos se indican con la etiqueta "<db>" mediante la nomenclatura: "NombreTabla.NombreColumna". El uso de este tipo de reglas está indicado en caso de que se necesite crear instancias nuevas en el repositorio.

```
<map>
 <type>DB2Class</type>
 <class><id>http://miuras.inf.um.es/ontologies/OGO.owl#Gene</id></class>
 <db><id>geneinf.geneId</id></db>
</map>
```

Tabla 8.1: Ejemplo de regla de correspondencia de tipo *DB2Class*, entre una tabla y el concepto de gen en la ontología.

Reglas de correspondencia DB2Prop

La tabla 8.2 describe el tipo de regla de correspondencia *DB2Prop*. Este tipo de regla relaciona una propiedad de una ontología con las columnas correspondientes de una tabla del repositorio a integrar. Esta regla está compuesta por tres tipos de etiquetas principales: (1) *source*, que identifica los sujetos asociados con la propiedad; (2) *predicate*, que identifica la propiedad de la ontología; y (3) *target*, que identifica los distintos valores de la propiedad para un sujeto concreto. De esta manera, mediante la etiqueta *source*, se describe la columna en la tabla que identifica las instancias sujeto relacionadas con la propiedad y su clase asociada en la ontología. Para ello, utiliza las etiquetas "<class>" y "<db>" del mismo modo descrito en la regla tipo *DB2Class*. La etiqueta "<class>" identifica la clase en la ontología y la etiqueta "<db>" la columna y la tabla que identifica al individuo, es decir, las filas en la tabla asociadas al identificador del individuo. La etiqueta *predicate* se corresponde con una propiedad o anotación de la ontología, y ésta es identificada mediante su URI. Por último, la etiqueta *target* define los parámetros para extraer los valores de la propiedad mediante el uso de la etiqueta "<db>". En esta etiqueta, se utiliza la misma nomenclatura "tabla.columna" descrita anteriormente para identificar las columnas de las filas de la tabla en la etiqueta *source*. Así, se obtienen los valores de la propiedad y se asocian a las instancias correspondientes. Además, es necesario que las columnas identificadas en la etiqueta "<db>" de *source* y *target* pertenezcan a la misma tabla, ya que los datos deben estar almacenados en la misma tabla.

Reglas de correspondencia DB2Rel

El último tipo de regla de correspondencia proporciona la información necesaria para relacionar dos individuos de la ontología mediante la información almacenada en una tabla del repositorio. En el ejemplo de la Tabla 8.3, se describe la regla de correspondencia para relacionar un gen con un individuo de la bio-ontología de Gene Ontology. Las instancias de las dos ontologías se conectan a través de la relación *participates_in*, definida en la ontología de

```
<map>
 <type>DB2Prop</type>
 <source>
  <class>
   <id>http://miuras.inf.um.es/ontologies/OGO.owl#Gene</id>
  </class>
  <db><id>geneinf.geneId</id></db>
 </source>
 <predicate>
  <property>
   <id>http://miuras.inf.um.es/ontologies/OGO.owl#Gene_name</id>
  </property>
 </predicate>
 <target>
  <db><id>geneinf.geneName</id></db>
 </target>
</map>
```

Tabla 8.2: Ejemplo del tipo de regla de correspondencia *DB2Prop* para la propiedad *Gene_name*.

relaciones en el dominio biomédico RO. Esta regla está compuesta, al igual que la regla tipo *DB2Prop*, de tres etiquetas principales: *source*, *predicate* y *target*. Así, la etiqueta *source* indica mediante las etiquetas "<class>" y "<db>", la clase de la ontología y la columna clave para seleccionar la información del sujeto de la relación. La etiqueta *target*, al igual que la etiqueta *source*, utiliza "<class>" y "<db>" para identificar los individuos objeto de la relación con la información de la tabla. La etiqueta *predicate* identifica la relación de la ontología mediante su URI en la etiqueta "<property>". La relación puede ser especializada según el valor de una columna de la tabla definido en la etiqueta "<db>", así es posible establecer relaciones diferentes entre individuos, aunque deben ser sub-relaciones de la definida en la etiqueta '<property>". Esta posibilidad requiere que las relaciones involucradas estén anotadas con el mismo valor de la columna indicada en la ontología mediante la anotación *"label"*. Por último, indicar que en cada regla de correspondencia solo se puede hacer referencia a una misma tabla, para evitar incoherencias.

En resumen, estos tres tipos de reglas son definidos para identificar los distintos individuos asociados a una clase de la ontología (*DB2Class*), a asignarles sus propiedades (*DB2Prop*), y a relacionar los individuos identificados entre sí (*DB2Rel*). Sin embargo, estas reglas de correspondencia no consi-

```
<map>
 <type>DB2Rel</type>
 <source>
  <class><id>http://miuras.inf.um.es/ontologies/OGO.owl#Gene</id></class>
  <db><id>gene2go.geneId</id></db>
 </source>
 <predicate>
  <property><id>http://miuras.inf.um.es/ontologies/OGO.owl#participates_in</id></property>
  <db><id>gene2go.ec</id></db>
 </predicate>
 <target>
  <class><id>http://um.es/go.owl#GO_0003674</id></class>
  <db><id>gene2go.goId</id></db>
 </target>
</map>
```

Tabla 8.3: Ejemplo de regla de correspondencia tipo *DB2Rel* para la relación de la ontología *participates_in*.

deran ningún criterio para evitar la redundancia en la información y, por lo tanto, es necesario establecer otros mecanismos que reduzcan la duplicidad de individuos, las reglas de identidad.

8.2.2. Reglas de identidad

El objetivo de las reglas de identidad es encontrar los individuos que representen a la misma entidad pero que posean diferentes URI en el repositorio semántico. Estas reglas utilizan las propiedades, anotaciones y relaciones que tienen definidas las clases de la ontología para definir cuándo una instancia es equivalente a otra. La semántica de las reglas de identidad está relacionada con los conceptos y propiedades de identificación descritos en la metodología OntoClean. Estas reglas se deben aplicar en el momento de crear un nuevo individuo en la ontología. De esta manera, se comprueba, antes de añadir un individuo, si ya existe uno equivalente en la ontología. Así se evita la duplicidad de instancias en el repositorio.

Para encontrar los individuos equivalentes en el repositorio semántico, las reglas de identidad establecen condiciones sobre las propiedades y relaciones definidas para una clase, y toman los valores para evaluar dichas condiciones de las propiedades asignadas a la instancia a comparar. Las condiciones son evaluadas según una expresión booleana definida, dando como resultado la relación de equivalencia para indicar si una instancia es o no equivalente a

otra. Por lo tanto, la evaluación de una regla de identidad con los valores de una instancia nos proporcionará las instancias equivalentes almacenadas en el repositorio semántico.

En el Anexo 13.3.2, se describe en detalle la estructura de las reglas de identidad. En concreto, en la tabla 2 se describen formalmente los elementos de la gramática definida para este tipo de reglas.

Por su parte, en la Tabla 8.4 se muestra un ejemplo de regla de identidad definida para la clase *Gene* de la ontología. Las URI de los conceptos de la ontología han sido simplificadas para facilitar la lectura del ejemplo, y sólo se muestran sus nombres locales. La regla de identidad está definida dentro de las etiquetas <condition> y </condition>. Los requisitos están descritos entre la etiqueta <requirement> y </requirement>. El primer requisito, o requisito raíz, sólo define la etiqueta "<object>", la cual indica la clase de la ontología descrita, que, en este caso, es la clase *Gene*. A continuación, en el ejemplo se definen dos etiquetas "<and>". La primera etiqueta indica que la evaluación de los sub-requisitos es obligatoria para la definición de la regla, y la segunda etiqueta se refiere a los dos sub-requisitos del mismo nivel. Así, el primer sub-requisito indica que para que dos genes sean equivalentes deben pertenecer a una misma especie, mientras que el segundo sub-requisito está vacío porque no define las etiquetas <predicate> ni <object>. Este tipo de requisitos es utilizado para agrupar otros requisitos y así redefinir el orden de evaluación de los requisitos. Por último, este requisito contiene otros dos sub-requisitos que son opcionales debido al uso de la etiqueta <or>. Estos requisitos indican que deberían compartir un nombre o un identificador del gen, respectivamente, para que dos genes sean equivalentes.

8.2.3. Integración semántica de información

El método de integración utiliza las reglas de correspondencia para obtener la información de los repositorios bioinformáticos y para crear las instancias correspondientes en el repositorio semántico. Cada repositorio a integrar debe tener asociado su conjunto de reglas de correspondencia. Por otro lado, las reglas de identidad tienen como objetivo identificar instancias equivalen-

```
<condition>
 <requirement><object><class>Gene</class></object>
 <and><and>
  <requirement>
   <predicate><scope>ALL</scope><property>fromSpecies</property></predicate>
   <object><value>EQUALS</value><class>Organism</class></object>
  </requirement>
  <requirement>
   <or>
    <requirement>
     <predicate><scope>SOME</scope><property>Gene_name</property></predicate>
     <object><value>EQUALS IGNORE CASE</value><class>Gene</class></object>
    </requirement>
    <requirement>
     <predicate><scope>SOME</scope><property>Gene_identifier</property></predicate>
     <object><value>EQUALS</value><class>Gene</class></object>
    </requirement>
   </or>
  </requirement>
 </and></and>
 </requirement>
</condition>
```

Tabla 8.4: Ejemplo de regla de identidad para la clase *Gene*.

tes en el repositorio semántico. Cada regla de identidad está asociada a un concepto de la ontología global y, por lo tanto, el documento XML de reglas de identidad es único para el repositorio semántico.

Para llevar a cabo el método de integración, se ha definido el algoritmo que se muestra en la Tabla 8.5. El método de integración recibe como parámetros la lista de repositorios a integrar (*listaBBDD*), la lista de reglas de identidad del repositorio semántico (*listaIR*) y la conexión con el repositorio semántico (*KB*).

El método definido integra, de manera secuencial, los repositorios, pero se podría realizar la integración de manera paralela si se pueden garantizar que no pueden ocurrir lecturas sucias en el repositorio semántico. Para cada repositorio a integrar, se obtienen sus reglas de correspondencia que serán clasificadas según su tipo (*DB2Class*, *DB2Prop* y *DB2Rel*).

Para recuperar las distintas instancias del repositorio, se deben empezar a utilizar las reglas *DB2Class*. Estas reglas identifican las claves primarias en el repositorio asociadas con las instancias de una clase de la ontología. Debido a que una instancia puede estar relacionada con otras instancias en la ontología, se debe ordenar el proceso de integración para evitar insertar

una instancia que tenga dependencias con otras instancias que aún no han sido creadas en el repositorio semántico. Por ejemplo, los genes pueden tener asociados instancias de proteínas, al modo que si intenta crear antes las instancias de gen, no es posible asociarle dicha información. Por lo tanto, la integración de información debe estar ordenada para evitar el problema descrito anteriormente. Como es posible que existan ciclos en el grafo, las instancias referenciadas se crearán en el momento que son asignadas a otra instancia hasta que todas las instancias del ciclo hayan sido creadas. Sin embargo, este método de inserción es menos eficiente, ya que debe guardar más información en memoria durante el proceso.

Una vez ordenadas las reglas *DB2Class*, para evitar dependencias, se procesarán en orden cada regla. Para cada regla seleccionada, se recuperará su clase de la ontología, las reglas de correspondencia del tipo *DB2Prop* y *DB2Rel* relacionadas, y la regla de identidad asociada a su clase o superclases.

A continuación, se obtendrá la lista de claves primarias de las instancias del repositorio a integrar. Así, para cada clave se creará una instancia candidata a ser integrada en el repositorio semántico. Se asignará a las instancias candidatas los valores de las propiedades y relaciones correspondientes, definidas en sus reglas *DB2Prop* y *DB2Rel*.

Una vez recopilada la información de una instancia candidata, se busca en el repositorio semántico las instancias equivalentes a la candidata mediante la utilización de su regla de identidad. Si se encontrara una instancia equivalente, se le asignarán a la instancia equivalente los valores de las propiedades y relaciones de la instancia candidata. En otro caso, se le asignará una nueva URI para la instancia candidata y se insertará en el repositorio.

```
01 //Lista de repositorios a integrar.
02 RepositoriesList listaBBDD = getRepositoriesList();
03 //Lista de reglas de identidad.
04 IdentityRulesList listaIR = getIdentityRulesList();
05
06 //Se integra cada repositorio en el KB.
07 while( listaBBDD.hasNext() ){
08   Repository repositorio = listaBBDD.next();
09   //Se recuperan las reglas de correspondencia del repositorio.
10   MRrepository repositorioMR = repositorio.getMRrepository();
11   //Se recuperan las reglas tipo DB2Class.
12   DB2ClassList listaReglasClase = repositorioMR.getDB2ClassRules();
13
14   //Se procesan las reglas DB2Class del repositorio.
15   while( listaReglasClase.hasNext() ){
16     DB2ClassRule reglaClase = listaReglasClase.next();
17     //Se recupera la clase de la ontología asociada a la regla DB2Class.
18     OntologyClass ontClase = reglaClase.getClass();
19     //Se recuperan las reglas de correspondencia asociadas a la clase.
20     DB2PropList listaReglasProp = repositorioMR.getDB2PropRules(ontClase);
21     DB2RelList listaReglasRel = repositorioMR.getDB2RelRules(ontClase);
22     //Se recupera la regla de identidad asociada a su clase.
23     IdentityRule reglaId = listaIR.getIdentityRule(ontClase);
24     //Se obtienen las claves con la consulta asociada a la regla DB2Class.
25     PKList listaPK = reglaClase.getPrimaryKeys();
26
27     //Se procesan las claves asociadas a una regla DB2Class.
28     while( listaPK.hasNext() ){
29       //Se recupera la información asociada a cada clave del repositorio.
30       PK claveId = listaPK.next();
31       //Se crea un candidato a instancia del KB.
32       Candidate candidato = createCandidato(claveId);
33       //Se Recupera la información del repositorio fuente asociada a cada candidato.
34       candidato.gatherPropertiesValues(listaReglasProp);
35       candidato.gatherRelationships(listaReglasRel);
36
37       //Se busca si existe una instancia equivalente en el KB.
38       Instance instancia = KB.findEquivalent(reglaId,candidato);
39       if( instancia.isEmpty() ){
40         //Si no hay se inserta el candidato como instancia.
41         KB.insert(candidato);
42       }else{
43         //Si existe, se combina su información con la del candidato.
44         KB.merge(instancia,candidato);
45       }
46     }
47   }
48 }
49 //Fin método
```

Tabla 8.5: Algoritmo del método de integración semántica.

El tiempo de ejecución del algoritmo descrito depende del número de repositorios a integrar y de la cantidad de instancias que se crearán. Para tener una estimación general del tiempo que requiere, a conitnuación se analizará la complejidad del algoritmo. En el mejor caso, el algoritmo puede paralelizarse para integrar todos los repositorios al mismo tiempo. Así, la complejidad estará influenciada por los bucles "while" de las líneas 15 y 28, y por la función de búsqueda definida en la línea 38. El primer bucle "while" depende del número de clases a poblar del repositorio, y el segundo bucle depende del número de instancias asociadas a cada clase. Estos dos bucles dependen del tamaño del repositorio a integrar, es decir, si aumentan el número de clases a poblar entonces el número de instancias por clase será menor, pero si el número de clases disminuye entonces el número de instancias por clase aumenta. Por lo tanto, se considerará que la complejidad de estos dos bucles es de O(n). Además, la función de búsqueda de la línea 38 tiene una complejidad de O(m), donde "m" es el tamaño del repositorio semántico. El tiempo de búsqueda dependerá también de la complejidad de las reglas de identidad que se utilicen. También influye que al avanzar el proceso de integración la cantidad de instancias irá aumentando, por lo que el tiempo de búsqueda por instancia aumentará. Así, se puede considerar que la complejidad total del algoritmo es de $O(n^2)$. En el peor caso, la integración de los repositorios será secuencial y por lo tanto la complejidad total será $O(n^3)$. En resumen, la complejidad de este algoritmo y el tamaño de los repositorios bioinformáticos hacen que el tiempo de ejecución pueda llegar a aumentar considerablemente y deberá ser tenido en cuenta a la hora de gestionar la integración de los repositorios.

Para recuperar la información de los repositorios fuente, las reglas de correspondencia proporcionan la información necesaria para definir las consultas SQL adecuadas y hacerlas corresponder con los elementos de la ontología global. Para cada tipo de regla, se obtendrán diferentes tipos de consultas. Así, en la Tabla 8.6 se muestra la consulta SQL obtenida de la regla de correspondencia definida en la Tabla 8.1. En este caso, cada valor de la columna *geneId*, representa una instancia de la clase *Gene*.

La regla de correspondencia tipo *DB2Prop* representada en la Tabla 8.2

se utiliza para generar la consulta SQL mostrada en la Tabla 8.7. Esta tabla muestra cómo se recupera el valor de la propiedad *Gene_name* para la instancia de la clase *Gene* utilizando "<clave>" como identificador de la instancia. El valor de la clave se obtiene a través de los valores resultado de la consulta de la Tabla 8.6.

Por último, las reglas tipo *DB2Rel* no recuperan valores de propiedad, sino los identificadores de instancias de las clases con las que tienen relación. La Tabla 8.8 muestra la consulta SQL asociada a la regla definida en la Tabla 8.3. Esta consulta SQL recupera los valores de la columna *goId* que identifican a las instancias de la clase *GO_0003674* de la Gene Ontology, relacionadas con la instancia de la clase *Gene*, identificada utilizando el valor de la clave obtenido en la consulta 8.6.

```
SELECT DISTINCT geneId
FROM geneinf ;
```

Tabla 8.6: Ejemplo de consulta SQL asociada a la regla de correspondencia de la Tabla 8.1.

```
SELECT geneName
FROM geneinf
WHERE geneId=<clave> ;
```

Tabla 8.7: Ejemplo de consulta SQL asociada a la regla de correspondencia tipo *DB2Prop* de la Tabla 8.2.

```
SELECT goId
FROM gene2go
WHERE geneId=<clave>
```

Tabla 8.8: Ejemplo de consulta SQL asociada a la regla de correspondencia tipo *DB2Rel* de la Tabla 8.3.

Ya que la información biomédica está integrada como individuos en el repositorio semántico, se utilizará un lenguaje semántico de consulta, como SPARQL, para acceder a dicha información. De esta manera, para evaluar las reglas de identidad en el repositorio semántico se traducirán éstas a consultas

```
PREFIX ogo: <http://miuras.inf.um.es/ontologies/OGO.owl#>
PREFIX tax: <http://um.es/ncbi.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?uri
WHERE {
 ?uri rdf:type ?type .
 ?type rdfs:subClassOf ogo:Gene OPTION(TRANSITIVE) .
 ?uri ogo:fromSpecies tax:organism1 .
 {
  {
  ?uri ogo:Gene_name ?label1 .
  FILTER regex(?label1, "name1", i) .
  }
   UNION
  {
  ?uri ogo:Gene_name ?label2 .
  FILTER regex(?label2, "name2", i) .
  }
   UNION
  { ?uri ogo:Gene_identifier "id" .}
 }
```

Tabla 8.9: Consulta SPARQL relacionada con la regla de identidad de la Tabla 8.4.

SPARQL. En la Tabla 8.9 muestra un ejemplo de consulta SPARQL definida utilizando la regla de identidad mostrada en la Tabla 8.4. Esta consulta busca en el repositorio semántico las URI de las instancias de la clase *Gene* que pertenezcan a la especie *"organism1"* y que contenga el nombre del gen *"name1"* o *"name2"*, o el identificador de gen *"id"*.

8.3. Repositorio semántico

Esta sección describe las características del repositorio semántico que se genera con la metodología de integración semántica propuesta en este capítulo.

El repositorio almacenará, de manera permanente, la información producida durante el proceso de integración. La Figura 8.3 muestra los elementos necesarios en la arquitectura del repositorio semántico. Los elementos son:

- API OWL/RDF: interfaz de programación para el manejo de ontologías OWL y lenguaje RDF. Esta interfaz se utiliza para la instanciación de la información en el repositorio mediante individuos de la ontología.

Además, debe permitir utilizar funciones de razonamiento o inferencia para verificar la consistencia del modelo ontológico que a utilizar.

- SPARQL: interfaz de consulta semántico basado en SPARQL. Esta interfaz se utiliza para la consulta del repositorio y para la utilización de las reglas de identidad definidas en la metodología.

- Almacén Persistente: el repositorio semántico utiliza un almacén persistente para el almacenamiento de la información integrada. Este almacén contiene tanto el modelo ontológico utilizado como los individuos instanciados. El almacén puede estar implementado utilizando diferentes tecnologías. Por ejemplo, pueden utilizarse ficheros de texto, repositorios de bases de datos o repositorios RDF. El tipo de almacén utilizado viene limitado por el API OWL/RDF elegido para implementar el repositorio semántico.

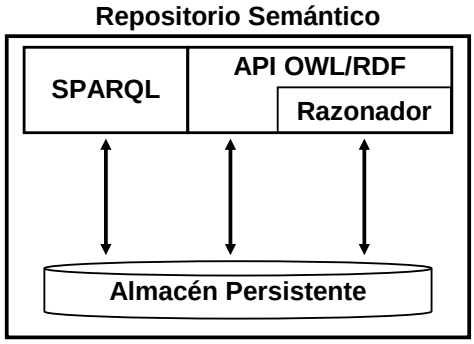

Figura 8.3: Arquitectura genérica de un repositorio semántico.

8.4. Ejemplo de integración semántica

Para desarrollar el ejemplo de integración semántica siguiendo el algoritmo propuesto, se ha definido el escenario que se describirá a continuación. La conceptualización correspondiente a este ejemplo está recogida en la Figura 8.4. El modelo ontológico que se muestra está compuesto por una clase

"Gen" y una clase *"Especie"*. Cada una de las cuales tienen dos propiedades asociadas, un identificador (*"Id"*) y un nombre (*"Nombre"*). Por último, las clases están enlazadas entre sí mediante la relación *"Pertenece_a"*, que indica a qué especie pertenece cada gen. Una vez se han seleccionado las fuentes de información y la ontología, se han de definir las reglas de correspondencia y de identidad correspondientes.

Desde la perspectiva de instancias de datos, la Tabla 8.10 contiene las instancias de genes, la Tabla 8.11 contiene los nombres o símbolos asociados a cada gen y la Tabla 8.12 contiene los nombre e identificadores de las especies que aparecen en el repositorio.

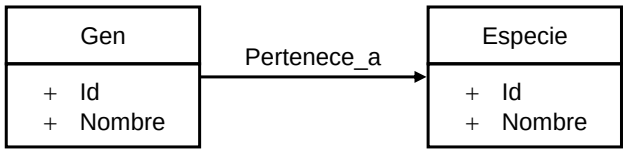

Figura 8.4: Ejemplo de ontología que conceptualiza el conocimiento asociado a las tablas 8.10, 8.11 y 8.12.

Gen_Id	Especie_Id
33013	7227
9448	9606
6866	9606

Tabla 8.10: Tabla *"Gen"* que contiene las instancias de genes.

Gen_Id	Gen_Nombre
33013	CG9572
9448	MAP4K4
9448	MEKKK4
6866	MAP4K4

Tabla 8.11: Tabla *"Gen_Detalles"* que contiene los símbolos de genes.

Para describir el funcionamiento del proceso de integración semántica, se definirán las reglas de correspondencia y de identidad correspondientes a

Especie_Id	Especie_Nombre
7227	Drosophila melanogaster
9606	Homo sapiens

Tabla 8.12: Tabla *"Especies"* que contiene las instancias de especies.

las tablas del ejemplo. En los siguientes apartados, se definirán dichas reglas y cómo se utilizan para llevar a cabo la integración de información en el repositorio semántico.

8.4.1. Ejemplo de reglas de correspondencia

Las reglas de correspondencia son presentadas según su relación con las clases "Gen" y "Especie". Se proporcionará una representación gráfica de su significado, así como una representación formal siguiendo el lenguaje definido en las secciones anteriores.

Reglas de correspondencia asociadas a la clase "Especie"

Las reglas de correspondencia asociadas a la clase "Especie" que deberán definirse son tres, una regla tipo "DB2Class" para indicar la columna que identifica las instancias de la clase y dos reglas tipo "DB2Prop" para las propiedades "Id" y "Nombre". Así, la Figura 8.5 muestra gráficamente las relaciones entre los elementos de la clase "Especie" en el modelo ontológico y sus correspondientes valores en el repositorio de información. A continuación, se describirán dichas reglas:

- Primera regla de correspondencia: es una regla tipo "DBClass" que identifica la columna "Especie_Id" de la tabla "Especies" como la que contiene los identificadores de las instancias de la clase "Especie". Esta regla está definida formalmente en la Tabla 8.13.

- Segunda regla de correspondencia: es una regla tipo "DBProp" que relaciona los valores de la columna "Especie_Id" de la tabla "Especies"

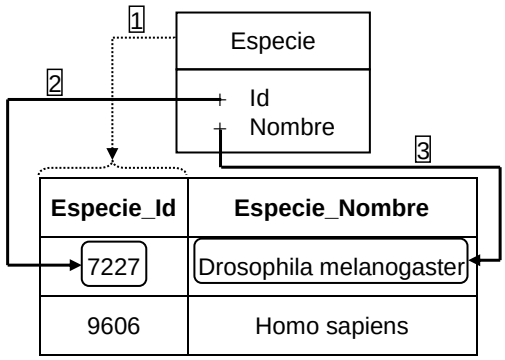

Figura 8.5: Gráfico de las reglas de correspondencia asociadas a la clase "Especie".

```
<map>
  <type>DB2Class</type>
  <class><id>Especie</id></class>
  <db><id>Especies.Especie_Id</id></db>
</map>
```

Tabla 8.13: Definición de la primera regla de correspondencia asociada a la clase "Especie".

con los valores de la propiedad "Id" de la clase "Especie". En la Figura 8.5, se asocia el valor "7227" a la propiedad "Id" de la instancia de especie con identificador del mismo valor. Esta regla está definida formalmente en la Tabla 8.14.

```
<map>
  <type>DB2Prop</type>
  <source><class><id>Especie</id></class><db><id>Especies.Especie_Id</id></db></source>
  <predicate><property><id>Id</id></property></predicate>
  <target><db><id>Especies.Especie_Id</id></db></target>
</map>
```

Tabla 8.14: Definición de la segunda regla de correspondencia asociada a la clase "Especie".

- Tercera regla de correspondencia: es una regla tipo "DBProp" que relaciona los valores de la columna "Especie_Nombre" de la tabla "Especies" con los valores de la propiedad "Nombre" de la clase "Especie". En la Figura 8.5 se asocia el valor "Drosophila melanogaster" para la propiedad "Nombre" de la instancia de especie con identificador "7227".

Esta regla está definida formalmente en la Tabla 8.15.

```
<map>
  <type>DB2Prop</type>
  <source><class><id>Especie</id></class><db><id>Especies.Especie_Id</id></db></source>
  <predicate><property><id>Nombre</id></property></predicate>
  <target><db><id>Especies.Especie_Nombre</id></db></target>
</map>
```

Tabla 8.15: Definición de la tercera regla de correspondencia asociada a la clase "Especie".

Reglas de correspondencia asociadas a la clase "Gen"

Las reglas de correspondencia definidas para la clase "Gen" son cuatro, una regla tipo "DB2Class", dos reglas tipo "DBProp" y una regla tipo "DBRel". Las Figuras 8.6 y 8.7 muestran gráficamente las relaciones entre los elementos del modelo ontológico, asociados a la clase "Gen", y sus correspondientes valores en el repositorio de información. A continuación, se describirán dichas reglas:

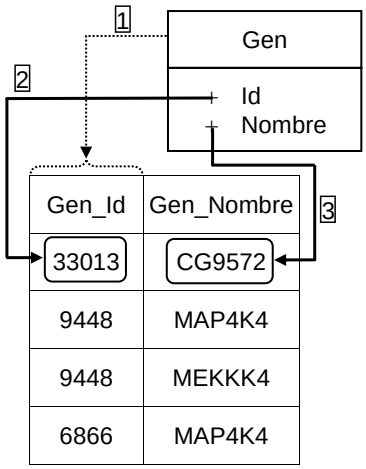

Figura 8.6: Gráfico de las reglas de correspondencia asociadas a la clase "Gen".

- Primera regla de correspondencia: regla "DBClass", que identifica la columna "Gen_Id" de la tabla "Gen_Detalles" como la que contiene los

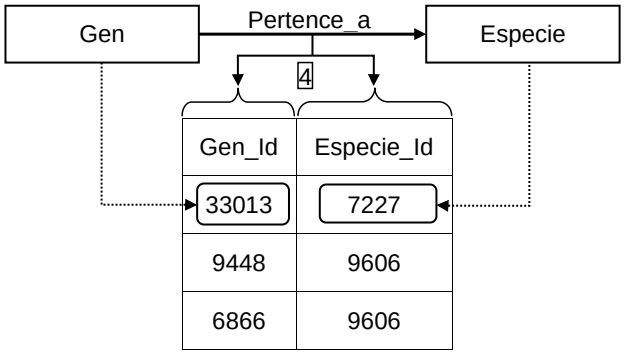

Figura 8.7: Gráfico de la regla de correspondencia que asocia instancias de la clase "Gen" con las instancias de la clase "Especie".

identificadores de las instancias de la clase "Gen". Esta regla está definida formalmente en la Tabla 8.16.

```
<map>
  <type>DB2Class</type>
  <class><id>Gen</id></class>
  <db><id>Gen_Detalles.Gen_Id</id></db>
</map>
```

Tabla 8.16: Primera regla de correspondencia asociada a la clase "Gen".

- Segunda regla de correspondencia: regla "DBProp", que relaciona los valores de la columna "Gen_Id" de la tabla "Gen_Detalles" con los valores de la propiedad "Id" de la clase "Gen". En la Figura 8.6 se asocia el valor "33013" para la propiedad "Id" de la instancia de gen con identificador del mismo valor. Esta regla está definida formalmente en la Tabla 8.17.

```
<map>
  <type>DB2Prop</type>
  <source><class><id>Gen</id></class><db><id>Gen_Detalles.Gen_Id</id></db></source>
  <predicate><property><id>Id</id></property></predicate>
  <target><db><id>Gen_Detalles.Gen_Id</id></db></target>
</map>
```

Tabla 8.17: Segunda regla de correspondencia asociada a la clase "Gen".

- Tercera regla de correspondencia: regla "DBProp", que relaciona los valores de la columna "Gen_Nombre" de la tabla "Gen_Detalles" con

los valores de la propiedad "Nombre" de la clase "Gen". En la Figura 8.6 asocia el valor "CG9572" para la propiedad "Nombre" de la instancia de gen con identificador "33013". Esta regla está definida formalmente en la Tabla 8.18.

```
<map>
 <type>DB2Prop</type>
 <source><class><id>Gen</id></class><db><id>Gen_Detalles.Gen_Id</id></db></source>
 <predicate><property><id>Nombre</id></property></predicate>
 <target><db><id>Gen_Detalles.Gen_Nombre</id></db></target>
</map>
```

Tabla 8.18: Tercera regla de correspondencia asociada a la clase "Gen".

- Cuarta regla de correspondencia: regla "DBRel", que enlaza instancias de la clase "Gen", identificadas por los valores de la columna "Gen_Id", con instancias de la clase "Especie", identificadas por los valores de la columna "Especie_Id" de la tabla "Gen". Esta regla está definida formalmente en la Tabla 8.19.

```
<map>
 <type>DB2Rel</type>
 <source><class><id>Gen</id></class><db><id>Gen.Gen_Id</id></db></source>
 <predicate><property><id>Pertenece_a</id></property></predicate>
 <target><class><id>Especie</id></class><db><id>Gen.Especie_Id</id></db></target>
</map>
```

Tabla 8.19: Cuarta regla de correspondencia asociada a la clase "Gen".

8.4.2. Ejemplo de reglas de identidad

Las reglas de identidad definidas para la ontología del ejemplo están representadas en las Tablas 8.20 y 8.21. Cada regla describe como identificar cada una de las clases de la ontología, es decir, se dispondrá de una regla de identidad para la clase "Gen" (Tabla 8.20) y otra para la clase "Especie" (Tabla 8.21). La regla de identidad asociada a la clase "Gen" tiene por objetivo encontrar las instancias de dicha clase que pertenezcan a la misma especie y que compartan, al menos, un valor de las propiedades "Id" o "Nombre" de gen. La regla de identidad asociada a la clase "Especie" identifica las

instancias de las especies que comparten el mismo valor en las propiedades "Id" o "Nombre".

```
<condition>
 <requirement><object><class>Gen</class></object>
 <and><and>
  <requirement>
   <predicate><scope>ALL</scope><property>Pertenece_a</property></predicate>
   <object><value>EQUALS</value><class>Especie</class></object>
  </requirement>
  <requirement>
   <or>
    <requirement>
     <predicate><scope>SOME</scope><property>Nombre</property></predicate>
     <object><value>EQUALS IGNORE CASE</value><class>Gen</class></object>
    </requirement>
    <requirement>
     <predicate><scope>SOME</scope><property>Id</property></predicate>
     <object><value>EQUALS</value><class>Gen</class></object>
    </requirement>
   </or>
  </requirement>
 </and></and>
 </requirement>
</condition>
```

Tabla 8.20: Definición de la regla de identidad de la clase *"Gen"* para la integración semántica.

```
<condition>
 <requirement><object><class>Especie</class></object>
 <and><or>
  <requirement>
   <predicate><scope>SOME</scope><property>Nombre</property></predicate>
   <object><value>EQUALS IGNORE CASE</value><class>Especie</class></object>
  </requirement>
  <requirement>
   <predicate><scope>SOME</scope><property>Id</property></predicate>
   <object><value>EQUALS</value><class>Especie</class></object>
  </requirement>
 </or></and>
 </requirement>
</condition>
```

Tabla 8.21: Definición de la regla de identidad de la clase *"Especie"* para la integración semántica.

8.4.3. Proceso de integración del ejemplo

Una vez definida la ontología que cubre el conocimiento de los repositorios a integrar, las reglas de correspondencia y las reglas de identidad, se puede

ejecutar el proceso de integración. Para ello, se deben ordenar las reglas de correspondecia para evitar las dependencias que existen entre instancias relacionadas. Esta ordenación evita que si una instancia se tiene que relacionar con otra instancia, ésta ya haya sido creada en el repositorio. Así, en este ejemplo se ordenan las reglas de correspondencia para empezar por las instancias de la clase "Especie" y a continuación por la clase "Gen" ya que los genes tienen asociadas las especies a las que pertenecen.

Para obtener las distintas instancias de la clase "Especie" se utilizará la consulta SQL mostrada en la Tabla 8.22. Dicha consulta está definida a partir de la regla de correspondencia de la Tabla 8.13. Los resultados de esta consulta serían las claves *"7227"* y *"9606"*. Para obtener las propiedades asociadas a cada instancia de especies se ha hecho uso de las consultas SQL mostradas en las Tablas 8.23 y 8.24, y que están definidas a partir de las reglas de correspondencia representadas en las Tablas 8.14 y 8.15. En estas consultas SQL, se sustituye la etiqueta "<key>" por los valores clave de la clase obtenidos por la consulta de la Tabla 8.22. En el caso de la instancia con clave "9606", se obtienen los valores *"9606"* para la propiedad "Id" y *"Homo sapiens"* para la propiedad "Nombre".

```
SELECT DISTINCT Especie_Id
FROM Especies ;
```

Tabla 8.22: Consulta SQL para obtener las claves de las instancias de la clase "Especie".

```
SELECT DISTINCT Especie_Id
FROM Especies
WHERE Especie_Id = <key> ;
```

Tabla 8.23: Consulta SQL para obtener los valores de la propiedad "Id" de las instancias de la clase "Especie".

```
SELECT DISTINCT Especie_Nombre
FROM Especies
WHERE Especie_Id = <key> ;
```

Tabla 8.24: Consulta SQL para obtener los valores de la propiedad "Nombre" de las instancias de la clase "Especie".

Con los datos obtenidos de estas consultas SQL se puede definir la consulta SPARQL asociada a la regla de identidad de la clase "Especie" (véase la Tabla 8.21). La Tabla 8.25 muestra la consulta asociada a la instancia de especie con las propiedades "Id = 9606" y "Nombre = Homo sapiens". En estas condiciones, la consulta SPARQL no devolverá ningún resultado, por lo que se insertan las instancias de las especies en el repositorio semántico.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?uri
WHERE {
 ?uri <rdf:type> <Especie> .
 {
  {
  ?URI <Nombre> ?label1 .
  FILTER regex(?label1, "Homo sapiens", i) .
  }
   UNION
  {
  ?uri <Id> "9606" .
  }
 }
```

Tabla 8.25: Consulta SPARQL de la regla de identidad para la instancia de Especie del ser humano.

Una vez integradas las dos especies en el repositorio semántico, se procederá a la integración de las instancias de genes. Para obtener los valores de las claves de las instancias, se utilizará la consulta SQL de la Tabla 8.26 asociada a la regla de correspondencia de la Tabla 8.16. Esta consulta devuelve los resultados *"33013"*, *"9448"* y *"6866"*. En este ejemplo, se muestra el proceso de integración de la última instancia de gen con clave *"6866"*, suponiendo que el resto de instancias ya han sido integradas. Para obtener los valores de las propiedades "Id" y "Nombre" del gen con clave *"6866"*, se han utilizado las consultas SQL de las Tablas 8.27 y 8.28 asociadas a las reglas 8.17 y 8.18 respectivamente. Los valores obtenidos en estas consultas son *"6866"*, para la propiedad "Id", y *"MAP4K4"*, para la propiedad "Nombre". La regla de correspondencia de tipo "DB2Rel" de la clase "Gen" (véase la Tabla 8.19) genera la consulta SQL mostrada en la Tabla 8.29 para la instancia con clave *"6866"*. El resultado es el valor *"9606"* de la clave de la instancia de especie

"Homo sapiens". Así, en caso de disponer de una asociación entre las claves de las instancias integradas y las URI de dichas instancias, se obtendrá la instancia correspondiente de la relación. En otro caso, se debería volver a obtener los valores de las propiedades de la instancia y consultar el repositorio semántico con la consulta SPARQL generada a partir de la regla de identidad de su clase. Una vez obtenidos los valores de las propiedades y relaciones de la instancia de gen *"6866"* se utilizará la consulta SPARQL de la Tabla 8.30, generada a partir de la regla de identidad de la clase "Gen" (véase la Tabla 8.20). En este caso se obtendrá la instancia de gen con clave "9448" que ya ha sido insertada en un paso anterior. Entonces, los valores de la instancia encontrada serán combinados con los valores de las propiedades de la instancia de gen con clave *"6866"*. De esta manera, se reduce la posible redundancia del repositorio semántico.

```
SELECT DISTINCT Gen_Id
FROM Gen ;
```

Tabla 8.26: Consulta SQL para obtener las claves de las instancias de la clase "Gen".

```
SELECT DISTINCT Gen_Id
FROM Gen
WHERE Gen_Id = "6866" ;
```

Tabla 8.27: Consulta SQL para obtener los valores de la propiedad "Id" de la instancia de gen con clave "6866".

```
SELECT DISTINCT Gen_Nombre
FROM Gen_Detalles
WHERE Gen_Id = "6866" ;
```

Tabla 8.28: Consulta SQL para obtener los valores de la propiedad "Nombre" de la instancia de gen con clave "6866".

