UNIVERSIDAD DE ALMERÍA

ESCUELA POLITÉCNICA SUPERIOR Y FACULTAD DE CIENCIAS EXPERIMENTALES

INGENIERÍA INFORMÁTICA

CREACIÓN DE UNA ONTOLOGÍA OWL PARA EL MODELO DE METADATOS CWM (COMMON WAREHOUSE MODEL)

El Alumno:

José Molina González

Almería, Febrero 2014

Director(es):

Manuel Torres Gil

Agradecimientos

Este trabajo ha sido posible gracias a mi tutor Manuel Torres Gil al que desde estas escuetas líneas quiero agradecer su dedicación y conocimiento. Tambien me gustaría agradecer la paciencia, cariño y apoyo a mis padres, familia y amigos que han dado durante este largo periodo en el que se ha extendido la realización de este proyecto. Especialmente a Vanessa, que día a día ha estado a mi lado en continuos fines de semana de trabajo siempre con su amor y dedicación.

Tabla de Contenidos

Capítulo 1.

Introducción

Contenido

1.1. Motivación del Proyecto

En la actualidad, la cantidad de datos que una organización maneja se incrementa a razón de dos veces su tamaño cada cinco años [1]. El origen de estos datos suelen provenir de sistemas heterogéneos, con diferentes fuentes de datos y metadatos, lo que dificulta en gran medida la gestión y explotación de la información para la toma de decisiones. La dificultad del intercambio de metadatos entre los diferentes modelos de datos de estas aplicaciones hace incrementar los costes a las organizaciones que los desarrollan, ya que cada solución requiere realizar adaptadores por cada origen de datos nuevos distintos. En sistemas altamente heterogéneos surgen otros problemas como son:

- Ambigüedades entre los metadatos.
- Diferentes procesos de negocio.
- Datos con un significado semántico común.

Existen en el mercado un alto número de herramientas que permiten la integración de diferentes fuentes de datos, como son las soluciones de Data Warehousing o almacenes de datos. La necesidad de definir un modelo de metadatos común entre las diferentes herramientas del mercado han dado origen al estándar Common Warehouse Metamodel (CWM) que unifica la organización de fuentes de datos, metadatos, transformaciones, análisis, procesos y operaciones [2] [3]. Sin embargo, la implementación de CWM por parte de la industria no ha conseguido resolver algunos de los problemas existentes como son las

carencias en el análisis semántico. Por ejemplo, en la Figura 1.1. se muestra un ejemplo de resolución de ambigüedades de las definiciones tales como ‘gato’ o ‘vaca’ mediante la adición de significado semántico dependiendo del dominio y contexto del uso, ya sea una granja de animales o un taller mecánico.

Figura 1.1. Resolución de carencias semánticas mediante ontologías.

Organizar e integrar los datos de una manera semántica, analizando el significado de los datos y las relaciones existentes en un dominio específico es lo que se formaliza en las Ontologías [4]. Una ontología necesita expresarse mediante un lenguaje que permita la representación e inferencia de estos conceptos basándose en sus construcciones [5]. Existen diferentes lenguajes que nos permiten definir y expresar una ontología, siendo Web Ontology Language (OWL) el más extendido, porque su uso facilita el desarrollo, la compatibilidad y la accesibilidad de una ontología a través de la Web, de una manera flexible y extensible.

1.2. Objetivos del Proyecto

El objetivo principal del proyecto es dotar de un significado semántico al modelo de metadatos propuesto por CWM, mediante ontologías expresadas en formato OWL que cubran las carencias de este tipo de aplicaciones mediante el análisis semántico de los datos sobre sistemas de datos altamente heterogéneos.

Para la consecución del objetivo principal, este proyecto tiene estos objetivos específicos:

- Analizar la especificación Common Warehouse Metadata (CWM).
- Investigar el uso de ontologías para añadir significado semántico.
- Expresar ontologías mediante Web Ontology Language 2 (OWL2).
- Estudiar el uso de herramientas de edición de ontologías (Protégé 4).
- Utilizar la librería de java OWL API desarrollada para gestionar las entidades, expresiones y axiomas definidos por las ontologías [6].
- Construir una ontología OWL que defina el paquete relacional de CWM 1.1.

- Desarrollar una aplicación capaz de expresar objetos CWM 1.1 como individuos de una ontología OWL.

1.3. Fases del Proyecto

El proyecto consiste en el enriquecimiento de los metadatos definidos por la especificación CWM (Common Warehouse Metamodel) extraídos de diferentes distribuciones de base de datos mediante el uso de ontologías OWL. Las principales fases que se ha seguido en la elaboración de este proyecto han sido:

1. Búsqueda de información (libros, artículos, revistas, e-books, webs…).
    a. Bibliografía sobre CWM.
    b. Bibliografía sobre Ontologías.
    c. Bibliografía sobre OWL2.

2. Estudios Previos.
    a. Definición de la especificación del modelo de metadatos CWM.
    Introducción a los conceptos, interfaces y relaciones establecidos en el estándar CWM y la importancia del desarrollo de aplicaciones basado en su modelo de metadatos para las empresas.

    b. Análisis y evolución de las ontologías.
    Estudio de las ontologías como herramientas que proveen a términos y relaciones de un dominio dado un contenido semántico y la evolución de los diferentes lenguajes utilizados para expresarlas.

    c. Descripción de los componentes del lenguaje OWL2.
    Exposición de las características, expresiones semánticas y sintaxis que OWL2 ofrece sobre el desarrollo y construcción de ontologías.

3. Aplicación de los contenidos.
    a. Implementación de una librería CWM 1.1.
    Creación de un proyecto Java con las clases que implementan el modelo de metadatos propuesto en el estándar CWM 1.1.

    b. Desarrollo de una ontología basada en el modelo de metadatos de CWM.
    Diseño y desarrollo de ontologías.

    c. Construcción y procesamiento de objetos CWM.
    Diseño y desarrollo de servicios web que ofrecen acceso a los objetos definidos en el modelo de metadatos CWM en diferentes instancias de bases de datos.

    d. Transformación de objetos CWM a individuos de una ontología CWM.
    Elaboración de una librería que utilice los objetos definidos por la librería de CWM y transforme a individuos definidos en las clases y ontologías descritas en OWL2.

e. Diseño de una interfaz de usuario Web.
   Implementación de la interfaz de usuario web que dará acceso a los servicios web de acceso a metadatos CWM de una base de datos relacional, así como a los individuos de la ontología CWM que lo define.

4. Estudios finales.
   a. Conclusiones.
      Análisis de los beneficios de la combinación de las tecnologías expuestas para llegar a las conclusiones adecuadas.

1.4. Estructura de la documentación

Este proyecto expone los beneficios de la creación de ontologías OWL para representar el conocimiento definido en el modelo de metadatos del estándar Common Warehousing Metamodel. Para ello, la estructura del proyecto se divide en 4 capítulos y una introducción.

En el **Capítulo 2** se exponen los conceptos y elementos del modelo de metadatos CWM del paquete relacional. Mediante este modelo se consigue definir y unificar los diferentes elementos y relaciones existentes en las diferentes bases de datos relacionales. El uso de este estándar ha sido promovido por diversos fabricantes para la creación de diferentes herramientas Warehouse, unificando así los diferentes modelos de metadatos, reduciendo costes en el desarrollo e integración de sistemas heterogéneos.

Junto el estándar CWM, se introduce la arquitectura dirigida por modelos (MDA) y los estándares (CWM, UML, XMI y MOF) que la componen. El uso de esta arquitectura permite la utilización de modelos como principal fuente de desarrollo, permitiendo un fácil desarrollo y transformación de los sistemas.

La integración y la interoperabilidad de los metadatos relacionales de las diferentes bases de datos al modelo propuesto en CWM, se realizan mediante una serie de transformaciones y adaptaciones entre los modelos, conocidos como ‘Puentes’ (bridges). A pesar de que estos ‘Puentes’ permiten la extracción y manejo de los metadatos de una manera automatizada, los metadatos en CWM no tienen un significado semántico de los mismos ni de las relaciones entre sí en los diferentes dominios de origen que provienen.

La creación de ontologías sobre los metadatos definidos en CWM, cubren esta necesidad, dotando a los metadatos de una mayor expresividad semántica para definir su significado dado el dominio.

El **Capítulo 3** de este proyecto, se definen las ontologías como herramientas semánticas y una breve descripción sobre la evolución de los lenguajes utilizados para expresarlas, centrándose en el Lenguaje de Ontologías Web (OWL) debido a su mayor expresividad y difusión en la industria.

El **Capítulo 4**, muestra utilización de las ontologías OWL basadas en el modelo de metadatos relacional definido en CWM mediante una solución de servicios web que permite el acceso a

los metadatos de bases de datos relacionales de diferentes fabricantes utilizando ‘Puentes’ a CWM y reglas de transformación sobre estos elementos CWM a individuos de las ontologías de una manera automatizada.

Por último, el **Capítulo 5** indicará las conclusiones del proyecto y las líneas futuras a seguir.

Capítulo 2.

Common Warehouse Model (CWM)

Contenido

2.1. Objetivos de CWM

Uno de los objetivos principales de CWM es definir un modelo de los metadatos de un almacén de datos (Datawarehouse).

2.2. Introducción a CWM

En la actualidad, la cantidad de tecnologías, herramientas y fabricantes que existen a la hora de establecer comunicaciones y operaciones entre diferentes bases de datos ha provocado que la interoperabilidad y la integración se conviertan en una prioridad en el diseño y desarrollo de sistemas heterogéneos. La
Figura **2.1** muestra la utilización de tres herramientas con diferentes propósitos sobre distintos fabricantes de bases de datos, que requieren de la fabricación de conectores específicos por cada modelo de metadatos específico de bases de datos.

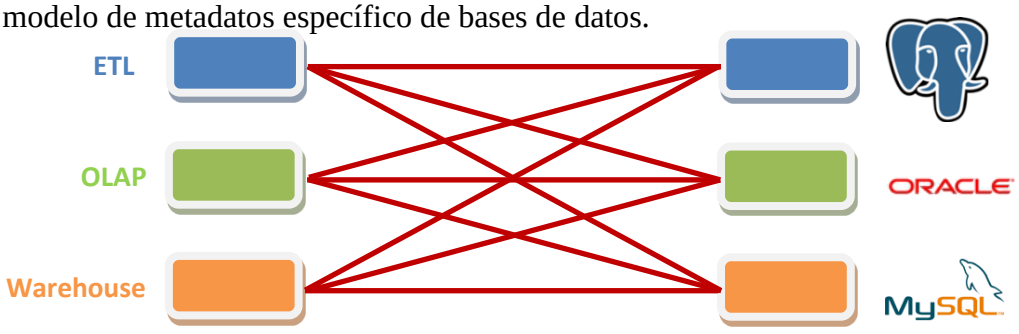


Figura 2.1. Comunicaciones establecidas entre diferentes herramientas y bases de datos.

Con el aumento de herramientas y soluciones de bases de datos que existen en la industria, elevan el desarrollo y coste de las integraciones entre diferentes modelos de fabricantes de una manera exponencial. Con el objetivo de reducir costes y aumentar la reutilización de desarrollos de transformación y acceso a modelos, surge Common Warehousing Metamodel (CWM). CWM es un lenguaje o framework para especificar representaciones externas de metadatos de almacenes de datos (Datawarehouse) con propósitos de intercambio. CWM es una especificación completa de sintaxis y semántica que herramientas de almacenes de datos y Business Intelligence pueden utilizar para un intercambio eficiente de metadatos.

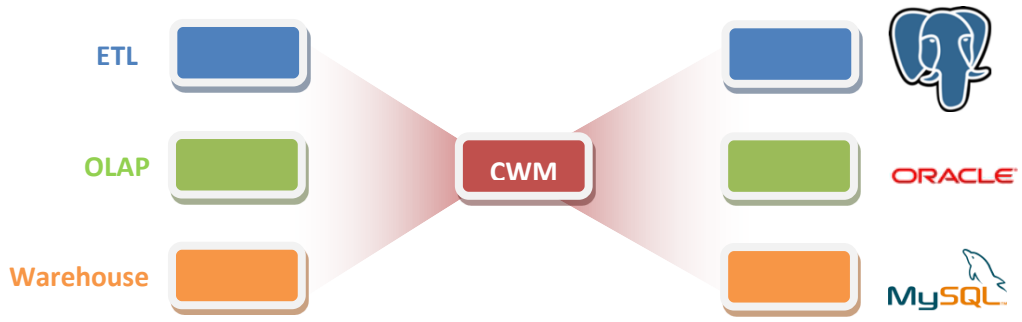


Figura 2.2. Puente de comunicación CWM entre herramientas y diferentes bases de datos.