```
SELECT DISTINCT Especie_Id
FROM Gen
WHERE Gen_Id = “6866” ;
```

Tabla 8.29: Consulta SQL para obtener los valores de la propiedad “Pertenece_a” que asocia la instancia de gen con clave “6866” con una especie.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?uri
WHERE {
 ?uri rdf:type <Gen> .
 ?uri <Pertenece_a> <9606> .
 {
  {
  ?uri <Nombre> ?label1 .
  FILTER regex(?label1, “MAP4K4”, i) .
  }
   UNION
  {
  ?uri <Id> “6866” .
  }
 }
```

Tabla 8.30: Consulta SPARQL de la regla de identidad para la instancia de gen *“6866”*.

Capítulo 9

Modelos de publicación y explotación de repositorios semánticos

En este capítulo se describen los modelos de publicación y explotación definidos para ser utilizados con el repositorio semántico resultado de la metodología de integración definida en el capítulo 8. En concreto, se definen dos sistemas: (1) una interfaz que permite explotar la información almacenada en el repositorio semántico a través de la definición de consultas avanzadas; y (2) un sistema para la publicación del repositorio semántico como conjuntos de datos enlazados en la nube LOD.

9.1. Interfaz de consultas avanzadas

Esta interfaz de consulta ha sido desarrollada para aprovechar las características semánticas del lenguaje SPARQL en la definición de consultas avanzadas sobre el repositorio semántico [214, 215].

Las interfaces desarrolladas actualmente y que permiten la definición de consultas SPARQL se limitan a ofrecer un conjunto de plantillas como ejemplos de consultas o, en el peor de los casos, un campo de texto donde los usuarios deberán introducir a mano las consultas SPARQL. Por ejemplo, la

interfaz de Linked Life Data ofrece ambas posibilidades al usuario. Sin embargo, estos tipos de interfaces requieren de un proceso de aprendizaje previo para poder definir las consultas personalizadas. Por lo tanto, para poder realizar una consulta, los usuarios deben conocer el modelo semántico utilizado para almacenar la información y además, tener los conocimientos suficientes del lenguaje SPARQL.

La interfaz desarrollada intenta facilitar a los usuarios la definición de consultas avanzadas sobre repositorios semánticos que utilicen SPARQL. Para ello, se define una vista con un formulario donde el usuario podrá definir los diferentes elementos de su consulta mediante un proceso interactivo. Este proceso interactivo, el cual ofrece a los usuarios las distintas opciones posibles en cada paso de la definición de la consulta avanzada, evitando de este modo que el usuario tenga que conocer la gramática del lenguaje SPARQL. Es decir, el sistema reduce al máximo posible la cantidad de información que los usuarios tengan que definir manualmente.

Además, la interfaz muestra las opciones según la definición del modelo ontológico utilizado en el repositorio semántico. La interfaz utiliza el modelo semántico no poblado, empleado por el repositorio semántico, para poder guiar el proceso de definición de consultas. De este modo, los usuarios de la interfaz ven reducida la exigencia de conocimiento del modelo ontológico del repositorio. El sistema restringe las opciones de definición de los elementos de la consulta, como las variables de búsqueda y condiciones, para que se ajusten al modelo del repositorio semántico.

Por lo tanto, los usuarios obtienen un doble beneficio Por un lado, la interfaz presenta el conocimiento del dominio de la consulta a los usuarios mediante las sugerencias de uso de conceptos, relaciones o propiedades de la ontología .Por otro lado, el usuario se abstrae de la gramática SPARQL para centrarse en la definición de la consulta.

La arquitectura del sistema se compone de diversos módulos encargados de: (1) guiar el proceso de definición de la consulta, (2) representar la consulta que se define, y (3) traducir esta representación siguiendo la gramática del lenguaje SPARQL. La Figura 9.1 muestra los diferentes elementos de la arquitectura y cómo interactúan entre sí. Así, el elemento central del sistema

es el *Módulo de Búsqueda Guiada*. Este módulo es el encargado de controlar la definición de la consulta mediante la habilitación o desactivación de elementos de la interfaz, y mediante la sugerencia de utilización de determinados conceptos, propiedades o relaciones de la ontología en cada paso de la definición. El módulo guía la definición de la consulta y asegura que las condiciones o restricciones definidas son posibles. Por otro lado, el módulo central utiliza la *Interfaz de Comunicación*, basada en AJAX, para enviar y recoger información del usuario de manera interactiva durante el proceso de definición.

La interfaz también utiliza el *Módulo de Representación de Consultas* para ir almacenando el estado de la definición de la consulta con su información. Con esta información, el módulo central puede realizar las sugerencias adecuadas. El módulo *Codificador SPARQL* es el encargado de representar la consulta definida por el usuario utilizando la gramática SPARQL. Este módulo es el responsable de asegurar que la consulta cumple con la gramática del lenguaje SPARQL. Por último, la interfaz de programación de Jena permite interactuar con la ontología global y con el repositorio semántico. El *Módulo de Búsqueda Guiada* utiliza la ontología global, no poblada, para extraer los conceptos, propiedades, relaciones y restricciones definidas sobre el modelo de manera más eficiente. Sin embargo, el módulo *Codificador SPARQL* accede al repositorio semántico para ejecutar la consulta y posteriormente envía al usuario a través de la *Interfaz de Comunicación* los resultados obtenidos.

La Figura 9.2 muestra la vista principal de la interfaz para la definición de consultas avanzadas. En dicha figura, se dispone de dos cuadros de texto y seis botones. El cuadro de texto etiquetado con *"Search for"* contiene los nombres de las variables de consulta, cada una de las cuales está asociada a un concepto de la ontología y sirve para extraer los individuos de esa clase que cumplan los requisitos definidos en la consulta. Para poder definir estas variables de consulta, se proporcionan los botones *"Select Concept"* y *"Delete Concept"*. Por un lado, al pulsar el botón *"Select Concept"*, se muestra en la interfaz la jerarquía de conceptos del modelo ontológico para que puedan seleccionar el que necesiten. Por otro lado, al pulsar el botón *"Delete Concept"*, se muestran las variables definidas para que el usuario

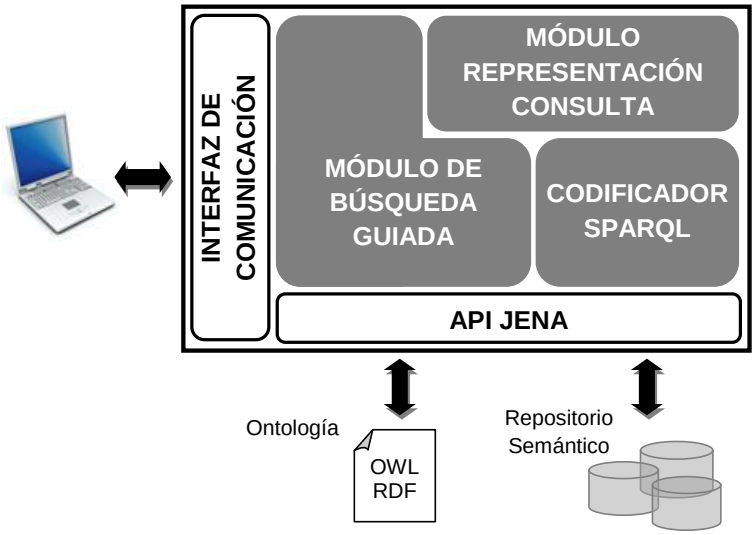

Figura 9.1: Arquitectura de la interfaz de consultas avanzadas.

seleccione las que quiera borrar de la lista. En caso de que una variable esté asignada en alguna condición de búsqueda definida en la consulta, ésta no puede ser eliminada hasta que se eliminen o modifiquen dichas condiciones.

Las condiciones de búsqueda de las consultas avanzadas se representadan en el cuadro de texto con la etiqueta *"Query Requirements"*. Las condiciones de búsqueda siguen la misma estructura que las tripletas (Sujeto, Predicado y Objeto). En concreto, el sujeto de las condiciones de consulta puede ser una variable de consulta definida, un concepto de la ontología u otra variable definida para su uso con las condiciones de consulta. El predicado de la consulta es una propiedad o relación definida en el modelo ontológico del repositorio semántico. Por último, según se haya seleccionado una propiedad o una relación en el predicado de las condiciones de consulta, el objeto puede ser un valor introducido por el usuario manualmente, o un concepto de la ontología o una variable, respectivamente. Para definir una condición de consulta, se debe utilizar el botón *"Add new requirements"*. Una vez seleccionado, se muestra al usuario los distintos tipos de condiciones de consulta posibles, agrupados según las variables definidas en la consulta avanzada. El usuario selecciona una condición y la edita si es necesario con los conceptos, variables o valores adecuados. Para borrar una condición de consulta, se debe

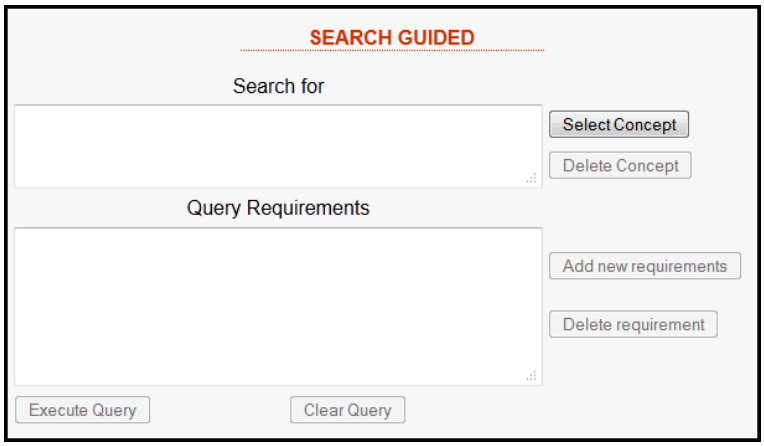

Figura 9.2: Vista para la definición de consultas avanzadas.

pulsar el botón *"Delete requirement"* y seleccionar una de las condiciones de consulta que se muestran.

Para borrar de manera completa la definición de una consulta avanzada, se ofrece el botón *"Clear Query"*, el cual permite que en cualquier momento de la definición de una consulta, ésta pueda ser borrada y empezar otra desde cero. Por último, el botón *"Execute Query"* permite ejecutar la consulta sobre el repositorio semántico. La interfaz transforma automáticamente al lenguaje SPARQL la consulta definida para poder ejecutarla sobre el repositorio semántico a través de su interfaz SPARQL.

El resultado obtenido por la interfaz se presenta al usuario en formato de tabla, donde las columnas se corresponden con las variables de consulta definidas y las filas son los individuos obtenidos del repositorio semántico que cumplen con las condiciones establecidas en la consulta avanzada.

En resumen, esta interfaz facilita la definición de las consultas sobre un repositorio semántico, ya que los usuarios no necesitan conocer en detalle las características de sus modelos ontológicos asociados. El sistema ofrece, mediante recomendaciones, las distintas posibilidades de definición de elementos de la consulta según su estado. Además, la interfaz puede utilizar las etiquetas *"label"* con las representaciones textuales de los elementos de la ontología en la definición de la consulta en vez de las URI, como en el caso de SPARQL, por lo que se favorece su legibilidad. Por otro lado, la interfaz

limita la cantidad de información que los usuarios pueden introducir manualmente en la consulta para garantizar que su definición sea correcta y pueda ser traducida a SPARQL. Esta restricción puede provocar que el tiempo requerido por los usuarios para definir las consultas sea mayor que con otras interfaces. Es decir, los usuarios deben seguir el procedimiento de definición de consultas establecido, no pudiendo introducir manualmente consultas. Sin embargo, este inconveniente puede ser considerado como una ventaja, ya que evita la definición de condiciones que no tienen sentido según la ontología utilizada por el repositorio semántico.

Por último, indicar que el usuario es el responsable de establecer las condiciones adecuadas para obtener los resultados correctos, mientras que el sistema sólo comprueba que las definiciones de las condiciones sean conformes a la ontología del repositorio, y a la sintaxis y gramática del lenguaje semántico de consultas SPARQL.

9.2. Publicación de repositorios semánticos en LOD

En esta sección, se describen las características que deben tener los sistemas para publicar información en la nube LOD. Además, se describe el proceso para publicar un repositorio semántico, generado mediante la metodología de integración semántica definida en el capítulo 8, siguiendo los principios de Linked Data.

La Figura 9.3 muestra un ejemplo de la arquitectura que debe seguir este tipo de sistemas. El primer elemento de la arquitectura es el repositorio RDF. Este repositorio almacena la información de manera que se puede consultar mediante SPARQL. El siguiente elemento de la arquitectura es el sistema gestor del repositorio RDF. Este sistema tiene como misión gestionar el repositorio RDF y ofrecer una interfaz de consulta basada en RDF. Un ejemplo de este tipo de sistemas es el servidor Virtuoso. El último elemento de la arquitectura es el servidor que se encarga de publicar el conjunto de datos enlazados. Para ello, debe disponer de, al menos, una interfaz HTML

con el que poder consultar sus datos mediante el uso de las URI de los recursos almacenados en el repositorio RDF. En esta figura, se señala al servidor Pubby [216] como ejemplo de este tipo de aplicaciones. El servidor Pubby es capaz de extraer la información del repositorio RDF utilizando consultas SPARQL y de ofrecer una interfaz HTML que pueda ser consultada desde un explorador Web con las URI de los recursos. Además, ofrece una interfaz de consulta basada en Linked Data que puede ser utilizada a través de exploradores específicos de RDF como Disco [217] o Tabulator [218]. Siguiendo la arquitectura presentada, la información del repositorio RDF puede ser consultada mediante tres tipos de interfaces, uno basado en consultas SPARQL, otro basado en exploradores específicos de RDF y por último, uno basado en HTML.

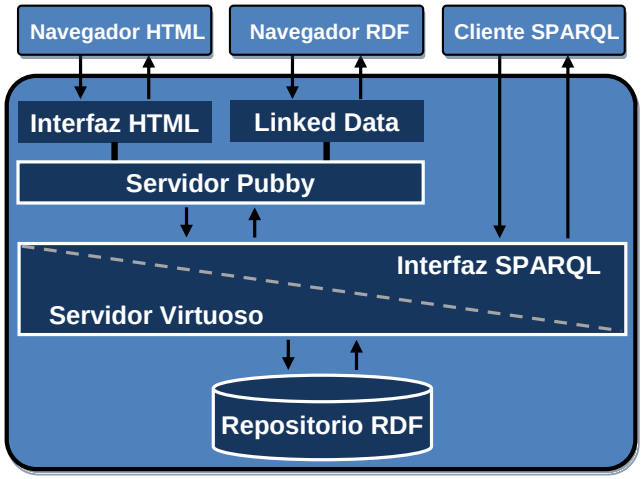

Figura 9.3: Arquitectura para la publicación de un conjunto de datos enlazados en LOD.

Por otro lado, para representar la información de un repositorio semántico en la nube LOD, éste debe cumplir con los requisitos o principios de Linked Data (véase la sección 3.2). A continuación, se exponen los tipos de cambios necesarios para transformar un repositorio semántico en un conjunto de datos enlazados según los principios de LD:

Cambio del espacio de nombres: para que las URI de los recursos del conjunto de datos a publicar siga el formato de jerarquía de etiquetas en las

direcciones URI [219] del protocolo HTTP, se deben de modificar los distintos espacios de nombre utilizados en los repositorios semánticos. Por ejemplo, un espacio de nombres *"http://servidor/subdirectorio/ontologia.owl#"*, por uno más adecuado como *"http://servidor/subdirectorio/ontologia/"*, que no utiliza caracteres especiales en su URI.

Espacio de nombre único: se debe utilizar un único espacio de nombres para que la información integrada forme parte del mismo conjunto y que no sea necesario publicar diferentes conjuntos de datos. Así, se consigue que recursos importados e integrados pertenezcan a un mismo conjunto de datos y que las URI de los recursos del repositorio sean uniformes.

Cambio en el formato de URI: esta transformación tiene como objetivo hacer más descriptivas las URI de los recursos del repositorio semántico. Para ello, se deben definir diferentes etiquetas que precedan al identificador local de los recursos a publicar. Así, para identificar un concepto, propiedad o relación, se puede utilizar la etiqueta *"ontology"*, porque estos elementos son definidos en el modelo ontológico del repositorio semántico. Sin embargo, para identificar las instancias que pueblan el repositorio semántico se puede utilizar la etiqueta *"resource"* seguida del nombre del concepto asociado y del identificador de la instancia.

Eliminación de recursos anónimos: en los repositorios semánticos se utiliza el lenguaje OWL para representar el modelo ontológico. OWL permite definir "blank nodes", que son recursos anónimos donde su URI sólo tiene sentido en el ámbito local de la ontología. Normalmente, se utilizan para representar la intersección o unión entre clases, o para la definición de restricciones sobre el dominio. Las guías para publicar los conjuntos de datos en la nube de LOD [220] desaconseja el uso de recursos anónimos porque es imposible establecer enlaces externos con ellos, y conectar datos entre diferentes fuentes de información es más complejo. Por eso, es necesario prescindir, cuando sea posible, de este tipo de información, ya que no puede ser publicada.

Una vez definidas las transformaciones necesarias del repositorio semánti-

co para adaptar su contenido a los principios de LD, se deben establecer enlaces entre el conjunto de datos obtenido y el resto de conjuntos de datos publicados en LOD. Estos enlaces dependen de la información publicada en LOD y de la que se quiere publicar. Los enlaces permitiren poder localizar la información en la nube LOD y navegar entre los diferentes conjuntos de datos.

Para enlazar la información del conjunto de datos obtenido de la transformación a partir del repositorio semántico, se añaden tripletas al repositorio RDF que establezcan las relaciones de igualdad o equivalencia entre los diferentes recursos. Así, en OWL se definen diferentes relaciones con dicho significado. La relación *owl:sameAs* puede ser utilizada para identificar dos instancias como iguales. La relación *owl:equivalentClass* puede ser utilizada para relacionar dos conceptos que sean equivalentes en dos conjuntos de datos. De esta manera, clases equivalentes tienen el mismo conjunto de instancias. Por último, la relación *owl:equivalentProperty* puede ser utilizada para establecer propiedades o relaciones equivalentes. Así, pueden ser consideradas sinónimas.

Capítulo 10

Herramientas para la integración semántica de repositorios bioinformáticos

En este capítulo, se presentan las diferentes herramientas implementadas para facilitar la integración de repositorios bioinformáticos mediante la metodología descrita en los capítulos anteriores.

10.1. Interfaz para la definición del proceso de integración

Para llevar a cabo el proceso de integración, se ha desarrollado una interfaz para facilitar la definición de las reglas de correspondencia y de identidad, así como para la ejecución del proceso de integración. Esta interfaz implementa los procedimientos descritos anteriormente en la sección de 8.2.

Esta interfaz ha sido desarrollada como una aplicación de escritorio. La vista principal permite seleccionar entre los distintos módulos del proceso de integración semántica (véase la Figura 10.1). Así, la primera opción *"DEFINE MAPPINGS"* permite la definición, edición y eliminación de reglas de correspondencia. La segunda opción *"DEFINE IDENTITIES"* facilita la definición, modificación y borrado de reglas de un fichero de reglas de identidad

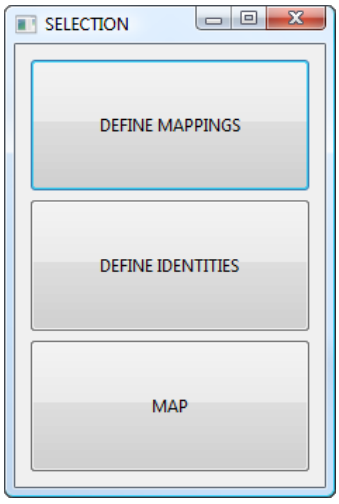

Figura 10.1: Vista de selección de operación del proceso de integración semántica.

y, por último, la tercera opción permite seleccionar los parámetros necesarios, así como las reglas de correspondencia e identidad, para la ejecución del proceso de integración.

En las siguientes subsecciones, se describe cada una de las vistas definidas en la interfaz para los distintos módulos del proceso de integración semántica.

10.1.1. Interfaz de definición de reglas de correspondencia

Esta interfaz facilita la definición de reglas de correspondencia entre un repositorio relacional y una ontología. Además, utiliza el formato de documento XML descrito en la sección 8.2.1 para almacenar de manera persistente las reglas definidas. Así, los documentos XML generados con esta herramienta pueden ser directamente utilizados por el proceso de integración semántica sin tener que ser definidos manualmente.

Para la definición de las reglas de correspondencia se ha implementado una interfaz compuesta de varias ventanas. La Figura 10.2 representa la vista principal mediante la cual se gestionan las reglas de correspondencia definidas. La barra de menú permite cerrar la ventana, cargar un repositorio y una

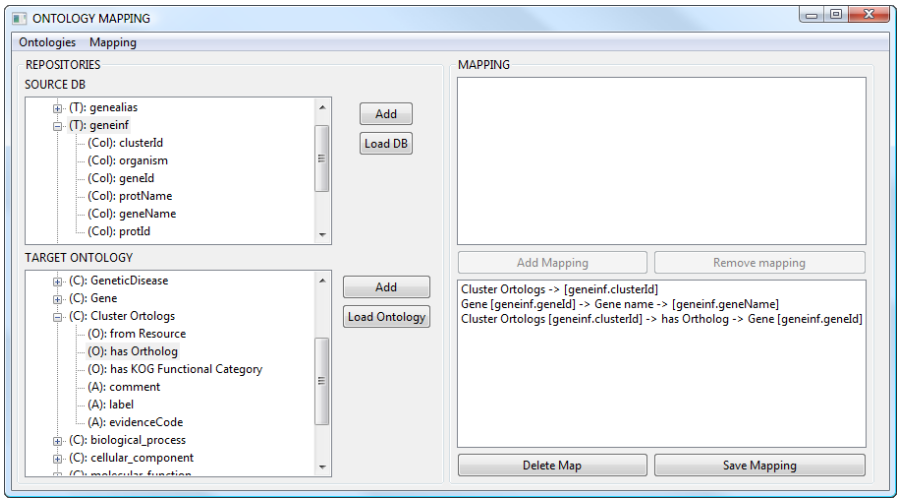

Figura 10.2: Vista principal para la definición de reglas de correspondencia.

ontología en la ventana principal, o importar ficheros de reglas de correspondencia previamente definidos. La parte izquierda de la ventana principal, identificada por la palabra *"REPOSITORIES"*, muestra los elementos que se podrán asociar de los distintos repositorios sobre los que se van a definir las correspondencias. La parte superior de la ventana, identificada por *"SOURCE DB"*, muestra las tablas del esquema del repositorio relacional cargado. Los nodos del primer nivel del árbol están caracterizados con el prefijo *"(T):"* que indica que se tratan de tablas relacionales, mientras que los demás nodos del árbol están caracterizados por *"(Col):"* e indican las columnas. Por otro lado, la parte inferior de la ventana, identificada por *"TARGET ONTOLOGY"*, está asociada a la jerarquía de clases, propiedades y relaciones de la ontología. Los nodos con el prefijo *"(C):"* indican que son clases de la ontología, los nodos con *"(I):"* están asociados a una instancia de la clase de su nivel superior, los nodos con *"(D):"* identifican a propiedades, los nodos con *"(O):"* indican las relaciones, las cuales tienen como dominio las clases de su nivel superior en el árbol, y por último, los nodos con *"(A):"* representan posibles anotaciones definidas en la ontología.

Para cargar un esquema de base de datos en el modelo se ha utilizado el botón *"Load DB"*. La Figura 10.3 muestra los distintos parámetros necesarios para realizar la conexión al repositorio. El primer elemento de la ventana

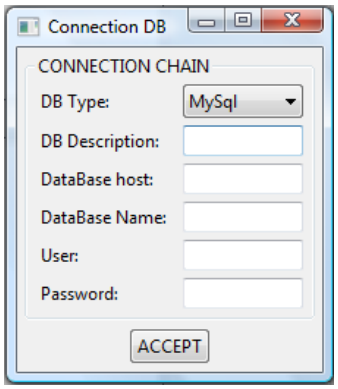

Figura 10.3: Vista para la especificación de los parámetros de conexión al repositorio.

es un botón para seleccionar entre distintos sistemas gestores de bases de datos. Los tipos de gestores de repositorios soportados son MySQL [221], PostgreSQL [222] y Oracle [223]. El segundo parámetro, *"DB Description"*, es una descripción textual del repositorio, que es opcional. El tercer parámetro, *"DataBase host"*, indica la URL del servidor de base de datos. El cuarto parámetro, *"DataBase Name"*, especifica el nombre del esquema de base de datos a importar. Por último, el quinto y sexto parámetros se corresponden con el usuario y contraseña para acceder al repositorio. Este usuario debe tener acceso de lectura al contenido y a los metadatos del esquema importado.

Por otro lado, para cargar la ontología con la que definir las correspondencias, se ha hecho uso del botón *"Load Ontology"*. Este botón abre una ventana para explorar los ficheros locales del sistema con el fin de seleccionar el fichero que contiene la definición de la ontología. Los botones *"Add"* permiten seleccionar los elementos de los árboles de los recursos de información. Así, el botón superior sirve para seleccionar los nodos del árbol del repositorio relacional y el botón inferior sirve para seleccionar los elementos de la ontología.

Una vez se han cargado el repositorio relacional y la ontología en la vista principal, se pueden definir las reglas de correspondencia. Se han descrito tres tipos diferentes de reglas de correspondencia, *DB2Class*, *DB2Prop* y *DB2Rel*. De este modo, la interfaz implementa tres procedimientos para definir cada tipo de regla.

La parte derecha de la vista principal, identificada por la etiqueta *"MAPPING"*, contiene dos cuadros de texto, donde el cuadro superior contiene los elementos temporales seleccionados de los recursos y el cuadro inferior muestra las reglas de correspondencia ya definidas. Con todo, el botón *"Add Mapping"* sirve para crear el tipo de regla de correspondencia adecuada según los elementos seleccionados de los recursos de información, y el botón *"Remove mapping"* sirve para borrar las selecciones realizadas sobre la regla actual. El botón *"Delete Map"* borra la regla de correspondencia seleccionada, y el botón *"Save Mapping"* se utiliza para localizar dónde se ha de almacenar el documento XML con las definiciones de la reglas de correspondencia.

A continuación, se describiren los procedimientos para definir los diferentes tipos de reglas de correspondencia utilizando la interfaz implementada.

Procedimiento para la definición de reglas de correspondencia DB2Class

Para definir las reglas tipo *DB2Class*, primero se debe seleccionar la clase correspondiente a la regla del árbol de elementos de la ontología. Después, se deben seleccionar las columnas de la tabla del repositorio que se corresponderán con la clave de la clase seleccionada. Una vez seleccionados los distintos elementos, se crea la regla automáticamente pulsando el botón *"Add Mapping"*. A continuación, ésta se inserta directamente en el cuadro de texto con las definiciones de las demás reglas de correspondencia, de modo que el cuadro de texto superior vuele a estar en blanco para la siguiente definición.

En la Figura 10.2, se muestra un ejemplo de regla de correspondencia del tipo *DB2Class*, *"Cluster Orthologs -> [geneinf.clusterId]"*, que indica que la clave de la clase de los grupos ortólogos se corresponde con la columna *clusterId* de la tabla *geneinf*.

Procedimiento para la definición de reglas de correspondencia DB2Prop

Para la definición de reglas tipo *DB2Prop*, se debe seleccionar primero una de las propiedades de la ontología que aparecen en la ventana principal

de la interfaz. Los elementos caracterizados con el prefijo *"(D):"* y *"(A):"* se corresponden con los elementos de la ontología seleccionables para este tipo de regla. Además, se debe seleccionar la tabla del repositorio en la que se relaciona la propiedad con la clase correspondiente. Una vez seleccionadas la tabla del repositorio y la propiedad de la ontología, se utiliza el botón *"Add Mapping"* para especificar el resto de parámetros necesarios para definir la regla en la Figura 10.4.

La Figura 10.4 está dividida en dos partes: la parte izquierda, identificada por la etiqueta *"RESOURCES"*, que proporciona la información sobre la tabla seleccionada (caracterizada con la etiqueta *"DATABASE"*) y el árbol de la ontología (caracterizado con la etiqueta *"ONTOLOGY"*); y la parte derecha, identificada por la etiqueta *"MAPPING PARAMETERS"*, que muestra los parámetros especificados para la creación de la regla.

En la parte izquierda de la ventana, se muestran las columnas de la tabla para que con el botón *"Add domain column"* se seleccionen las columnas correspondientes a la clave de la clase, y el botón *"Add value column"* para seleccionar las columnas que contienen los valores de la propiedad. La clase de la ontología asociada a la propiedad se selecciona del árbol de la ontología mediante el botón *"Add class"*. Esta información es proporcionada en la definición de las reglas de correspondencia para las propiedades, ya que una propiedad puede estar asociada a más de una clase de la ontología. El botón *"Remove annotation"* permite borrar las anotaciones seleccionadas en la ventana.

Por último, el botón *"Add mapping"* permite crear la definición de la regla de correspondencia y volver a la ventana de la vista principal para continuar con la definición del resto de reglas. Un ejemplo de cómo una regla de este tipo es representada en la interfaz (véase la Figura 10.2) es: *"Gene[geneinf.geneId] -> Gene name -> [geneinf.geneName]"*. Este ejemplo ilustra cómo la propiedad *Gene name* se asocia a instancias de la clase gen, que tiene como clave primaria los valores de la columna *geneId* y como nombre de los genes los valores de la columna *geneName* de la tabla *geneinf*.

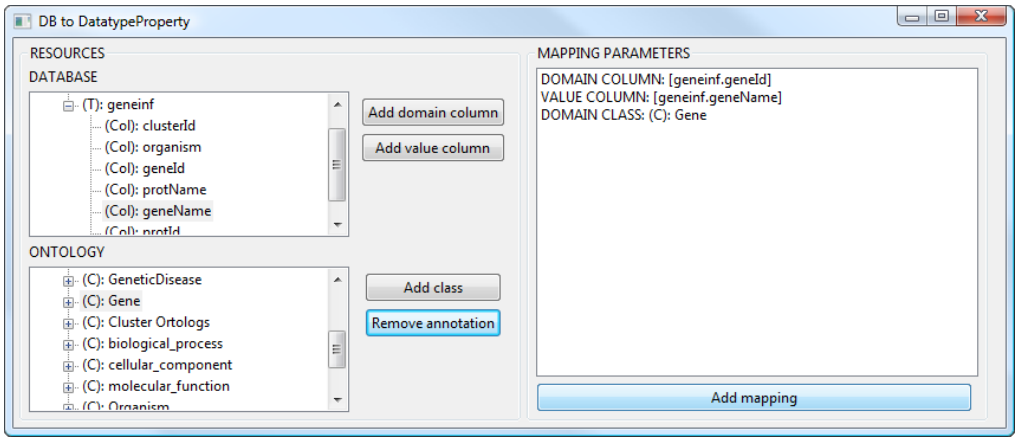

Figura 10.4: Ejemplo de vista para la definición de reglas de correspondencia tipo *DB2Prop*.

Procedimiento para la definición de reglas de correspondencia DB2Rel

Para la definición de las reglas de correspondencia tipo *DB2Rel*, se debe seleccionar una relación de la ontología y la tabla del repositorio que contiene la información a utilizar. Las relaciones están caracterizadas por la etiqueta *"(O):"* en la vista principal. Una vez seleccionados estos elementos de la vista principal y tras presionar el botón *"Add Mapping"*, se muestra la ventana para especificar los parámetros de la regla.

La Figura 10.5 muestra un ejemplo de la vista para la definición de los parámetros de las reglas de correspondencia DB2Rel. Esta ventana está compuesta por dos partes: la parte izquierda, identificada por la etiqueta *"RESOURCES"*, que proporciona la información sobre la tabla seleccionada (caracterizada con la etiqueta *"DATABASE"*) y el árbol de la ontología (caracterizado con la etiqueta *"ONTOLOGY"*); y la parte derecha, identificada por la etiqueta *"MAPPING PARAMETERS"*, que indica los parámetros especificados para la creación de la regla.

En la parte izquierda, se dispone del botón *"Add domain columns"* que sirve para seleccionar las columnas de la tabla asociada a la clase del dominio de la relación y que se corresponden con la clave de la clase. El botón *"Add range columns"* especifica las columnas que están relacionadas con la clave del

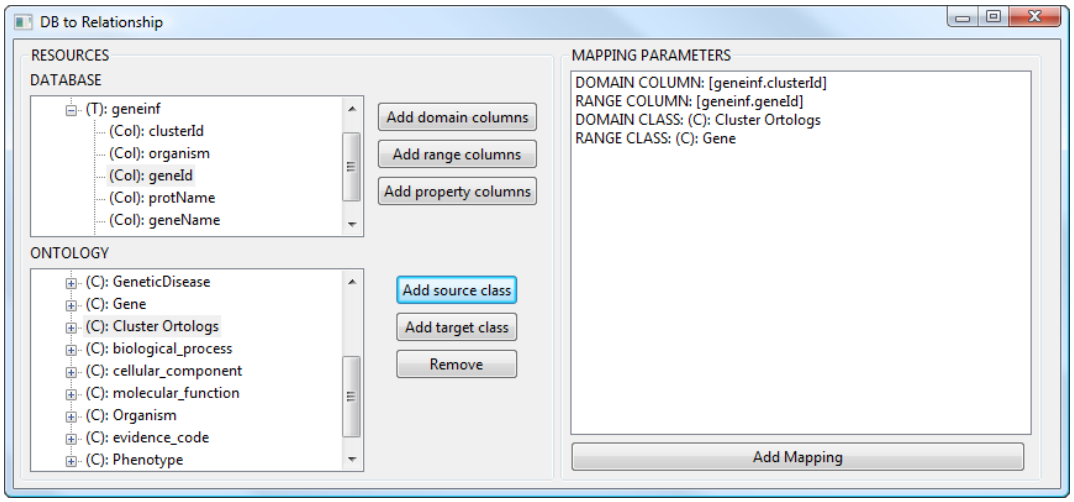

Figura 10.5: Ejemplo de vista para la definición de reglas de correspondencia tipo *DB2Rel*.

rango de la relación. El botón *"Add property columns"* puede ser utilizado para seleccionar una sub-relación de la elegida. Los valores de la columna seleccionada se corresponde con algún valor de la etiqueta *"label"* de una sub-relación de la relación elegida.

Por otro lado, el botón *"Add source class"* es utilizado para seleccionar la clase sujeto de la relación dentro del árbol de la ontología. El botón *"Add target class"* sirve para seleccionar la clase objetivo de la relación de la regla. El botón *"Remove"* permite borrar los parámetros de la regla seleccionados del cuadro de texto de la parte derecha de la ventana.

Por último, el botón *"Add Mapping"*, sirve para crear la definición de la regla de correspondencia y volver a la ventana de la vista principal de definición de reglas. El ejemplo de este tipo de reglas, presente en la Figura 10.2, está representado por: *"Cluster Orthologs [geneinf.clusterId] -> has Ortholog -> Gene [geneinf.geneId]"*, indicando que la relación *has Ortholog* asocia una instancia de la clase de grupos ortólogos, con su clave primaria la columna *clusterId*, con una instancia de la clase gen, y siendo su clave primaria la columna *geneId* de la tabla *geneinf*.

Una vez definidas todas las reglas de correspondencia entre el repositorio relacional y la ontología, se usa el botón *"Save Mapping"* para almacenar

en un documento XML: (1) los parámetros de conexión de la base de datos relacional; (2) la localización de la ontología; (3) cada una de las reglas definidas, según la gramática de reglas de correspondencia definida en el Anexo 13.3.2.

10.1.2. Interfaz de definición de reglas de identidad

Para facilitar la definición de las reglas de identidad, se ha desarrollado una interfaz que se corresponde con la opción *"DEFINE IDENTITIES"* de la Figura 10.1. Esta interfaz permite al usuario definir las reglas de identidad sin necesidad de conocer la gramática de su lenguaje, la cual está descrita en el Anexo 13.3.2. La interfaz evita definir las reglas manualmente y, por lo tanto, permite reducir muchos errores y coste en la definición de éstas.

La vista principal de la interfaz (véase la Figura 10.6) se dividide en dos secciones principales. La sección superior de la ventana, identificada mediante la etiqueta *"IDENTITIES CONDITIONS"*, contiene un cuadro de texto donde se muestran las reglas ya definidas, y un conjunto de botones para la configuración de la interfaz y la gestión de las reglas. La sección inferior de la ventana tiene como finalidad guiar el proceso de definición de las reglas, y está compuesto por dos elementos. Por un lado, la parte izquierda se compone de un cuadro de texto que describe la regla que se está definiendo, y, por otro lado, en la parte derecha se ofrecen las funcionalidades para definir los requisitos de la regla actual. Además, la barra de menú de la ventana permite abrir y cerrar la vista de interfaz, así como cargar un documento XML con la definición de reglas de identidad.

Las reglas de identidad definidas se muestran en la parte superior de la ventana principal. Estas reglas tienen la forma de expresión booleana, que se corresponden con las condiciones para considerar dos instancias de una ontología como equivalentes. Por ejemplo, en la Figura 10.6 se describe la regla:

"(Gene AND ((ALL fromSpecies -> EQUAL Organism) AND ((SOME Gene_name -> EQUALS IGNORE CASE) OR (SOME Gene_identifier -> EQUALS IGNORE CASE))))"

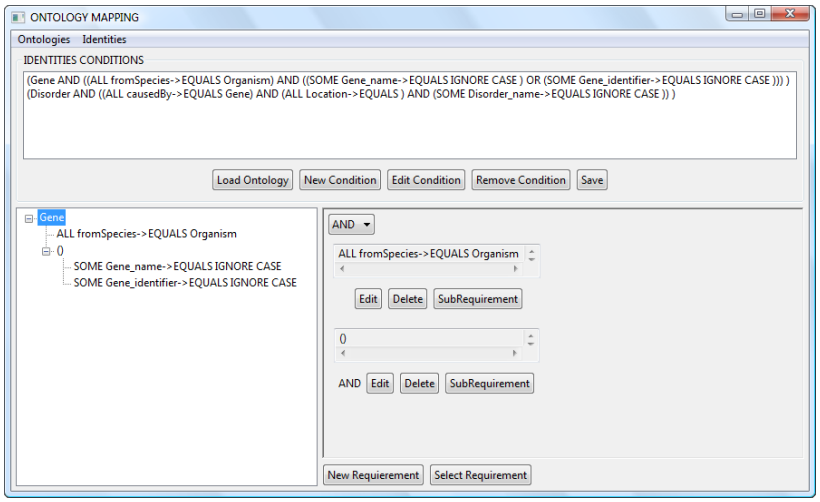

Figura 10.6: Vista principal para la definición de reglas de identidad.

Esta regla indica que dos instancias son equivalentes si pertenecen a la clase gen, tienen relación con la misma especie y comparten algún nombre o identificador de instancia.

Para comenzar a definir las reglas de identidad, primero se debe inicializar la interfaz, indicando la localización de la ontología que se va a utilizar. Para ello, la interfaz dispone del botón *"Load Ontology"*, el cual muestra la ventana para explorar y seleccionar el documento de la ontología en el sistema de ficheros local. Para crear una nueva regla de identidad, se utiliza el botón *"New Condition"*, que muestra una ventana como la de la Figura 10.7.

En la vista de la Figura 10.7, se puede seleccionar el concepto de la ontología que va a estar relacionado con la definición de los requisitos de la regla. Esta figura muestra la jerarquía de conceptos de la ontología cargada en la interfaz, y el botón *"ACCEPT"* permite asignar la clase seleccionada a la definición de la regla de identidad y volver a la ventana principal. Para editar una regla ya definida, se utiliza el botón *"Edit Condition"*, que carga en la parte inferior de la ventana la descripción de la regla seleccionada del cuadro de texto de la parte superior.

Para eliminar una regla definida, se utiliza el botón *"Remove Condition"*. Por último, el botón *"Save"* se utiliza para guardar las reglas definidas en la interfaz en un documento XML de manera persistente.

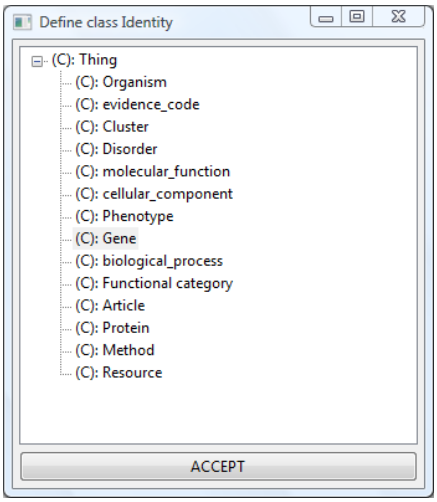

Figura 10.7: Vista para la selección de la clase de la ontología asociada a una regla de identidad.

En la parte inferior de la ventana principal, se encuentran los elementos para la definición de las reglas de identidad. En su parte izquierda, se muestra un árbol con los requisitos definidos en la regla. Además, para definir los requisitos de la regla de identidad, se proporcionan los elementos necesarios en la parte derecha de la vista.

La Figura 10.6 muestra en su parte inferior-derecha los elementos para definir dichos requisitos. El primer elemento que aparece en la vista, siguiendo un orden descendente, es el botón de selección con las opciones *AND* y *OR*. Este botón caracterizará si los requisitos del concepto seleccionado son opcionales u obligatorios. Los siguientes elementos son utilizados para especificar los requisitos. Cada uno estará compuesto de un cuadro de texto donde se mostrará el requisito definido y tres botones para gestionarlo: el botón *"Edit"* para editar la definición del requisito, el botón *"Delete"* para borrar el requisito, y el botón *"SubRequirement"* para definir sub-requisitos relacionados con el concepto al que hace referencia. Por último, en la parte inferior de la ventana hay dos botones, *"New Requirement"* y *"Select Requirement"*. El primer botón permite añadir los elementos a la vista con los que definir un nuevo requisito: su cuadro de texto y sus botones correspondientes. El segundo botón es utilizado para centrar la definición de la regla, entre los

distintos niveles de requisitos y sub-requisitos definidos, y así poder definir expresiones booleanas más complejas.

La Figura 10.8 muestra la ventana utilizada para definir los parámetros de los requisitos. La parte izquierda de esta ventana muestra la información de la ontología necesaria para definir el requisito, y la parte derecha contiene los cuadros de texto para definir los parámetros. El primer parámetro se corresponderá con el botón etiquetado con *"Boolean operator:"*, que se utiliza para indicar el tipo de relación con el anterior requisito (en caso de ser el primer requisito definido, este parámetro no tiene valor). El siguiente parámetro, con la etiqueta *"Property Scope:"*, debe tomar un valor entre *ALL* o *SOME*, e indica que todos o algunos de los valores asociados a una propiedad o relación de las instancias deben ser equivalentes, respectivamente. El parámetro etiquetado con *"Select Property:"* está relacionado con el botón *"Add Property"*, que es utilizado para seleccionar una propiedad o relación de la ontología mostrada. El siguiente parámetro, *"Requirement"*, es el que se utilizará para establecer el tipo de comparación entre valores de las instancias de la ontología para la propiedad seleccionada. Los valores pueden ser *EQUALS*, si los valores deben ser iguales, o *EQUALS IGNORE CASE*, si los valores deben ser iguales sin distinguir entre mayúsculas y minúsculas.

En caso de seleccionar una propiedad, se puede definir el parámetro *"Value type:"* para indicar el tipo de datos de la propiedad. En otro caso, se debe seleccionar una clase de la ontología. Esta clase es el objeto de la relación y se asocia al requisito mediante el botón *"Add Class"* con etiqueta *"Select Class:"*.

Además, se dispone de tres botones de la parte inferior de la ventana: (1) el botón *"ACCEPT"*, para finalizar el proceso de definición de los parámetros y volver a la ventana principal; (2) el botón *"CLEAR"*, para borrar los valores de los parámetros del requisito; y (3) el botón *"CANCEL"*, para volver a la ventana principal sin que se tengan en cuenta los cambios en el requisito.

Por último, la información sobre la ontología mostrada en el cuadro de texto de la parte izquierda de la ventana va cambiando según los parámetros del requisito que se está editando. De esta manera, cuando se debe seleccionar una propiedad o relación de la ontología, ésta mostrará las propiedades

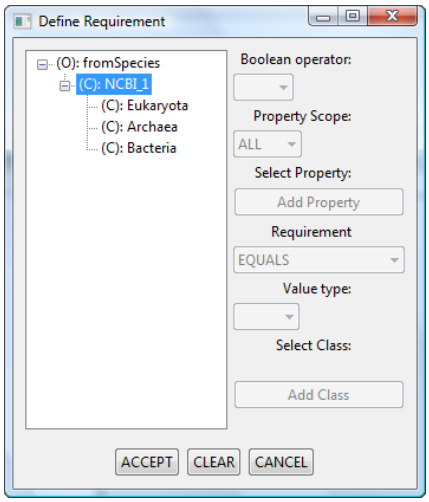

Figura 10.8: Vista para la definición de un requisito en una regla de identidad.

y relaciones asociadas a la clase de la ontología sobre la que se define el requisito. Por otro lado, cuando se tiene que seleccionar la clase objeto de la relación, ésta mostrará el rango de clases correspondientes a la relación seleccionada de la ontología. Por lo tanto, el usuario debe definir los parámetros del requisito en orden descendente y además evitar posibles incoherencias en la definición de la regla.

Durante la definición de una regla, el cuadro de texto de la ventana principal con las definiciones de los requisitos va cambiando automáticamente según se van definiendo los requisitos. Este cuadro de texto contiene, en su primera fila, la última regla de identidad editada, la cual va cambiando automáticamente a medida que se definen los requisitos.

Además, la definición de requisitos de una regla tiene forma de árbol, donde el primer nivel se corresponde con la clase de la ontología asociada a la regla de identidad, el segundo nivel se corresponde con los requisitos relacionados con propiedades o relaciones de la clase del primer nivel, el tercer nivel contiene las definiciones de sub-requisitos de la clase objeto del requisito del nivel superior, y así sucesivamente.

10.1.3. Interfaz de configuración del proceso de integración semántica

Para ejecutar el proceso de integración automático de los distintos repositorios bioinformáticos, se deben configurar ciertos parámetros de la aplicación desarrollada. Los parámetros a configurar son los siguientes:

- El fichero con las descripciones de las reglas de identidad asociadas a la ontología. El documento de reglas debe contener la referencia al documento OWL de la ontología.
- El esquema vacío del repositorio, donde se crea el repositorio semántico y, por lo tanto, donde se almacena la información integrada.
- La lista de ficheros con las descripciones de las reglas de correspondencia entre los repositorios que contienen la información a integrar y la descripción de la ontología global del repositorio semántico. La ontología indicada en los ficheros de reglas de correspondencia y la indicada en el fichero de reglas de identidad deben coincidir para que pueda llevarse a cabo la integración semántica de la información.

Se ha desarrollado una interfaz para la configuración y ejecución del proceso de integración automático. La Figura 10.9 muestra la vista principal de la interfaz de configuración donde se especifican los parámetros descritos anteriormente.

El botón etiquetado con *"Add Identities File"* de la vista principal tiene como finalidad seleccionar el documento XML con la definición de las reglas de identidad. El botón *"Add KB Parameters"* se utiliza para especificar los parámetros asociados al repositorio semántico. Para seleccionar la lista de documentos XML con las reglas de correspondencia, se proporciona el botón *"Add Mapping"*, el cual muestra una nueva ventana con la que seleccionar los documentos a utilizar (véase la Figura 10.10).

Cada documento seleccionado, se muestra en la ventana principal junto con dos botones, *"Add"* y *"Delete"*. El primer botón sirve para editar la selección del documento, mientras que el segundo botón permitire eliminar

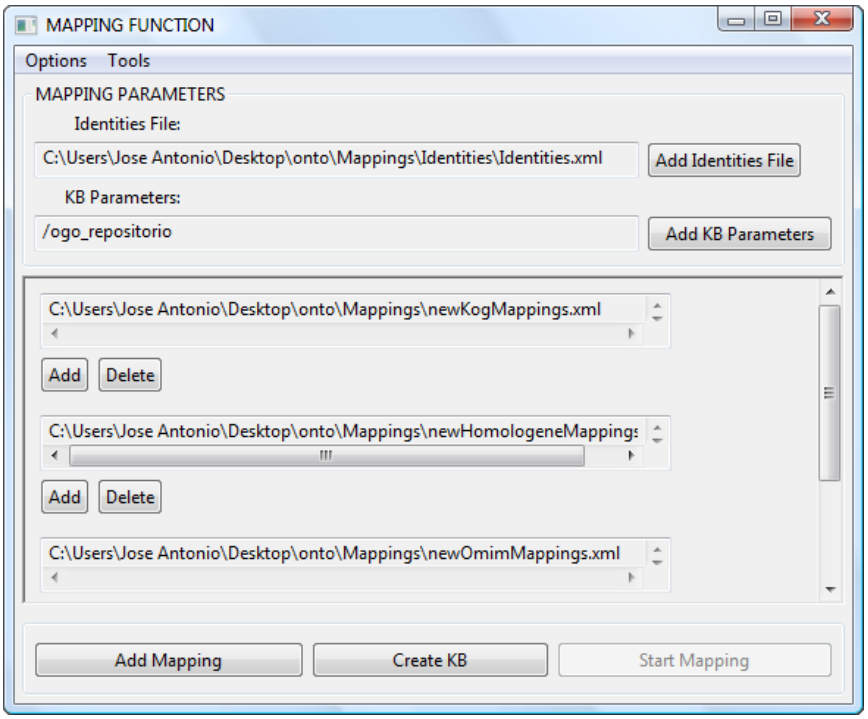

Figura 10.9: Vista de la interfaz para indicar los parámetros de configuración del proceso de integración.

un documento seleccionado. El botón *"Create KB"*, en la parte inferior de la ventana principal, se utiliza para inicializar el repositorio semántico en el esquema indicado antes del proceso de integración. Para iniciar el proceso de integración semántico, se dispone del botón *"Start Mapping"* que llama al método de integración con los parámetros especificados en la interfaz.

Por último, para especificar los parámetros asociados a los documentos XML de reglas de correspondencia se desarrolló la vista mostrada en la Figura 10.10. En esta vista, se debe especificar el documento XML con las reglas de correspondencia asociadas a un repositorio bioinformático concreto.

Sin embargo, para asociar los conceptos de las bio-ontologías importadas en la ontología global con las instancias insertadas en el repositorio semántico, se ofrece la posibilidad de combinar varios documentos de reglas de correspondencia para su integración en un mismo repositorio semántico. Esto sólo es posible si los repositorios bioinformáticos asociados a cada fichero de reglas de correspondencia comparten los identificadores de las mismas claves en sus

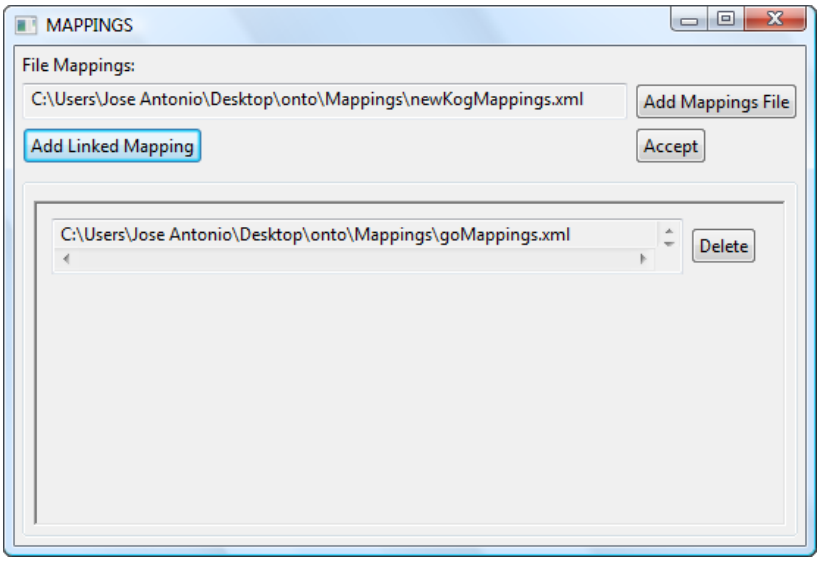

Figura 10.10: Vista donde se especifican los ficheros de reglas de correspondencia asociados a un repositorio bioinformático.

esquemas. Esta característica permite mejorar la eficiencia del proceso ya que realiza la integración de varios repositorios bio-informáticos compatibles en un sólo paso.

Así, la Figura 10.10 muestra el botón *"Add Mapping File"*, para asignar el documento de reglas de correspondencia principal, y el botón *"Add Linked Mapping"*, para asociar los documentos de reglas relacionados. Los documentos de reglas de correspondencia relacionados con el documento principal se muestran en un lista en la parte inferior de la vista. El botón *"Delete"*, que está asociado a cada documento, permite eliminarlo de la lista. *"Accept"* finaliza el proceso de asignación de documentos de reglas de correspondencia y se vuelve a la ventana principal de configuración del proceso de integración.

Bloque III

Validación: Sistema OGO

Capítulo 11

Creación del Sistema OGO

En este capítulo, se describen las características de los elementos del Sistema OGO que se han desarrollado como resultado de haber seguido la metodología de integración semántica, definida en el capítulo 8.

La Figura 8.1 muestra los elementos genéricos de la metodología de integración utilizados para la creación del sistema OGO [224]. En concreto, el sistema OGO relaciona información sobre genes y proteínas ortólogas con información sobre enfermedades genéticas humanas. Ya que la información está integrada en un único repositorio semántico, éste facilitará el desarrollo de investigaciones traslacionales mediante la explotación de su contenido.

Los repositorios biológicos y biomédicos utilizados en el proceso de integración son los relacionados con la información de genes y proteínas ortólogas KOG, OrthoMCL, HomoloGene e Inparanoid (véase la sección 2.2); y los relacionados con la información de enfermedades genéticas humanas OMIM (véase la sección 2.3).

En las siguientes secciones, se describirán en detalle la ontología global del Sistema OGO y el repositorio semántico o base de conocimiento OGO.

11.1. Definición de la ontología global OGO

Esta sección describe la definición de la ontología global utilizada en la construcción del sistema OGO, así como las relaciones establecidas con otras

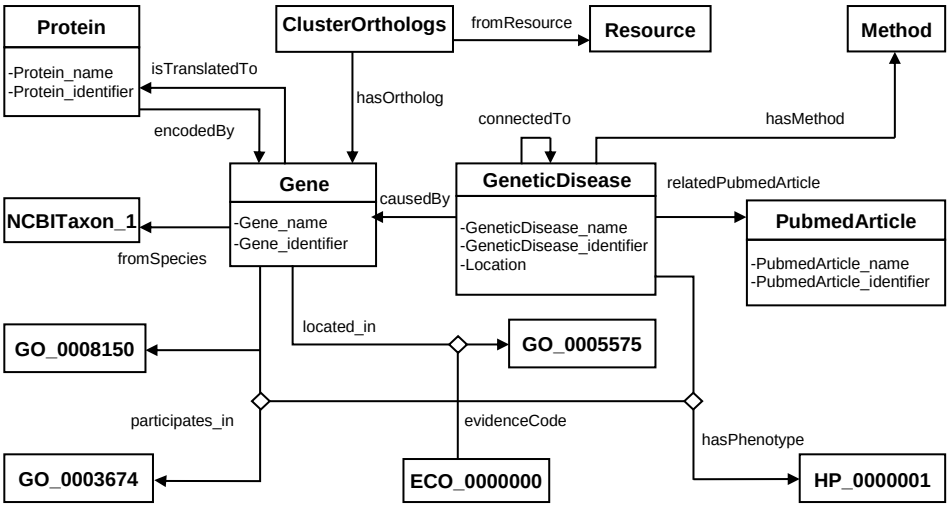

Figura 11.1: La ontología del Sistema OGO.

bio-ontologías importadas. La ontología global abarca el conocimiento de los dominios de grupos de genes ortólogos y de las enfermedades genéticas humanas presentes en los repositorios de información utilizados como entrada para el proceso de integración. El espacio de nombres utilizado es *http://miuras.inf.um.es/ontologias/OGO.owl#*. La Figura 11.1 muestra los conceptos raíz de la ontología, sus propiedades y las relaciones establecidas entre ellos. También muestra los conceptos y relaciones asociados con las ontologías biomédicas importadas en la ontología global OGO. Las ontologías importadas son: la ontología Gene Ontology (véase la sección 4.1.2); la ontología ECO (véase la sección 4.1.3); la ontología RO (véase la sección 4.1.4); la ontología NCBITaxonomy (véase la sección 4.1.5);y la ontología de HPO (véase la sección 4.1.6).

En la representación de los conceptos de la ontología OGO, destacan dos partes, una sobre grupos ortólogos y otra sobre enfermedades genéticas. Las dos partes tienen en común el concepto de gen. Así, mediante la identificación e integración de las instancias comunes del concepto gen, se pueden establecer las relaciones entre ambas partes. Además, la ontología tiene otros conceptos y relaciones asociados pertenecientes a otras ontologías biomédicas. Las siguientes secciones describen las diferentes partes de esta ontología.

Por otro lado, se puede destacar que la coherencia y consistencia de la

definición de la ontología global OGO han sido verificadas utilizando razonadores de OWL, como Pellet y HermiT. Estos razonadores permiten verificar si existen incoherencias en la definición de los conceptos, propiedades, relaciones y restricciones de la ontología. De esta manera, se comprueba de manera formal que la definición de la ontología cumple las restricciones de la lógica descriptiva del lenguaje OWL. Sin embargo, esta verificación no asegura que la ontología represente correctamente el conocimiento del dominio. La verificación de la calidad de la definición viene por la representación del conocimiento de los repositorios a integrar y por su revisión por expertos del dominio. Así, la reutilización de conceptos y relaciones de otras bio-ontologías verificadas por la comunidad científica permite la mejora y completitud del conocimiento representado.

11.1.1. Conceptualización del dominio de grupos ortólogos

La semántica de los grupos de genes y proteínas ortólogos ha sido representada en la definición en la ontología global (véase la Figura 11.1) mediante los siguientes conceptos: *Gene*, que representa la entidad gen y posee las propiedades *Gene_name* o nombre descriptivo del gen, y *Gene_identifier* que es su identificador; *Protein*, que representa a la entidad proteína con las propiedades *Protein_name* y *Protein_identifier* que se corresponden con el nombre e identificador de la proteína; *ClusterOrthologs*, que modela los grupos de genes ortólogos; y *Resource*, que identifica los distintos repositorios de información sobre grupos ortólogos que se integran.

Los genes y proteínas están enlazados a través de la relación *isTranslatedTo*, para indicar que un gen codifica una proteína determinada, y mediante *encodedBy* se indica su relación inversa. La relación *isTranslatedTo* tiene una cardinalidad mínima cero, ya que no todos los repositorios establecen la relación entre la secuencia del gen y de la proteína asociada para todas sus instancias. La relación *fromResource* asocia a un grupo de genes ortólogos con el repositorio bioinformático del que se recopiló la información y su cardinalidad es uno. Cada repositorio bioinformático utiliza un algoritmo diferente

para la obtención de los grupos y, por lo tanto, la información de los grupos debe permanecer diferenciada. La relación *hasOrtholog* relaciona un grupo con los genes ortólogos asociados. Un grupo de ortólogos, por definición, debe contener información de más de un gen y, por lo tanto, su cardinalidad mínima es dos.

11.1.2. Conceptualización del dominio de enfermedades genéticas

La semántica asociada a las enfermedades genéticas es representada en la ontología global mediante los conceptos *GeneticDisease*, *Method* y *PubMedArticle*. El concepto *GeneticDisease* representa una enfermedad genética y posee las propiedades *GeneticDisease_name*, *GeneticDisease_identifier* y *Location*, las cuales representan el nombre de la enfermedad, su identificador del repositorio OMIM y la localización en el cromosoma de la alteración del gen causante de la enfermedad, respectivamente. El concepto *Method* representa los métodos por los que se obtuvo la información sobre la enfermedad, como "la deducción por la secuencia de aminoácidos de la proteína". Por último, el concepto *PubMedArticle* representa las referencias a los artículos del repositorio PubMed [225] relacionados con las enfermedades genéticas. En este concepto, se definen las propiedades *PubmedArticle_name* y *PubmedArticle_identifier*, que representan el título del artículo y su identificador en la base de datos PubMed.

Las instancias del concepto de enfermedades genéticas están conectadas con los genes causantes de las enfermedades a través de la relación *causedBy*, siendo la asignación de un valor para esta relación obligatoria para cada enfermedad genética. La relación *connectedTo* asocia la información de dos enfermedades genéticas relacionadas, bien porque comparten origen genético o bien por sus características fenotípicas comunes. Las enfermedades genéticas están relacionadas con los métodos seguidos en la investigación clínica de las enfermedades mediante la relación *hasMethod*. La relación *relatedPubmedArticle* asocia las enfermedades genéticas con los artículos que los referencian de la base de datos PubMed.

11.1.3. Reutilización de ontologías biomédicas

La ontología global definida para el sistema OGO reutiliza el conocimiento presente en otras ontologías biomédicas. La reutilización de conceptos y relaciones descritos en otras bio-ontologías facilita una búsqueda uniforme de información, evita la duplicidad en la definición de recursos ontológicos y mejora la calidad de la conceptualización de la ontología, ya que el conocimiento que describen ha sido revisado y aceptado por la comunidad científica.

Las bio-ontologías reutilizadas forman parte del proyecto OBO Foundry. Uno de los objetivos de OBO Foundry es la evolución de ontologías con el objetivo de la integración de información biomédica. Sus ontologías definen, principalmente, jerarquías o taxonomías de conceptos con pocas relaciones entre ellos. Sin embargo, para llevar a cabo el proceso de integración semántica de la información del Sistema OGO con el conocimiento modelado en dichas bio-ontologías, ha sido necesario utilizar el método, definido en el estándar OWL2.0, *"Punning"*. Utilizando esta técnica, se han redefinido las bio-ontologías importadas, añadiendo las instancias asociadas a cada uno de los conceptos que contienen. De esta manera, la información integrada en el repositorio semántico como instancias puede ser enlazada con las instancias insertadas a las bio-ontologías. La Figura 11.2 muestra un ejemplo de uso de la técnica de punning. El recurso *go:GO_0005623* es utilizado como clase al relacionarlo mediante la relación *rdfs:subClassOf*, con la clase *go:GO_0005575*. Además, *go:GO_0005623* es utilizado como instancia al asignarlo a la instancia *ogo:Gene/4852* de la clase *Gene* a través de la relación *ro:located_in*.

A continuación, se describe cómo los conceptos y relaciones de las bio-ontologías son reutilizadas en la ontología global del Sistema OGO (véase la Figura 11.1).

Gene Ontology (GO): La ontología GO está compuesta por tres sub-ontologías que abarcan el conocimiento sobre componentes celulares, funciones moleculares y procesos biológicos. Cada sub-ontología define un concepto raíz, el concepto *GO_0005575* está asociado a la ontología de componentes celulares, el concepto *GO_0003674* se corresponde con la

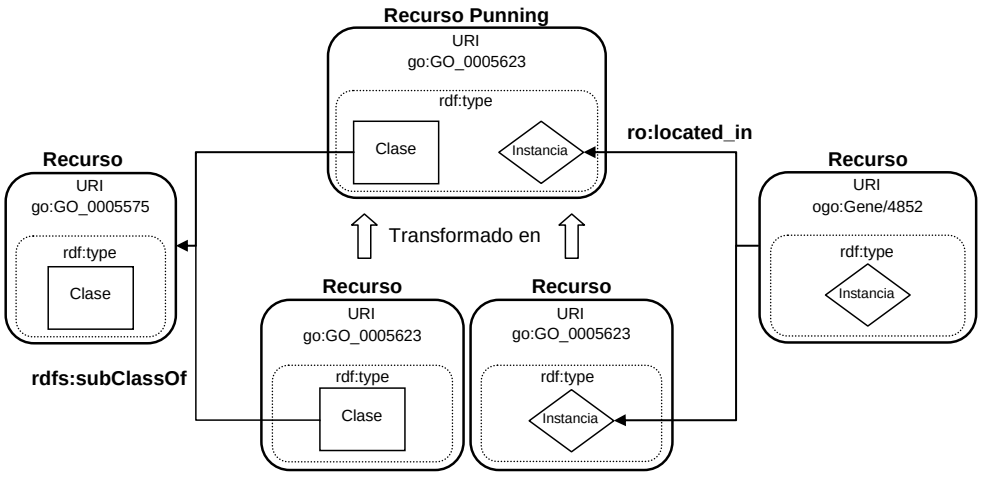

Figura 11.2: Ejemplo de uso de la técnica Punning para modelar el recurso *go:GO_0005623*.

ontología de funciones moleculares y el concepto *GO_0008150* se corresponde con la ontología de los procesos biológicos. Los genes están asociados con los conceptos de la Gene Ontology a través de las relaciones definidas en la ontología de relaciones biomédicas (RO) *participates_in* y *located_in*. Estas relaciones están anotadas por los conceptos de la ontología de códigos de evidencia (ECO). Se ha definido una jerarquía de sub-relaciones paralela a la jerarquía de códigos de evidencia y que establecen la relación ternaria entre una instancia de gen, un término de GO y su código de evidencia correspondiente. El espacio de nombres de esta ontología es *http://miuras.inf.um.es/ontologias/GO.owl#* y sustituye al espacio de nombres *http://purl.org/obo/owl/GO#*. Sin embargo, los nombres locales de los recursos de la ontología son idénticos. Además, se han establecido relaciones *owl:sameAs* para relacionar los conceptos originales con los redefinidos para este trabajo.

Ontología de Códigos de Evidencia (ECO): La ontología ECO proporciona una jerarquía de conceptos que describe una serie de códigos de evidencia, donde la raíz de la ontología es el concepto *ECO_0000000*. Ejemplos de códigos de evidencia son: la *Evidencia del Alineamiento de Secuencia*, o *ISA* en la Gene Ontology, que se corresponde con el

concepto *ECO_0000200*; y el *Contexto Genómico*, o *IGC* en la Gene Ontology, que se corresponde con *ECO_0000177*. Los conceptos de esta ontología son utilizados para caracterizar, mediante la anotación *evidenceCode*, las relaciones *participates_in* y *located_in* de la ontología de relaciones biomédicas y la relación *hasPhenotype* asociada a la ontología de fenotipos humanos. El espacio de nombres de esta ontología es *http://miuras.inf.um.es/ontologias/ECO.owl#* y sustituye al espacio de nombres *http://purl.org/obo/owl/ECO#*.

Ontología de Relaciones Biomédicas (RO): La ontología RO ha sido definida para proporcionar un conjunto consensuado de relaciones para utilizar al definir bio-ontologías. La ontología global reutiliza las relaciones *participates_in* y *located_in* que relacionan las instancias de genes con instancias de términos de GO. El espacio de nombres de esta ontología es el de la ontología global, *http://miuras.inf.um.es/ontologias/OGO.owl#* y sustituye al espacio de nombres *http://purl.org/obo/owl/relationship#*. Para poder establecer la relación ternaria entre genes, términos de GO y los códigos de evidencia, se han definido jerarquías de sub-propiedades para *participates_in* y *located_in*, donde cada sub-propiedad está relacionada con un código de evidencia a través de la anotación *evidenceCode*.

Taxonomía de Especies del NCBI (NCBITaxon): La ontología NCBITaxon representa la clasificación taxonómica de los organismos definida por el NCBI. Esta ontología está formada por una jerarquía de conceptos que establecen las relaciones entre las distintas especies de organismos, especialmente la clasificación taxonómica de Vida, Dominio, Reino, Filo, Clase, Orden, Familia, Género y Especie. Esta ontología no pretende ser una clasificación completa de las especies, ya que representa sólo el diez por ciento de las especies del planeta. La raíz de esta ontología es el concepto *NCBITaxon_1*, y representa a todos los seres vivos. Las instancias de genes tienen asignadas una instancia de un organismo concreto mediante la relación *fromSpecies*. El espacio de nombres de esta ontología es *http://miuras.inf.um.es/ontologias/NCBITaxon.owl#*,

y sustituye al espacio de nombres *http://purl.org/obo/owl/NCBITaxon#*. El nombre local de los recursos de la ontología son iguales en las dos ontologías y, además, se han establecido relaciones tipo *owl:sameAs* para relacionar los conceptos originales con los redefinidos para este trabajo.

Ontología de Fenotipos Humanos (HPO): La ontología HPO contiene una jerarquía de conceptos que describen diferentes fenotipos en humanos que pueden ser causados por enfermedades genéticas. El concepto raíz de esta ontología es *HP_0000001*. Estos conceptos están relacionados con las instancias de las enfermedades genéticas humanas, clase *GeneticDisease*, a través de la relación *hasPhenotype*. Esta relación también está anotada con los códigos de evidencia de la ontología ECO. Así, se formará una jerarquía de sub-relaciones de *hasPhenotype*, paralela a la jerarquía de códigos de evidencia, que al asignarla a los fenotipos establecen la relación entre la enfermedad genética, el fenotipo y el código de evidencia correspondiente. El espacio de nombres de esta ontología es *http://miuras.inf.um.es/ontologias/HP.owl#*, y sustituye al espacio de nombres *http://purl.org/obo/owl/HP#*. El nombre local de los recursos de la ontología son iguales en las dos ontologías, pero se han establecido las relaciones tipo *owl:sameAs* para relacionar los conceptos originales con los redefinidos para este trabajo.

11.2. Repositorio semántico del sistema OGO

En esta sección, se describen las características técnicas del repositorio semántico creado en el sistema OGO. La Figura 11.3 muestra la arquitectura del repositorio semántico del sistema OGO. Este repositorio utiliza como base el sistema gestor de base de datos MySQL para almacenar la información. Por encima del sistema gestor de base de datos se sitúa la capa intermedia de Jena, la cual ofrece al gestor del repositorio las herramientas e interfaces de programación para manipular y consultar el repositorio semántico. Además, Jena permite explotar la información del modelo mediante el uso de razonadores, y mediante una interfaz de consulta SPARQL.

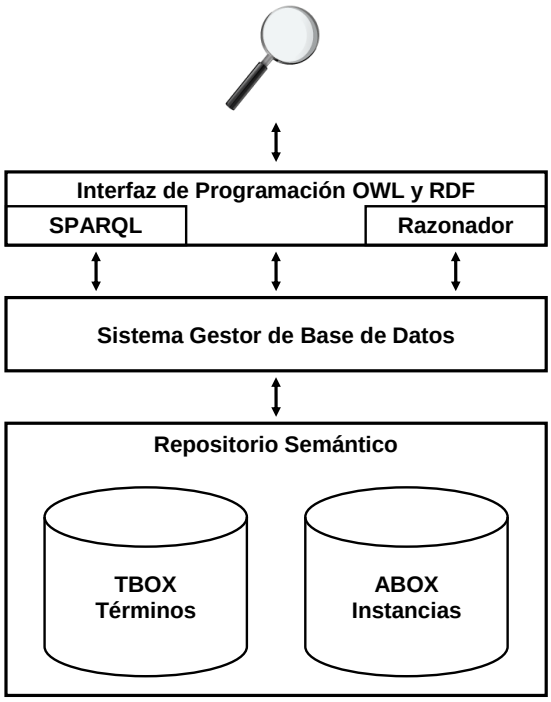

Figura 11.3: Arquitectura del Repositorio Semántico de OGO.

El modelo semántico, se basa en el lenguaje ontológico OWL para obtener una representación de la información modelada mediante lógica descriptiva. Este modelo semántico está dividido en un TBox y un ABox. El TBox, se corresponde con los conceptos definidos en la ontología global OGO, mientras que el ABox contiene las instancias resultado de la integración semántica. Con la distinción entre ambos elementos del repositorio semántico, un razonador puede realizar distintas tareas de inferencia según el nivel sobre el que se lleve a cabo su tarea. Así, a nivel de TBox se utiliza el término de *clasificación* para las definiciones conceptuales de la ontología, pero a nivel de ABox se utiliza el término *chequeo* para las instancias.

Para crear el repositorio semántico utilizando Jena son necesarios tres requisitos: (1) un esquema vacío en la base de datos donde se va a crear el repositorio; (2) tener un usuario de la base de datos con los permisos suficientes para crear, modificar y borrar el esquema correspondiente; (3) y utilizar la interfaz de programación de Jena.

Para crear el repositorio semántico y que Jena genere el modelo inferido

Concepto	Cantidad
Gene	666775
Protein	1048576
ClusterOrthologs	51307
GeneticDisease	18141
PubMedArticle	111028

Tabla 11.1: Tabla de las estadísticas de la cantidad de individuos almacenados en el repositorio semántico del Sistema OGO para los conceptos *Gene*, *Protein*, *ClusterOrthologs*, *GeneticDisease* y *PubMedArticle*.

asociado, se utiliza el perfil *PelletReasonerFactory.THE_SPEC*, que indica que se va a utilizar el razonador Pellet. Además, para utilizar el repositorio semántico sin funcionalidades de inferencia se tiene que usar el perfil *"OntModelSpec.OWL_DL_MEM"*, que indica que el modelo se corresponde con descripción de una ontología que sigue la especificación *OWL DL* y es almacenado en memoria.

Para la incorporación del modelo ontológico al repositorio semántico se ha utilizado el método *"read"* de la clase *"OntModel"* de Jena, la cual permite el contenido de ficheros OWL de manera sencilla. Sin embargo, para el proceso de integración se han utilizado los métodos *"createIndividual"* de *"OntModel"* y *"addProperty"* de la clase *"Resource"*.

Por último, como resultado del proceso de integración semántica llevado a cabo para la creación del Sistema OGO se obtuvo un repositorio semántico con la información descrita en la tabla 11.1.

Capítulo 12

Consulta, publicación e integración con aplicaciones externas

En este capítulo, se presentan las herramientas e interfaces de consulta desarrolladas para el Sistema OGO. Estas interfaces tienen como principales objetivos facilitar la búsqueda de la información integrada en el repositorio, y ofrecer la posibilidad de explotar el conocimiento que contiene. Estos objetivos se han llevado a cabo mediante el uso de tecnologías orientadas al desarrollo de aplicaciones y servicios Web, como JSP [226], SOAP o WSDL, y, por otro lado, con tecnologías de la Web semántica como OWL, RDF, SPARQL o Jena.

En este capítulo, se presenta una aplicación Web para la búsqueda de información, la cual permite a los usuarios obtener información del repositorio semántico OGO mediante formularios Web (sección 12.1). A continuación, se describen las características específicas de la publicación del contenido del repositorio semántico de OGO en la nube de LOD (sección 12.2). En la siguiente sección, se describe la utilización de la interfaz de consultas avanzadas, descrita en 9.1, junto con el repositorio semántico del Sistema OGO (sección 12.3). Esta interfaz utiliza la semántica del repositorio para guiar el proceso de definición de consultas avanzadas. Por último, se presentan los

servicios Web desarrollados para el acceso a la información disponible en el repositorio semántico de OGO por aplicaciones externas, a dicho sistema (sección 12.4). Como ejemplo de uso de estos servicios, se describirá su utilización por el sistema ONCOdata. ONCOdata es un sistema de recomendaciones de diagnóstico y tratamiento oncológico basado en evidencias.

12.1. Interfaz de consulta mediante formularios Web

Esta interfaz ha sido desarrollada para la consulta de la información del repositorio semántico OGO [227]. Por lo tanto, se ha buscado desarrollar una interfaz intuitiva, y que no requiera de grandes conocimientos de bioinformática, biología o biomedicina para su uso. Esta aplicación tiene como finalidad poder explotar las relaciones definidas entre instancias del repositorio para obtener la información sobre ortólogos y sobre enfermedades genéticas. Esta aplicación dispone de dos perspectivas de consulta diferentes, una centrada en los genes ortólogos y otra en las enfermedades genéticas. Desde cualquier perspectiva, se permite navegar por la información del repositorio a través de los enlaces proporcionados en los resultados de las búsquedas. La primera perspectiva de consulta obtiene la información a partir de la identificación de genes ortólogos, mientras que la otra lo hace a partir de las enfermedades genéticas.

Esta interfaz utiliza modelo de la ontología global del sistema OGO para poder extraer la información del repositorio semántico. El sistema es capaz de interrelacionar la información y facilitar la investigación traslacional, ya que la ontología global del sistema define un grafo donde la información está integrada mediante individuos, y estos están conectados a través de las relaciones definidas.

El primer método de consulta utiliza un identificador o nombre de un gen proporcionado por el usuario para recuperar la instancia correspondiente del repositorio semántico y posteriormente devolver la información solicitada por éste. En la figura 12.1, se muestran los componentes del formulario Web, de

la interfaz de búsqueda, desde la perspectiva de los genes ortólogos. El primer elemento que se puede observar en la esquina superior-izquierda es el selector, para cambiar entre la interfaz de búsqueda de la perspectiva sobre ortólogos o sobre las enfermedades genéticas. El siguiente elemento de la interfaz es el cuadro de texto, donde el usuario introducirá el nombre o identificador del gen sobre el que quiere realizar la consulta. Los nombres e identificadores han sido obtenidos de los repositorios de información biológica utilizados en el proceso de integración. A continuación, se proporcionan diversos elementos con los que los usuarios pueden filtrar la información de su búsqueda sobre genes ortólogos. Así, se proporciona una lista, *"Select Resources"*, con la que el usuario puede indicar los repositorios origen sobre los que realizar la búsqueda. Si se selecciona el elemento de la lista *"ALL_RESOURCES"*, se utilizará toda la información de grupos de ortólogos integrados en el repositorio. Por otro lado, la lista, *"Select Organisms"*, permitirá al usuario seleccionar un conjunto de especies sobre las que realizar la búsqueda, así, el usuario podrá restringir la búsqueda a ciertas especies. Al igual que ocurre con la anterior lista, se proporcionará un elemento para seleccionar todas las especies, *"ALL_ORGANISMS"*, el cual estará seleccionado por defecto. Por último, se proporcionará un conjunto de elementos en el formulario para indicar el tipo de información que se quiere obtener en la búsqueda. Así, se dispone de cinco elementos seleccionables: (1) *"Show synonyms"*, que proporciona el conjunto de sinónimos de los genes ortólogos encontrados en la consulta; (2) *"Show proteins"*, que devuelve la información sobre las proteínas asociadas a los genes seleccionados; (3) *"Show Disorders"*, que proporciona un enlace con la información de las enfermedades genéticas relacionadas con cada gen obtenido en las consultas; (4) *"Show Gene Ontology terms"*, que muestra para cada gen el conjunto de términos de la Gene Ontology que caracterizan las funciones moleculares, componentes celulares y/o procesos biológicos relacionados; y (5) *"Show resource"*, que muestra para cada gen los repositorios de información de donde ha sido obtenido.

La información devuelta por la interfaz de búsqueda depende de los elementos y filtros seleccionados. Para buscar las instancias de los genes ortólogos en el repositorio semántico, se utilizará el lenguaje de consultas SPARQL.