La Figura 2.2. muestra cómo el uso de un modelo de metadatos común como puente de comunicación entre las diferentes tecnologías de bases de dato reduce el número de desarrollos para comunicarse entre los diferentes modelos, así como [7]:

- **Portabilidad**
  Aumentando el re-uso de las aplicaciones y reduciendo el costo y complejidad del desarrollo y administración de las aplicaciones.

- **Interoperabilidad entre plataformas**
  Usando métodos rigurosos para garantizar que los estándares basados en implementaciones de tecnologías múltiples tengan todas idénticas reglas de negocio.

- **Independencia de plataforma**
  Reduciendo el tiempo, costo y complejidad asociada con aplicaciones desplegadas en diferentes tecnologías.

- **Especificación del dominio**
  A través de modelos específicos del dominio, que permiten implementaciones rápidas de aplicaciones nuevas, en una industria específica sobre diversas plataformas.

- **Productividad**
  Permitiendo a los desarrolladores, diseñadores y administradores de sistemas usar lenguajes y conceptos con los que se sienten cómodos, facilitando la comunicación e integración transparente entre los equipos de trabajo.

2.3. Model Driven Architecture (MDA)

CWM forma parte de los estándares (UML, XMI y MOF) que dan soporte a la arquitectura dirigida por modelos (MDA). Esta arquitectura utiliza los modelos como principal fuente de desarrollo, permitiendo una fácil utilización y transformación en sistemas altamente cambiantes con desarrollos o plataformas no heterogéneas (diferentes lenguajes de programación, fabricantes, etc.) (Figura 2.3.).

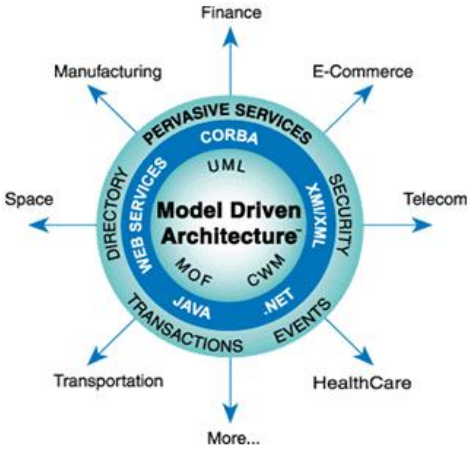


Figura 2.3. Arquitectura dirigida por modelos (MDA).

Un modelo no es más que una representación abstracta de una parte de una aplicación o sistema. Podemos modelar algo tan específico como las clases que componen la interfaz de usuario, algo tan amplio como la distribución de los datos y la funcionalidad en toda la red, o cualquier otra cosa [8]. La administración y uso inteligente de los metadatos de aplicaciones, plataformas, herramientas y bases de datos son la clave para la integración e interoperabilidad del sistema.

> *Si algo va a cambiar mucho, hazlo fácil de cambiar*
>
> - [9] J. Clauberg, 1647

MDA contiene cuatro principios fundamentales [10] [8]:

- Los modelos expresados en una buena notación son la base para entender los sistemas y soluciones escalables de una empresa.
- El desarrollo de sistemas puede ser organizado en un conjunto de modelos y la realización de una serie de transformaciones entre los modelos.
- La generalización en la descripción de modelos en un conjunto de modelado de metadatos, permite la automatización e integración de herramientas entre los diferentes modelos.
- La integración de estos modelos estándares en los diferentes fabricantes, permiten una apertura al modelo a los consumidores y diferentes herramientas, favoreciendo la interacción entre diferentes fabricantes.

A continuación vamos a describir los diferentes estándares de esta arquitectura y la interacción que estos tienen con el estándar CWM.

2.3.1. MetaObject Facility (MOF)

MOF es el estándar adoptado por el grupo OMG's para la representación de meta-modelos. El meta-modelo definido por CWM se basa en las especificaciones establecidas en MOF, permitiendo el uso de XMI en el intercambio de metadatos en almacenes de datos (Warehouse) y el acceso a los metadatos definidos utilizando Interface Description Language (IDL) como lenguaje de programación (Figura 2.4.).

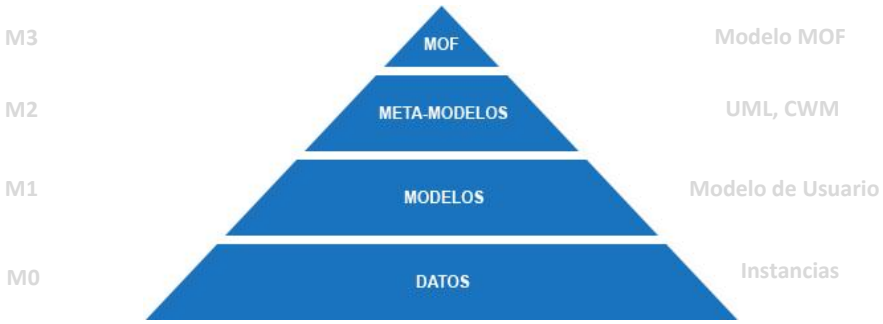


Figura 2.4. Arquitectura MOF de cuatro capas [7].

La utilización de IDL es relevante en la construcción de modelos de objetos CWM en memoria o guardados en repositorios, permitiendo la definición de interfaces, métodos y la estructura necesaria que el modelo debe soportar [11] (Ejemplo 2.1.).

```
interface Dimension : DimensionClass, Foundation::Core::Class
{
   boolean is time() raises (Reflective::MofError);
   void set_is_time(in boolean new_value)
   raises (Reflective::MofError);
   boolean is_measure ()
   raises (Reflective::MofError);
   void set is measure (in boolean new value)
   raises (Reflective::MofError);
};
```

Ejemplo 2.1. Definición con IDL de exclusión mutua en el meta objeto Dimension OLAP CWM.

MOF soporta cualquier tipo de metadatos que puedan ser descritos utilizando técnicas de Modelado de Objetos. Los metadatos pueden describir cualquier aspecto de un sistema y la información que este contiene y puede llegar a describirlo en cualquier nivel de detalle dependiendo de los requerimientos de estos metadatos.

La especificación de MOF provee [7]:

- Un modelo abstracto de los objetos y sus relaciones.
- Un conjunto de reglas para mapear un meta-modelo a interfaces independientes del lenguaje.
- Reglas definiendo el ciclo de vida, composición y semántica de los modelos basados en MOF.
- Una jerarquía de interfaces reflexivas definiendo operaciones genéricas para descubrir y manipular las propiedades de los meta-modelos basados en MOF.

2.3.2. Unified Modeling Language (UML)

UML, la especificación de OMG más usada, define un lenguaje de modelado que a través de MOF unifica cada proceso de desarrollo e integración de un modelo de negocio, aplicación, desarrollo, implantación y evolución [12] (Figura 2.5.).