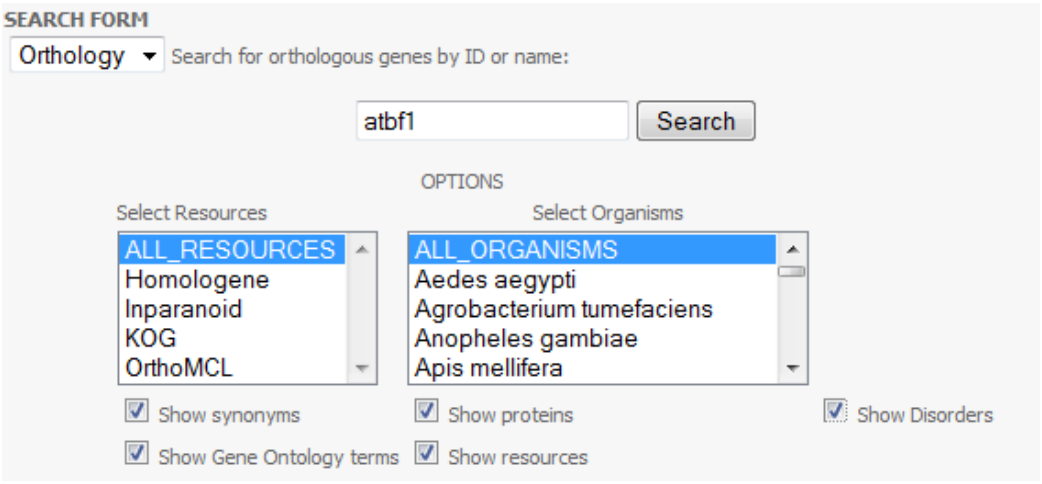

Figura 12.1: Interfaz de consulta de información sobre ortólogos.

La interfaz utiliza una plantilla para definir la consulta SPARQL correspondientes a los parámetros definidos por el usuario. La Figura 12.2 muestra la plantilla definida para la interfaz de búsqueda de información sobre genes ortólogos. En ella, se definen dos variables en el *SELECT*, *"?cluster"* y *"?gene"*, que se corresponden con las instancias de grupos de ortólogos y con instancias de genes respectivamente. Los grupos de ortólogos están relacionados con los genes a través de la relación *<ogo:hasOrtholog>*, definida en la ontología OGO. Además, se asociará a *"?cluster"* la variable *"?gene1"*, que representa la instancia de gen que el usuario define para buscar sus genes ortólogos. La variable *"?gene1"* utiliza el elemento *UNION*, de SPARQL, para buscar las diferentes instancias de gen que tengan asociado el identificador o nombre *"token"*, que es proporcionado por el usuario en el formulario Web de la interfaz. Los identificadores y los nombres de los genes están modelados en la ontología como *"<ogo:Gene_identifier>"* y *"<ogo:Gene_name>"*, respectivamente. Los filtros de búsqueda están definidos combinando los elementos *OPTIONAL* y *UNION*. Para el caso del filtro sobre el tipo de repositorio biológico origen de la información sobre grupos de ortólogos, se utilizará la relación *"<ogo:fromResource>"*. Sin embargo, para las especies sobre las que se centra la búsqueda se utiliza la relación *"<ogo:fromSpecies>"*. Por otro lado, los elementos *"<uri_Resource1>"* y *"<uri_Resource2>"* se correspon-

den con las URI de las instancias de los repositorios proporcionados en el formulario Web. Además, los elementos *"<uri_Organism1>"* y *"<uri_Organism2>"* representan las URI de las especies definidas en la ontología. Si el usuario no selecciona ningún filtro en el formulario Web, estas opciones no son incluidas en la consulta SPARQL final. Una vez obtenidas las URI de los genes ortólogos, se recuperan los individuos asociados y sus propiedades según los parámetros que el usuario haya seleccionado en el formulario Web.

```
PREFIX ogo:   <http://miuras.inf.um.es/ontologias/OGO.owl#> .
SELECT ?cluster ?gene
WHERE {
  ?cluster <ogo:hasOrtholog> ?gene .
  ?cluster <ogo:hasOrtholog> ?gene1 .
  {
    ?gene1 <ogo:Gene_identifier> ‘‘token’’ .
   UNION
    ?gene1 <ogo:Gene_name> ‘‘token’’ .
  }
  OPTIONAL {
   {
     ?cluster <ogo:fromResource> <uri_Resource1> .
    UNION
     ?cluster <ogo:fromResource> <uri_Resource2> .
   }
  }
  OPTIONAL {
    {
      ?gene <ogo:fromSpecies> <uri_Organism1> .
     UNION
      ?gene <ogo:fromSpecies> <uri_Organism2> .
    }
  }
}
```

Figura 12.2: Plantilla SPARQL para la consulta de información sobre ortólogos.

Los resultados de las búsquedas se devuelven en una única página Web, donde la información de los genes ortólogos obtenidos es mostrada en forma de lista. Por ejemplo, la Figura 12.3 muestra la información relacionada con el gen *"ATBF1"*. Los elementos de la descripción se corresponden con los elementos seleccionables para filtrar la información en la interfaz de búsqueda. El nombre y el organismo al que pertenece el gen es la información mínima aportada para cada instancia. El siguiente elemento de la descripción son los sinónimos e identificadores asignados a dicha instancia. A continuación, se

Gene:
ATBF1
Organism:
Homo sapiens

Synonyms
ID:463
ATBT
ZFHX3

Protein Id	**Protein Name**
118498345	NP_008816.3

Resources
Homologene

Disorder
Prostate cancer, susceptibility to [104155]

GO Term	**Evidence Code**	**GO Aspect**
[GO:0000122] negative regulation of transcription from RNA polymerase II promoter	IGI	Biological process
[GO:0006355] regulation of transcription, DNA-dependent	TAS	Biological process
[GO:0005667] transcription factor complex	IDA	Cellular component
[GO:0005622] intracellular	IEA	Cellular component
[GO:0005634] nucleus	TAS	Cellular component
[GO:0008270] zinc ion binding	IEA	Molecular function
[GO:0043565] sequence-specific DNA binding	IEA	Molecular function
[GO:0046872] metal ion binding	IEA	Molecular function
[GO:0016564] transcription repressor activity	IGI	Molecular function
[GO:0003705] RNA polymerase II transcription factor activity, enhancer binding	TAS	Molecular function

Figura 12.3: Ejemplo de descripción del gen ortólogo *"ATBF1"* dado por el método de consulta de ortólogos.

proporcionan el identificador y nombre de la proteína relacionada con el gen, además, es posible seleccionar su identificador para obtener la descripción de la proteína del repositorios de información biológica de donde fue obtenido, como Entrez o Ensembl. A continuación, se indica el repositorio del que se extrajo la información biológica, en este ejemplo es del repositorio de información *Homologene*. La siguiente sección, es la relacionada con las enfermedades genéticas donde se proporciona una descripción de la enfermedad, la cual está relacionada mediante un enlace con la página Web del repositorio OMIM que la describe en más detalle. Por último, se muestra la información de los términos de Gene Ontology, junto con su código de evidencia y la descripción del espacio de nombres al que pertenece el término.

La interfaz de búsqueda orientada a enfermedades genéticas facilita la búsqueda de información, y permite obtener la información de los genes ortólogos asociados con la información genética de dicha enfermedad. A través de la relación de la ontología *"causedBy"*, las instancias de enfer-

medades genéticas integradas en el repositorio semántico están relacionadas con instancias de genes. La interfaz de consulta, es compartida con el método de búsquedas de información sobre genes ortólogos. Sin embargo, desde esta perspectiva no hay definidos filtros ni opciones de búsqueda. El formulario sólo dispone de un cuadro de texto, donde los usuarios introducen la descripción de la enfermedad genética a consultar. La Figura 12.4, muestra un ejemplo de búsqueda de información relacionada con el cáncer de próstata. Los resultados de la búsqueda pueden ser múltiples, por lo que el conjunto de instancias encontradas es devuelto para que el usuario seleccione la que más le interesa.

Figura 12.4: Ejemplo de búsqueda de información de "cáncer de próstata" mediante el método de consulta de enfermedades genéticas.

La Figura 12.5 muestra el ejemplo de la descripción de la enfermedad del cáncer de próstata con el identificador *"104155"* del repositorio OMIM. Para describir una enfermedad se proporciona la información almacenada en el repositorio semántico que está conectada con la instancia de la enferme-

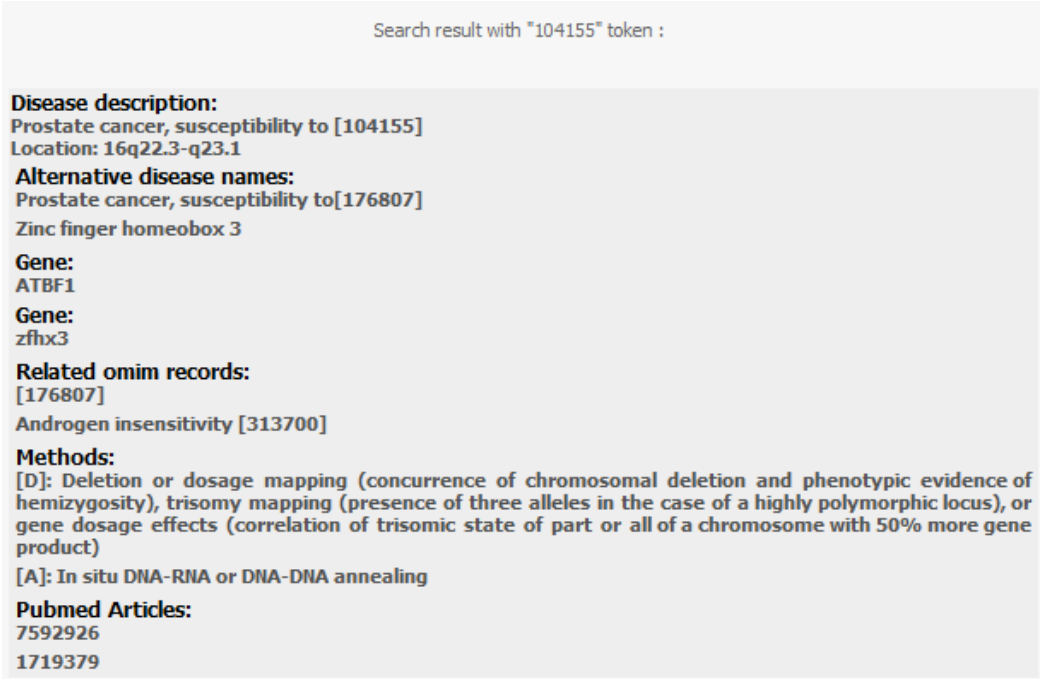

Figura 12.5: Descripción de la instancia de la enfermedad del cáncer de próstata con identificador "104155".

dad. Así, en la descripción se indican: (1) el nombre de la enfermedad; (2) la localización en el cromosoma de la secuencia genética responsable de la enfermedad; (3) los nombres alternativos para identificar la enfermedad; (4) los nombres de los genes relacionados con la enfermedad; (5) los enlaces a otras descripciones de enfermedades genéticas que tienen relación con la enfermedad seleccionada; (6) los métodos por los que se obtuvo la evidencia de la enfermedad; y (7) los enlaces a artículos científicos del repositorio PubMed que hacen referencia a la enfermedad seleccionada.

Para recuperar las instancias de las enfermedades genéticas del repositorio semántico definimos una plantilla SPARQL, para usarla como plantilla de consulta, y así, definir la consulta final del usuario con la información que es proporcionada a través del formulario Web. La Figura 12.6 representa la plantilla SPARQL utilizada en el desarrollo de la interfaz de consulta. La plantilla utiliza las propiedades *"GeneticDisease_identifier"* y *"GeneticDisease_name* para identificar la instancia de la enfermedad genética que el

usuario busca mediante la comparación de esos valores con el *"token"* dado por el usuario. Para permitir la búsqueda de las enfermedades por su nombre, se ha utilizado el elemento *"FILTER"* y *"regex"* del lenguaje SPARQL en la condición *"FILTER regex(?label,token,"i")"* que proporcionan la funcionalidad de usar expresiones regulares en las comparaciones de cadenas de texto. Así, la información de búsqueda proporcionada por los usuarios es utilizada como una expresión regular y comparada con los valores de los nombres de las enfermedades genéticas sin tener en cuenta mayúsculas o minúsculas. Las instancias de las enfermedades genéticas encontradas se recuperan mediante la variable *"?disease"*. Por último, las propiedades y relaciones asociadas con las enfermedades se obtienen a través del modelo de la ontología para proporcionar su descripción.

```
PREFIX ogo:   <http://miuras.inf.um.es/ontologias/OGO.owl#> .
SELECT ?disease
WHERE {
  {
    ?disease <ogo:GeneticDisease_identifier> ‘‘token’’ .
   UNION
    {
      ?disease <ogo:GeneticDisease_name> ?label .
      FILTER regex(?label,‘‘token’’,’’i’’) .
    }
  }
}
```

Figura 12.6: Plantilla SPARQL para la consulta de información sobre enfermedades genéticas.

Esta interfaz de consulta ha sido desarrollada para facilitar la búsqueda de información en el repositorio semántico. La aplicación ha sido diseñada para un uso intuitivo por personas y con una funcionalidad similar a los sistemas de búsqueda de información biológica existentes en la Web, como en los repositorios de Entrez. Sin embargo, este sistema aprovecha las ventajas de la ontología global del Sistema OGO para facilitar la exploración de los datos del repositorio. Ya que la información de las enfermedades genéticas y los grupos de ortólogos está integrada en el mismo repositorio semántico, es posible navegar de una instancia a otra de la ontología, permitiendo, de esta manera, extraer mediante consultas semánticas nuevas relaciones entre

ambos dominios. Por ejemplo, un investigador interesado en buscar la funcionalidad de un gen concreto podría consultar los repositorios de grupos de genes ortólogos en busca de coincidencias y posteriormente buscar en repositorios qué enfermedades genéticas están relacionadas con los genes ortólogos. Con esta interfaz, la búsqueda en los distintos repositorios de ortólogos y su relación con enfermedades genéticas se realiza en un único paso. De esta manera, se reduce el tiempo de búsqueda y se evita al investigador un trabajo manual que no requiere de su supervisión como experto.

12.2. Publicación del repositorio semántico OGO en LOD

Esta sección describe el trabajo relacionado con la publicación del conjunto de datos del repositorio semántico del sistema OGO en la nube LOD, y denominado OGOLOD [228]. La referencia al conjunto de datos OGOLOD[1] fue publicada en el archivo de conjuntos de datos de conocimiento abierto "the Data Hub" [229] de CKAN en la versión de septiembre de 2011. Este conjunto de datos proporciona a investigadores la posibilidad de consultar información anotada sobre genes ortólogos y enfermedades genéticas. Además, esta información está conectada con otros conjuntos de datos en la nube de LOD y que siguen las recomendaciones de Linked Data (LD).

La arquitectura del sistema OGOLOD sigue la definida en la sección 9.2 por la figura 9.3. Así, el servidor Virtuoso gestionará el repositorio RDF. La información del repositorio RDF ha sido obtenida mediante la transformación del repositorio semántico OGO a los principios de LD. También se ha desplegado una instancia de la aplicación Pubby en el servidor *"miuras.inf.um.es"* para la consulta del repositorio RDF mediante una interfaz HTML.

En las siguientes subsecciones, se describirán los detalles del conjunto de datos OGOLOD y de las interfaces de consulta disponibles para su consulta.

[1] `http://thedatahub.org/dataset/ogolod`

12.2.1. Conjunto de datos OGOLOD

La información principal ofrecida por el repositorio OGOLOD es el conjunto integrado de información de genes ortólogos y enfermedades genéticas disponible en el repositorio semántico OGO y cuya ontología global describe el conjunto de datos disponible (véase la Figura 11.1). Además, el conjunto de datos de OGOLOD contiene referencias entre las instancias definidas en él y las instancias de conjuntos de datos externos. También contiene enlaces entre las definiciones de conceptos y propiedades de la ontología del repositorio OGO y otras ontologías externas.

Para obtener el conjunto de datos OGOLOD, primero se tuvo que transformar la información del repositorio semántico de OGO conforme a los principios y recomendaciones de LD, y posteriormente se enlazaron los recursos del conjunto de datos obtenidos con conjuntos de datos externos y publicados en la nube LOD.

A continuación, se describen las transformaciones necesarias para la obtención del conjunto de datos OGOLOD:

- Se modificó el espacio de nombres del modelo ontológico del Sistema OGO *"http://miuras.inf.um.es/ontologias/OGO.owl#"* por *"http://-miuras.inf.um.es/ogolod"*, para evitar el uso del carácter "#".

- Se modificaron los espacios de nombres de las ontologías importadas en la ontología global OGO para que el conjunto de datos a publicar posea el mismo espacio de nombres.

- Se añadieron diferentes etiquetas como prefijo del identificador local de los recursos del conjunto de datos. Por ejemplo, la URI <*ogolod:ontology/GeneticDisease*> se corresponde con la URI del concepto *"GeneticDisease"* de la ontología OGO. Por otro lado, la URI <*ogolod:-resource/GeneticDisease/100070*>, se corresponde con la instancia de una enfermedad genética con identificador *"100070"*.

- Se eliminaron los recursos anónimos definidos en la ontología OGO.

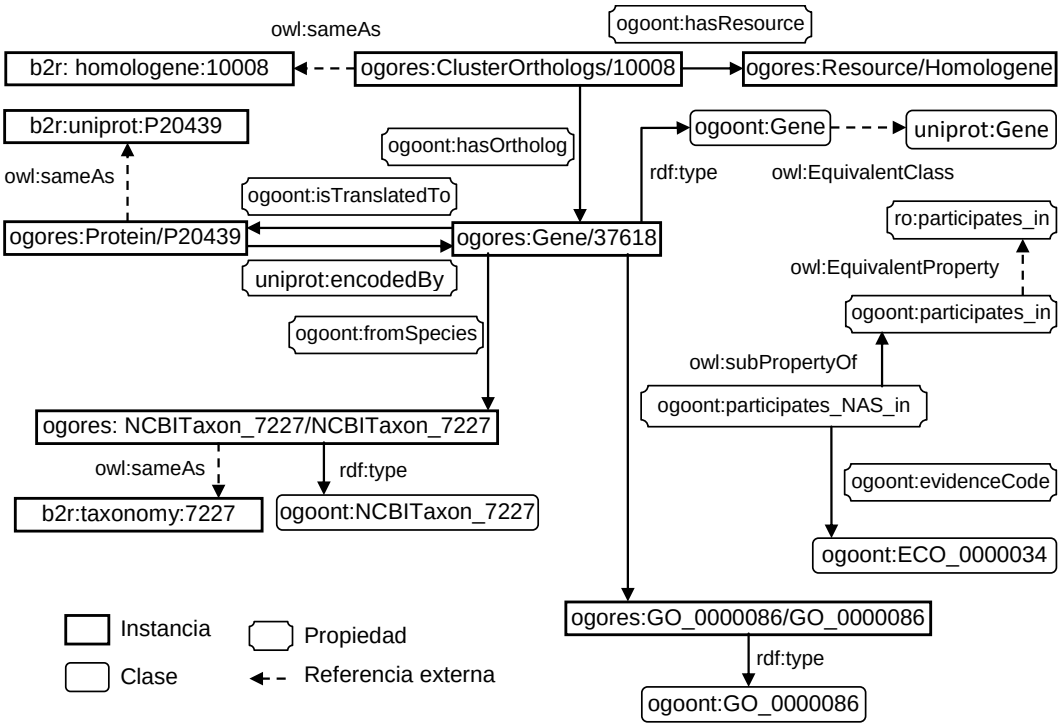

Figura 12.7: Extracto del conjunto de datos de OGOLOD

- Se modificaron las URI de las instancias de los recursos que utilizaban el método *"Punning"* para que éstas se diferenciaran de sus conceptos. Por ejemplo, la URI *<ogo:ontology/NCBI_1>* corresponde al concepto raíz de la taxonomías de las especies se generaron las URI: *<ogolod:ontology/NCBI_1>* para identificar a la clase y *<ogolod:resource/NCBI_1/NCBI_1>* para identificar a la instancia.

Para establecer las referencias con recursos externos ya publicados en la nube LOD, se utilizaron las relaciones *owl:sameAs*, *owl:EquivalentClass* y *EquivalentProperty*. La Figura 12.7 muestra un ejemplo de cómo se han utilizado cada uno de los tipos de enlaces comentados anteriormente.

Por otro lado, para que los datos sean publicados en la nube LOD, éstos deben estar enlazados con los máximos recursos posibles ya publicados en ella. Así, la Tabla 12.1 proporciona ejemplos de los tipos de enlaces a conjuntos de datos externos definidos para el repositorio OGOLOD. Los conjuntos de datos externos referenciados contienen información sobre los repositorios:

Prefix	IRI
ogoont:	<http:miuras.inf.um.es/ogolod/ontology/>
ogores:	<http:miuras.inf.um.es/ogolod/resource/>
uniprot:	<http://purl.uniprot.org/core/>
ro:	<http://www.obofoundry.org/ro/ro.owl#>
b2r:	<http://bio2rdf.org/>
llfd:	<http://linkedlifedata.com/resource/phenotype/>

URI de OGOLOD	URI externa
ogoont:Gene	uniprot:Gene
ogoont:Protein	uniprot:Protein
ogoont:causedBy	ro:has_agent
ogores:ClusterOrthologs/10008	b2r:homologene:10008
ogores:Gene/67440	b2r:page/geneid:67440
ogores:PubmedArticle/15902763	b2r:pubmed:15902763
ogores:GeneticDisease/100070	b2r:omim:100070
ogores:GO_0032283/GO_0032283	b2r:go:0032283
ogores:Protein/Q2IHD7	b2r:uniprot:Q2IHD7
ogores:NCBITaxon_9606/NCBITaxon_9606	b2r:taxonomy:9606
ogores:Protein/285813127	b2r:protein:285813127
ogores:HP_0000082/HP_0000082	llfd:id/HP:0000082

Tabla 12.1: Ejemplos de cada tipo de enlaces a conjuntos de datos externos en OGOLOD

Homologene, *Genes de Entrez*, *Pubmed*, *OMIM*, *Gene Ontology*, *taxonomía de NCBI*, *Uniprot KB*, *Proteínas de Entrez* y *Human Phenotype Ontology*. Estos conjuntos de datos forman parte de las plataformas Bio2RDF y Linked Life Data, asociadas con el dominio biomédico.

12.2.2. Interfaces de consulta de OGOLOD

Las interfaces proporcionadas para consultar el repositorio OGOLOD son tres: interfaz SPARQL, interfaz HTML e interfaz Linked Data (véase la Figura 9.3). La interfaz SPARQL[2] permite realizar consultas SPARQL sobre el repositorio RDF del sistema OGOLOD. El servidor Virtuoso proporciona una implementación del lenguaje SPARQL con diversas extensiones que

[2] `http://miuras.inf.um.es/sparql`

añaden funcionalidades extra al lenguaje. La Figura 12.8 muestra un ejemplo de una consulta definida para la interfaz SPARQL de Virtuoso donde se utiliza la propiedad *TRANSITIVE* para explotar la jerarquía de términos de GO.

```
PREFIX ogolod:<http://miuras.inf.um.es/ogolod/ontology/>
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT DISTINCT ?gene_ID ?relationship ?GO_Name ?eco_Name
WHERE {
  ?gene ogolod:Name "4F9" .
  ?gene ogolod:Identifier ?gene_ID .
  ?gene ?participates_in ?goTerm .
  ?participates_in rdfs:label ?relationship .
  OPTIONAL {
     ?participates_in ogolod:evidenceCode ?eco .
     ?eco rdfs:comment ?eco_Name .
  }
  ?goTerm rdfs:label ?GO_Name .
  ?goTerm rdf:type ?type .
  ?type rdfs:subClassOf ogolod:GO_0003674 OPTION ( TRANSITIVE ) .
}
```

Figura 12.8: Consulta SPARQL para obtener los términos de la Gene Ontology asociados a la función molecular y los códigos de evidencia relacionados con el gen, “4F9”.

La información de las tripletas almacenadas en el repositorio RDF pueden ser exploradas a través de la interfaz HTML proporcionada por Pubby. Los datos se representan en páginas HTML (véase la Figura 12.9), que pueden ser obtenidas mediante navegadores Web. Además, Pubby ofrece la interfaz de Linked Data para recuperar la información de las tripletas RDF del repositorio a través de navegadores RDF. Estos interfaces permiten navegar mediante los enlaces a recursos internos definidos en el repositorio y enlaces externos a recursos de otros repositorios.

12.3. Interfaz de consultas avanzadas con OGO

La interfaz desarrollada utiliza la ontología global OGO para guiar a los usuarios durante el proceso de definición de una consulta semántica. La consulta definida se transforma automáticamente al lenguaje SPARQL para

at **miuras.inf.um.es**

http://miuras.inf.um.es/ogolod/resource/GeneticDisease/100070

Property	Value
ogolodonto:Identifier	100070
ogolodonto:Location	19q13
ogolodonto:Name	AORTIC ANEURYSM, FAMILIAL ABDOMINAL 1
ogolodonto:causedBy	<http://miuras.inf.um.es/ogolod/resource/Gene/100329167>
ogolodonto:connectedTo	<http://miuras.inf.um.es/ogolod/resource/GeneticDisease/122455> <http://miuras.inf.um.es/ogolod/resource/GeneticDisease/173360> <http://miuras.inf.um.es/ogolod/resource/GeneticDisease/601046> <http://miuras.inf.um.es/ogolod/resource/GeneticDisease/601158> <http://miuras.inf.um.es/ogolod/resource/GeneticDisease/604312> <http://miuras.inf.um.es/ogolod/resource/GeneticDisease/607086> <http://miuras.inf.um.es/ogolod/resource/GeneticDisease/609781> <http://miuras.inf.um.es/ogolod/resource/GeneticDisease/609782> <http://miuras.inf.um.es/ogolod/resource/GeneticDisease/611891>
ogolodonto:hasMethod	<http://miuras.inf.um.es/ogolod/resource/Method/A> <http://miuras.inf.um.es/ogolod/resource/Method/D> <http://miuras.inf.um.es/ogolod/resource/Method/Fd> <http://miuras.inf.um.es/ogolod/resource/Method/H> <http://miuras.inf.um.es/ogolod/resource/Method/REa> <http://miuras.inf.um.es/ogolod/resource/Method/REb> <http://miuras.inf.um.es/ogolod/resource/Method/REn>
ogolodonto:hasPhenotype_IEA	<http://miuras.inf.um.es/ogolod/resource/HP_0001455/HP_0001455> <http://miuras.inf.um.es/ogolod/resource/HP_0004942/HP_0004942> <http://miuras.inf.um.es/ogolod/resource/HP_0004953/HP_0004953>
ogolodonto:relatedPubmedArticle	<http://miuras.inf.um.es/ogolod/resource/PubmedArticle/10558909> <http://miuras.inf.um.es/ogolod/resource/PubmedArticle/10805895> <http://miuras.inf.um.es/ogolod/resource/PubmedArticle/12563204> <http://miuras.inf.um.es/ogolod/resource/PubmedArticle/15096456> <http://miuras.inf.um.es/ogolod/resource/PubmedArticle/15156361> <http://miuras.inf.um.es/ogolod/resource/PubmedArticle/15944607> <http://miuras.inf.um.es/ogolod/resource/PubmedArticle/16311603> <http://miuras.inf.um.es/ogolod/resource/PubmedArticle/1642543> <http://miuras.inf.um.es/ogolod/resource/PubmedArticle/19497516> <http://miuras.inf.um.es/ogolod/resource/PubmedArticle/1985458> <http://miuras.inf.um.es/ogolod/resource/PubmedArticle/2787414> <http://miuras.inf.um.es/ogolod/resource/PubmedArticle/3047388> <http://miuras.inf.um.es/ogolod/resource/PubmedArticle/3418662> <http://miuras.inf.um.es/ogolod/resource/PubmedArticle/3761500> <http://miuras.inf.um.es/ogolod/resource/PubmedArticle/3855680> <http://miuras.inf.um.es/ogolod/resource/PubmedArticle/4038055> <http://miuras.inf.um.es/ogolod/resource/PubmedArticle/588966> <http://miuras.inf.um.es/ogolod/resource/PubmedArticle/6538765> <http://miuras.inf.um.es/ogolod/resource/PubmedArticle/6729702> <http://miuras.inf.um.es/ogolod/resource/PubmedArticle/6845177> <http://miuras.inf.um.es/ogolod/resource/PubmedArticle/7563401> <http://miuras.inf.um.es/ogolod/resource/PubmedArticle/7613891> <http://miuras.inf.um.es/ogolod/resource/PubmedArticle/7651004> <http://miuras.inf.um.es/ogolod/resource/PubmedArticle/7956205> <http://miuras.inf.um.es/ogolod/resource/PubmedArticle/8048013> <http://miuras.inf.um.es/ogolod/resource/PubmedArticle/8455684> <http://miuras.inf.um.es/ogolod/resource/PubmedArticle/9786252>
owl:sameAs	<http://bio2rdf.org/page/omim:100070>
rdf:type	ogolodonto:GeneticDisease

This page shows information obtained from the SPARQL endpoint at http://miuras.inf.um.es/sparql.
As Turtle | As RDF/XML | Browse in Disco | Browse in Tabulator | Browse in OpenLink Browser

Figura 12.9: Ejemplo de página HTML obtenida a través de la interfaz HTML del sistema OGOLOD. Esta interfaz puede ser utilizada para navegar a través del conjunto de datos del repositorio vía relaciones internas, como *ogolodonto:causedBy*, o externas, como *owl:sameAs*

consultar el repositorio semántico OGO.

A continuación, se describe el proceso de definición de una consulta a través de esta interfaz en un ejemplo: "un usuario quiere conocer los genes pertenecientes a la especie *Rattus norvegicus* que sean ortólogos de los genes humanos responsables del cáncer de próstata". Esta información puede ser útil para obtener genes candidatos de estudio responsables del cáncer de próstata en ratas.

La Figura 12.10 muestra la ventana principal de la interfaz de consulta avanzada con la definición de la consulta de ejemplo. Esta ventana está compuesta por dos cuadros de texto, que contienen la definición del ejemplo de consulta, y seis botones, para introducir los parámetros de búsqueda. Los nombres de los elementos de la consulta se obtienen de las etiquetas *"label"* asociadas a los conceptos, propiedades y relaciones definidas en la ontología. Para definir las variables de consulta *Disorder[1]*, asociada con las instancias de enfermedades genéticas, y *Gene[1]*, asociada a las instancias de genes en el repositorio semántico, se utiliza el botón *"Select Concept"*. Así, la interfaz muestra la jerarquía de conceptos de la ontología OGO y el usuario debe seleccionar los conceptos *Disorder* y *Gene* para crear cada variable (véase la Figura 12.11).

El cuadro de texto etiquetado con *"Query Requirements"* contiene las condiciones de la consulta. Para añadir nuevas condiciones, se utiliza el botón *"Add new requirements"*, de modo que la vista muestra la lista de posibles condiciones que se pueden asociar a las variables definidas (véase la Figura 12.12). Las condiciones mostradas dependen del dominio y rango de las propiedades y relaciones, y de las restricciones definidas en la ontología OGO. Así, mediante las sugerencias del sistema, el usuario es guiado para definir correctamente las condiciones.

Por último, la Figura 12.13 muestra el resultado de la consulta definida en la Figura 12.10, donde la primera columna representa a la variable de las enfermedades genéticas, *"Disorder[0]"*, y la segunda columna representa a la variable de los genes, *"Gene[1]"*.

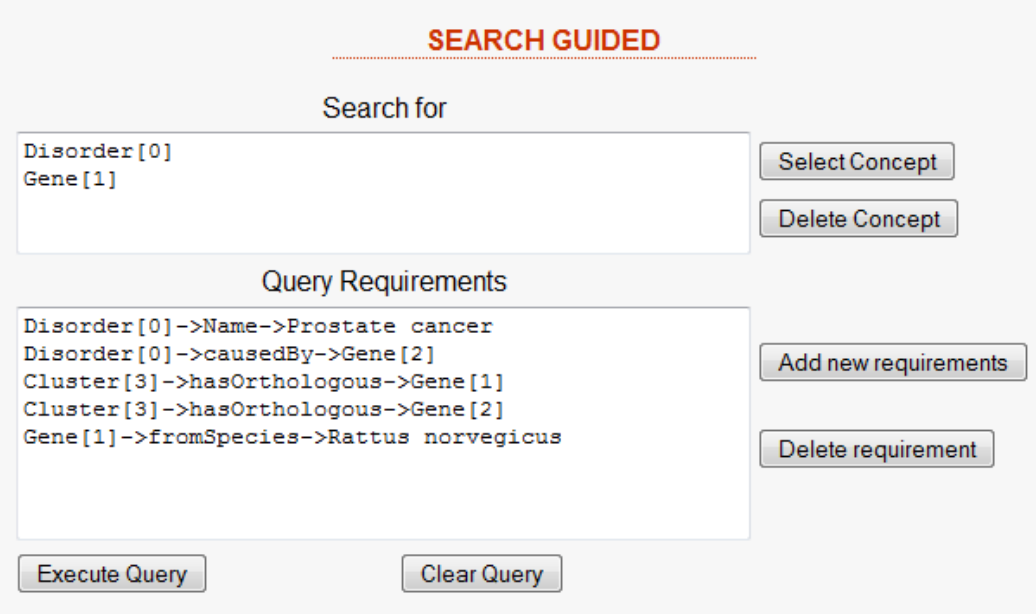

Figura 12.10: Ejemplo de consulta definida a través de la interfaz de consultas avanzadas.

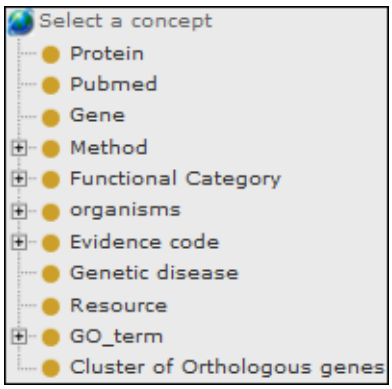

Figura 12.11: Jerarquía de conceptos de la ontología OGO mostrada para seleccionar las variables de la interfaz de consultas avanzadas.

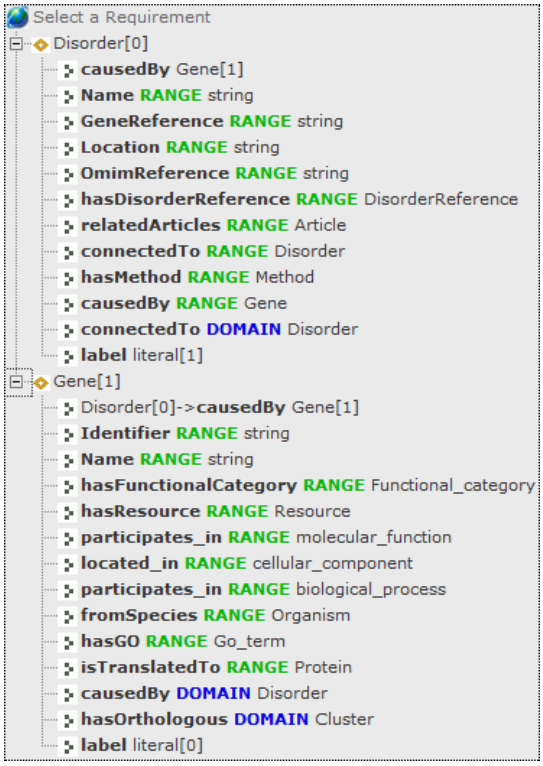

Figura 12.12: Ejemplo de los distintos requisitos disponibles para las variables definidas a través de la interfaz de consultas avanzadas.

Disorder[0]	Gene[1]
104155, prostate cancer, susceptibility to, zinc finger homeobox 3	pex12, 116718
	rgd1560268, zfhx3.predicted, 307829
	rgd1563022, zfhx4.predicted, 310250
600020, neurofibrosarcoma, prostate cancer, susceptibility to, max-interacting protein 1	clec5a, 679787
	ensrnog00000026306, loc684510
	loc689617, 689617
	loc689629, 689629
	max, mgc124611, 60661
	mxi.wr, mxi1, 25701
	tgap1, 294892

Figura 12.13: Resultado de la consulta de la Figura 12.10.

12.4. Servicios Web para la explotación del sistema OGO

Esta sección presenta los servicios Web desarrollados para la consulta del repositorio semántico de OGO. El objetivo de estos servicios es ofrecer una interfaz para que aplicaciones independientes puedan explotar el contenido del repositorio semántico de manera autónoma. Por lo tanto, esta interfaz permitirá definir consultas y devolver datos de manera automática a aplicaciones independientes mediante la utilización de los diferentes servicios Web desarrollados.

Estos servicios fueron desarrollados como parte del trabajo de integración [230] con el sistema ONCOdata, pero pueden ser reutilizados por cualquier otro sistema. ONCOdata es una herramienta comercial para la recomendación de tratamientos oncológicos. Desde un punto de vista técnico, ONCOdata es un sistema de ayuda a la decisión que utiliza el conocimiento presente en guías clínicas sobre tratamientos del cáncer y los datos de los pacientes, para inferir los tratamientos más adecuados durante las distintas fases de evolución de la enfermedad. La Figura 12.14 muestra el escenario donde ambos sistemas intercambian mensajes para permitir la consulta del repositorio semántico a través de una relación cliente-servidor, donde el sistema ONCOdata es el cliente que solicita una consulta y el sistema OGO es el servidor que ejecuta la consulta y devuelve los datos al cliente.

Para implementar la interfaz de consulta de la información del repositorio semántico OGO, se a hecho uso de servicios Web, ya que proporcionan una solución de máxima interoperabilidad entre sistemas con un mínimo acoplamiento. Además, permiten que la aplicación que hace de servidor en la comunicación sea independiente del sistema OGO, por lo que facilita su diseño e implementación. Los servicios Web se basan en estándares y protocolos abiertos como WSDL, SOAP, HTTP y XML, que facilitan el intercambio de datos entre diferentes aplicaciones, independientemente de su ubicación geográfica. La adopción de estos estándares reduce el tiempo, el coste de desarrollo y despliegue de aplicaciones porque los sistemas involucrados son independientes de lenguaje de implementación utilizado y el entorno de eje-

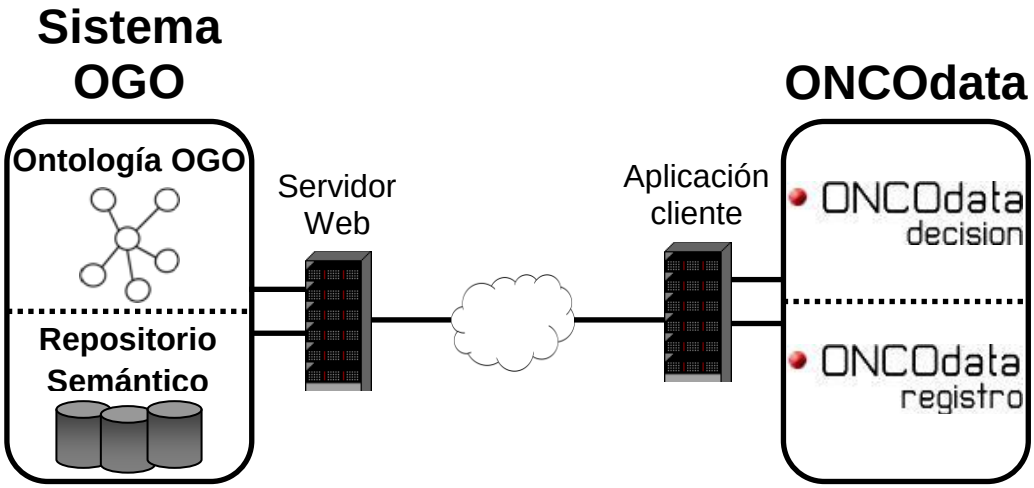

Figura 12.14: Escenario de la integración del Sistema OGO con ONCOdata.

cución de su plataforma. Otra ventaja de los servicios Web es la codificación de los mensajes en formato texto, ya que facilita la legibilidad de los mismos y su procesamiento.