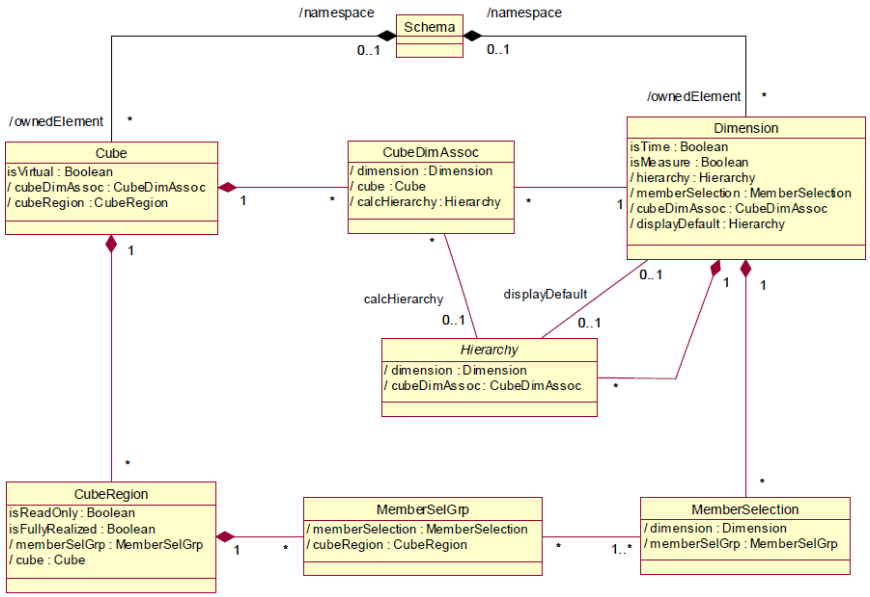


Figura 2.5. Diagrama de clases UML del paquete OLAP.

El modelo de metadatos CWM es una extensión del modelo de UML, incluyendo en éste todos los componentes relacionados con objetos manejados para diseñar un almacén de datos (warehouse). La notación UML es usada como representación del diagrama del modelo de metadatos CWM. Algunas restricciones expresadas en el modelo de metadatos de CWM están recogidas en Object Constraint Language (OCL), definido en la especificación de UML.

2.3.3. XML Metadata Interchange (XMI)

XMI es el mecanismo de intercambio de metadatos usado por CWM, por lo que no requiere ninguna extensión en este sentido. XMI define un formato de intercambio de modelos usando XML. XMI define la mayoría de los aspectos relacionados con descripción de objetos en XML (Ejemplo 2.2.), como son:

- La representación de objetos en XML.
- Incluye mecanismos estándares para la referencia de objetos en uno o a través de varios ficheros.
- Identificación de objetos mediante identificadores IDs e identificadores universales UUIDs.
- Validación de documentos XMI usando XML Schemas.

```
<xsd:complexType name="Documentation">
  <xsd:choice minOccurs="0" maxOccurs="unbounded">
   <xsd:element name="contact" type="xsd:string"/>
   <xsd:element name="exporter" type="xsd:string"/>
   <xsd:element name="exporterVersion" type="xsd:string"/>
   <xsd:element name="longDescription" type="xsd:string"/>
   <xsd:element name="shortDescription" type="xsd:string"/>
   <xsd:element name="notice" type="xsd:string"/>
   <xsd:element name="owner" type="xsd:string"/>
   <xsd:element name=timestamp" type="xsd:datetime"/>
   <xsd:element ref="Extension"/>
  </xsd:choice>
  <xsd:attribute ref="id"/>
  <xsd:attributeGroup ref="ObjectAttribs"/>
  <xsd:attribute name="contact" type="xsd:string" use="optional"/>
  <xsd:attribute name="exporter" type="xsd:string" use="optional"/>
  <xsd:attribute name="exporterVersion" type="xsd:string" use="optional"/>
     12 MOF 2 /XMI Mapping Specification, v2.4.1
  <xsd:attribute name="longDescription" type="xsd:string" use="optional"/>
  <xsd:attribute name="shortDescription" type="xsd:string" use="optional"/>
  <xsd:attribute name="notice" type="xsd:string" use="optional"/>
  <xsd:attribute name="owner" type="xsd:string" use="optional"/>
</xsd:complexType>
<xsd:element name="Documentation" type="Documentation"/>
```

Ejemplo 2.2. Ejemplo documento XMI.

XMI define reglas de producción EBNF para crear documentos XML y esquemas que comparten objetos constantemente. MOF es la tecnología base para estos modelos de metadatos, siendo a su vez UML (un subconjunto limitado) la base de éste (Figura 2.6.).

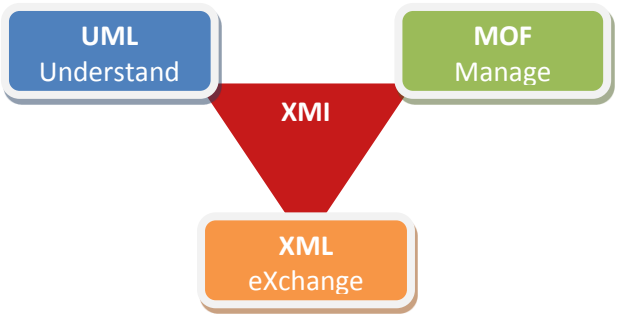


Figura 2.6. Integración de estándares en XMI.

Un documento estándar XML del modelo de metadatos CWM puede ser generado a través de las reglas de producción de documentos XMI basadas en MOF DTD.

2.4. Organización de CWM

El modelo de metadatos de CWM utiliza una estructura jerárquica de paquetes para mejorar su uso, comprensión y reutilización. En la Figura 2.7. se muestran los diferentes paquetes agrupados por funcionalidad.

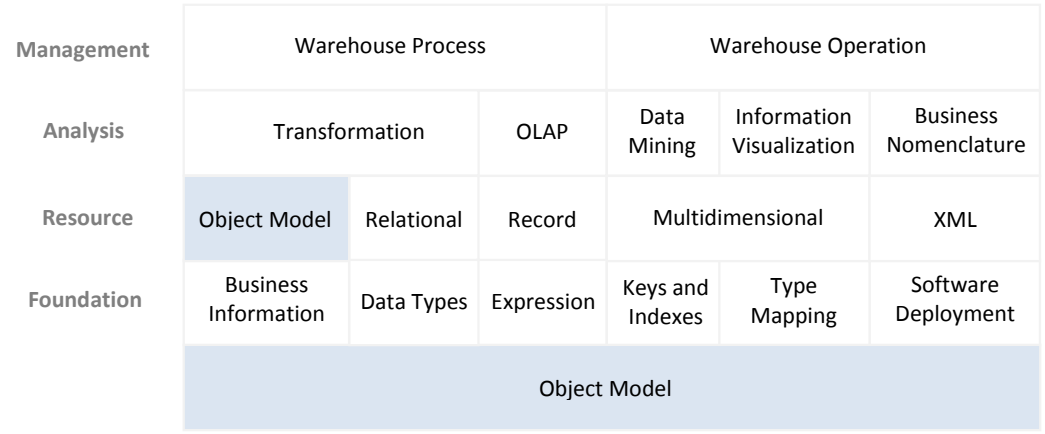


Figura 2.7. Modelo de metadatos CWM.

En las siguientes tablas se definen de forma general el contenido y propósito de cada uno de los paquetes especificados en CWM, centrándonos más específicamente en el paquete relacional en el apartado 2.4 de este capítulo sobre CWM.

ObjectModel

Paquete	Descripción
Core	Contiene clases y relaciones que forman la base del modelo de objetos CWM, el cual es usado por el resto de paquetes definidos en CWM.
Behavioral	Contiene clases y relaciones que describen el comportamiento de los objetos en CWM.
Relationships	Contiene clases y relaciones que describen las relaciones entre objetos CWM.
Instance	Contiene clases y relaciones que representan instancias de clasificaciones en CWM.

Tabla 2.1. Descripción de paquetes del modelo de objetos en CWM.

Foundation

Paquete	Descripción
Business Information	Contiene clases y relaciones que representan la lógica de negocio de los elementos del modelo.
Data Types	Contiene clases y relaciones que representan las construcciones necesarias para crear tipos de datos necesarios en el diseño de modelos.
Expressions	Contiene clases y relaciones que representan expresiones
Keys and Indexes	Contiene clases y relaciones que representan claves e índices
Software Deployment	Contiene clases y relaciones que representan cómo desplegar el software en un almacén de datos.
Type Mapping	Contiene clases y relaciones que representan mapeos de tipos de datos entre diferentes sistemas.

Tabla 2.2. Descripción de paquetes del modelo Fundación en CWM.

Resource

Paquete	Descripción
Relational	Contiene clases y relaciones que representan metadatos de fuentes de datos relacionales.
Record	Contiene clases y relaciones que representan metadatos de fuentes de datos de registros.
Multidimensional	Contiene clases y relaciones que representan los metadatos de fuentes de datos multidimensionales.
XML	Contiene las clases y relaciones que representan los metadatos de fuentes de datos XML.

Tabla 2. 3. Descripción de paquetes del modelo Resource en CWM.

Analysis

Paquete	Descripción
Transformation	Contiene clases y relaciones que representan los metadatos de herramientas de transformación de datos.
OLAP	Contiene clases y relaciones que representan los metadatos de herramientas analíticas de procesamiento online.
Data Mining	Contiene clases y relaciones que representan los metadatos de herramientas para la explotación de datos.
Information Visualization	Contiene clases y relaciones que representan los metadatos de herramientas de visualización de la información.
Business Nomenclature	Contiene clases y relaciones que representan los metadatos en taxonomías y glosarios.

Tabla 2.4. Descripción de paquetes del modelo Análisis en CWM.

Management

Paquete	Descripción
Warehouse Process	Contiene clases y relaciones que representan metadatos de procesos en almacenes de datos
Warehouse Operation	Contiene clases y relaciones que representan metadatos de resultados de operaciones sobre almacenes de datos.

Tabla 2.5. Descripción de paquetes del modelo Management en CWM.

2.5. Paquete Relacional de CWM

En el marco de este proyecto nos vamos a centrar en los paquetes que define CWM para la modelización de metadatos de una base de datos relacional. CWM se basa en el estándar SQL para el diseño y modelización de los metadatos de una base de datos relacional.

Figura 2.8. Principales fabricantes de bases de datos relacionales.

Los diferentes fabricantes de bases de datos relacionales contienen modificaciones sobre este estándar SQL, interpretando y añadiendo sus propios elementos en el modelado, aumentando

así el coste de desarrollo de soluciones en sistemas heterogéneos. En la Figura 2.8. vemos algunos de los principales fabricantes de bases de datos.

Cada fabricante de bases de datos relacional modela sus propios catálogos o diccionarios de datos donde almacenan los metadatos que definen cada instancia de los elementos definidos (Tablas, Procedimientos, Usuarios, Índices, Esquemas…etc.). CWM modela los metadatos necesarios para la modelización de una base de datos relacional unificando la construcción y transformación de estos modelos de manera que se reduzcan costes y aumente la integración y reutilización en sistemas altamente cambiantes o heterogéneos.

En los siguientes apartados se describen los modelos de metadatos relacionados con el paquete relacional en CWM.

2.5.1. Modelo de metadatos de Objetos

El modelo de metadatos de Objetos definido en CWM es utilizado por todos los paquetes establecidos en CWM como base para la modelización de los metadatos de sus clases. Este modelo es un subconjunto del modelo definido en UML, conteniendo las clases que ofrecen la potencia de modelado.

El modelo de metadatos de Objetos propuesto por CWM se compone de cuatro paquetes como se expone en la Figura 2.9. A continuación se describen en detalle los paquetes necesarios para poder diseñar un modelo de metadatos de una base de datos relacional mediante CWM.

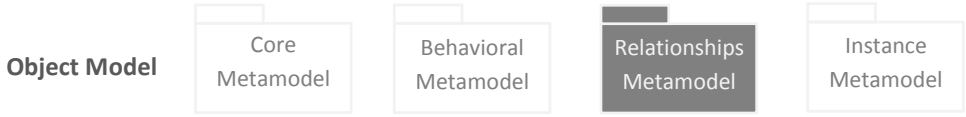


Figura 2.9. Paquetes del modelo de Objetos definidos en CWM.

Core Metamodel

El paquete del modelo de metadatos del Core es el único paquete de CWM que no tiene dependencias con otros paquetes de CWM. El paquete de objetos del Core contiene un modelo básico de clases y relaciones usadas por el resto de paquetes declarados en CWM, incluido los otros paquetes del modelo de objetos. La jerarquía de clases y relaciones que se declaran en el modelo del Core son mostradas en la Figura 2.10. y Figura 2.11.