Desde un punto de vista técnico, se ha empleado el estándar XML para la codificación de los mensajes a intercambiar, facilitando así el procesamiento de su información por las aplicaciones mediante el uso de analizadores XML. Los mensajes se intercambian a través de la Web mediante el protocolo de comunicación HTTP, y se utiliza el estándar SOAP para describir los formatos de los mensajes de los distintos servicios Web. Por último se han definido los documentos WSDL correspondientes con las descripciones de la gramáticas de los servicios Web definidos para el intercambio de mensajes.

En concreto, se han definido tres tipos diferentes de servicios Web para acceder a la información presente en el sistema OGO. La Figura 12.15 muestra los distintos métodos definidos en las interfaces de los servicios Web desplegados para intercambiar datos entre el sistema OGO y otras aplicaciones independientes.

1. Servicio Web enfocado en la consulta de información relacionada con las enfermedades genéticas humanas.

2. Servicio Web especializado en la consulta de información desde la pers-

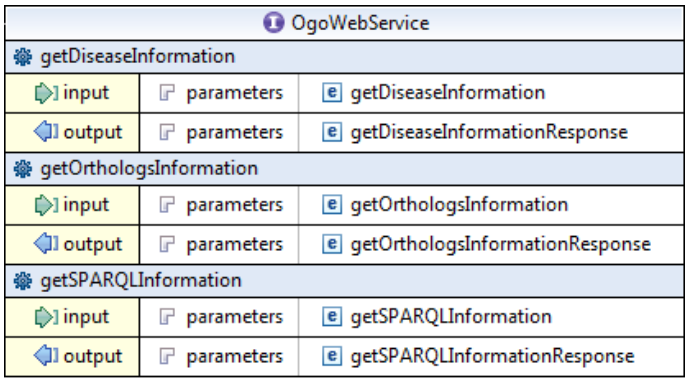

Figura 12.15: Interfaces de los servicios Web desplegados para la consulta del sistema OGO.

pectiva de los grupos de genes ortólogos.

3. Servicio Web que define una interfaz para la utilización del lenguaje SPARQL en la realización de consultas sobre el repositorio semántico.

Los dos servicios Web especializados en la consulta de información sobre enfermedades genéticas y sobre grupos de genes ortólogos están implementados de manera que el uso de SPARQL para consultar los datos integrados en el sistema OGO sea transparente al cliente de los servicios. Así, para consultar la información sobre enfermedades genéticas, se requiere el paso del parámetro con el nombre de la enfermedad, mientras que para consultar la información relacionada con grupos de genes ortólogos se necesita el identificador del gen. Para obtener la información del repositorio semántico, cada servicio Web tiene definido una plantilla con el esquema de consulta SPARQL adecuado. Las plantillas 12.2 y 12.6 descritas en la sección 12.1 son las utilizadas para obtener la información sobre genes ortólogos y enfermedades genéticas, respectivamente. Por otro lado, la combinación de los resultados de un servicio con el otro permite explotar las relaciones entre los datos para la investigación traslacional. Sin embargo, el tercer servicio Web permite pasar como parámetro la definición de una consulta SPARQL, siendo responsable el usuario de su correcta definición. Este servicio permite una mayor libertad para la definición de consultas avanzadas, pero requiere de mayor conocimiento del lenguaje de consultas y del modelo ontológico

utilizado en el repositorio semántico.

La Figura 12.16 muestra un extracto del documento WSDL donde se indican los parámetros y los métodos para utilizar el servicio Web relacionado con la consultas sobre enfermedades genéticas. Este documento describe los detalles de los métodos para intercambiar los mensajes, sus parámetros y tipos de datos, así como la localización del servicio, *"http://miuras.inf.um.es:9080/-OgoWS/services/OgoDisease"*. Por lo tanto, con la información presente en el documento WSDL, la aplicación cliente puede definir los mensajes SOAP, qué métodos utilizar para obtener las respuestas y dónde localizar el servicio Web.

Por otro lado, la Figura 12.17 muestra un ejemplo del resultado de una búsqueda de información sobre la enfermedad del cáncer de mama. La estructura del documento XML resultado contiene la información de las distintas propiedades y relaciones asociadas a cada instancia de las enfermedades encontradas en el repositorio con la descripción de la enfermedad proporcionada por el usuario. Así, para las enfermedades genéticas, se utiliza la etiqueta *"<geneName>"* con el fin de obtener la información de la propiedad de la ontología *"<ogo:gene_name>"* asociada al gen asociado a la enfermedad a través de la relación *"<ogo:causedBy>"*; la etiqueta *"<location>"* contiene el valor de la propiedad *"<ogo:location>"*; la etiqueta *"<name>"* contiene el valor de la propiedad *"<ogo:geneticDisease_name>"*; la etiqueta *"<method>"* describe el método relacionado con la obtención de información sobre la enfermedad y asociado a ésta a través de la relación *"<ogo:hasMethod>"*; y la etiqueta *"<pubmed>"* contiene el valor de la propiedad *"<PubmedArticle_identifier>"* de la instancia de artículos del repositorio PubMed relacionados con la instancia de la enfermedad mediante la relación *"<ogo:relatedPubmedArticle>"*.

```
<?xml version=''1.0'' encoding=''UTF-8''?>
<wsdl:definitions targetNamespace=''http://model''>
 <wsdl:types><schema elementFormDefault="qualified">
   <element name="getDiseaseInformation"><complexType>
    <sequence><element name="diseaseName" type="xsd:string"/></sequence>
   </complexType></element>
   <element name="getDiseaseInformationResponse"><complexType>
    <sequence><element name="getDiseaseInformationReturn" type="xsd:string"/></sequence>
   </complexType></element>
 </schema></wsdl:types>
 <wsdl:message name="getDiseaseInformationRequest">
  <wsdl:part element="impl:getDiseaseInformation" name="parameters"/>
 </wsdl:message>
 <wsdl:message name="getDiseaseInformationResponse">
  <wsdl:part element="impl:getDiseaseInformationResponse" name="parameters"/>
 </wsdl:message>
 <wsdl:portType name="OgoDisease"><wsdl:operation name="getDiseaseInformation">
  <wsdl:input message="impl:getDiseaseInformationRequest"
   name="getDiseaseInformationRequest"/>
  <wsdl:output message="impl:getDiseaseInformationResponse"
   name="getDiseaseInformationResponse"/>
 </wsdl:operation></wsdl:portType>
 <wsdl:binding name="OgoDiseaseSoapBinding" type="impl:OgoDisease">
  <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="getDiseaseInformation">
   <wsdlsoap:operation soapAction=""/>
   <wsdl:input name="getDiseaseInformationRequest">
    <wsdlsoap:body use="literal"/></wsdl:input>
   <wsdl:output name="getDiseaseInformationResponse">
    <wsdlsoap:body use="literal"/></wsdl:output>
  </wsdl:operation>
 </wsdl:binding>
 <wsdl:service name="OgoDiseaseService">
  <wsdl:port binding="impl:OgoDiseaseSoapBinding" name="OgoDisease">
   <wsdlsoap:address location="http://miuras.inf.um.es:9080/OgoWS/services/OgoDisease"/>
  </wsdl:port>
 </wsdl:service>
</wsdl:definitions>
```

Figura 12.16: Extracto del documento WSDL que define el servicio Web para consultar información sobre enfermedades genéticas.

```
<?xml version='1.0' encoding='iso-8859-1'?>
<reply>
 <disease>
  <geneName>SH2D3B</geneName>
  <geneName>NSP2</geneName>
  <location>1p22.1</location>
  <name>breast cancer anti-estrogen resistance 3</name>
  <pubmed>9582273</geneName>
 </disease>
 <disease>
  <geneName>DKFZP564A063</geneName>
  <geneName>brms1</geneName>
  <location>11q13.1-q13.2</location>
  <name>breast cancer metastasis suppressor 1</name>
  <method>In situ DNA-RNA or DNA-DNA annealing</method>
  <pubmed>10850410</geneName>
 </disease>
```

Figura 12.17: Ejemplo de resultado de una consulta sobre la enfermedad del cáncer de mama.

El servicio Web relacionado con la obtención de la información sobre los grupos de genes ortólogos es similar a la descripción realizada del servicio asociado a las enfermedades genéticas. En este caso, el resultado de la consulta es el grupo de ortólogos que contiene la información del gen pasado por parámetro. En concreto, el resultado contiene la descripción de las propiedades y relaciones de los genes asociados al grupo. Por último, el servicio Web asociado a las consultas SPARQL recibe por parámetro la definición de la consulta y devuelve la información obtenida en un documento XML. El documento resultado consiste en un listado de elementos que contienen los valores asociados a las variables utilizadas en la consulta.

Bloque IV

Contribuciones, conclusiones y trabajo futuro

Capítulo 13

Contribuciones, conclusiones y trabajo futuro

13.1. Contribuciones

Esta tesis ha tenido como objetivo el desarrollo de un entorno para la gestión de información biológica y biomédica, con el cual se genere un repositorio semántico que permita explotar la información y conocimiento almacenados y facilite la investigación traslacional. Los resultados obtenidos durante la tesis y su validación con el Sistema OGO demuestran el potencial del repositorio semántico respecto de los repositorios bioinformáticos tradicionales.

A continuación se enumerarán las principales aportaciones del trabajo realizado:

- Definición de una metodología para la integración de repositorios bioinformáticos en un repositorio semántico. Dicho repositorio estará basado en la especificación de una ontología global, donde los datos integrados tendrán asociados conceptos, relaciones y propiedades, y que facilitarán la comprensión del repositorio a usuarios y aplicacionesa autónomas.

- Diseño e implementación de un método para la integración de repositorios relacionales a un repositorio semántico. Este método estará dirigido a través de la definición de reglas de correspondencia y de integración,

y una ontología global que conceptualice el conocimiento de los repositorios relacionales. La definición de estas reglas permitirán establecer la relación entre los repositorios relacionales y una ontología global. Además, permitirán reducir la redundancia de información en el proceso de integración entre las distintas fuentes de información. Como resultado de este método, se obtiene un repositorio integrado a nivel semántico.

- Diseño e implementación de una interfaz de definición de consultas avanzadas que permite explotar el conocimiento de la ontología global del repositorio semántico. Esta interfaz guiará a los usuarios en la definición de las consultas semánticas avanzadas. Además, evitará que los usuarios cometan errores sintácticos asociados al lenguaje SPARQL, proporcionará las relaciones con las que asociar los términos de la consulta y mostrará la información ontológica necesaria para la definición de la consulta.

- Definición de las transformaciones necesarias para la publicación y relación de datos, de un repositorio semántico basado en ontologías OWL, en la nube LOD.

- Definición de una bio-ontología que representa el conocimiento de manera global de los dominios de los genes ortólogos y de las enfermedades genéticas humanas. Esta ontología establecerá una relación entre los conceptos de los distintos repositorios, facilitando así, la investigación traslacional.

- Diseño e implementación de una interfaz de consulta mediante formularios Web para obtener información de un repositorio semántico. Esta interfaz permite realizar consultas a los usuarios a partir de los parámetros y filtros de consulta proporcionados. Estos datos se utilizan posteriormente para la definición de una consulta SPARQL de manera automática con la que obtener la información adecuada.

- Diseño e implementación de servicios Web para la consulta de un repositorio semántico por agentes software autónomos, permitiendo la

consulta de la información mediante el lenguaje SPARQL o mediante los métodos de consulta establecidos.

13.2. Conclusiones y trabajo futuro

Las disciplinas biológicas y biomédicas demandan requieren cada vez más de herramientas software que faciliten a los investigadores la gestión y relación de la información que generan. Además, esta información está caracterizada por su gran complejidad, heterogeneidad y volumen. La informática, como disciplina transversal, es aplicada cada vez más para dar soluciones avanzadas a las necesidades computacionales de las citadas disciplinas. Esto ha dado lugar al desarrollo de la bioinformática y la informática médica.

La evolución de los sistemas hardware y software actuales, así como el desarrollo de nuevas tecnologías como la Web semántica, permitirán el diseño e implementación de sistemas de gestión avanzada de grandes cantidades de información biológica y biomédica. Estos sistemas reducirán el tiempo que expertos del dominio deben utilizar en tareas automatizables, como la comparación o búsqueda de información entre distintos repositorios bioinformáticos.

Conseguir que la información biológica y biomédica almacenada en repositorios bioinformáticos esté representada de manera que pueda ser analizada y explotada por aplicaciones software de manera autónoma es esencial para el desarrollo de nuevas herramientas que procesen dicha información y faciliten el trabajo de los investigadores. Con este objetivo, las bio-ontologías pueden ser consideradas como la pieza central que describa el conocimiento asociado a los repositorios bioinformáticos y que proporcionen un vocabulario común con el que facilitar el procesamiento e interpretanción de la información publicada.

Una de las aportaciones más importantes de los modelos ontológicos es la definición de las relaciones entre conceptos que permiten la exploración del modelo y su descubrimiento por aplicaciones autónomas. Las instancias con las que se pueblas dichos modelos están asociados a diversas clases que permiten su clasificación y consulta desde diferentes perspectivas de mane-

ra más intuitiva. Esta flexibilidad puede ser explotada por investigaciones traslacionales para la extracción de asociaciones implícitas en el modelo de información. Además, la reutilización del conocimiento presente en las bio-ontologías ya definidas (véase la sección 11.1) permite mejorar la calidad de los modelos ontológicos de los sistemas semánticos que las importan.

Para la integración de repositorios bioinformáticos se ha definido en esta tesis una metodología de integración semántica. Dicha metodología requiere la definición de la ontología global que relacione las descripciones conceptuales de los distintos repostorios entre sí. La ontología global facilitará el proceso de integración automático y la explotación del repositorio semántico en la que se integrará la información. Así, la metodología tiene como objetivo obtener un repositorio semántico que utilice la ontología global para proporcionar el conocimiento de cómo la información está integrada de manera que permita explotar su contenido. Esta metodología es genérica y puede ser aplicada a otros dominios de información o puede servir para extender el contenido de un repositorio semántico con otros repositorios existentes.

Para poder aplicar la metodología de integración definida fue necesario diseñar e implementar un proceso automático que permita integrar la información de distintos repositorios bioinformáticos. Para la definción del método de integración se afrontaron diversos problemas. Por un lado, los métodos de extracción de información de los repositorios existentes están diseñados para su uso por exclusivo por personas, es decir, la información no están representadas de manera que puedan ser procesadas automáticamente, y además, utilizan claves no compartidas para la identificación de la información que almacenan.

Por otro lado, la información disponible en los repositorios está representada normalmente utilizando el lenguaje natural y documentos semi-estructurados. Dicha información no contiene referencias a otros elementos de repositorios diferentes que faciliten su integración. Por esta razón, para poder definir un método genérico de integración de información entre repositorios bioinformáticos, es necesario realizar un pre-procesamiento de la información para que ésta esté disponible de manera más eficiente para su integración y se puedan extraer relaciones entre repositorios. Con todo, el método de

integración semántica de información desarrollado en esta tesis parte de que la información esté representada en repositorios relacionales, para que las correspondencias entre los repositorios y la ontología global puedan ser definidas de manera uniforme. Por lo tanto, para utilizar la herramienta implementada los repositorios bioinformáticos deben estar representados en bases de datos relacionales. Como trabajo futuro se podría flexibilizar el formato en que los repositorios bioinformáticos deben estar para poder ser integrados en el repositorio semántico.

El método de integración semántica implementado se basa en la definición de tres tipos reglas de correspondencia para la asociación entre los valores de las tablas de los repositorios fuente y los recursos de la ontología. En concreto, cada tipo de regla tiene como objetivo establecer las correspondencias entre las columnas de las tablas y los conceptos, relaciones y propiedades de la ontología asociada.

Por otro lado, el método de integración utiliza reglas de identidad para el descubrimiento de instancias equivalentes en el repositorio semántico que genera durante el proceso de integración. Esta funcionalidad permite la descripción de las identidades de las instancias a través de sus propiedades y relaciones asociadas a su concepto correspondiente en la ontología global. Esto permite describir de manera más avanzada la identidad de las instancias que mediante el uso de un identificador o URI en la ontología. Los sistemas de publicación e integración estudiados no definen de manera formal esta funcionalidad, por lo que la asociación entre instancias equivalentes se establecía manualmente, a menos que compartieran identificadores comunes. Cuando se puebla una ontología con la información de un solo repositorio, las claves de los repositorios son únicas, pero al integrar información de varios repositorios, es necesario establecer métodos más avanzados de identificación. Se ha demostrado en el trabajo de esta tesis que la definición de las reglas de identidad permiten lograr este objetivo.

Los repositorios semánticos generados utilizando la metodología de integración definida están preparados para poder ser explotados como modelos ontológicos o como almacenes RDF. Un modelo ontológico asociado al repositorio semántico puede ser explotado mediante los conceptos, las instancias,

las relaciones y las propiedades definidas en la ontología. Así, la información es accedida desde un nivel superior de abstracción y, por lo tanto, facilita su gestión. Sin embargo, el repositorio semántico accedido como conjunto de tripletas RDF permite consultar y obtener la información de manera más eficiente. Así, ya que el uso del repositorio semántico como un almacén RDF no permite generar el modelo ontológico asociado, la información debe ser extraída utilizando las URI de los datos o sus tripletas RDF. La consulta de información puede ser realizada a través de lenguajes semánticos, como SPARQL. Sin embargo, para definir las consultas en este tipo de lenguajes es necesario que los usuarios tengan conocimientos de la sintaxis y gramática correspondientes, además de los conocimientos del grafo RDF utilizado para representar la información.

Para validar la metodología y herramientas desarrolladas, éstas se han aplicado en la construcción del Sistema OGO. Este sistema se basa en un repositorio semántico que relaciona información de grupos de genes ortólogos con información sobre enfermedades genéticas humanas para facilitar la investigación traslacional. Para generar dicho repositorio se utilizó la metodología y herramientas de integración semánticas definidas para integrar diversos repositorios bioinformáticos. La integración de repositorios bioinformáticos del Sistema OGO se realizó en dos fases. En la primera versión del Sistema OGO, sólo se integraron los datos de los repositorios relacionados con los grupos de genes y proteínas ortólogas. En la segunda versión, se extendió el Sistema OGO para integrar la información sobre enfermedades genéticas humanas.

La ontología global del Sistema OGO facilita la integración semántica de la información. Esta ontología representa el conocimiento del dominio de los genes ortólogos y enfermedades genéticas humanas y además proporciona una representación formal y no ambigua de dichos dominios de información. Así, ésta permite explotar, mediante razonadores, la semántica asociada a los repositorios bioinformáticos correspondientes. Esta funcionalidad ofrece la posibilidad de verificar la consistencia de la definición de la ontología y de la información que almacena. Sin embargo, existe una limitación práctica al realizar tareas de inferencia con grandes cantidades de información. Los

razonadores disponibles, en el momento de redacción de esta tesis, demandan grandes cantidades de memoria virtual y tiempo de procesamiento que limitan las funcionalidades que proporcionan.

Para la consulta de la información del Sistema OGO se han desarrollado tres tipos de interfaces diferentes. Estas interfaces han sido diseñadas para ofrecer acceso a la información del repositorio semántico para personas y aplicaciones.

El primer tipo de interfaz de consulta fue diseñado para facilitar la consulta de la información del repositorio a los usuarios. La información del repositorio puede ser consultada a través de dos perspectivas diferentes, una mediante la perspectiva de los grupos de genes ortólogos y otra mediante la información de las enfermedades genéticas. Esta interfaz define un formulario Web por cada perspectiva para que los usuarios introduzcan los parámetros y filtros de búsqueda. La información introducida por los usuarios en los formularios Web es utilizada para definir, de manera automática, las consultas SPARQL correspondientes. Con las consultas SPARQL definidas se obtendrá la información adecuada del repositorio. Los resultados de las consultas serán devueltos a los usuarios en una página Web donde a través de enlaces pueda navegarse entre las instancias del repositorios semántico encontradas.

El segundo tipo de interfaz de consulta fue diseñado para proporcionar a los usuarios un método que les facilitará la definición avanzada de consultas sobre un repositorio semántico. Esta interfaz de consulta podría ser aplicada a cualquier tipo de repositorio semántico que ofrezca una interfaz SPARQL y utilice un modelo ontológico para almacenar la información. Los fallos sintácticos y gramaticales relacionados con la definición de consultas en el lenguaje SPARQL son evitados ya que la interfaz guiará el proceso de definición de éstas. Es decir, durante el proceso de definición de las consultas, la interfaz ofrecerá a los usuarios las posibles relaciones entre los conceptos con las que definir las condiciones de consulta. De esta manera, se reduce la complejidad y se facilita la búsqueda de información con SPARQL. Debido a que la interfaz restringe la definición de las consultas, los usuarios deberán introducir las condiciones mediante menús, listas y botones ofrecidas por esta

interfaz. Así, la definición de una consulta mediante esta interfaz requiere de más tiempo, en caso de usuarios expertos, que con las interfaces de consulta SPARQL tradicionales, donde los usuarios definen las consultas manualmente. Las funcionalidades de esta interfaz han sido verificadas mediante su uso en el Sistema OGO.

El tercer tipo de método de acceso ha sido diseñado para la consulta y explotación de la información del repositorio OGO por aplicaciones autónomas a través de servicios Web. En concreto se han definido tres servicios Web con los que acceder a la información. El primer y segundo tipo de servicios Web definen métodos de consulta a través de parámetros con la misma funcionalidad que en las dos perspectivas diferentes de la primera interfaz de consulta descrita. El servidor Web utiliza los parámetros de los métodos asociados a cada servicio para definir las consultas SPARQL con las que obtener la información del repositorio. Sin embargo, el tercer tipo de servicio se basa en la definición de la consulta SPARQL por parte de la aplicación cliente del servicio, de manera que se permite la definición de consultas avanzadas por otras aplicaciones. La información recuperada es devuelta a los usuarios a través de documentos XML, utilizando los nombres de las variables de consulta como etiquetas para anotar la información. Este tipo de interfaz ha sido verificado mediante su utilización por el sistema ONCOdata.

Dotar a la Web de una mayor semántica para facilitar el acceso a la información por parte de aplicaciones software es uno de los objetivos más importantes de la Web semántica. Con este fin se realizó la publicación de la información del repositorio semántico OGO en la nube LOD. Dicho proyecto de publicación de información se le ha denominado OGOLOD. Este proyecto permite acceder y navegar, a través de un explorador o mediante una interfaz SPARQL, entre la información generada en el repositorio semántico OGO. Por ejemplo, si se indica en un navegador HTML la URI de un recursos creado en OGOLOD, éste devuelve todas las propiedades y relaciones asociadas al recurso indicado. Además, la información del repositorio está enlazada con otros conceptos equivalentes y publicados, permitiendo de esta manera relacionarlo con otros conjuntos de datos de la nube LOD. Para publicar la información del repositorio OGO, éste tuvo que ser adaptado

y transformado, ya que algunas de las características de OWL no cumplen con los requisitos para su publicación en LOD. Esta adaptación ha obligado a reducir la semántica del Sistema OGO en OGOLOD. Por otro lado, las consultas definidas en SPARQL no son capaces de utilizar ningún tipo de inferencia, por lo que la explotación de la semántica es también más reducida. Por lo tanto, este proyecto viene a proporciona un conjunto de datos nuevo en LOD para la evolución de la Web actual a la Web semántica. Una de las características de la construcción de OGOLOD es la definición manual de los tipos de enlaces entre conjuntos de datos externos que provoca que estos puedan quedar incoherentes con el paso del tiempo. Esto implica que es responsabilidad de cada gestor mantener actualizado el estado de los enlaces a otros conjuntos de datos que publica y que la exploración de los distintos conjuntos de datos no está exenta de errores. Sin embargo, facilita la gestión y la creación de nuevo contenido.

Las limitaciones actuales de las herramientas y tecnologías de la Web semántica impiden que se puedan aprovechar las funcionalidades de inferencia cuando se maneja una gran cantidad de información. Por lo tanto, la funcionalidad de inferencia con el repositorio semántico OGO queda restringida. Las herramientas para procesar ontologías y razonadores actuales consumen una gran cantidad de recursos, por lo que el tiempo de respuesta y la demanda de memoria virtual son dos aspectos importantes a tener en cuenta al diseñar las herramientas que exploten la información del repositorio semántico y al definir los requisitos del sistema donde se desplieguen.

La información integrada en el Sistema OGO relaciona información entre genotipos y fenotipos asociados a enfermedades genéticas. Así, es interesante extender el Sistema OGO con información de repositorios sobre tratamientos clínicos o medicamentos relacionados con las enfermedades genéticas especificadas. Esta información puede ser explotada para definir ensayos o investigaciones clínicas en distintas especies.

Otra vía futura de extensión del repositorio semántico puede ser integrar datos capturados mediante arquetipos. Un arquetipo se define como un modelo formal de un concepto clínico en uso, es decir, un conjunto de restricciones sobre un dominio de información, que definen un concepto clínico. Una

primera aproximación ha sido llevada a cabo mediante el uso de la interfaz de consultas avanzadas desarrollada en esta tesis con una base de conocimientos de arquetipos clínicos modelados en OWL [231, 232].

El Sistema OGO permite gestionar la actualización de la información del repositorio semántico mediante la ejecución completa del procedimiento de integración semántico con la información actualizada de los distintos repositorios bioinformáticos. El proceso de integración requiere un tiempo considerable para su ejecución, debido al orden de complejidad del algoritmo de integración y del gran volumen de información a integrar. En este sentido, se puede optimizar las actualizaciones mediante la definición de un procedimiento diferente para la integración y otro para la actualización del repositorio, con el que reducir el tiempo de ejecución del mismo. En este caso, se debe tener en cuenta tanto la nueva información generada, la información obsoleta y la información modificada de los repositorios. Sin embargo, los repositorios bioinformáticos analizados sólo ofrecen las versiones completas de su contenido y no las versiones incrementales que pueden facilitar esta labor.

En esta tesis, se han creado bases de datos con la información de los ficheros ofrecidos por los repositorios bioinformáticos utilizados durante el proceso de integración. Para esta transformación, se han desarrollado aplicaciones específicas que son dependientes de la estructura de los ficheros y, por lo tanto, son susceptibles de quedar obsoletas con cada actualización del repositorio fuente. Un posible trabajo futuro, puede ser incluir herramientas más genéricas que permitan extraer los datos de repositorios bioinformáticos que utilicen distintos sistemas para la publicación de datos, como servicios Web, ficheros semi-estructurados, interfaces Web, arquetipos u otros formatos de representación de información.

El uso del repositorio semántico como un grafo RDF en el que se pueda definir consultas SPARQL limita la semántica utilizada del repositorio. Además, los usuarios del mismo deben tener conocimientos y experiencia con SPARQL y con el grafo RDF asociado al repositorio. En esta tesis, se ha desarrollado un método para facilitar esta labor mediante la interfaz de consultas avanzadas que guía a los usuarios en la definición de éstas. Las

consultas definidias utilizando esta interfaz son muy próximas al lenguaje SPARQL. Una posible línea de investigación, puede ser avanzar en el uso del conocimiento definido sobre el repositorio para desarrollar sistemas más avanzados que asistan a los usuarios en el acceso a la información sin que requieran conocimientos previos de SPARQL y del modelo semántico que se consulta.

Para concluir, los resultados obtenidos durante el trabajo de tesis proporcionan una solución para obtener un entorno en el que facilitar la gestión de información biológica y biomédica a través de la especificación de la semántica asociada a ambos dominios. Tanto la definición de una metodología de integración semántica, como el desarrollo e implementación de un proceso automático de integración permiten conseguir un método para mejorar la gestión y mantenimiento de la información de los repositorios bioinformáticos. Como resultado de la integración, se obtiene un repositorio semántico que sirve como base para el desarrollo de interfaces que exploten la información almacenada de modo que faciliten la investigación traslacional entre distintas disciplinas. Así, esta metodología puede ser reutilizada para integrar información de dominios distintos a los utilizados en el trabajo de esta tesis, o extender la información integrada del sistema desarrollado con información de otros repositorios bioinformáticos.

13.3. Publicaciones y contribuciones en congresos

13.3.1. Publicaciones JCR

- Jose A. Miñarro Giménez, Marisa Madrid y Jesualdo T. Fernández Breis. OGO: an ontological approach for integrating knowledge about orthology. *BMC Bioinformatics*, 10(Suppl 10):S13, 2009.

- Jose A. Miñarro Giménez, Mikel Egaña Aranguren, Rodrigo Martínez Béjar, Marisa Madrid y Jesualdo T. Fernández Breis. Semantic integration of information about orthologs and diseases: The OGO system.

Journal of Biomedical Informatics, 44: 1020-1031, 2011.

- Jose Antonio Miñarro-Giménez, Mikel Egaña-Aranguren, Boris Villazón-Terrazas y Jesualdo Tomás Fernández-Breis. Publishing Orthology and Diseases Information in the Linked Open Data cloud. *Journal of Current Bioinformatics*, 2012.

13.3.2. Congresos internacionales

- Jose A. Miñarro Giménez, Marisa Madrid, y Jesualdo T. Fernández Breis. An integrated ontological knowledge base about orthologous genes and proteins. *In Proceedings on 2nd Workshop Semantic Web Applications and Tools for Life Sciences, SWAT4LS*, Edinburgh, Scotland, volume 435, 2008.

- Jose A. Miñarro Giménez, Marisa Madrid, y Jesualdo T. Fernández Breis. Semantic integration and exploitation of orthology information and genetic disorders. *In Proceedings on 2nd Workshop Semantic Web Applications and Tools for Life Sciences, SWAT4LS*, Amsterdam, Netherlands, volume 20, 2009.

- Jose A. Miñarro Giménez, Mikel Egaña Aranguren, Francisco García Sánchez, y Jesualdo T. Fernández Breis. A semantic query interface for the OGO platform. *In Information Technology in Bio and Medical Informatics, ITBAM 2010*, pages 128-142. Springer, 2010.

- Jose A. Miñarro Giménez, Teddy Miranda Mena, Rodrigo Martínez Béjar, y Jesualdo T. Fernández Breis. Exploitation of traslational bioinformatics for decision-making on cancer treatments. *In Information Technology in Bio and Medical Informatics, ITBAM 2011*, pages 1-15. Springer, 2011.

- Jose A. Miñarro Giménez, y Jesualdo T. Fernández Breis. Ontology-driven Method for Integrating Biomedical Repositories. *1st Workshop on Artificial Intelligence in Healthcare and Biomedical Applications, CAEPIA 2011*, pages 473 - 482. LNAI 7023.

Bloque V

Resumen en inglés

Chapter 14

English summary

14.1. Introduction

Traslational research [1] can be defined as the movement of discoveries in basic research to application at the clinical level. This discipline aims to connect basic biomedical researches with clinical research in order to reach new conclusions based on biomedical evidences. To facilitate the traslational research, biological and biomedical information must be related. So, we need to integrate biological and biomedical repositories. However, the heterogeneous and complex information that is contained in such repositories hinder the integration process. Besides, the large amount of stored data and the diversity of information retrieval methods are the kind of obstacles that we have to overcome.

Life sciences is a knowledge based discipline, in which the production of knowledge from data (e.g. the cellular location of a concrete gene product) is a daily activity. However, such knowledge is represented through vast amounts of complex and changing information stored in disparate resources [11] and in machine-unfriendly formats like natural language annotations or scientific literature [17]. The availability of computational methods for organizing, accessing and retrieving information in a systematic way has become crucial, as well as the development of methods that allow the definition and maintenance of shared domain models is paramount for the progress

of research in life sciences [7], and this goal requires such knowledge to be integrated for humans and not by humans. Life scientists waste a lot of time trying to find the inter-related information, and manually removing the redundancy of the results obtained from querying different, independent information resources, because traditional biological resources were designed to be accessed and processed by humans. In addition to this, the results provided by most of the currently available repositories are complete records rather than the concrete information units of interest for the users.

A large number of biological databases have been developed in the last years. The 2012 update of the Molecular Biology Database collection reveals the existence of more than 1380 databases [6]. There are databases for almost any biological field of study, although most of them contain information about genes and proteins from different organisms. Some examples are the Saccharomyces Genome Database, the Mouse Genome Informatics, Flybase or Wormbase. Most of the initial development efforts were done by small research communities, which defined their own terminologies. One major limitation of such approach was that research results could not be efficiently used by and shared with other communities. Consequently, new databases were designed to integrate that disperse information in order to provide a common reference for genes and proteins, such as NCBI Entrez [151] or UniProt [8]. Moreover, other initiatives such as identifiers.org [40], also try to define links between different bioinformatics repositories and to generate unique identifiers at a global level.

One of the most promising endeavours for dealing with such knowledge management problem is the Semantic Web [9]. The Semantic Web is a vision for the next generation web, driven by the World Wide Web Consortium (W3C [10]), that advocates for a web made of data, rather than documents. The application of the Semantic Web on life sciences is steadily growing, through the so called Life Sciences Semantic Web [12] (LSSW). The LSSW is based on the use of Semantic Web oriented W3C standards like RDF [14] (Resource Description Framework), SPARQL [15] (SPARQL Query Language for RDF) and OWL [16] (Web Ontology Language) to codify biological knowledge in a distributed and machine-friendly fashion. Due to the terminological

heterogeneity, different groups worked together to develop common vocabularies. That was the origin of the Gene Ontology [19] (GO), which solves the semantic heterogeneity associated to the annotation of gene products between different databases. An ontology is a formal, explicit specification of a shared conceptualization [148], which provides a shared vocabulary and can be used as a domain model. The success of GO provoked a huge interest in designing, developing and using bio-ontologies, whose number has rapidly increased. Projects such as the OBO Foundry [20] promote the development and use of bio-ontologies. A substantial (and growing) group of current Knowledge Bases (KBs) and bio-ontologies exploits the LSSW approach [13].

In this thesis, we present a framework for the semantic management of biological and biomedical information and thus to facilitate the traslational research in bioinformatics. As a result we have developed the Ontological Gene Orthology (OGO) system [215]. In the OGO system, Semantic Web technologies assist life scientists in the exploration of ortholog/genetic diseases research paths by providing a precise, explicit meaning for information units and intertwining such information. The first version of the OGO system exploited Semantic Web technologies to integrate and manage orthologs information from different repositories. The next version of the OGO system was improved by extending its KB with genetic disease information and reusing standardized bio-ontologies. The fact of reusing the knowledge from standardized bio-ontologies enriched the OGO system and makes it more interoperable. In addition to this, the classes that we are reusing from such bio-ontologies have a more precise definition and have a series of properties that were not covered in our original ontology, so the semantic infrastructure has also more knowledge now.

Orthology information is mainly used in phylogenetic studies and for taxonomic classification [22]. It is also valuable to generate new research hypotheses; i.e. it is likely that an ortholog of a given sequence, being in another taxon, has the same function of that sequence [23]. Therefore, orthologs are used in research about genetic diseases, in particular for understanding their genetic causes [24]. However, resources that produce and store orthologous

clusters, that is, groups of related orthologs, like OrthoMCL [51], do not offer links to genetic diseases and, consequently, the user needs to manually integrate those two kinds of information by using different tools and interfaces. As a result, biomedical researchers need to perform a series of mainly manual tasks to retrieve this kind of traslational information. For instance, retrieving the genes related to those that cause Prostate cancer requires performing the following actions: (1) query the Online Mendelian Inheritance in Man resource (OMIM [78]) for obtaining information about the genes that cause Prostate cancer; (2) query the existing orthology information resources, such as KOG [50], Inparanoid [53], OrthoMCL and Homologene [52] for obtaining the orthologous genes; and (3) manual combination and analysis of the information retrieved from all orthology resources. Consequently, biomedical researchers cannot easily find this information because they are required to know: (1) which resources are available and contain the desired information; (2) how such resources can be accessed and queried; (3) the meaning of the data types and fields used in each resource. In our case, when searching for orthologs some resources are protein centered, like KOG, whereas others are gene centered, like Homologene, so this diverse of granularity may hinder the search by users. An incorrect interpretation at this level may invalidate any further analysis performed by the researcher. Thus, there is a clear need for methods and tools that help researchers in finding the right information.

So, the integration of biomedical resources is necessary due to the huge amount of information that is continuously being generated and considering its inherent diversity and heterogeneity. Otherwise, the mentioned caveats would limit the efficient use of such resources. Our Semantic Web-based approach for the integration of biological knowledge will be applied to orthology and genetic diseases. The integration will be facilitated by a global ontology, which will be used for mapping the different resources to achieve a common machine processable representation. This will allow the generation of an integrated ontological KB, to which biologists and machines will have access through web interface.

The next sections provide a summary of the main sections of this thesis. Next, section 14.2 will present the main objectives, whereas the section 14.3

will sum up state of the art technologies and systems which are related to this thesis. Then, the section 14.4 will indicate the results obtained and finally the section 14.6 will summarize the conclusions and future work.

14.2. Aims

This thesis aims to facilitate the traslation research by defining an integration methodology of bioinformatics repositories and by taking advantage of the semantic annotation of data to implement more advanced tools for managing and exploiting such information. The result of the integration methodology is a Knowledge Base (KB) which use a global bio-ontology to represent the associated knowledge. The KB allows us to retrieved precise information for traslational bioinformatics research.

Therefore, the main aims of this thesis are:

- Definition of a methodology to integrate bioinformatics repositories of biology and biomedical domain into a KB.
- Design and implementation of a method for integrating separate repositories avoiding redundancy and sharing a global ontology.
- Design and implementation of several methods to search and exploit the KB to facilitate traslational bioinformatics research.
- Validation of the thesis results by integrating bioinformatics repositories related to orthologous genes and genetic diseases.

14.3. State of the art

This section will introduce the latest advances of the standards and technologies related to this thesis. The thesis uses bioinformatics repositories to obtain the information to be annotated and integrated into the KB. Besides, it takes advantage of ontologies to conceptualize the biological and biomedical domains. In particular, bio-ontologies play a dominant role in the development of the work of the thesis because their concepts and properties are used

to annotate the integrated information, they are use as repository schemas, and also provide a common terminology to exchange information with other systems. Semantic Web technologies are used to manipulate and exploit the generated KB and defined bio-ontologies. Whereas to integrate the information we need to use a mapping language to connect relational schemas to ontology models. So, we have analyzed the available mapping languages. Besides, we analyze other publishing data method in RDF format, such as Linking Data.

14.3.1. Bioinformatics Repositories

Bioinformatics repositories provide biological and biomedical information to researchers. The available information comes from the results of previous experiments and published scientific articles. The way information is stored and organized is designed to be accessed only by users. Therefore, users have to manually search, retrieve and analyze such data. If the information were available to be accessed and processed by applications, tasks that could be automated would reduce the tedious work of researchers. So, their productivity would be improved by reducing time and effort of mechanical tasks.