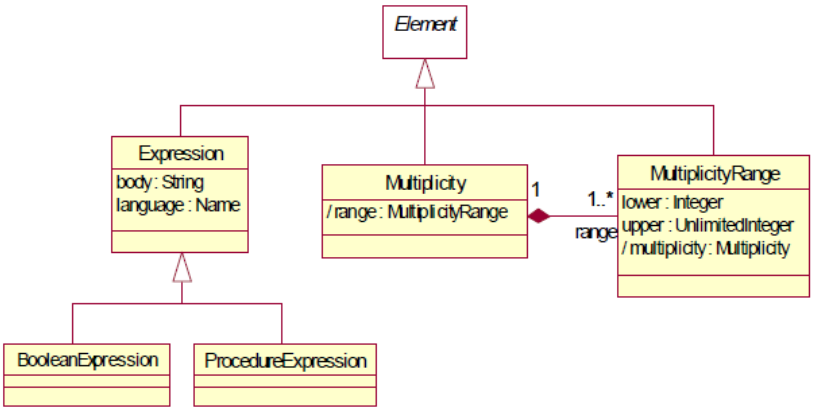


Figura 2.10. Diagrama de clases del Core CWM (Expression, Multiplicity, MultiplicityRange).

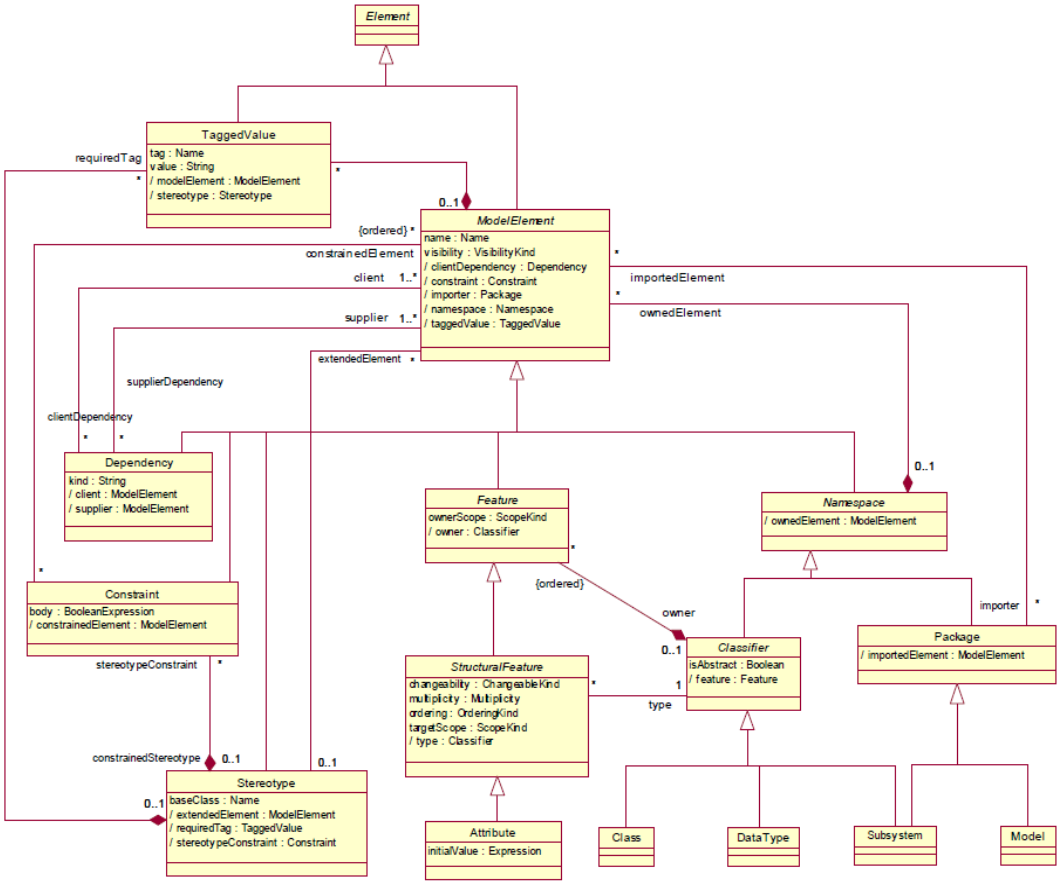


Figura 2.11.Diagrama de clases del paquete CWM Core.

Por ejemplo, la definición del paquete Core de las clases Paquete, Clase y Atributo, definidas en la Figura 2.11, son la base que usa el modelo de metadatos relacional para modelar en una base de datos relacional los conceptos como Catálogo/Esquema, Tabla y Columna.

Además de establecer las clases y relaciones sobre las que se basan los demás paquetes de CWM, el paquete Core también establece los tipos de datos básicos y enumerados usados en CWM, como se muestra en la Tabla 2.6.

Tipos Básicos		Típos enumerados
Any	Boolean	ChangeableKind
Float	Integer	OrderingKind
Name	String	ScopeKind
Time	UnlimitedInteger	VisibilityKind

Tabla 2.6. Tipos definidos en el paquete Core de CWM.

Behavioral Metamodel

El modelo de metadatos Behavioral contiene las clases y relaciones que describen el comportamiento de los tipos de CWM y provee un conjunto de invocaciones de determinados patrones. Este paquete del modelo de metadatos de objetos depende a su vez del paquete Core del mismo modelo como se puede ver en la Figura 2.12.

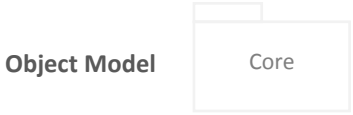


Figura 2.12. Dependencia con el paquete Core en el modelo de metadatos Behavioral definido en CWM.

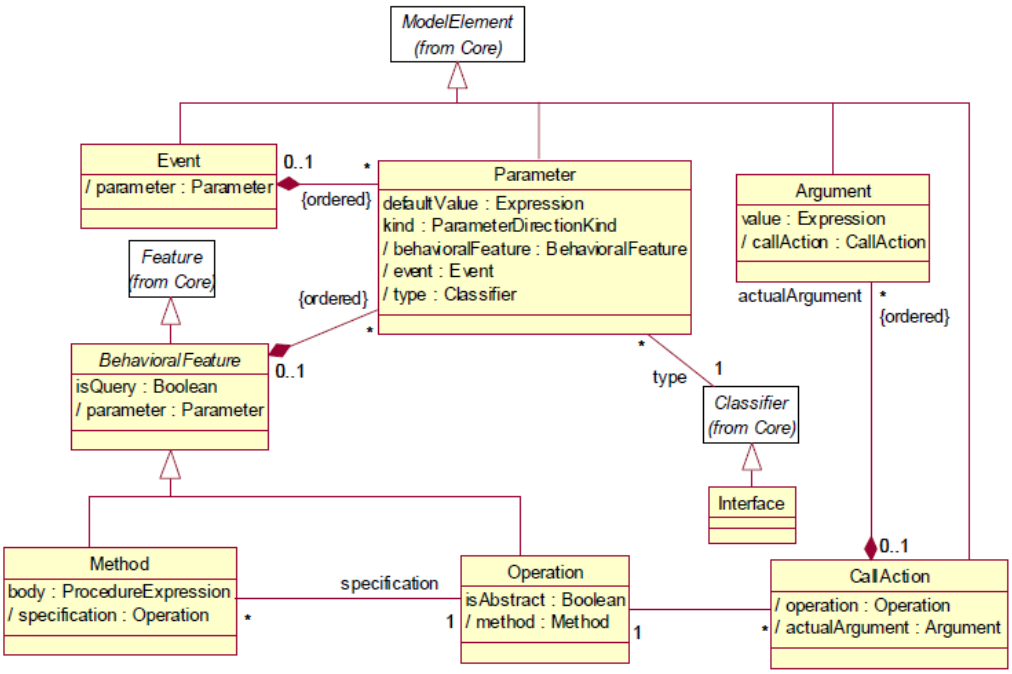


Figura 2.13. Diagrama de clases del paquete Behavioral CWM.

Las definiciones y relaciones entre las clases para este paquete, se muestran en la Figura 2.13, expresando conceptos de comportamiento, como son los argumentos de una operación, que pueden encontrarse, por ejemplo, en la definición de Procedimientos Almacenados de una base de datos relacional, ya sean como entrada o salida.

Instance Metamodel

El paquete de Instancias del modelo de metadatos definido en CWM contiene clases y relaciones que son utilizadas para la representación de instancias de clasificadores. Estas instancias son también utilizadas para el intercambio de datos específicos, permitiendo así la inclusión de valores de datos junto a los metadatos (Figura 2.14.).

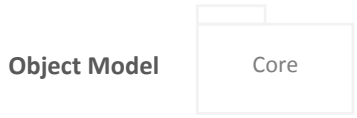


Figura 2.14. Dependencia con el Paquete Core del Modelo de Metadatos de Instancias definido en CWM.

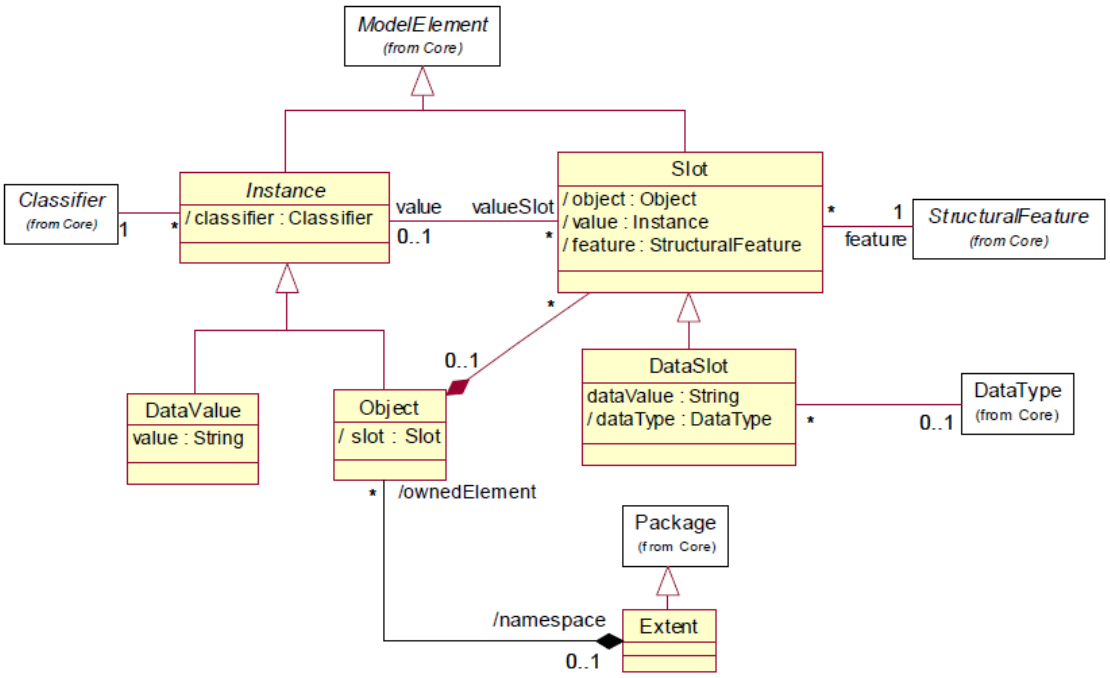


Figura 2.15. Diagrama de clases del paquete Instance CWM.

La clase Slot, representada en la Figura 2.15, es un contenedor genérico que almacena datos de dos formas diferentes, ya sean valores de datos o instancias de objetos. La primera manera es creando una instancia de esta clase y asignando el valor a través de la relación SlotValue, mientras que la alternativa, es crear una instancia de la clase heredada DataSlot, y guardar el valor en su atributo dataValue.

2.5.2. Modelo de metadatos de Fundación

El modelo de CWM Foundation es una colección de paquetes (Figura 2.16.) que contienen elementos de modelado que representan conceptos y estructuras con un propósito general, que son compartidas por otros paquetes en CWM.

Los elementos del modelo de Foundation difieren de los del modelo de objetos porque éstos son específicos para conseguir objetivos y metas de CWM mientras que los elementos del modelo de Objetos, en contraste, son de un propósito general y aplicable a diversas áreas.