The current bioinformatics repositories are accessible via the Internet. They consist on data bases that are able to manage large amount of information and also make it available efficiently. The current "Nucleic Acids Research Database Issue and the online Molecular Biology Database Collection" indicates the existence of 1380 bioinformatics data bases, where 96 are new and 83 have been updated.

There are a wide variety of types of bioinformatics repositories depending on different criteria. Depending on the type of information they contain, we can indicate, for example, the Genbank [27], ENA [26] or DDBJ [28] for nucleotide sequencing repositories, or the Uniprot for proteins repository. Moreover, the repositories can contain specialized information of a particular species such as FlyBase [32], MGD [33] or SGD [31], or the repository may contain information from several repositories such as NCBI [36], EBI [37] or

KEGG [38].

We validate the work of this thesis by using bioinformatics repositories that contains information about clusters of orthologs or genetic diseases. The gathered repositories that contain information about orthologous genes and proteins are KOG, HomoloGene, Inparanoid and OrthoMCL, whereas OMIM contains genetic diseases information.

The ortholog concept specifies a type of evolutionary event. It describes two DNA sequences that belong to different species and the species were made different by a speciation event. We can relate the ortholog concept with other evolutionary concepts such as the homolog, paralog and xenolog concepts. So, if two DNA sequences from different or the same organism are similar, then the sequences are homologous. Therefore, if two sequences are orthologous, then they are also homologous. However, two paralogous sequences are two homologous sequences, from the same species, that have been made different by a duplication event. Finally, two xenologous sequences are homologs that have been made different by an horizontal transfer in the same species genome.

The KOG repository consists of clusters of orthologous genes and proteins from seven eukaryotic species. In particular, the species are *Caenorhabditis elegans*, *Drosophila melanogaster*, *Homo sapiens*, *Arabidpsis thaliana*, *Saccharomyces cerevisiae*, *Schizosaccharomyces pombe* and *Encephalitozoon cuniculi*. It contains 4852 clusters of orthologs that indicates 110655 genes and 59838 proteins. The Table 2.1 shows in the first column the files that were downloaded and in the second column a brief description of each file. The Figure 2.2 shows an example of a cluster of orthologs, how it is represented and the information that can be gathered.

The OrthoMCL repository contains clusters of orthologous proteins. The first version was issued in 2005 and it contained 70388 clusters from 55 species. The second version of this bioinformatics repository was published in 2008 with information about orthologs from 87 species. In the third version, there were 128 species, the fourth version contained information from 138 species and the current version contains information from 150 species. We could gather information of 124740 cluster of orthologs and 1192387 orthol-

ogous proteins. The Table 2.2 describes the files that the repository make available to download. The Figure 2.3 shows an example of a cluster of orthologs, where each row indicates an organism and a protein name. The codes for each species is described in the repository files.

HomoloGene is a bioinformatics repository which is focused not only in clusters of orthologs but also in paralogs. It uses the synteny [54] method to obtain the clusters. So, this method compares genome sections from different species by trying to find a set of similar sequences in the same chromosome in different species. The repository analyzes 20 species to group 244950 genes into 43998 groups. The Table 2.3 shows the files that are available to download.

The Inparanoid repository stores information from inparalogs genes. It uses an algorithm to differentiate the inparalogs and outparalogs between two species or among a group of them. Inparalogs are genome sequences that have been duplicated after a speciation event. Therefore, they can be considered orthologs to one or more orthologs in other species if the have a common ancestor. On the other hand, outparalogs are the genome sequences that have been duplicated before a speciation event and therefore they cannot be considered orthologs. Inparanoid repository analyze 1687023 sequences from 100 species to obtain the clusters of inparalogs. The species are analyzed in pairs and the sequences are grouped in pairs, so the repository provides 4950 files where the groups are stored.

Finally, the OMIM repository consists of 20757 records of human genetics diseases. Each record provides a description of a disease with a reference to the gene is related to. This relationship allows us to connect the phenotype with the genotype perspective. The Table 2.4 describes the files with the information of the repository. This repository was initiated in the 60's by Dr. Victor McKusick. The Figure 2.6 shows three different descriptions of OMIM records where the relationship between the disease and the gene is represented.

14.3.2. Semantic Web

The Semantic Web is an evolution from the current Web, where the data is semantically annotated and it is represented in order to be understood by humans and to be automatically processed by machines. To solve the knowledge management related to life sciences and bioinformatics repositories, Semantic Web technologies have been applied. For example, YeastHub [173] utilizes semantic web technologies for integrating data in the life sciences domain.

We have utilized the Semantic Web Technologies to manage and exploit the semantic and the information of bioinformatics repositories. The use of OWL language for managing the knowledge associated to the repositories is the cornerstone of the thesis. Having the knowledge conceptualized and the data annotated allows us to exploit the repositories via their semantics. There are specialized parser designed to handle OWL ontologies, such as OWL-API [90] or Jena [91], in order to create an ontology, add new resources, delete or update them and to navigate through the ontology graph. They are also able to read and write different format of ontologies files, such as RDF, OWL or Turtle. We have used Jena in this thesis because of persistence functionality. This parser allows to store the information in many different types of repositories.

OWL ontologies are based on description logic and therefore the knowledge defined can be analyzed to infer new information. So, reasoners can be used together with the ontology representation to process the information model. Pellet [112], FaCT++ [110], HermiT [111] and RACER [113] are examples of such reasoners, which implements different algorithms to process the semantics. Reasoners can verify ontology consistency and also infer new information depending on the ontology axioms and properties defined.

SPARQL is also used in this thesis to search for instances in KBs. There are other RDF query languages such as SquishQL [123], RDQL [124] o TriQL [125], but SPARQL is the query language accepted by W3C. Its grammar is similar to SQL but relational repositories are stored as tables whereas RDF repositories are stored as triples (Subject, Predicate and Object).

The SPARQL query language defines three components: URI, literal, and variable. URIs identify the resources and are surrounded by "<" y ">", literals are enclosed by quotation marks and variables are identified by a "?" in front of the variable name. The main function in the language is *SELECT*, which returns all instances that match the variables defined in the SELECT. The Figure 3.6 shows an example of SPARQL query that search for all titles of books in alphabetical order in the repository. SPARQL can also define optional restrictions or filter the values of a variable depending on different criteria, such as a regular expression.

However, SPARQL does not provide any method to manipulate a RDF graph. So, SPARUL [92] is an extension of the SPARQL grammar to implement INSERT (see Figure 3.8), DELETE and MODIFY (see Figure 3.11) RDF triples in a repository and to CREATE or DROP a RDF graph (see Figure 3.12). This language was first defined by Hewlett-Packard and nowadays it is being developed by the W3C SPARQL Working Group.

Linked Data [130, 131] is the term which describes a set of methods and principles for exposing, sharing, and connecting pieces of data, information, and knowledge on the Semantic Web using URIs and RDF. The information is shared via Web and stored in RDF repositories, therefore the data can be referenced from other repositories allowing interrelating repositories.

The basic recommendations and best practices to publish information in the Web of data were proposed by Berners-Lee in [141] and adopted by the Linked Data initiative. Four main principles were defined to make the Web of data grow:

1. Use URIs as names for resources.
2. Use HTTP URIs so resources can seek for.
3. Provide useful information, using the standards, RDF and SPARQL, when seeking for resources.
4. Include links to other URIs, so that they can discover more resources.

One of the most outstanding projects to promote publishing information in the Web of data using Linked Data principles is the Linking Open Data

(LOD) community project. This project aims to make data freely available to everyone via Web. There are many open data sets available on the Web (see Figure 3.13) and they comprise the LOD cloud. CKAN [142] is a repository where the metadata of the published repositories are stored. So, users can search for the description of each repository which form the LOD cloud.

14.3.3. Biomedical Semantic Web

In biomedical discipline, the information stored in repositories is complex, heterogeneous, dispersed and rapidly evolving. These features are exacerbated by a natural language representation of the information and therefore susceptible to ambiguity in interpretation. Therefore, it is necessary to define the biomedical domain-specific knowledge to interpret the values present in the repositories with precision. Thus, biomedical ontologies are used to make available to users the knowledge needed to understand the information repositories.

Bio-ontologies provides a common terminology to harmonize the knowledge from a particular biological or biomedical domain. So, bio-ontologies can be used to annotate information in order to be easily understood by others biologists that could not be expert in such domain. Traditional bioinformatics repositories do not provide any knowledge to the information so the users are assumed to have enough expertise to understand them. But, this is not the case for applications, which have to be implemented with the knowledge to process the data integrated in the code. However, bio-ontologies allows to have the knowledge independent from the code and also to reuse it in other systems.

OBO Foundry is an initiative which aims to create a group of bio-ontologies to cover a wide range of biological and biomedical knowledge domain. It defines a principles that lead the definition of the bio-ontologies. The core of the OBO are the following ontologies:

- CHEBI [154] (Chemical Entities of Biological Interest): A structured classification of chemical compounds of biological relevance.

- GO (Gene Ontology): Provides structured controlled vocabularies for the annotation of gene products with respect to their cellular location, their biological process, or their molecular function.

- PATO [155] (Phenotypic Quality Ontology): This ontology can be used in conjunction with other ontologies such as GO or anatomical ontologies to refer to phenotypes.

- PRO [156] (Protein Ontology): This ontology has been designed to describe the relationships of proteins and protein evolutionary classes, to delineate the multiple protein forms of a gene.

- XAO [157] (Xenopus Anatomy Ontology): A structured controlled vocabulary of the anatomy and development of the Xenopus laevis.

- ZFA [158] (Zebrafish Anatomy Ontology): A structured controlled vocabulary of the anatomy and development of the Danio rerio.

Moreover, OBO Foundry contains some ontologies as candidates which are ontologies of interest under revision. Bio-ontologies are coded in OBO format (see Figure 4.1) which is an ontology language to define their ontologies, but there are also some parser, such as OBOEdit [161] or OBOInOWL [163], to transform this representation into an OWL ontologies.

The bio-ontologies reused in this thesis belongs to the OBO Foundry. The ontologies are the GO, Evidence Codes Ontology [164] (ECO), OBO Relationship Types [165] (RO), NCBI Organismal Classification [166] and Human Phenotype Ontology [167] (HPO).

The Gene Ontology permits to describe gene products depending on their cellular location, their biological process, or their molecular function. Each criterion is represented as a different sub ontology and therefore, providing three ontologies: Cellular Component, Biological Process and Molecular Function.

The Evidence Codes Ontology provides a wide vocabulary to annotate experimental evidences. It is being used in the BioPortal [171] of NCBO. It came up with GO in order to indicate the type of evidence of when linking a

GO term with a gene product. The Figure 4.3 shows the *ECO:0000002* code which indicates the evidence "direct assay result".

The Relationship Ontology aims to define a set of independent relationships to associate the concepts define in the ontologies of the OBO Foundry. The defined relationships are listed in the Table 4.2, but they can be specialized and extended with other relationships to improve their meaning in other ontologies.

The NCBI Organismal Classification contains a hierarchy of concepts which represents the taxonomy of species in the NCBI. There are no relationships defined for the concepts of the ontology and it is not a complete classification of species. It provides a classification of the most frequent species. For example, the hierarchy of concepts related to *"Homo sapiens"* is: *cellular organisms*, *Eukaryota*, *Opisthokonta*, *Metazoa*, *Eumetazoa*, *Bilateria*, *Coelomata*, *Deuterostomia*, *Chordata*, *Craniata*, *Vertebrata*, *Gnathostomata*, *Teleostomi*, *Euteleostomi*, *Sarcopterygii*, *Tetrapoda*, *Amniota*, *Mammalia*, *Theria*, *Eutheria*, *Euarchontoglires*, *Primates*, *Haplorrhini*, *Simiiformes*, *Catarrhini*, *Hominoidea*, *Hominidae*, *Homininae* y *Homo*. The Figure 4.4 shows an example of the *"Homo sapiens"* concept definition with the properties and rank associated.

The Human Phenotype Ontology provides a wide vocabulary, about 9500 terms, which can be used to describe phenotypic features encountered in human hereditary and other disease. Currently, it is focused on diseases listed in the OMIM repository, for which annotations are also provided. The ontology is divided into three sub-ontologies: (1) The Organ abnormality ontology; (2) The Inheritance ontology; and (3) The Onset and Clinical course Ontology. The Figure 4.5 shows the excerpt of the ontology, in OWL format, associated with the definition of the concept *HP:0000002*.

Regarding the Linked Data initiative, biomedical data sets are one of the domains of information that are having more activity in the development of LOD cloud. So, some bioinformatics repositories such as Uniprot

[1], Protein Data Bank [2], DrugBank [3] or Gene Ontology Annotations [4], have published their content following the principles of the Linked Data. These repositories can be referenced by other repositories and therefore enrich the content of the LOD cloud. One example of research group that is publishing biomedical data sets and establishing links between them in the LOD cloud is the Bio2RDF [174]. Bio2RDF integrates information from 42 bioinformatics repositories that belongs to many organizations, such as NCBI, KEGG, Gene Ontology, PDB, Mesh, BioPAX, etc. Another example is Linked Life Data [233]. It encompass 27 bioinformatics repositories that are integrated in LOD cloud and also provides search methods to retrieve the information published.

14.3.4. Methodologies for The Integration of Biomedical Information

Owing to the proliferation of bioinformatics repositories have been developed several types of integration methodologies. This section describes the most important ones. The integration of information is based on the sharing data among different repositories allowing the identification of mappings [176]. The Figure 5.1 shows the classification of the analyzed integration methodologies according to their information integration level and scope. The integration methods are listed below.

1. Warehouse Integration
2. Unified View Integration
3. Links Integration
4. Mashups
5. Service Oriented Architectures Integration

[1] http://www.uniprot.org/uniprot/
[2] http://semanticscience.org/projects/pdb2rdf/
[3] http://www4.wiwiss.fu-berlin.de/drugbank/
[4] http://www.bioontology.org/wiki/index.php/OBD:SPARQL-GO

6. Model Driven Architectures Integration

7. Application Integration

8. Workflow Integration.

The first methodology is the warehouse integration. It is based in the creation of global warehouse where the information from several information sources are integrated. This methodology demands a lot of computing resources during the integration process, but its results are considered a real integration. So, the resulted repository is an unique repository which retrieves integrated information. Examples of repositories that were created following this integration regime are ATLAS [177] and Columba [178]. Besides, there are tools that implement this methodology, such as BioMART [179] and BioWarehouse [180].

The unified view integration methodology is based on developing a common infrastructure that allows consulting and retrieving information from different repositories as if they were one. So, the information is not integrated until the users do not provide their queries. This method depends on the availability and reliability of source repositories. However, it facilitates the maintenance of the repository because updating the data does not affect the unified view. Only changes in the schema or access method might hinder its consultation. Moreover, this method is detrimental to the response time on the unified view. Examples of bioinformatics systems that use this methodology are TAMBIS [181] and BIOZON [182].

The third methodology is the Link Integration. It uses links to indicate the correspondences between instances of two repositories. Defining links that connect one piece of information with its equivalent one in other repository are the basis of this methodology. Hence, the integration involve the definition of links that indicate the relationships among repositories. The project *identifiers.org* allows to search biomedical information and provides links to several biomedical repositories where such information can be found. So, the repositories may refer to the same information from different perspectives, thus enriching the search result.

Mahsups are the integration regime that combines several information sources and services to offer new contents and services by means of aggregation. They mainly exploit Web Services and REST services, and gather the data from the sources repositories and RSS services. They usually combine biomedical information with the services to provide new Web resource. The complexity and implementation efforts are minimized compared to other integration methodologies. These systems are usually developed for a particular use and to solve a specific need, so its application is restricted.

Service Oriented Architectures (SOA), Model Driven Architectures (MDA), Application and Workflow are integration methods that can be combined with the integration methodologies discuss above. These methods defines different processes that facilitate the integration of information. SOA and MDA are two architectures of software development that can be used to implement the integration method. These types of architectures are widely used in the bioinformatics domain [190], such BioCatalogue [191] and Gaggle [192].

The application integration method is based on developing new applications that integrate applications which are focused on the same information domain. Integrated applications are designed in a particular domain and therefore it is difficult to be extended to other domains. Utopia [193] is an project which use this integration approach.

Finally, the workflow integration method consists of a series of process that are interrelated, to achieve the integration of information. Each process is specialized in a particular task and the workflow coordinates the inputs and outputs of each process. Taverna [194] is an example which uses this type of integration method to coordinate bioinformatics web services.

14.3.5. Methodologies for The Semantic Publishing of Biomedical Information

Much of the bioinformatics repositories use data bases to store and manage their content, such as Entrez. Also, much of the information available for downloading is represented in tabular format files which can be easily

exported to tables in relational repositories. But this representation of information is not suitable for being consulted by autonomous applications because there are no domain semantics related to the information in order to be interpreted or processed, however RDF and OWL languages provide features to annotate data with semantics. Therefore, defining methodologies to publish the content of such repositories into RDF repositories is crucial to extend the repositories with their semantics. In 2007, W3C organized a workshop on RDF access to relational databases. From that workshop, the RDB2RDF Incubator Group [146] was created to develop some proposals. Two reports were published in 2009, the first one [195] analyzed the state of the art in the area, and the other [196] provided a recommendation to standardize a mapping language to map repositories from relational schemas to RDF or OWL.

We have classified the most important publishing methodologies in three categories: (1) Direct publication; (2) Customized publication ; and (3) Virtual publication.

The direct publication aims to publish information stored in traditional repositories in RDF automatically. This method neither required users to indicate the mappings nor RDF schema. Examples of this method is the Direct Mapping method (DM) [199]. This method defines autonomously direct mappings between the relational tables to a RDF model. So, the RDF model match the relational schema and there is no need to define the mappings manually. The semantics associated to this is very limited but it is the simplest approach. OntoGrate [200] is also a platform that utilizes this publishing method. It was designed to unify database schemas and ontologies by defining mappings between them. To find the mappings automatically, a system which uses data mining and alignment of terms was designed. Ontograte also allows the translation of relational queries across different repositories through ontological models to improve the scalability of the system when they contain a large set of instances.

The customized publication is based on the manual definition of the mappings between a existing database schema and a RDF graph. Examples of this type of methodology are: R2RML [204], D2R [205] and R2O [206].

R2RML allows to define mapping rules and provides the features to customize the mappings between a database and a defined RDF graph. The mapping rules are also RDF graph and it use Turtle to define the rules. The set of defined mapping rules are called mapping graph. This language is independent from its implementation so it can be used to develop virtual or real RDF repositories from a traditional repository. The Figure 6.1 shows the architecture of the mapping rules.

D2R is a declarative language which allows to define the mappings between a database schema and a OWL ontology or RDF graph. It uses XML to codify the mapping rules (see Figure 6.2). This language provides sufficient flexibility to define the mappings between the different models. This flexibility is achieved by using SQL queries on the definition of rules. The result of the different SQL queries are gathered and then the results are associated with instances of the ontology model. This language allows to define relationships with different degrees, properties with multiple values, and complex patterns resulting from standardization of tables in the database.

R2O is also a declarative and extensible language for defining mapping rules between relational repositories and ontologies. It provides a set of simple axioms, coded in XML, with a well defined semantics which facilitates mapping rule definitions.

Finally, the virtual publication methodology allows to define mappings between databases and RDF graphs but the RDF dataset is produced on demand. The data is not published in a different repository but virtually generated by an application. Examples of systems which implement this methodology are: D2RQ [210] and Virtuoso RDF Views [211].

D2RQ language lets specify the relationships between databases schemas and OWL/RDF ontologies. This language is part of D2RQ platform, which provides the D2RQ engine that allows to create a RDF views over a relational database and thus be able to search with Jena, Sesame or SPARQL query language. The D2RQ platform has the D2R server that aims to publish on the web interface to browse the repository information using Linked Data and SPARQL queries.

Virtuoso RDF Views is a feature offered by OpenLink Virtuoso Universal

Server. This feature allows the server to dynamically transform the content of the database to RDF and made it available for consultation via a SPARQL query interface. Thus, virtuoso allows to define SPARQL queries on RDF views and on local RDF repositories simultaneously.

14.4. Results

This thesis is focused on the knowledge management to facilitate relational research by integrating and exploiting life sciences semantic and information. It uses Semantic Web technologies to manage and exploit the knowledge and data from bioinformatics repositories. The main result of this thesis is the definition of a semantic integration methodology for bioinformatics repositories. This methodology requires the definition of a global ontology which conceptualizes the knowledge and relates the concepts of bioinformatics repositories. As a result of the integration process a Knowledge Base (KB) is generated. This KB stores the global ontology definitions and contains the integrated and annotated information. We have also designed and implemented different systems to exploit the KB and to publish its content in the LOD cloud. Finally, we have also designed and implemented an integration application that follows the principles of the proposed methodology for semantic integration.

14.4.1. Semantic Integration Methodology

The defined methodology (see Figure 8.1) aims to guide the integration process of independent bioinformatics repositories by using the semantics associated with the information domains. As a result of the methodology, a KB is obtained. The KB uses the knowledge of the information domains as its repository schema, i.e. the data are annotated with the concepts of its domain and also interrelated by using formally defined relationships.

Bioinformatics repositories The bioinformatics repositories are gathered depending on the information to integrate. We have to analyze their content and how the information can be obtained. The repositories

have to provide an interface to process its data or provide its data to create a local duplication.

Global Ontology The global ontology is the cornerstone of this methodology. We need to define an ontology which conceptualizes the information domain of the selected bioinformatics repositories. It has to represents the concepts, properties and relationships of the information of the repositories, and also the restrictions of its domains. We can also import other bio-ontologies, which overlap the semantics of the global ontology, to enrich its conceptualization.

Mapping rules We define a grammar for mapping the repositories and the global ontology. Thus, we obtain the mapping rules that allow to automate the annotation of data and to create the KB.

Identity rules We define a grammar is associated with identities rules of the global ontology concepts to integrate instances of the repository. These rules allow more advanced identity definition than a simple identifier or property value. Since bioinformatics repositories can use different identifiers to describe the same real resources, this feature is needed to identify the resources through their descriptions of instances and thus eliminate duplicates.

Integration process Once, we have accomplished the previous methodology phases we can perform the integration process. It uses the mapping rules to insert data into the KB and the identity rules to search for duplications to combine them.

Knowledge Base Finally, we create a KB which stores the integrated information of several repositories. The KB uses the global ontology to manage its content, making it available for consultation from a semantic approach.

The semantic integration methodology takes advantages of the domain knowledge of the global ontology to guide the integration process. The mapping rules use the ontology to associate the schema of the bioinformatics

repositories with concepts, properties and relationships of the ontology. The identity rules use the ontology descriptions to provide more advanced identity definition than a single key property. So, the integration process can use the defined rules to integrate the data into a KB automatically.

We defined the grammar of mapping rules (see Table 1) which allow to code the mapping rules in XML documents. Each XML document defines the rules to map a relational repository to an ontology. Therefore, each rules document contains the information to connect to the particular database, to load the particular ontology model and the definition of the rules. Mapping rules can be differentiate between rules to map database keys to classes of the ontology, to map property values to defined properties of the ontology, and to map foreign keys to relationships of the ontology. Relational tables declare primary keys to indicate the identity columns for the rows of such tables. So, this schema can consider equivalent to ontology classes and URIs of instances of the ontology. But we have to consider that a table might not represent any concept of the ontology. This kind of rules are identified by the *"DB2Class"* label. The Table 8.1 shows an example where a table is identified with the Gene class of the ontology. Besides, relational tables can have more columns that are not key columns and usually they represent values of properties. So, we can use the key used in the *"DB2Class"* rule to associate its property values. We defined the *"DB2Prop"* rule to relate a column of a table with the property of its ontology class. Table 8.2 shows the rule to associate the gene names property with the values of a column. Finally, to indicate the relationships between two instances in relational tables we use foreign keys. Therefore, we have defined the *"DB2Rel"* rule to associate the relationships of the ontology with the key column and foreign key of a relational table. The Table 8.3 provides an example of this *"DB2Rel"* mapping rule where it associates gene instances with GO terms through the *"participates_in"* relationship.

Identities rules aim to identify in the KB instances that were already created but are equivalent to the instances to be created. These rules facilitate the definition of the requirements to consider equivalent instances. If we found an equivalent instance in the KB with these rules, we could com-

bine both instances and then avoid redundancy in the repository. Identity rules like mapping rules are coded using XML documents and its grammar is showed in Table 2. An identity rules document contains the rules related to an ontology, where each rule is associated with a class (and with its subclasses by means of hereditary) of the ontology. The requirements defined in each rule are used to find the equivalent instances. These requirements are evaluated as a Boolean expression. Hence, we use *"AND"* and *"OR"* operators to indicate the type of Boolean expression in a requirement. In particular, each requirement in a rule specifies a set of conditions over the properties and relationships values of instances of a particular class. The conditions can indicate that all values of a property have to be equal or just some of them. If the property value is an *string*, we can also indicate that the evaluation of equality is not case sensitive. The Table 8.4 shows an example of the identity rule of the Gene class. In particular, we consider two genes equals when some names or identifiers are equal and both instances are related to the same species. It means that other properties or relationships are not considered important or necessary to identify a gene.

The Table 8.5 shows the algorithm associated to the integration process defined for the integration methodology. This algorithm uses the mapping rules to annotate the data from the bioinformatics repositories with the global ontology concepts, whereas the identity rules help to reduce the ambiguity and redundancy of the information sources. The complexity of the algorithm depends on the number of repositories to integrate, the amount of data of each repository and the size of the KB. The case performing of the algorithm is $O(n^2)$ in the best case and in the worst case is $O(n^3)$.

14.4.2. Knowledge Base

The KB contains the integrated information of the bioinformatics repositories but modeled using the defined global ontology. The Figure 8.3 shows the generic elements of the architecture of the KB. The first element is the OWL and RDF API. This element is required to create the instances in the KB and to manage its semantics and reasoning features. The next element is

the SPARQL endpoint. Users needs to retrieve its content through a semantic query language. This feature is important to be able to implement the identity rules functionality. The last element is the repository management system which stores the integrated information. The type of repositories depend on the repositories supported by the selected ontology API.

14.4.3. Integration process

The semantic integration methodology defines the integration process which uses the global ontology, the bioinformatics repositories, and the mapping and identity rules to integrate the information into the KB. To facilitate the integration process we designed and implemented an application which support: (1) the definition of mapping rules, (2) the definition of identity rules, and (3) the configuration of the integration process.

The main interface for defining mapping rules is showed in Figure 10.2. This interface allows users to indicate the repository connection parameters and the ontology file location. Once we have defined this parameters, the interface loads the information and makes it available in the left side of the window. The right side of the window shows the rules definition. Depending on the elements selected in the left side of the window, the interface create the corresponding mapping rule type definition. If users select a key column of a table of the database and class of the ontology, they obtain a *"DB2Class"* rule definition. If users select a table and a property of the ontology, a new windows is showed (see Figure 10.4) to define the parameters of a *"DB2Prop"* rule definition, such as the column to identify the property values and the ontology class of the related instances. If users select a table and a relationship of the ontology, another windows is showed (see Figure 10.5) to define the *"DB2Rel"* parameters, such as the columns keys to identify the instances and their ontology classes.

To assist users in the identity rules definition we developed an application which is showed in Figure 10.6. This interface shows the identity rules definitions in the upper section of the window. The lower section shows the requirements associated to a particular rule definition. Users can indicate

the ontology file to load for helping the rule definition. The ontology is used to indicate the class related to a new identity rule (see Figure 10.7) and to define the requirements parameters (see Figure 10.8), such as the related property, its Boolean operator or the class of the instances associated with a relationship.

To run the integration process we have to configure the suitable parameters. The parameters are:

- The list locations of the mapping rules files. Each mapping rule describes the connection to a particular repository and the location of its related ontology file.

- The location of the definition of identity rules document. The XML document must contain the location of its related ontology file.

- The connection parameters to the repository related to the KB. The KB is supported by a database management system and Jena API.

We have developed an application to configure these parameters and to perform the integration method. First, the Figure 10.9 shows the main interface of the application. It provides the *"Add Identities File"* button to select the identity rules file. The *"Add KB Parameters"* button is used to define the connection parameters associated with the KB. To indicate the list of mapping rules documents, the main interface provides the *"Add Mapping"* button. This button shows other window interface (see Figure 10.10) where users can define the parameters associated with a particular repository. One repository can have more than one mapping rules document associated but a single document can only describe the mappings between a database schema and an ontology. So, if the information of a repository consists of different schemas we need to define one mapping rules document for each schema. This interface provides the *"Add Mapping File"* button to indicate the main mapping document of the repository, the *"Add Linked Mapping"* button allows to add to the list the mapping rules document related to the main document, the *"Delete"* button remove a selected document from the

list, and finally the *"Accept"* allows to continue the configuration of the integration method in the main interface. The ontology used to define the mapping rules and identity rules documents must be equal in all documents in order to use this ontology to create the KB. Besides, the main interface provides the *"Create KB"* button, to create or clear the KB repository ,and finally the *"Start Mapping"* allows to run the integration method.

The defined algorithm that represents the integration method is showed in the Table 8.5. This method gets the list of bioinformatics repositories to be integrated, the list identities rules and the connection to the KB. Then, for each repository we obtain its mapping rules. To gather the instances of the selected repository we must use the *DB2Class* rules. These rules identify the primary keys of the instances in the repository which are associated with a class of the ontology. Due to the fact that instances might be related to other instances in the ontology, we must sort out the order of insertion of instances into the KB. For example, genes might be associated with proteins, if we try first to insert the gene instances into the KB, at that very moment we do not have created its related proteins so this kind of relationships can be created. Therefore, sorting the insertion of instances we can avoid dependency problems. In case of having cycles in the ontology graph, we must create the instances in the moment they are referenced by other instances. Once we sorted the *DB2Class* rules to avoid dependency problems, we can select the rules in order. For each selected rule we need to obtain its ontology class, its mapping rules (*DB2Prop* and *DB2Rel*) and the identity rule associated with its ontology class. Next, we can gather the primary keys of the instances that are stored in the repository. We can create a candidate instance for each key from the repository. We assign the properties and relationships values to each candidate instance by means of its *DB2Prop* and *DB2Rel* rules. Then, we can search for equivalent instances of the candidate instance in the KB using its identity rule. If we find an equivalent instance we merge the information of the candidate instance and the equivalent instance, if not we assign a new URI to the candidate instance and insert it into the KB.

To collect the data from repositories, the integration method defines SQL queries based on the description of the mapping rules. Depending on the

type of the mapping rule we can obtain different types of SQL queries. In case of the *"DB2Class"* type we define a SQL query to obtain the key values of a table which are associated with instances of a particular class. The Table 8.6 shows the SQL query defined for the example of *"DB2Class"* rule of the Table 8.1. In case of the *"DB2Prop"* type of mapping rule, we define the corresponding SQL query to gather together all values of a specific property associated with an instance. This type of queries use the key values obtained by the corresponding SQL query of the *"DB2Class"* rule. The Table 8.7 shows the SQL query associated with the example of *"DB2Prop"* rule of Table 8.2. Finally, the *"DB2Rel"* rule type is used to collect the information of a particular relationship. Its corresponding SQL query gather the key values of instances related to a particular instance which is identified by the key value obtained by a *"DB2Class"* rule. The Table 8.8 shows an example of the SQL query generated from the mapping rule of the Table 8.8. Once we have collected the data from repositories, we are able to annotate them with the concepts, properties and relationships defined in the mapping rules.

Because the key values used in different bioinformatics repositories are usually not equal, we must define the rules that describe when two instances are equivalent. The identity rules aims to provide such knowledge. We use the annotated data to create candidates instances which may be inserted in the KB. The identities rules and a candidate instance is used to define SPARQL queries. We use these queries to search for instances that are already integrated, and which are equivalent to the instance candidate. The Table 8.9 shows an example of a SPARQL query generated from the identity rule of Table 8.4. This query search for the instances of the class *Gene*, that belong to the species *"orgnanism1"* and are associated with the name of the gene *"name1"* or *"name2*, or also the identifier of the gene *"id"*.

14.4.4. Exploiting the semantics of the KB

They exploit the semantic of the KB to gather and filter its content, however, they do not allow users to define their own queries. To allow users to retrieve precise information from a KB, we need to develop an interface

which provides users with a tool where they can define advanced queries [214]. The interface guides the users through the query definition by providing them with recommendation based on the knowledge of the KB. The KB is consulted with SPARQL queries which are generated with the restrictions defined by users. The advanced query interface facilitates the query definition because users do not have to take into account the SPARQL grammar and do not need to know exactly how information is interrelated in the KB. The interface avoids SPARQL grammar and syntax mistakes since users do not have to define manually the query due to the fact that the interface guides the query definition process. Users with a limited knowledge on the KB structure and content can define queries because the interface provides them with recommendations of the properties and restrictions that are allowed for each concept and how they are interrelated in the KB.

The Figure 12.10 shows the main view of the advanced query interface with an example of query definition. The interface provides two text boxes. The top text box contains the variables that are defined to collect information, whereas the bottom text box contains the defined constraints. The query variables are associated with concepts of the KB, and they are used to retrieve instances of such concepts. The interface provides two buttons, *"Select Concept"* and *"Delete Concept"*, to add or remove a defined variable from the query. The restrictions are similar to SPARQL restrictions definition. The restrictions consist of three parts, subject, predicate and object. Subjects can be any resource of the KB or variables, predicates correspond to relationships or properties of the KB, and finally objects can be anything, such a resource, a variable or a literal value. To define the query restrictions the users are provided with recommendations. The recommendations are showed as a list of allowed restrictions for the defined variables. The Figure 12.12It shows an example of the restrictions that can be defined for a variable associated with the genetic disease concept and for a variable associated with the gene concept. The main view provides the *"Add new requirements"* and *"Delete requirement"* buttons to add and remove restrictions in the query. Finally, we use the *"Execute Query"* of the main view to obtain the query results.

14.4.5. Publishing data into LOD cloud

In this section we describe the requirements for transforming KB information into RDF triples in order to publish it in the Linking Open Data(LOD) cloud initiative. The transformation of a KB into a RDF dataset follows the Linked Data (LD) principles: (1) use URIs to name entities; (2) use HTTP URIs so that those names can be resolved; (3) when a user looks up a URI, provide useful information, using RDF and SPARQL; and (4) include links to other URIs to increase the chance of discovering new related data.

Thus, we have analyzed the main differences between both types of repository format to be able to define a transformation method for the integrated information in a KB. The aim of this transformation method is to translate an ontological model and its instances into RDF triples that meet the LOD principles. The KB provides a formal representation of a domain knowledge, which can be processed and consulted by third-party applications. Therefore, applications can gather the instances of a particular concept, and search through the ontology model for its related instances. Since a KB is defined by using ontology languages, such as OWL, its transformation into a RDF dataset is less laborious than from other types of repositories like relational databases. However, we had to adapt its content to fulfill the LD principles, which required performing the tasks described next: Avoiding blank nodes, Transformation of URIs, Linking KB instances with LOD instances, and refactoring OWL punning.

Blank nodes are resources that lack a public URI. Guidelines for publishing information as LD [220] discourage the use of blank nodes because it is impossible to set external RDF links to a blank node, and merging data from different sources becomes difficult. The URIs of RDF datasets must be defined to be accessed with the HTTP protocol. The URIs of resources had to be changed in order to deploy the Web server.

Besides, to get more manageable URIs we can add labels in the URIs for distinguishing between instances and concepts, depending whether the data belongs to the schema or to the instances level. For example the URI *http://miuras.inf.um.es/ogolod/ontology/Gene* indicates the class *Gene* of a

ontology and the URI *http://miuras.inf.um.es/ogolod/resource/Gene/67440* indicates the instance of the class *Gene* with the gene identifier 67440. By adding this information to the URIs the users can understand the type of information they are accessing. Moreover, we have included the class of the instances in its URIs to identify the type of the instances. We also replaced some characters with special meaning in the HTTP URL specification, such as "#" or ":", to prevent the use of characters with different meaning to the one established in the document RFC3986 [219]. When translating KB URIs we have also to refactor the OWL punning resources. OWL punning allows us to use the same URI in an instance and a class. Both resources are the same but depending on the context it can be used in a different way. Because RDF-based data exploitation cannot use the axiomatic context, we should assign a different URI to the class and the instance resource.

To link the KB instances, which are translated into the RDF dataset, to another datasets in the LOD cloud we had to look for appropriate instances in such datasets. This task can manually performed by searching in the Data Hub [229] search interface of CKAN for other repositories with the same instances. To link an instance definition to another instance definition in another dataset, we can generate an external connection through the *owl:sameAs* relationship. In case of class definition we can use the *owl:EquivalentClass* relationship, whereas the *owl:EquivalentProperty* relationship can be used for properties definitions.

The generic architecture of a RDF dataset which is published in the LOD is represented in the Figure 9.3. The requirements are that the data has been published in RDF format, and also it can be consulted by a HTML interface. Therefore, we need a RDF management server to store the RDF triples, such as Virtuoso server [213] which is able to manage RDF dataset repositories. The RDF server must provide a SPARQL endpoint interface to access its content. Moreover, we require a server that offers the HTML interface for data consultation. The Pubby [216] server is an example of a server which provides search interfaces based on HTML and LD to publish the data. This type of servers use the SPARQL endpoint offered by the RDF management server to provide the information of the search interface.

14.5. Validation

We have validated the results of this thesis by applying them on the information domains of clusters of orthologous genes and human genetic diseases. As a result of this validation we have obtained the OGO system, which encompasses all the results developed in this thesis and in particular represents the system which allows searching and exploiting the information.

The construction of the OGO system was developed during some phases. This first step was to apply the integration methodology to several bioinformatics repositories about orthologs [224]. Then, the OGO system was also extended to integrate a genetic disease repository [227]. The implemented integration process was used to automatically integrate such bioinformatics repositories into the KB model by allowing the customization of mappings and avoiding duplicates.

The repositories used to gather the knowledge about orthologous genes domain were KOG, HomoloGene, Inparanoid and OrthoMCL, whereas OMIM was used to obtain the knowledge about genetic diseases domain. Moreover, the definition of the global ontology for the semantic integration methodology entailed the reuse of some existing bio-ontologies: the GO, the ECO, the RO, the NCBI Organismal Classification and the HPO.

14.5.1. The OGO ontology

During the construction of the OGO system a global ontology was defined: The OGO ontology. The Figure 11.1 shows the root concepts of this global ontology and the properties and relationships defined. The Namespace of the defined ontology is *http://miuras.inf.um.es/ontologias/OGO.owl#*. This ontology imports the bio-ontologies mentioned above and conceptualizes the orthologs and genetic diseases domains related to the indicated biomedical repositories. The imported bio-ontologies enrich the data annotation and promote the reuse of a shared terminology.

The main knowledge about orthologs in the OGO ontology is related these concepts: *Gene*, which represents a gene and has the properties *Gene_name* and *Gene_identifier*; *Protein*, which represents a protein and has the proper-

ties *Protein_name* y *Protein_identifier*; *ClusterOrthologs*, which corresponds to a cluster of orthologous genes; and the concept *Resource*, which represents the different bioinformatics repositories related to this thesis. The genes and proteins are linked through the relationship *isTranslatedTo*, to indicate that a particular gene is translated into a particular protein. And by *encodedBy* we indicate its inverse relationship. The *isTranslatedTo* relationship is defined with a minimum cardinality of zero because it is possible that a gene does not codify any protein. The *fromResource* relationship associates an instance of a cluster with the repository where the information was gathered and every cluster must have one resource associated. The *hasOrtholog* relationship connects the cluster with their orthologous genes. Moreover, a cluster must contain more than one ortholog.

The knowledge associated to the genetic disease domain was represented in the global ontology by: the *GeneticDisease* concept, which represents a genetic disease record, and has the *GeneticDisease_name*, *GeneticDisease_identifier* and *Location* properties; the *Method* concept indicates the methods used to obtain the information about the disease, such as "Deductions from the amino acid sequence of protein"; the *PubMedArticle* concept, which represents the instances of scientific reports published in PubMed [225], and it has the properties *PubmedArticle_name* and *PubmedArticle_identifier*. The instances of genetic diseases must be connected to the gene which cause the disorder by the *causedBy* relationship. The *connectedTo* relationship associates among themselves. Genetic diseases are connected to the methods instances by the *hasMethod* relationship. Finally, the *relatedPubmedArticle* relationship links the instances of genetic diseases with the instances of PubMed papers.

Regarding the imported bio-ontologies, we incorporated their knowledge into the global ontology but we adapted the bio-ontologies in order to make its knowledge available in the KB. The main advantages of sharing bio-ontologies is that knowledge has been reviewed by many users, so it has a minimum level of quality, plus it allows links between ontology models and avoids some redundancy in the definition of the ontology. The imported bio-ontologies are part of the OBO Foundry, they are candidates and members ontologies that

have been developed and reviewed by domain experts. These bio-ontologies mainly defines concept hierarchies and some relationships among themselves. However, the integration process creates instances of the global ontology with the information sources, so it needs instances of the bio-ontologies concepts to relate both resources. In order to provide such instances we used the Punning [100] feature of OWL2 to achieve our goal. This feature permits to associate the same resource URI to an ontology class and an ontology instance, but the instance has to have the class as its type class. Hence, the resource is liable to be a class or an instance depending on its context. The Figure 11.2 shows an example of the Punning feature. The *go:GO_0005623* resource is considered a class through the *rdfs:subClassOf* relationships with the class *go:GO_0005575*. On the other hand, *go:GO_0005623* is considered an instance through the *ro:located_in* relationship with the instance of the Gene class, *ogo:Gene/4852*.

Next, I will remark upon the contributions of these bio-ontologies:

Gene Ontology (GO): Each GO perspective defines hierarchy of concepts where the *GO_0005575* concept is the root of the cellular component ontology, the *GO_0003674* concept is the root of the molecular function ontology, and the *GO_0008150* concept is the root of the biological process ontology. Gene instances are related to GO terms through the *participates_in* and *located_in* relationships. We have created a hierarchy of relationships where each relationship is annotated with a concept of the Evidence Code Ontology (ECO). Therefore, the hierarchy of relationships is similar to the hierarchy of concepts in ECO. The Namespace of the adapted resources from the GO is *http://miuras.inf.um.es/ontologias/GO.owl#* instead of *http://purl.org/obo/owl/GO#*. Besides, there are links between both models through the *owl:sameAs* relationship.

Evidence Code Ontology (ECO): This ontology provides a hierarchy of concepts and the root concept is *ECO_0000000*. The concepts of this ontology were used to annotate the hierarchy of relationships of *participates_in*, *located_in* and *hasPhenotype*, through the annotation prop-

erty,*evidenceCode*. Its new Namespace is *http://miuras.inf.um.es/ontologias/ECO.owl#* and it replaces the Namespace *http://purl.org/obo/owl/ECO#* but the concepts between both models are related through the *owl:sameAs* relationship.

OBO Relationship Types (RO): RO defines a set of relationships to be used by other bio-ontologies. For example, the *is-a* relationship is semantically equivalent in GO and in the Celltype ontology [172]. We imported the *participates_in* and *located_in* relationship of this ontology. The Namespace of such relationships is the same as the global ontology and they are related to the original definition through the *owl:sameAs* relationship.

NCBI Organismal Classification (NCBITaxon): This ontology represents a hierarchy of classes of a species taxonomy. The root concept is *NCBITaxon_1* and represents any species. The instances of this ontology are related to genes instances through *fromSpecies* relationship. Its Namespace is *http://miuras.inf.um.es/ontologias/NCBITaxon.owl#* and it replaces *http://purl.org/obo/owl/NCBITaxon#*.

Humans Phenotype Ontology (HPO): The HPO contains a hierarchy of concepts which describes humans phenotypes owing to a genetic disease. The *HP_0000001* concept is the root concept of the HPO. Their concepts are related to genetic disease instances of the KB through the *hasPhenotype* relationship. This relationship was extended with a hierarchy of relationships which were annotated with the concepts of the ECO. The Namespace *http://miuras.inf.um.es/ontologias/HP.owl#* replaced the original *http://purl.org/obo/owl/HP#* namespace.

The reasoning features of OWL were exploited for checking the consistency of the OGO ontology model. However, due to the size of the bioinformatics repositories, current reason engines, such as Pellet or HermiT, cannot support the consistency or inference features over the entire OGO KB.

14.5.2. The OGO KB

The integration process requires the definition of the mapping and identity rules which take into account the classes and relationships of this ontology. The execution of the integration process generated the OGO KB. The architecture of the OGO KB is showed in Figure 11.3. It uses the Jena API to manage the OGO ontology and to create and consult the OGO KB. Jena provides an SPARQL interface to retrieve information and is also able to utilize some reason engines, such as Pellet. Besides, it uses the MySQL database management server to store the annotated data.

The volume of the resulted KB is summarized in the Table 11.1. The OGO KB contains nearly 2 million annotated instances and also the imported bio-ontologies resources. The integrated instances is associated to the ABox of the KB whereas the OGO ontology terms belong to the TBox.

14.5.3. Exploitation of the OGO KB

This section presents the interfaces developed for searching and exploiting the content of the OGO KB. We have developed some search interfaces in order to the KB be consulted by users and applications. The developed interfaces aim exploit the knowledge of the OGO KB. We used Semantic Web Technologies, such as OWL, RDF or SPARQL, to exploit its knowledge and annotated data. The search interfaces are focused on the Web to access and retrieve the integrated information. Because the KB consists of orthologs and genetic disease information, the search interfaces provide two different approaches, one for each type of information domain. But from each perspective we are able to retrieve information from the other perspective allowing traslational researches.

Firstly, we present the search interface which allows retrieving orthologs and genetic disease by using web forms. Secondly, we present the use of the advanced search interface (see Section 14.4.4) to facilitate the definition of SPARQL queries in the OGO KB. Thirdly, we present some implemented Web Services that were developed to retrieve information from the OGO KB by autonomous applications.

We developed a web interface to search the information of the KB via Internet from two different perspectives: orthologs and genetic diseases. So, we implemented a different search form for each perspective. The search methods consist of text box where users can indicate their search parameters, and check box and drop-down list where users can indicate the parameters to filter the results. The interfaces utilize the parameters to automatically generate the corresponding SPARQL to consult the KB. The results are showed as a table in a web page.

The first web form is aimed to retrieve orthologs-centered information. The Figure 12.1 shows the ortholog form. This form provides a text box to introduce the name or identifier of the orthologous gene. Besides, it provides two drop-down lists to limit the search space to particular bioinformatics repositories, where the information was collected, and to a particular set of species. To specify which information the users are interested in, the form provides check boxes to retrieve information about alternative gene names, proteins information, GO terms, its repository sources and its related genetic disorder, if they exist. The related SPARQL template used to build the final query to search the KB for orthologs information is showed in Figure 12.2. This SPARQL template utilizes the search parameters that are provides by users to generate its query.

On the other hand, the second web form is focused on genetic diseases perspective. This form provides a text box to indicate the genetic disease name that users are searching for. The name is considered as a search token, so the interface can find more than one disease as a result in the KB. These results are showed a list where users can select one. A genetic disease result retrieves the information related to the genetic disease concept of the global ontology of the OGO KB. The results of both forms provide links associated with information descriptions in order to navigate from one resource to another in the OGO KB and to link them to other bioinformatics repositories. Its SPARQL template is showed in Figure 12.6 and it uses the genetic disease name or identifier to retrieve genetic disease records from the OGO KB.

These two web forms permit users to access the information of the OGO KB without defining its own SPARQL query. These interfaces were designed

to be used by users with a limited knowledge about Semantic Web Technologies. They exploit the semantic of the OGO KB to gather and filter its content, however, they do not allow users to define their own queries. To allow this functionality, we developed another web interface which provide users with a tool where they can define advanced queries.

The advanced query interface guides the users through the query definition by providing them with recommendation based on the knowledge of the OGO KB. The users do not have to know SPARQL grammar as in traditional SPARQL endpoint and ontological model of the OGO KB.

The Figure 12.10 shows the main view of the advanced query interface with an example of query definition. To select the variables for the query the interface provides the hierarchy of classes of the ontology (see Figure 12.11). Once variables have been defined, the users can start defining the conditions of the query. The Figure 12.12 shows the different requirements possible for defining the query. These possibilities guides the query definition and hinder mistakes in defining requirements.

The interfaces described above were designed to be used only by users. However, we also offered autonomous applications with access to the content of the OGO KB. So, during the project *"Exploitation of Traslational Bioinformatics for Decision-Making on Cancer Treatments"* [230], we developed search interfaces which were designed to be used by applications to consult the OGO KB. These interfaces offer the same functionalities to applications as the interfaces described above offer to users. In this project, we provided the ONCOdata system with access to our KB. The ONCOdata is a decision support system which gives recommendations for the treatment of cancer diseases based on clinical evidences. It gathers the clinical evidences from the patient's clinical health record, and provides the most suitable recommendations for its cancer disease. The project aim was to expand the functionality of the ONCOdata system by adding biomedical justification about the provided disease diagnosis and treatment recommendations. Hence, we developed three web services that allow applications to access the OGO KB via Internet.

1. The web service that can provide information about cluster of orthologous genes from its gene name or identifier as parameter.

2. The web service which allows gathering information about genetic diseases from its identifier or name as parameter.

3. The web service that provides an SPARQL endpoint in order to applications can define their own queries to consult the OGO KB.

We have used Web services because such technology offer loosely coupled communication, and text-encoded data and messages. The widespread adoption of SOAP and WSDL standards together with HTTP and XML facilitate developers to adopt and less costly to deploy web services.

To implement the first two web services (orthologs and genetic disease search interfaces) we defined two separate SPARQL templates (see Figures 12.2 and 12.6). Such templates can be combined with the parameters passed through the web services to generate the corresponding SPARQL queries. The results are returned as a XML document to the client of the web service. For example, the Figure 12.17 shows an example of the results obtained from the genetic disease centered web service. These results are related to the "breast cancer" genetic disease. On the other hand, the SPARQL endpoint web service receives the SPARQL query as parameter. This web service allows advanced query definition although the client applications are responsible for the query accuracy. So, applications should process the global ontology knowledge and be able to define SPARQL queries in order to consult the KB. Its results consist of a list of XML elements associated with a SPARQL query variable.

To sum up, these web services allow applications to navigate the content of the KB. The first two web services use search tokens as parameter to gather information. The results and these web services can be used in traslational bioinformatics research. An application can exploit these web services to search the KB in order to discover new relationships among the annotated data. The last web service provides more freedom in the query definition than the others web services. Besides, applications can take advantage of semantic web technologies to exploit the knowledge of the KB to define suitable

SPARQL queries. Therefore, these web services are provided for searching and exploiting the content of the KB by autonomous applications.

14.5.4. Publishing the OGO KB into the LOD cloud

In this subsection we describe the transformation of the OGO KB into a RDF dataset, called OGOLOD. Besides, this dataset was published in the Linking Open Data(LOD) initiative. OGOLOD [5] was first published in the Data Hub from September 2011, a public registry of open datasets which is powered by CKAN. The OGOLOD dataset meets the Linked Data principles.

To transform the content of our KB, we removed the blank nodes to facilitate navigating the resulted RDF dataset. In case of OGOLOD, the triple links between genes, evidence codes and GO terms were refactored to satisfy this requirement. Gene products are related to GO terms through *participates_in* and *located_in* relationships. Furthermore, evidence codes represent the experimental evidence for these relationships: a gene product participates in a biological process with a certain evidence. This structure was represented in the OGO KB using blank nodes, where gene instances were related to blank nodes and the blank nodes were linked to the corresponding GO terms and evidence codes. However, we extended the *participates_in* and *located_in* relationships by adding a hierarchy of subproperties that represent each type of evidence code. Thus, *participates_EXP_in*, which is a subproperty of *participates_in*, is used for a connection between a gene, a GO term and the evidence code *EXP* (Inferred from experiment). This solution was also applied to the triple relationship between a genetic disease, a human phenotype and an evidence code.

The URIs of the OGOLOD dataset must be defined to be accessed with the HTTP protocol. The URIs of the resources had to be changed in order to deploy the Web server of the OGOLOD system. Besides, to get more manageable URIs, we added in the definition of the URIs the label *ontology/* or the label *resource/*, depending whether the data belongs to the schema or to the instances. For example the URI *http://miuras.inf.um.es/-*

[5] `http://thedatahub.org/dataset/ogolod`

ogolod/ontology/Gene indicates the class *Gene* of the ontology and the URI *http://miuras.inf.um.es/ogolod/resource/Gene/67440* indicates the instance of the class *Gene* with the gene identifier 67440.

To link the OGOLOD dataset resources to other datasets in the LOD cloud we had to look for appropriate instances in such datasets. We found the datasets which contain instances from Homologene, Entrez Gene, Pubmed, OMIM, GO, NCBI taxonomy, Uniprot KB, Entrez Protein and HPO bioinformatics repositories and bio-ontologies. These external repositories are part of the Bio2RDF and Linked Life Data projects. Table 12.1 shows an example of the URIs of external datasets and the equivalent URIs of the OGOLOD repository for each integrated bioinformatics repository and bio-ontology. Since the OGOLOD dataset and the external datasets share the identifiers and accession codes used by the biomedical resources, we can automatically transform the resource URIs of the OGOLOD into the external datasets URIs. Figure 12.7 shows examples of the different types of links of the OGOLOD dataset to external datasets.

Besides, we used punning was to import OBO ontologies in the OGO ontology. Therefore, to create the OGOLOD dataset we had to use different URIs for punned instances and classes. Thus, we extended the imported bio-ontologies with their corresponding instances for each concept in the OGOLOD dataset.

The SPARQL endpoint associated to the OGOLOD dataset is provided in [6]. Virtuoso server was used to store the generated RDF dataset. This server extends the SPARQL grammar implementation to provide SPARQL language with more semantic functionalities. The Figure 12.8 shows an example of a SPARQL query which uses the *TRANSITIVE* option. This functionality allows to declare transitive any relationship of the ODOLOD dataset. So, the example declare transitive the *"rdfs:subClassOf"* relationship in order to exploit the hierarchy of concepts of the OGOLOD dataset.

The HTML interface of the Pubby server allows to consult the OGOLOD resources through a web browser. The Figure *Fig:navigation* shows an example of the web page which contains the information about the *"http://miu-*

[6] `http://miuras.inf.um.es/sparql`

ras.inf.um.es/ogolod/resource/GeneticDisease/100070" resource. Besides, the OGOLOD dataset can be consulted through the Linked Data format by using RDF browser such as Disco [217] o Tabulator [218]. We can also navigate from one dataset to another dataset in the LOD cloud because we provided OGOLOD with links to external repositories. Therefore, we can enrich the content of the OGOLOD with the connections provided by other datasets.

14.6. Conclusions and future work

In this thesis a methodology for the semantic management and integration of biological and biomedical information was presented. We utilized a knowledge base (KB) to store the integrated information and its corresponding knowledge. This KB allows us to use Semantic Web technologies to exploit its knowledge and to consult its content via SPARQL queries. The increasing size and complexity of bioinformatics repositories can be handled from a semantic approach, as we have presented in this thesis. Having powerful enough tools to allow advanced queries definitions on bioinformatics data repositories is a main requirement in current biological research. Furthermore, since we integrated bioinformatics repositories, we defined a formal representation of its knowledge domain and we provide a set of search interfaces designed to be used by humans and autonomous applications, we can facilitate traslational research between the domains of biology and biomedicine.

One of the most important contribution of this thesis is the definition of a methodology to integrated traditional bioinformatics repositories into semantics repositories. The traditional repositories usually provide their data in a human friendly formats that are difficult for autonomous application to analyze and process. To process their data we have to deal with heterogeneous repositories and search interfaces that were designed to be consulted by humans. So, the aim of our methodology is to integrate such bioinformatics repositories into a new repository which allows to exploit its content by applications automatically and to avoid ambiguity problems. The methodology requires the definition of rules to map the bioinformatics repositories and a global ontology, which represent the knowledge of such repositories. So, the

mapping rules connect the bioinformatics repository schemas with the global ontology that can be considered as the schema of the KB. We can extend the content of the KB by extending the definition of the global ontology and defining the corresponding mapping rules. So, we facilitate the maintenance and updating the created semantics repositories.

To integrate different repositories that reference similar instances and to avoid such redundancy, we need to define how the instances can be identified to find their equivalents. For example, bioinformatics repositories usually define accession numbers that are used to identify genes instances in their local repositories, but also they use names that are usually used to identify the instances by biologists, such as the *ZFHX3* gene symbol which corresponds to *463* gene identifier from Entrez Gene repository. So, we added identity rules to the integration methodology in order to define the requirements that two instances must meet to be considered equivalent. These identity rules are based on properties and relationships of the ontology definition, i.e. we define the requirements that the instances of Gene class of our global ontology must meet to be consider equals. So, we require that the gene instances belongs to the same organism and also share at least one of their gene names or gene identifiers.

One of the main problems we have found when retrieving data from bioinformatics repositories was the heterogeneous data and the abundance of different representation formats. Besides, the information is usually represented in semi-structured formats or in natural language. In order to design and develop a method for integrating bioinformatics repositories we need to consider a limited set of data storage formats. Since each bioinformatics repository, that we integrated, defines its own data structure we decided to start from a common data representation format. We decided to use relational repository format because of its efficiency and scalability. Relational repositories can be easily consulted by applications but they lack for their data semantics in order to allow applications to autonomously consult their data. Therefore, in order to integrate the information of the bioinformatics repositories, firstly we have to transform them into relational repositories.

As a result of the integration methodology we obtained our KB. This KB

uses the global ontology as its information schema. We can use the ontology to automatically navigate through the instances of the KB. Therefore, we new relationships that were not provided individually by the integrated repositories. The KB can be managed at a semantic level because of the global ontology and Semantic Web technologies functionalities.

In particular, we use the ontological model APIs to access the KB like an ontology model, such as Jena. So, we can retrieve information from a higher abstraction level than traditional bioinformatics repositories. Thus, we can reduce ambiguity errors among different instances of the KB when applications access its content, whereas traditional repositories do not formally indicate the type of data that could be collected from them. Moreover, we can use semantic query languages, such as SPARQL to consult the repository. SPARQL is focused on querying RDF graphs, so it defines RDF triples and variables that match the triples of the repository. However, SPARQL does not provide an inference layer to exploit the semantics of the OWL vocabulary. Besides, users need to learn how to use the language and the structure of the RDF graphs like as learning how to use SQL and the structure of relational schemas. Therefore, biologists might find hard to use SPARQL directly when querying the KB.

In this thesis we introduce the OGO system, the platform which brings together all the methodologies and tools defined to validate the work of the thesis. In particular, it was generated by following our semantic integration methodology. We obtained as a result of this methodology the OGO global ontology and the OGO KB. Besides, we designed and implemented some search interfaces to exploit the content of the OGO KB. As we provide a set of tools to exploit our KB by autonomous applications, we laid the foundations of new tools that avoid carrying out manually tasks that can be automated, such as comparing results data of their experiments or finding new relationships between genome sequences and humans phenotypes based on previous experiment results.

The OGO global ontology conceptualizes the knowledge of the clusters of orthologous genes and genetic diseases domains. It connects instances through relationships that permit to navigate the integrated information.

The aim of this ontology is to provide a formal representation to avoid ambiguity problems during the process of the construction of the OGO KB. Therefore, we can use reason engines in order to exploit the semantics of the integrated repository. However, to process large amounts of information using reasoners, we need large amounts of virtual memory and processing time that may limit its use in a final product scenario. Currently, such technologies are not powerful enough to process large amount of data and have lacks of scalability. On the other hand, our global ontology imports other bio-ontologies which improve the quality of its domain knowledge conceptualization. These imported bio-ontologies provide a shared terminology that might be used by third party applications in order to automatically explore our KB. Moreover, the global ontology is used to guide the integration process and hence it facilitates the management of the OGO KB.

In order to exploit the integrated data of our KB, we have developed various query interfaces. So, we first designed two types of query interfaces depending on their target users. Users are provided with two query interfaces based on web forms and one system for guidance on definition of advanced queries. If the target users are applications we have deployed web services for the consultation through query parameters and through SPARQL queries directly.

We developed two query interface for allowing biologists consult the KB from two different perspectives using web forms. The first perspective is orthologous gene centered whereas the second perspective is genetic disease centered. The first perspective allows us to search for orthologous gene related information in the OGO KB. The second perspective permits to search the information related to genetic disease concept in the KB. These interfaces allows users to gather information about genetic diseases from orthologous genes and the other way around. Therefore, they facilitates the biomedical traslational research. However, query interfaces based on web forms are not flexible enough to define more advanced queries. So, we utilized the designed query interface that allows users to define their own advanced SPARQL queries with our OGO KB. This interface exploit the OGO global ontology related to the KB to guide users through the definition of advanced

queries. Thus, the interface uses the description of the ontological model to provide recommendations to users during the query definition. Such recommendations limit the query definition by providing only the allowed triples and variables to the users in the advanced query. Consequently, users do not have to be aware of the grammar of the SPARQL language and also the interface suggests the classes, properties and relationships to users. Therefore, they do not have to know all the ontological model. However, the major disadvantage of this interface is that users are required to follow step by step the query definition so that the process can be longer than directly define their queries manually.

To allow autonomous applications to access the content of the OGO KB we developed three web services. The first and second web service were developed to query the repository through the parameters that describe orthologous genes and genetic diseases respectively. The parameters were used to define the corresponding SPARQL queries and to retrieve their related information. These web services do not require users to use any semantics query language, so client applications can consult and process the results by using XML documents. The third web service was developed to allow applications to define their own SPARQL queries, so it provides a SPARQL endpoint to consult the OGO KB. Autonomous applications can use this web service to navigate through the KB and to obtain the precise information they require.

Providing the Web with semantics to information can be understood by both applications and humans is one of the goals behind the Semantic Web. In order to contribute to the Semantic Web, we published the data of the OGO KB into the Linking Open Data (LOD) cloud as. The resulted dataset is called the OGOLOD. Thus, the information of the OGO KB can be consulted from the web and to find the relationships with other published datasets. The data are accessible through a HTML browser using their instance URIs. We had to transform the content of the OGO KB to meet the requirements of the LOD cloud. The LOD provides some principles that the dataset must achieve in order to be published, such as the data URIs have to be accessible via Web, the instances should contain descriptive information or the dataset has to be linked to other datasets in order to extend the

connections among published data. The published data is stored in a RDF graph which can be only consulted through SPARQL queries. Unlike the OGO KB repository, the OGOLOD dataset cannot be exploited using reasoners because it is not related to an ontology model that can be processed. Thus, although the semantics of the information has been defined it cannot be exploited by inference engines. This published data set provides the basis to develop new tools that can process the information about orthologs and genetic diseases.

The current version of the OGO system provides relationships between genome and phenotype information. But, as future work we could extend the content of the OGO KB with related medical treatments and drugs. This information could be used to suggest clinical trials on alternative species for traslational researches.

At present, when updating the OGO KB, we have to integrate again all bioinformatics repositories. As the repositories have different update times, we could take this into account and design a method to update the different repositories without the need to update all the OGO KB. Therefore, each repository which has been updated may contain new, obsolete and / or modified information that must be treated differently. This method could reduce the time of each update, and also allow the KB to keep as updated as possible.

One of the results of this thesis is the query interface which guides users during the definition of advanced SPARQL queries to consult a KB. As a future work, we could research the design of user interfaces that facilitates the definition of semantic queries on semantic repositories. The disadvantages of using semantic languages, such as SPARQL, are the difficulties that users have in order to define their own queries when they are not experts in the language or in the repository schema. So, if we could provide a friendlier query interface, we may facilitate the use of OGO KB by biologists and physicians.

Another future extension of the OGO KB might be the integration of data from archetypes. An archetype is defined as a formal model of a clinical concept. Each archetype defines a set of restrictions on a particular

information domain, which represent a clinical concept. We have already proposed an initiative to exploit the semantics of archetypes by applying the advanced search interface that guides users for the definition of semantic queries [231, 232].

To sum up, we have designed a framework to facilitate the management of biological and biomedical information by means of using Semantic Web Technologies for integrating and publishing data. We defined a methodology to integrate into a knowledge base the information and the semantics of the domains of the bioinformatics repositories. This knowledge base allows us to maintain and exploit its content from a semantics level that facilitates the development of advanced applications. We also developed an automated method for integrating and avoiding redundancy of information in the repository. This method was used to create the OGO KB. Furthermore, we developed some query interfaces that were designed to be used by humans an applications in order to consult the integrated information of the OGO KB. These interfaces are based on the use of SPARQL. This language provides an efficient method for navigating the repository through the classes, properties and relationships of our defined global ontology. The global ontology, associated to our OGO system, conceptualizes the domains of the bioinformatics repositories. Besides, the global ontology imports other bio-ontologies which enrich its knowledge and allows sharing a common vocabulary to explore the OGO KB by third party applications. The relationships defined through the gene and genetic disease concepts allows us to connect the genomic information with the clinical information collected from the bioinformatics repositories. Therefore, the OGO KB may facilitate traslational researches between biological and biomedical domains.

Bibliografía

[1] Alan Ruttenberg, Tim Clark, William Bug, Matthias Samwald, Olivier Bodenreider, Helen Chen, Donald Doherty, Kerstin Forsberg, Yong Gao, Vipul Kashyap, June Kinoshita, Joanne Luciano, M. Scott Marshall, Chimezie Ogbuji, Jonathan Rees, Susie Stephens, Gwendolyn T. Wong, Elizabeth Wu, Davide Zaccagnini, Tonya Hongsermeier, Eric Neumann, Ivan Herman, and Kei-Hoi Cheung. Advancing translational research with the semantic web. *BMC Bioinformatics*, 8(3):S2, 2007.

[2] F. Horn, G. Vriend, and F.E. Cohen. Collecting and harvesting biological data: the gpcrdb and nucleardb information systems. *Nucleic acids research*, 29(1):346, 2001.

[3] Su Yun Chung and Limsoon Wong. Kleisli: A new tool for data integration in biology. *Trends in Biotechnology*, 17(9):[351,355], 1999.

[4] M.K. Kerr and G.A. Churchill. Statistical design and the analysis of gene expression microarray data. *Genet. Res. Camb*, 77:[123,128], 2001.

[5] Jun Zhang, Rod Chiodini, Ahmed Badr, and Genfa Zhang. The impact of next-generation sequencing on genomics. *Journal of Genetics and Genomics*, 38(3):[95,109], 2011.

[6] M.Y. Galperin and X.M. Fernández-Suárez. The 2012 nucleic acids research database issue and the online molecular biology database collection. *Nucleic Acids Research*, 40:[D1,D8], 2012.

[7] R. Altman, A. Valencia, S. Miyano, and S. Ranganathan. Challenges for intelligent systems in biology. *IEEE Intelligent Systems*, 16(6):[14,20], 2001.

[8] Uniprot Consortium. Uniprot. Recurso disponible on-line: `http://www.uniprot.org/`.

[9] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):[34,43], Mayo 2001.

[10] W3C. World wide web consortium. Recurso on-line: `www.w3c.org`.

[11] O. Bodenreider and R. Stevens. Bio-ontologies: current trends and future directions. *Briefings in Bioinformatics*, 7:[256,274], 2006.

[12] B. Good and M. Wilkinson. The life sciences semantic web is full of creeps. *Briefings in Bioinformatics*, 7(3):[257,286], 2006.

[13] E. Antezana, M. Kuiper, and V. Mironov. Biological knowledge management: the emerging role of the semantic web technologies. *Briefings in Bioinformatics in Bioinformatics*, 10(4):[392,407], 2009.

[14] W3C. Resource description framework (rdf). Recurso disponible on-line: `http://www.w3.org/RDF/`.

[15] W3C. Lenguaje de consultas sparql. Recurso on-line: `http://www.w3.org/TR/rdf-sparql-query`.

[16] W3C. Owl 2 web ontology language. Recurso on-line: `http://www.w3.org/TR/owl2-quick-reference/`.

[17] T. Attwood, D. Kell, P. McDermott, J. Marsh, S. Pettifer, and D. Thorne. Rescuing knowledge lost in literature and data. *Biochemical Journal*, 424:[317,333], 2009.

[18] C. Brewster, K. O'Hara, S. Fuller, Y. Wilks, E. Franconi, M. A. Musen, J. Ellman, and S. Buckingham Shum. Knowledge representation with ontologies: The present and future. *IEEE Intelligent Systems*, 19(1):[72,81], 2004.

[19] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P Davis., K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J.C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nat Genet*, 25(1):[25,29], Mayo 2000.

[20] Barry Smith, Michael Ashburner, Cornelius Rosse, Jonathan Bard, William Bug, Werner Ceusters, Louis Goldberg, Karen Eilbeck, Amelia Ireland, Christopher Mungall, The OBI Consortium, Neocles Leontis, Philippe Rocca-Serra, Alan Ruttenberg, Susanna-Assunta Sansone, Richard Scheuermann, Nigam Shah, Patricia Whetzel, and Suzanna Lewis. The obo foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology*, 25:[1251,1255], 2007.

[21] Jose A. Miñarro-Gimenez, Marisa Madrid, and Jesualdo T. Fernández-Breis. Ogo: an ontological approach for integrating knowledge about orthology. *BMC Bioinformatics*, 10(Suppl 10):S13, 2009.

[22] F. Wu, L. Mueller, D. Crouzillat, V. Petiard, and S. Tanksley. Combining bioinformatics and phylogenetics to identify large sets of single-copy orthologous genes (cosii) for comparative, evolutionary and systematic studies: a test case in the euasterid plant clade. *Genetics*, 174(3):[1407,1420], 2006.

[23] Erick Antezana, Mikel Egaña, Ward Blondé, Aitzol Illarramendi, Iñaki Bilbao, Bernard De Baets, Robert Stevens, Vladimir Mironov, and Martin Kuiper. The cell cycle ontology: an application ontology for the representation and integrated analysis of the cell cycle process. *Genome Biology*, 10(5):R58, 2009.

[24] K. Harvey, C. Pfleger, and I. Hariharan. The drosophila mst ortholog, hippo restricts growth and cell proliferation and promotes apoptosis. *Cell*, 114:[457,467], 2003.

[25] W3C SWEO Community Project. Linking open data. Recurso disponible on-line: `http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData`.

[26] ENA. Eurpean nucleotide archive. Recurso disponible on-line: `http://www.ebi.ac.uk/ena/`.

[27] D.A. Benson, I. Karsch-Mizrachi, D.J. Lipman, J. Ostell, and E.W. Sayers. Genbank. *Nucleic Acids Research*, 37:[D26,D31], 2009.

[28] Y. Tateno, T. Imanishi, S. Miyazaki, K. Fukami-Kobayashi, N. Saitou, H. Sugawara, and T. Gojobori. Dna data bank of japan (ddbj) for genome scale research in life science. *Nucleic Acids Research*, 30:[27,30], 2002.

[29] Carola Kanz, Philippe Aldebert, Nicola Althorpe, Wendy Baker, Alastair Baldwin, Kirsty Bates, Paul Browne, Alexandra van den Broek, Matias Castro, Guy Cochrane, Karyn Duggan, Ruth Eberhardt, Nadeem Faruque, John Gamble, Federico Garcia Diez, Nicola Harte, Tamara Kulikova, Quan Lin, Vincent Lombard, Rodrigo Lopez, Renato Mancuso, Michelle McHale, Francesco Nardone, Ville Silventoinen, Siamak Sobhany, Peter Stoehr, Mary Ann Tuli, Katerina Tzouvara, Robert Vaughan, Dan Wu, Weimin Zhu, and Rolf Apweiler. The embl nucleotide sequence database. *Nucleic Acids Research*, 33:[D29,D33], 2005.

[30] International INSDC Advisory Committee. International nucleotide sequence database collaboration. Recurso disponible on-line: `http://insdc.org/`.

[31] SGD. Saccharomyces genome database. Recurso disponible on-line: `http://www.yeastgenome.org/`.

[32] FlyBase. A database of droshophila genes & genomes. Recurso disponible on-line: `http://flybase.org/`.

[33] MGI. Mouse genome database. Recurso disponible on-line: `http://www.informatics.jax.org/`.

[34] Barbara R. Jasny and Donald Kennedy. The human genome. *Science*, 291:1153, 2001.

[35] The Jackson Laboratory. The jackson laboratory. Recurso disponible on-line: `http://www.jax.org/`.

[36] E.W. Sayers, T. Barrett, D.A. Benson, S.H. Bryant, K. Canese, V. Chetvernin, D.M. Church, M. DiCuccio, R. Edgar, S. Federhen, M. Feolo, L.Y. Geer, W. Helmberg, Y. Kapustin, D. Landsman, D.J. Lipman, T.L. Madden, D.R. Maglott, V. Miller, I. Mizrachi, J. Ostell, K.D. Pruitt, G.D. Schuler, E. Sequeira, S.T. Sherry, M. Shumway, K. Sirotkin, A. Souvorov, G. Starchenko, T.A. Tatusova, L. Wagner, E. Yaschenko, and J. Ye. Database resources of the national center for biotechnology information. *Nucleic Acids Research*, 37:[D5,D15], 2009.

[37] EBI. European bioinformatics institute. Recurso disponible on-line: `http://www.ebi.ac.uk/`.

[38] H. Ogata, S. Goto, K. Sato, W. Fujibuchi, H. Bono, and M. Kanehisa. Kegg: kyoto encyclopedia of genes and genomes. *Nucleic Acids Research*, 27:[29,34], 1999.

[39] Minoru Kanehisa. Kanehisa laboratories. Recurso disponible on-line: `http://www.kanehisa.jp/`.

[40] EMB. Identifiers.org. Recurso disponible on-line: `www.identifiers.org`.

[41] Gang Fang, Nitin Bhardwaj, Rebecca Robilotto, and Mark B. Gerstein. Getting started in gene orthology and functional analysis. *PLoS Comput Biol*, 6(3):e1000703, 03 2010.

[42] J.W. Schopf, A.B. Kudryavtsev, D.G. Agresti, T.J. Wdowiak, and A.D. Czaja. Laser-Raman Imagery of Earth's Earliest Fossils. *Nature*, 416:[73,76], 2002.

[43] Jaime E. Blair, Prachi Shah, and S. Blair Hedges. Evolutionary sequence analysis of complete eukaryote genomes. *BMC Bioinformatics*, 6:53, 2005.

[44] W.M. Fitch. Homology a personal view on some of the problems. *Trends in Genetics*, 16:[227,231], 2000.

[45] C.H. House, B. Runnegar, and S.T. Fitz-Gibbon. Geobiological analysis using whole genome-based tree building applied to the bacteria, archaea, and eukarya. *Geobiology*, 1:[15,26], 2003.

[46] W.M. Fitch. Uses for evolutionary trees. *Philosophical Transactions of the Royal Society: Biological Sciences*, 349:[93,102], 1995.

[47] R.A. Jensen. Orthologs and paralogs we need to get it right. *Genome Biology*, 2:interactions1002, 2001.

[48] E.V. Koonin. An apology for orthologs or brave new memes. *Genome Biology*, 2:[1005.1,1005.2], 2001.

[49] C. Patterson. Homology in classical and molecular biology. *Molecular Biology Evolution*, 5:[603,625], 1988.

[50] R.L. Tatusov, N.D. Fedorova, J.D. Jackson, A.R. Jacobs, B. Kiryutin, E.V. Koonin, D.M. Krylov, R. Mazumder, S.L. Mekhedov, A.N. Nikolskaya, B.S. Rao, S. Smirnov, A.V. Sverdlov, S. Vasudevan, Y.I. Wolf, J.J. Yin, and D.A. Natale. The cog database: an updated version includes eukaryotes. *BMC Bioinformatics*, 4:41, 2003.

[51] Feng Chen, Aaron J. Mackey, Christian J. Stoeckert Jr, and David S. Roos. Orthomcl db: querying a comprehensive multi-species collection of ortholog groups. *Nucleic Acids Research*, 34:[D363,D368], 2006.

[52] http://www.ncbi.nlm.nih.gov/homologene. Homologene database.

[53] M. Remm, C.V. Storm, and E.L. Sonnhammer. Automatic clustering of orthologs and in-paralogs from pairwise species comparisons. *Journal of Molecular Biology*, 314(5):[1041,1052], 2001.

[54] A. Heger and C.P. Ponting. Evolutionary rate analyses of orthologs and paralogs from 12 drosophila genomes. *Genome Research*, 17:[1837,1849], 2007.

[55] A. Alexeyenko, J. Lindberg, A. Perez-Bercoff, and E.L. Sonnhammer. Overview and comparison of ortholog databases. *Drug Discovery Today Technology*, 3:[137,143], 2006.

[56] T. Hulsen, M.A. Huynen, J. de Vlieg, and P.M. Groenen. Benchmarking ortholog identification methods using functional genomics data. *Genome Biology*, 7:R31, 2006.

[57] K. Dolinski and D. Botstein. Orthology and functional conservation in eukaryotes. *Annual Review of Genetics*, 41:[465,507], 2007.

[58] F. Chen, A.J. Mackey, J.K. Vermunt, and D.S. Roos. Assessing performance of orthology detection strategies applied to eukaryotic genomes. *PLoS ONE*, 2:e383, 2007.

[59] T. Gabaldon. Large-scale assignment of orthology: back to phylogenetics? *Genome Biology*, 9:235, 2008.

[60] A. Kuzniar, R.C. van Ham, S. Pongor, and J.A. Leunissen. The quest for orthologs: finding the corresponding gene across genomes. *Trends Genetic*, 24:[539,551], 2008.

[61] A.M. Altenhoff and C. Dessimoz. Phylogenetic and functional assessment of orthologs inference projects and methods. *PLoS Computational Biology*, 5(1):e1000262, 2009.

[62] T.J. Hubbard, B.L. Aken, S. Ayling, B. Ballester, K. Beal, E. Bragin, S. Brent, Y. Chen, P. Clapham, and L. Clarke. Ensembl 2009. *Nucleic Acids Research*, 37:[D690,D697], 2009.

[63] S. Tweedie, M. Ashburner, K. Falls, P. Leyland, P. McQuilton, S. Marygold, G. Millburn, D. Osumi-Sutherland, A. Schroeder, and R. Seal. Flybase: enhancing drosophila gene ontology annotations. *Nucleic Acids Research*, 37:[D555,D559], 2009.

[64] K.D. Pruitt, T. Tatusova, and D.R. Maglott. Ncbi reference sequence (refseq): a curated non-redundant sequence database of genomes, transcripts and proteins. *Nucleic Acids Research*, 35:[D61,D65], 2007.

[65] T. Bieri, D. Blasiar, P. Ozersky, I. Antoshechkin, C. Bastiani, P. Canaran, J. Chan, N. Chen, W.J. Chen, and P. Davis. Wormbase: new content and better access. *Nucleic Acids Research*, 35:[D506,D510], 2007.

[66] D.J. Sherman, T. Martin, M. Nikolski, C. Cayla, J.L. Souciet, P. Durrens, and Genolevures Consortium. Genolevures: protein families and synteny among complete hemiascomycetous yeast proteomes and genomes. *Nucleic Acids Research*, 37:[D550,D554], 2009.

[67] D. Lawson, P. Arensburger, P. Atkinson, N.J. Besansky, R.V. Bruggner, R. Butler, K.S. Campbell, G.K. Christophides, S. Christley, and E. Dialynas. Vectorbase: a home for invertebrate vectors of human pathogens. *Nucleic Acids Research*, 35:[D503,D505], 2007.

[68] C. Aurrecoechea, J. Brestelli, B.P. Brunk, J. Dommer, S. Fischer, B. Gajria, X. Gao, A. Gingle, G. Grant, and O.S. Harb. Plasmodb: a functional genomic database for malaria parasites. *Nucleic Acids Research*, 37:[D539,D543], 2009.

[69] M. Heiges, H. Wang, E. Robinson, C. Aurrecoechea, X. Gao, N. Kaluskar, P. Rhodes, S. Wang, C.Z. He, and Y. Su. Cryptodb: a cryptosporidium bioinformatics resource update. *Nucleic Acids Research*, 34:[D412,D422], 2006.

[70] C. Aurrecoechea, J. Brestelli, B.P. Brunk, J.M. Carlton, J. Dommer, S. Fischer, B. Gajria, X. Gao, A. Gingle, and G. Grant. Giardiadb and trichdb: integrated genomic resources for the eukaryotic protist pathogens giardia lamblia and trichomonas vaginalis. *Nucleic Acids Research*, 37:[D526,D530], 2009.

[71] M. Mi, N. Guo, A. Kejariwal, and P.D. Thomas. Panther version 6: protein sequence and function evolution data with expanded representation of biological pathways. *Nucleic Acids Research*, 35:[D247,D252], 2007.

[72] S. Ouyang, W. Zhu, Hamilton J, H. Lin, M. Campbell, K. Childs, F. Thibaud-Nissen, R.L. Malek, Y. Lee, and L. Zheng. The tigr rice genome annotation resource: improvements and new features. *Nucleic Acids Research*, 35:[D883,D887], 2007.

[73] P. Fey, P. Gaudet, T. Curk, B. Zupan, E.M. Just, S. Basu, S.N. Merchant, Y.A. Bushmanova, G. Shaulsky, and W.A. Kibbe. Dictybase: a dictyostelium bioinformatics resource update. *Nucleic Acids Research*, 37:[D515,D519], 2009.

[74] M.B. Arnaud, M.C. Costanzo, M.S. Skrzypek, P. Shah, G. Binkley, C. Lane, S.R. Miyasato, and G. Sherlock . Sequence resources at the candida genome database (cgd). *Nucleic Acids Research*, 35:[D452,D456], 2007.

[75] J. Wang, Q. Xia, X. He, M. Dai, J. Ruan, J. Chen, G. Yu, H. Yuan, Y. Hu, and R. Li. Silkdb: a knowledgebase for silkworm biology and genomics. *Nucleic Acids Research*, 33:[D399,D402], 2005.

[76] E.L. Hong, R. Balakrishnan, Q. Dong, K.R. Christie, J. Park, G. Binkley, M.C. Costanzo, S.S. Dwight, S.R. Engel, and D.G. Fisk. Gene ontology annotations at sgd: new data sources and annotation methods. *Nucleic Acids Research*, 36:[D577,D581], 2008.

[77] D. Swarbreck, C. Wilks, P. Lamesch, T.Z. Berardini, M. Garcia-Hernandez, H. Foerster, D. Li, T. Meyer, R. Muller, and L. Ploetz. The arabidopsis information resource (tair): gene structure and function annotation. *Nucleic Acids Research*, 36:[D1009,D1014], 2008.

[78] Johns Hopkins University. Online mendelian inheritance in man. Recurso disponible on-line: `http://omim.org/`.

[79] Sue Povey, Ruth Lovering, Elspeth Bruford, Mathew Wright, Michael Lush, and Hester Wain. The hugo gene nomenclature committee (hgnc). *Human Genetics*, 109:[678,680], 2001.

[80] CERN. Organización europea para la investigación nuclear. Recurso disponible on-line: `http://public.web.cern.ch/public/`.

[81] W3C. Uniform resource identifiers (uri). Recurso disponible on-line: `http://tools.ietf.org/html/rfc2396`.

[82] W3C. Html: Hyper text markup language. Recurso disponible on-line: `http://www.w3.org/html/`.

[83] W3C. Owl web ontology language. Recurso on-line: `http://www.w3.org/TR/owl-ref/`.

[84] Nicola Guarino and Christopher Welty. *The Handbook on Ontologies*, chapter An Overview of OntoClean, pages [151,172]. Springer-Verlag, 2004.

[85] Nicola Guarino and Chris Welty. Identity, unity, and individuation: Towards a formal toolkit for ontological analysis. In *Proceedings of ECAI-2000: The European Conference on Artificial Intelligence*, 2000.

[86] Nicola Guarino and Chris Welty. A formal ontology of properties. In *Proceedings of EKAW-2000: The 12th International Conference on Knowledge Engineering and Knowledge Management*, 2000.

[87] Nicola Guarino and Chris Welty. Ontological analysis of taxonomic relationships. In *Proceedings of ER-2000: The 19th International Conference on Conceptual Modeling*, 2000.

[88] G. Antoniou and F. van Harmelen. *Handbook on Ontologies*, chapter Web Ontology Language: OWL, pages [67,92]. Springer, 2004.

[89] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. *The Description Logic Handbook*. Cambridge University Press, New York, NY, USA, 2007.

[90] Matthew Horridge and Sean Bechhofer. The owl api: A java api for working with owl 2 ontologies. In *OWLED 2009, 6th OWL Experienced and Directions Workshop, Chantilly, Virginia*, 2009.

[91] HP. Jena. Recurso on-line: `http://jena.sourceforge.net/`.

[92] Hewlett-Packard Development Company. Sparql update. Recurso disponible on-line: `http://www.w3.org/Submission/SPARQL-Update/`, 2008.

[93] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition. Special issue: Current issues in knowledge*, 5(2)(2):[199,220], 1993.

[94] G. van Heijst, A.T. Schreiber, and B.J. Wielinga. Using explicit ontologies in kbs development. *Int. J. Hum.-Comput. Stud.*, 46(2-3):[183,292], March 1997.

[95] Rudi Studer, V. Richard Benjamins, and Dieter Fensel. Knowledge engineering: Principles and methods. *Data & Knowledge Engineering*, 25:[161,197], 1998.

[96] J. T. Fernández-Breis and R. Martínez-Béjar. A cooperative framework for integrating ontologies. *Int. J. Hum.-Comput. Stud.*, 56(6):[665,720], 2002.

[97] S. Luke, L. Spector, D. Rager, and J. Hendler. Ontology-based web agents. In *In Proceedings of the First International Conference on Autonomous Agents (Agents97)*, pages 59–66, 1997.

[98] N.D. Duque-Méndez and J.C. Chavarro-Porras. Evolución de los lenguajes utilizados en la construcción de la web semántica. *Sciencia et Technica*, 32:[381,386], 2006.

[99] W3C. Daml+oil. Recurso disponible on-line: `http://www.w3.org/TR/daml+oil-reference`.

[100] C. Golbreich, E.K. Wallace, and P.F. Patel-Schneider. Owl 2 web ontology language: New features and rationale. *W3C working draft*, 21:[1,55], 2009.

[101] W3C. Owl2 rdf/xml syntax specification. Recurso disponible on-line: `http://www.w3.org/TR/rdf-syntax-grammar/`.

[102] W3C. Owl 2 xml serialization. Recurso disponible on-line: `http://www.w3.org/TR/owl2-xml-serialization/`.

[103] W3C. Owl 2 functional syntax. Recurso disponible on-line: `http://www.w3.org/TR/owl2-syntax/`.

[104] W3C. W3c: Owl 2 web ontology language manchester syntax. Recurso on-line: `http://www.w3.org/TR/owl2-manchester-syntax/`.

[105] David Beckett. Turtle: Terse rdf triple language. Recurso disponible on-line: `http://www.w3.org/TeamSubmission/turtle/`.

[106] Raphael Volz, Daniel Oberle, Steffen Staab, and Rudi Studer. Ontolift prototype. wonderweb project. Technical report, Technical Report D11, 2002.

[107] GNU. Lgpl: Lesser general public license. Recurso disponible on-line: `http://www.gnu.org/copyleft/lesser.html`.

[108] The Apache Software Foundation. Apache license. Recurso disponible on-line: `http://www.apache.org/licenses/LICENSE-2.0.html`.

[109] OBO Foundry. Obo file format specification. Recurso disponible on-line: `http://www.geneontology.org/GO.format.obo-1_2.shtml`.

[110] Dmitry Tsarkov. Fact++. Recurso disponible on-line: `http://owl.man.ac.uk/factplusplus/`.

[111] Information Systems Group. Hermit owl reasoner. Recurso disponible on-line: `http://hermit-reasoner.com/`.

[112] Clark & Parsia. Pellet: Owl 2 reasoner for java. Recurso disponible on-line: `http://clarkparsia.com/pellet`.

[113] Racer Systems GmbH & Co. KG. Racer pro. Recurso disponible on-line: `http://www.racer-systems.com`.

[114] Hewlett-Packard Development Company. Hp lab semantic web research. Recurso disponible on-line: `http://www.hpl.hp.com/semweb`.

[115] Hewlett Packard. Jena SDB. Recurso disponible on-line: `http://openjena.org/wiki/SDB`.

[116] Hewlett Packard. Jena TDB. Recurso disponible on-line: `http://openjena.org/wiki/TDB`.

[117] Stanford Center for Biomedical Informatics Research. Protégé. Recurso on-line: `http://protege.standford.edu/`.

[118] Martin Giese and Arild Waaler. Automated reasoning with analytic tableaux and related methods. In *18th International Conference, TABLEAUX, Oslo, Norway*, 2009.

[119] Boris Motik, Rob Shearer, and Ian Horrocks. Hypertableau reasoning for description logics. *Journal of Artificial Intelligence Research*, 36:[165,228], 2009.

[120] Franz Baader, Ian Horrocks, and Ulrike Sattler. Description logics as ontology languages for the semantic web. In *Festschrift in honor of Jörg Siekmann, Lecture Notes in Artificial Intelligence*, pages [228,248]. Springer-Verlag, 2003.

[121] Thorsten Liebig, Marko Luther, Olaf Noppens, and Michael Wessel. Owllink. *Semantic Web – Interoperability, Usability, Applicability*, 2(1):23–32, 2011.

[122] Franz Inc. Allegrograph. Recurso disponible on-line: `http://www.franz.com/agraph/allegrograph/`.

[123] L. Miller, A. Seaborne, and A. Reggiori. Three implementations of squishql, a simple rdf query language. In *International Semantic Web Conference*, 2002.

[124] Hewlett-Packard. Rdql: A query language for rdf. Recurso disponible on-line: `http://www.w3.org/Submission/2004/SUBM-RDQL-20040109`, 2003.

[125] Christian Bizer. Triql: A query language for named graphs. Technical report, Freie Universität Berlin, Germany, 2004.

[126] W3C. Sparql protocol for rdf. Recurso on-line: `http://www.w3.org/TR/rdf-sparql-protocol/`.

[127] W3C. Wsdl. Recurso disponible on-line: `http://www.w3.org/TR/2007/REC-wsdl20-primer-20070626/`.

[128] W3C. Sparql query results xml format. Recurso on-line: `http://www.w3.org/TR/rdf-sparql-XMLres/`.

[129] ISO/IEC. Structured query language: Sql (iso/iec 9075 1:2008). Recurso disponible on-line: `http://www.iso.org/iso/catalogue_detail.htm?csnumber=45498`.

[130] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data - the story so far. *Special Issue on Linked Data, International Journal on Semantic Web and Information Systems (IJSWIS)*, 5:[1,22], 2009.

[131] Christian Bizer, Tom Heath, Tim Berners-Lee, and Michael Hausenblas. 4th linked data on the web workshop (ldow2011). In *World Wide Web Conference Series*, pages [303,304], 2011.

[132] Microsoft. Open government data initiatives. Recurso disponible on-line: `http://www.microsoft.com/industry/government/opengovdata`.

[133] Comisión Europea. Agenda digital europea. Recurso disponible on-line: `http://ec.europa.eu/information_society/digital-agenda/index_en.htm`.

[134] Comisión Europea. Fp7 - séptimo programa marco. Recurso disponible on-line: `http://cordis.europa.eu/fp7`.

[135] LATC Consortium. Latc: Linked open data around-the-clock. Recurso disponible on-line: `http://latc-project.eu/`.

[136] LOD2 Consortium. Lod2: Creating knowledge out of interlinked data. Recurso disponible on-line: `http://lod2.eu/`.

[137] STI International. Planet data. Recurso disponible on-line: `http://www.planet-data.eu/`.

[138] United State Government. Data.gov. Recurso disponible on-line: `http://data.gov`.

[139] HM Government. Opening up government. Recurso disponible on-line: `http://data.gov.uk`.

[140] France Government. Portail du gouvernement. Recurso disponible on-line: `http://data.gouv.fr`.

[141] T. Berners-Lee. Linked data - design issues. Recurso disponible on-line: `http://www.w3.org/DesignIssues/LinkedData.html`, July 2006.

[142] CKAN. Ckan linking open data cloud. Recurso disponible on-line: `http://ckan.net/group/lodcloud`.

[143] B. Smith and M. Brochhausen. Putting biomedical ontologies to work. *Methods Inf Med*, 49:[135,140], 2010.

[144] W3C. Semantic web health care and life sciences (hcls). Recurso disponible on-line: `http://www.w3.org/blog/hcls/`.

[145] W3C. Sparql working group. Recurso disponible on-line: `http://www.w3.org/2009/sparql/wiki/Main_Page`.

[146] W3C. Rdb2rdf working group. Recurso on-line: http://www.w3.org/2001/sw/rdb2rdf/, 2009.

[147] W3C. Linking open drug data (lodd). Recurso disponible on-line: http://www.w3.org/wiki/HCLSIG/LODD.

[148] T.R. Gruber. Towards principles for the design of ontologies used for knowledge sharing. In *International Workshop on Formal Ontology, Padova, Italy*, 1993.

[149] USA. National library of medicine. Recurso disponible on-line: http://www.nlm.nih.gov/.

[150] National Library of Medicine. Umls: Unified medical language system. Recurso disponible on-line: http://www.nlm.nih.gov/research/umls/.

[151] NCBI. Ncbi databases. Recurso on-line: http://www.ncbi.nlm.nih.gov/guide/all/.

[152] OBO Foundry. The open biological and biomedical ontologies. Recurso disponible on-line: http://www.obofoundry.org/.

[153] Barry Smith, Werner Ceusters, Bert Klagges, Jacob Köhler, Anand Kumar, Jane Lomax, Chris Mungall, Fabian Neuhaus, Alan L Rector, and Cornelius Rosse. Relations in biomedical ontologies. *Genome Biology*, 6:R46, 2005.

[154] EMBL-EBI. Chebi - chemical entities of biological interest. Recurso disponible on-line: http://www.ebi.ac.uk/chebi/.

[155] Christopher Mungall, Georgios Gkoutos, Cynthia Smith, Melissa Haendel, Suzanna Lewis, and Michael Ashburner. Integrating phenotype ontologies across multiple species. *Genome Biology*, 11:R2, 2010.

[156] D.A. Natale, C.N. Arighi, W.C. Barker, J.A. Blake, C.J. Bult, M. Caudy, H.J. Drabkin, P. D'Eustachio, A.V. Evsikov, H. Huang,

J. Nchoutmboube, N.V. Roberts, B. Smith, J. Zhang, and C.H. Wu. The protein ontology: a structured representation of protein forms and complexes. *Nucleic Acids Res.*, 39:[D539,D545], 2011.

[157] J.B. Bowes, K.A. Snyder, E. Segerdell, C.J. Jarabek, K. Azam, A.M. Zorn, and P.D. Vize. Xenbase: gene expression and improved integration. *Nucleic Acids Res.*, 38:[D607,D612], 2010.

[158] ZFA workgroup. Zebrafish anatomy ontology. Recurso disponible on-line: `http://obo.cvs.sourceforge.net/viewvc/obo/obo/ontology/anatomy/gross_anatomy/animal_gross_anatomy/fish/zebrafish_anatomy.obo`.

[159] Berkeley Bioinformatics and Open Source Projects Group. The obo ontology editor. Recurso disponible on-line: `http://www.oboedit.org`.

[160] Stuart Aitken, Yin Chen, and Jonathan Bard. Obo explorer: an editor for open biomedical ontologies in owl. *Bioinformatics*, 24(3):443–444, 2008.

[161] OBO Foundry. Oboedit owl plugin. Recurso disponible on-line: `http://smi-protege.stanford.edu/~nigam/OboEditPlugin.zip`.

[162] Stuart Aitken, Yin Chen, Bonnie Webber, Wenfei Fan, and Jonathan Bard. Managing the transition from obo to owl: The cobra-ct bio-ontology tools. In *Proceedings of the UK e-Science All-Hands Meeting*, pages 94–101, 2007.

[163] Erik Antezana. Onto perl. Recurso disponible on-line: `http://search.cpan.org/dist/ONTO-PERL/`.

[164] Michelle Gwinn Giglio. Evidence codes ontology. Recurso disponible on-line: `http://obo.cvs.sourceforge.net/viewvc/obo/obo/ontology/evidence_code.obo`.

[165] B. Smith, W. Ceusters, B. Klagges, J. Kohler, A. Kumar, J. Lomax, C.J. Mungall, F. Neuhaus, A. Rector, and C. Rosse. Relations in biomedical ontologies. *Genome Biology*, 6:R46, 2005.

[166] National Center for Biotechnology Information. Ncbi taxonomy. Recurso disponible on-line: `http://www.berkeleybop.org/ontologies/obo-all/ncbi_taxonomy/ncbi_taxonomy.obo`.

[167] P.N. Robinson, S. Köhler, S. Bauer, D. Seelow, D. Horn, and S. Mundlos. The human phenotype ontology: a tool for annotating and analyzing human hereditary disease. *The American Journal of Human Genetics*, 83:[610,615], 2008.

[168] J. Blake. Bio-ontologies-fast and furious. *Nature Biotechnology*, 22:[773,774], 2004.

[169] C. Jonquet, N.H. Shah, and M.A. Musen. The open biomedical annotator. In *AMIA Summit on Translational Bioinformatics, San Francisco, CA, USA*, 2009.

[170] Clement Jonquet, Mark A. Musen, and Nigam Shah. A system for ontology-based annotation of biomedical data. *Lecture Notes in BioInformatics*, 5109:[144,152], 2008.

[171] The National Center for Biomedical Ontology. Bioportal. Recurso disponible on-line: `http://bioportal.bioontology.org/`.

[172] Jonathan Bard, Seung Y Rhee, and Michael Ashburner. An ontology for cell types. *Genome Biology*, 6:R21, 2005.

[173] Kei-Hoi Cheung, Kevin Y. Yip, Andrew Smith, Remko deKnikker, Andy Masiar, and Mark Gerstein. Yeasthub: a semantic web use case for integrating data in the life sciences domain. *Briefings in Bioinformatics*, 21(suppl 1):i85–i96, 2005.

[174] Francois Belleau, Marc-Alexandre Nolin, Nicole Tourigny, Philippe Rigault, and Jean Morissette. Bio2rdf: towards a mashup to build bioinformatics knowledge systems. *Journal of Biomedical Informatics*, 41(5):[706,716], 2008.

[175] LarKC. Larkc: The large knowledge collider. Recurso disponible online: `http://www.larkc.eu/`.

[176] C. Goble and R. Stevens. State of the nation in data integration for bioinformatics. *Journal of Biomedical Informatics*, 41(5):[687,693], 2008.

[177] Sohrab P Shah, Yong Huang, Tao Xu, Macaire MS Yuen, John Ling, and BF Francis Ouellette. Atlas: a data warehouse for integrative bioinformatics. *BMC Bioinformatics*, 6(1):34, 2005.

[178] Silke TriSZl, Kristina Rother, Heiko Muller, Thomas Steinke, Ina Koch, Robert Preissner, Cornelius Frommel, and Ulf Leser. Columba: an integrated database of proteins, structures, and annotations. *BMC Bioinformatics*, 6(1):81, 2005.

[179] S Haider, B Ballester, D Smedley, J Zhang, P Rice, and Arek Kasprzyk. Biomart central portal–unified access to biological data. *Nucleic Acids Res.*, 37:23–27, 2009.

[180] TJ Lee, Y Pouliot, V Wagner, P Gupta, DW Stringer-Calvert, JD Tenenbaum, and PD Karp. Biowarehouse: a bioinformatics database warehouse toolkit. *BMC Bioinformatics*, 7:170, 2006.

[181] Robert Stevens, Norman W Paton, Sean Bechhofer, Gary Ng, Martin Peim, Patricia Baker, Carole Goble, and Andy Brass. *TAMBIS: transparent access to multiple bioinformatics services*, pages [184,185]. Bioinformatics, 2004.

[182] Aaron Birkland and Golan Yona. Biozon: a system for unification, management and analysis of heterogeneous biological data. *BMC Bioinformatics*, 7(1):70, 2006.

[183] C. Penkett, J. Morris, V. Wood, and J. Bähler. Yogy: a web-based, integrated database to retrieve protein orthologs and associated gene ontology terms. *Nucleic Acids Research*, 34:[330,334], 2006.

[184] Satya S. Sahoo, Olivier Bodenreider, Joni L. Rutter, Karen J. Skinner, and Amit P. Sheth. An ontology-driven semantic mashup of gene and biological pathway information: Application to the domain of nicotine

dependence. *Journal of Biomedical Informatics*, 41(5):[752,765], 2008. Semantic Mashup of Biomedical Data.

[185] Roy Thomas Fielding. Representational state transfer (rest). Recurso disponible on-line: `http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm`, 2000.

[186] W3C. Really simple syndication (rss). Recurso disponible on-line: `http://feed2.w3.org/docs/rss2.html#whatIsRss`.

[187] Robin Dowell, Rodney Jokerst, Allen Day, Sean Eddy, and Lincoln Stein. The distributed annotation system. *BMC Bioinformatics*, 2(1):7, 2001.

[188] Philip Jones, Nisha Vinod, Thomas Down, Andre Hackmann, Andreas Kahari, Ernst Kretschmann, Antony Quinn, Daniela Wieser, Henning Hermjakob, and Rolf Apweiler. Dasty and uniprot das: a perfect pair for protein feature visualization. *Bioinformatics*, 21(14):[3198,3199], 2005.

[189] W3C. Soap. Recurso disponible on-line: `http://www.w3.org/TR/2007/REC-soap12-part0-20070427/`.

[190] Pieter B.T. Neerincx and Jack A. M. Leunissen. Evolution of web services in bioinformatics. *Briefings in Bioinformatics*, 6(2):178–188, 2005.

[191] Jiten Bhagat, Franck Tanoh, Eric Nzuobontane, Thomas Laurent, Jerzy Orlowski, Marco Roos, Katy Wolstencroft, Sergejs Aleksejevs, Robert Stevens, Steve Pettifer, Rodrigo Lopez, and Carole A. Goble. Biocatalogue: a universal catalogue of web services for the life sciences. *Nucleic Acids Research*, 38(suppl 2):W689–W694, 2010.

[192] Paul Shannon, David Reiss, Richard Bonneau, and Nitin Baliga. The gaggle: An open-source software system for integrating bioinformatics software and data sources. *BMC Bioinformatics*, 7(1):176, 2006.

[193] S Pettifer, TK Attwood, and J Sinnott. *Integrated approaches for bioinformatics data analysis and visualisation, challenges, opportunities and new solutions*. Data Analysis and Visualisation in Genomics and Proteomics. John Wiley and Sons, Ltd, ISBN 0-470-09439-7, JAN 2005.

[194] Duncan Hull, Katherine Wolstencroft, Robert Stevens, Carole Goble, Matthew Pocock, Peter Li, and Thomas Oinn. Taverna: a tool for building and running workflows of services. *Nucleic Acids Research*, 34(Web Server issue):[729,732], July 2006.

[195] Satya S. Sahoo, Wolfgang Halb, Sebastian Hellmann, Kingsley Idehen, Ted Thibodeau Jr, Sören Auer, Juan Sequeda, and Ahmed Ezzat. A survey of current approaches for mapping of relational databases to rdf. Recurso disponible on-line: `http://www.w3.org/2005/Incubator/rdb2rdf/RDB2RDF_SurveyReport.pdf`.

[196] Ashok Malhotra. Incubator group report. Recurso disponible on-line: `http://www.w3.org/2005/Incubator/rdb2rdf/XGR/`, 2009.

[197] L. Stojanovic, N. Stojanovic, and R. Volz. Migrating data intensive web sites into the semantic web. In *Symposium on Applied Computing, Madrid, Spain*, 2002.

[198] N. Stojanovic, L. Stojanovic, and R. Volz. A reverse engineering approach for migrating dataintensive web sites to the semantic web. In *Intelligent Information Processing, Montreal*, 2002.

[199] W3C. A direct mapping of relational data to rdf. Recurso on-line: `http://www.w3.org/TR/rdb-direct-mapping/`.

[200] Dejing Dou, Han Qin, and Paea LePendu. Ontograte: Towards automatic integration for relational databases and the semantic web through an ontology based framework. *International Journal of Semantic Computing (IJSC)*, 4:[123,151], 2010.

[201] IETF. Rfc3987: Internationalized resource identifiers (iris). Recurso disponible on-line: `http://www.ietf.org/rfc/rfc3987.txt`.

[202] Richard Fikes, Pat Hayes, and Ian Horrocks. Owl ql: A language for deductive query answering on the semantic web. Technical Report 03-14, Knowledge System Laboratory, Stanford University, Stanford, CA, 2003.

[203] D. V. McDermott and D. Dou. Representing disjunction and quantifiers in rdf. In *Proceedings of International Semantic Web Conference*, pages [250,263], 2002.

[204] W3C. R2rml: Rdb to rdf mapping language. Recurso on-line: `http://www.w3.org/TR/r2rml/`, 2011.

[205] Christian Bizer. D2r map - a database to rdf mapping language. In *The 12th International World Wide Web Conferenc (www2003)*, 2003.

[206] Jesus Barrasa and Asunción Gómez-Pérez. R2o, an extensible and semantically based database to ontology mapping language. In *In Proceedings of the 2nd Workshop on Semantic Web and Databases, Toronto, (Can) Aug.*, pages [1069,1070], 2004.

[207] J. Madhavan, P. Bernstein, and E. Rahm. Generic schema matching with cupid. In *Proceedings of the 27th VLDB Conference, Rome, Italy*, 2001.

[208] E. Rahm and P. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10:[334,450], 2001.

[209] Nadine Cullot, Raji Ghawi, and Kokou Yétongnon. Db2owl: A tool for automatic database-to-ontology mapping. Recurso disponible on-line: `http://vision.u-bourgogne.fr/le2i/user_data/publications/DB2OWL1.pdf`.

[210] Christian Bizer, Richard Cyganiak, Jörg Garbers, Oliver Maresch, and Christian Becker. D2rq platform. Recurso on-line: `http://www4.wiwiss.fu-berlin.de/bizer/D2RQ/spec/`.

[211] Openlink software. Vistas rdf de virtuoso. Recurso disponible on-line: `http://virtuoso.openlinksw.com/whitepapers/relational%20rdf%20views%20mapping.html`.

[212] Open RDF. Sesame. Recurso disponible on-line: `http://www.openrdf.org/`.

[213] Openlink software. Servidor universal openlink virtuoso. Recurso disponible on-line: `http://virtuoso.openlinksw.com/`.

[214] Jose Antonio Miñarro-Giménez, Mikel Egaña Aranguren, Francisco García Sánchez, and Jesualdo Tomás Fernández-Breis. A semantic query interface for the ogo platform. In *Information Technology in Bio-and Medical Informatics, ITBAM 2010*, pages 128–142. Springer, 2010.

[215] Jose Antonio Miñarro-Gimenez, Mikel Egaña Aranguren, Rodrigo Martínez-Béjar, Marisa Madrid, and Jesualdo Tomás Fernández-Breis. Semantic integration of information about orthologs and diseases: The ogo system. *Journal of Biomedical Informatics*, 44:[1020,1031], 2011.

[216] Richard Cyganiak and Chris Bizer. Pubby - a linked data frontend for sparql endpoints. Recurso disponible on-line: `http://www4.wiwiss.fu-berlin.de/pubby/`.

[217] Chris Bizer and Tobias Gauß. Disco: Hyperdata browser. Recurso disponible on-line: `http://www4.wiwiss.fu-berlin.de/bizer/ng4j/disco/`.

[218] T. Berners Lee. Tabulator. Recurso disponible on-line: `http://www.w3.org/2005/ajar/tab`.

[219] Tim Berners-Lee, Roy T. Fielding, and Larry Masinter. Uniform resource identifier (uri): Generic syntax. Recurso disponible on-line: `http://tools.ietf.org/html/rfc3986`.

[220] Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Number 1 in Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan and Claypool, 1st edition edition, 2011.

[221] ORACLE. Mysql. Recurso disponible on-line: `http://www.mysql.com/`.

[222] PostgreSQL Global Development Group. Postgresql. Recurso disponible on-line: `http://www.postgresql.org/`.

[223] ORACLE. Oracle database. Recurso disponible on-line: `http://www.oracle.com/us/products/database/index.html`.

[224] Jose Antonio Miñarro-Giménez, Marisa Madrid, and Jesualdo Tomás Fernández-Breis. An integrated ontological knowledge base about orthologous genes and proteins. In *Proceedings on 1st Workshop Semantic Web Applications and Tools for Life Sciences, Edinburgh, Scotland*, volume 435, 2008.

[225] NCBI. Pubmed. Recurso disponible on-line: `http://www.ncbi.nlm.nih.gov/pubmed`.

[226] ORACLE. Java server pages. Recurso disponible on-line: `http://jcp.org/en/jsr/detail?id=245`.

[227] Jose Antonio Miñarro-Giménez, Marisa Madrid, and Jesualdo Tomás Fernández-Breis. Semantic integration and exploitation of orthology information and genetic disorders. In *Proceedings on 2nd Workshop Semantic Web Applications and Tools for Life Sciences, Amsterdam, Netherlands*, volume 20, 2009.

[228] Jose Antonio Miñarro-Giménez, Mikel Egaña Aranguren, Boris Villazón-Terrazas, and Jesualdo Tomás Fernández-Breis. Ogo linking open data. Recurso disponible on-line: `http://thedatahub.org/dataset/ogolod`, 2011.

[229] CKAN. The Data Hub – The easy way to get, use and share data. Recurso disponible on-line: `http://thedatahub.org`.

[230] Jose Antonio Miñarro-Giménez, Teddy Miranda-Mena, Rodrigo Martínez-Béjar, and Jesualdo Tomás Fernández-Breis. Exploitation of translational bioinformatics for decision-making on cancer treatments. In *Information Technology in Bio- and Medical Informatics, ITBAM 2011*, pages 1–15. Springer, 2011.

[231] Catalina Martínez-Costa, Jose A. Miñarro-Giménez, Marcos Menárguez-Tortosa, Rafael Valencia-García, and Jesualdo Fernández-Breis. Flexible semantic querying of clinical archetypes. In Rossitza Setchi, Ivan Jordanov, Robert Howlett, and Lakhmi Jain, editors, *Knowledge-Based and Intelligent Information and Engineering Systems*, volume 6276 of *Lecture Notes in Computer Science*, pages 597–606. Springer Berlin / Heidelberg, 2010.

[232] José Alberto Maldonado, Catalina Martínez-Costa, David Moner, Marcos Menárguez-Tortosa, Diego Boscá, Jose A. Miñarro-Gimenez, Jesualdo T. Fernández-Breis, and Montserrat Robles. Using the researchehr platform to facilitate the practical application of the ehr standards. *Journal of Biomedical Informatics*, 0(0):–, 2011.

[233] ONTOTEXT. Linked life data: a semantic data integration platform for the biomedical domain. Recurso disponible on-line: `http://linkedlifedata.com/`.

Anexo

Gramática de las reglas de correspondencia

En la Tabla 1, se muestra la gramática definida para representar las reglas de correspondencia. En caso de que un mismo repositorio estuviera asociado con distintas ontologías, o que una misma ontología estuviera relacionada con distintos repositorios, se deberá definir un documento por cada combinación repositorio-ontología. Así, dentro de cada documento de reglas de correspondencia se indicarán el esquema del repositorio y la ontología correspondientes.

En la gramática se utiliza la etiqueta "<Alignment>" como etiqueta raíz de todo el documento de reglas de correspondencia en el documento XML. Para identificar el esquema del repositorio se utiliza la etiqueta "<dbsource>" y para identificar la ontología se define la etiqueta "<ontotarget>". Bajo estas etiquetas definimos las propiedades necesarias para localizar y acceder a las fuentes de información que se utilizarán en el proceso de integración.

MappingsDocument	::= '<Alignment>' **Repositorio Ontología Map*** '</Alignment>'
Repositorio	::= '<dbsource ' **Parámetros** '>' **Descripción** '</dbsource>'
Parámetros	::= **Tipo** ' ' **Conexión** ' ' **Usuario** ' ' **Contraseña**
Ontología	::= '<ontotarget>' **Fichero** '</ontotarget>'
Map	::= **MapClase** | **MapPropiedad** | **MapRelacion**
MapClase	::= '<map>' **TipoClase Clase DB** '</map>'
MapPropiedad	::= '<map>' **TipoProp Sujeto Predicado Literal** '</map>'
MapRelacion	::= '<map>' **TipoRel Sujeto Predicado Objeto** '</map>'
Sujeto	::= '<source>' **Clase DB** '</source>'
Predicado	::= '<predicate>' **Propiedad DB** '</predicate>'
Literal	::= '<target>' **DB** '</target>'
Objeto	::= '<target>' **Clase DB** '</target>'
Propiedad	::= '<property>¡*id*¿' **URI** '</id></property>'
Clase	::= '<class><id>' **URI** '</id></class>'
DB	::= '<db><id>' **Tabla** '.' **Columna** '</id></db>'
TipoClase	::= '<type>DB2Class</type>'
TipoProp	::= '<type>DB2Prop</type>'
TipoRel	::= '<type>DB2Prop</type>'
Tipo	::= 'type="' *String* '"'
Conexión	::= 'connection="' *String* '"'
Usuario	::= 'user="' *String* '"'
Contraseña	::= 'password="' *String* '"'
Descripción	::= *String*
Fichero	::= *String*
URI	::= *String*
Tabla	::= *String*
Columna	::= *String*

Tabla 1: Gramática del lenguaje para la definición de las reglas de correspondencia.

Gramática de las reglas de identidad

En la Tabla 2, se muestra la gramática definida para representar las reglas de identidad. Las reglas de identidad son codificadas utilizando documentos XML, al igual que las reglas de correspondencia. Cada regla de identidad, define los requisitos de equivalencia entre instancias para una clase de la ontología basándose en los valores de sus propiedades y relaciones. La definición de una regla de identidad de una clase es aplicable a sus sub clases de la ontología ya las reglas definen condiciones sobre propiedades y relaciones de las clases, y éstas se heredan de la clase padre a las clases hijas. Por otro lado, las reglas de identidad solo están relacionadas con la definición de las clases de una ontología, y por lo tanto no referencia a los repositorios relacionales de donde se extrae la información a integrar.

En la gramática se utiliza la etiqueta "<Identities>" como etiqueta raíz de todo el documento de reglas de identidad. Para identificar la ontología se define la etiqueta "<ontosource>". Bajo estas etiquetas se definien las condiciones necesarias para localizar las instancias equivalentes. Estas condiciones utilizan la etiqueta '<condition>'.

Una regla de identidad está definida como una expresión booleana y representada como un árbol de requisitos que deben cumplir las propiedades y relaciones de las instancias. La raíz del árbol indica la clase de la ontología que se describe en la regla. Los requisitos de la regla de identidad están definidos en la etiqueta <requirement>, donde cada uno describe una tripleta (sujeto, predicado y objeto) asociada a una relación o propiedad concreta. Las tripletas están definidas utilizando las etiquetas "<predicate>" y "<object>" para describir el predicado y el objeto respectivamente. El sujeto asociado a la tripleta se corresponde con el objeto definido por el requisito padre dentro del árbol de requisitos. Así, las tripletas definidas en cada requisito están conectadas con su sujeto mediante la relación padre-hijo.

Cada relación padre-hijo entre requisitos tiene asignada una las etiquetas *"AND"* y *"OR"*, su uso indica la obligatoriedad u opcionalidad de los sub requisitos descritos, respectivamente para evaluar la identidad de un individuo. Esto es debido a que las propiedades no siempre tienen que tener valores

asignados. Como es habitual, la precedencia de “AND” es mayor que la de “OR”.

Los requisitos sobre el predicado están definidos mediante dos etiquetas: “<scope>” y “<property>”. El “<scope>” establece el ámbito de aplicación de la propiedad o relación definida mediante la etiqueta “<property>”. El ámbito de aplicación del predicado puede tomar los valores *“ALL”* y *“SOME”*, que determina si todas los valores de una propiedad deben existir en la instancia equivalente o solo algunas de ellas respectivamente.

Por otro lado, el objeto del requisito contiene otros dos tipos de etiquetas “<value>” y “<class>”. Los posibles valores de la etiqueta “<value>” son *EQUALS* y *EQUALS IGNORE CASE*. Estos valores definen como evaluar los valores de las propiedades o relaciones. En caso de evaluar cadenas de texto ambos tipos son posibles, e indican que los valores deben ser iguales o iguales sin distinguir entre mayúsculas y minúsculas. En otro tipo de propiedades solo el tipo de evaluación *EQUALS* es posible. La etiqueta “<class>” especifica la clase perteneciente al rango de la relación definida en el predicado, en caso de propiedades esta etiqueta puede indicar la clase del dominio de la propiedad. Si el requisito define sub requisitos, el sujeto de los nuevos requisitos será la clase definida en la etiqueta “<class>”.

DocIdentidades	::= '<Identities>'**Ontología Identidad*** '</Identities>'
Ontología	::= '<ontosource>' **Fichero** '</ontosource>'
Identidad	::= '<condition>' **RequisitoRaiz** '</condition>'
RequisitoRaiz	::= '<requirement>' **ObjetoId SubRequisitos** '</requirement>'
ObjetoId	::= '<object>' **Clase** '</object>'
SubRequisitos	::= **ANDRequisitos** | **ORRequisitos**
ANDRequisitos	::= '<and>' **Expresion** '</and>'
ORRequisitos	::= '<or>' **Expresion** '</or>'
Expresion	::= **Requisito** | **ANDExpresion** | **ORExpresion**
ANDExpresion	::= '<and>' **Requisito Expresion**? '</and>'
ORExpresion	::= '<or>' **Requisito Expresion**? '</or>'
Requisito	::= (**Predicado Objeto SubRequisitos**?) | ((**Predicado Objeto**)? **SubRequisitos**)
Predicado	::= '<predicate>' **Ambito Propiedad** '</predicate>'
Objeto	::= '<object>' **Evaluacion Tipo Clase** '</object>'
Clase	::= '<class>' **URI** '</class>'
Ambito	::= '<scope>' **AmbitoValor** '</scope>'
Propiedad	::= '<property>' **URI** '</property>'
Evaluacion	::= '<value>' **EvalValor** '</value>'
Tipo	::= '<type>' **URI** '</type>'
AmbitoValor	::= 'SOME' | 'ALL'
EvalValor	::= 'EQUALS' | 'EQUALS IGNORE CASE'
Fichero	::= *String*
URI	::= *String*

Tabla 2: Gramática del lenguaje para la definición de las reglas de identidad.