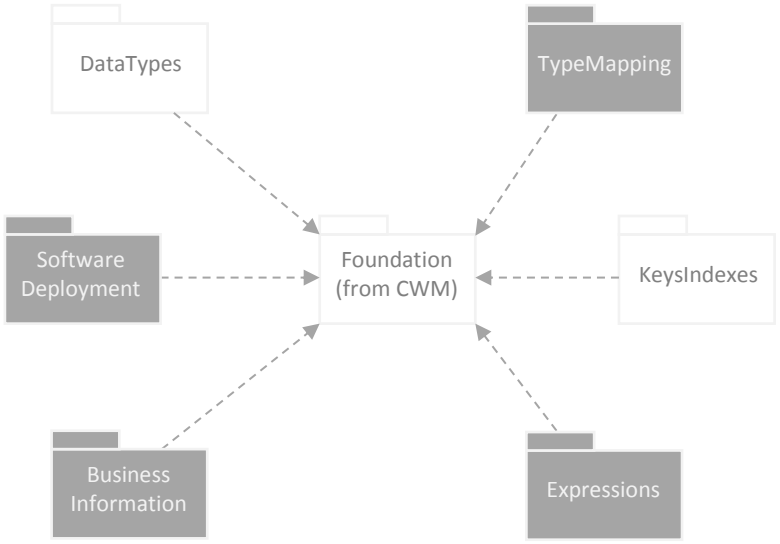


Figura 2.16. Diagrama de Paquetes que componen Foundation CWM.

En el ámbito de este proyecto, nos centraremos en los paquetes del modelo de Foundation que sirvan de base del modelo Relacional, es decir, los paquetes DataType y KeyIndexes.

DataType Metamodel

El paquete DataTypes definido en CWM contiene los conceptos y relaciones de clases que permiten modelar construcciones de metadatos que otros modelos pueden usar para crear tipos de datos específicos que éstos necesiten (Figura 2.18.). En el modelo Foundation no se establece ninguna definición de tipos de datos específicos. Este paquete tiene una dependencia con el paquete Core del modelo de Objetos CWM, como se muestra en la Figura 2.17.

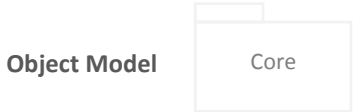


Figura 2.17. Dependencia del paquete DataType Metamodel definido en CWM.

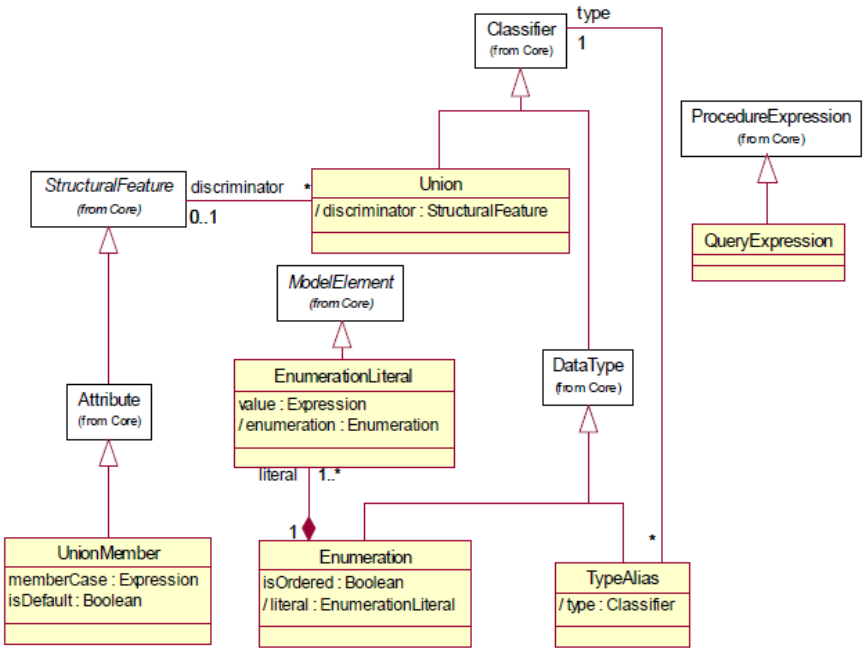


Figura 2.18. Diagrama de clases del paquete DataType CWM.

El paquete Relacional utiliza como clases base las de este paquete por ejemplo, la definición de tipos de datos requeridos en SQL. Por ejemplo, una Tabla de bases de datos contiene columnas que tienen asociado un tipo de dato específico, como puede ser VARCHAR, INTEGER o FLOAT. Estas definiciones pueden conseguirse a través de la clase SQLSimpleType parametrizando los valores de sus atributos, como se muestra en el ejemplo de la Figura 2.19. y se describe en la Tabla 2.7. Los valores de longitud o precisión de los tipos, serán almacenados en la columna como precisión o longitud de ese tipo. Por ejemplo, la columna Nombre tiene un tipo VARCHAR (45), se almacena la longitud del atributo de la columna con valor 45, mientras que en la instancia del tipo estará almacenado en el atributo characterOctetLength el valor 8 indicando que un carácter se codifica en 8 bits.

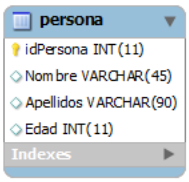


Figura 2.19. Representación de una tabla Persona.

Columns	SQLType	Instancia de	Atributos
idPersona	INTEGER	SQLSimpleType	characterMaximumLength = IDV characterOctetLength = null numericPrecision = IDV numericPrecisionRadix = 2 or 10 (IDV) numericScale = 0 dateTimePrecision = null
Nombre	VARCHAR	SQLSimpleType	characterMaximumLength = IDV characterOctetLength = null (defined in Column) numericPrecision = null numericPrecisionRadix = null numericScale = null dateTimePrecision = null
Apellidos	VARCHAR	SQLSimpleType	characterMaximumLength = IDV characterOctetLength = null (defined in Column) numericPrecision = null

			numericPrecisionRadix = null numericScale = null dateTimePrecision = null
Edad	SMALLINT	SQLSimpleType	characterMaximumLength = null characterOctetLength = null numericPrecision = IDV numericPrecisionRadix = 2 or 10 (IDV) numericScale = 0 dateTimePrecision = null

Tabla 2.7. Definición de tipos SQL asociados a las columnas de la tabla Persona.

KeyIndexes Metamodel

El paquete de modelado de metadatos "Keys and Indexes" en CWM contiene las clases que modelan las identificaciones y ordenaciones de instancias (Figura 2.21). El modelo Foundation CWM define los conceptos base de unicidad y relaciones implementadas como claves (Keys) cuyas estructuras son utilizadas, entre otros paquetes o herramientas de CWM, como clases base por el paquete Relacional para definir elementos y relaciones que se definen en una base de datos relacional, como son los índices y columnas clave de una tabla. El paquete KeyIndexes tiene como dependencias el paquete Core del modelo de objetos CWM (Figura 2.20.).

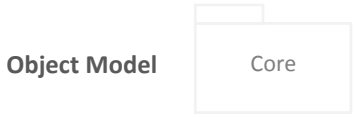


Figura 2.20. Dependencia del modelo de metadatos KeyIndexes definido en CWM.

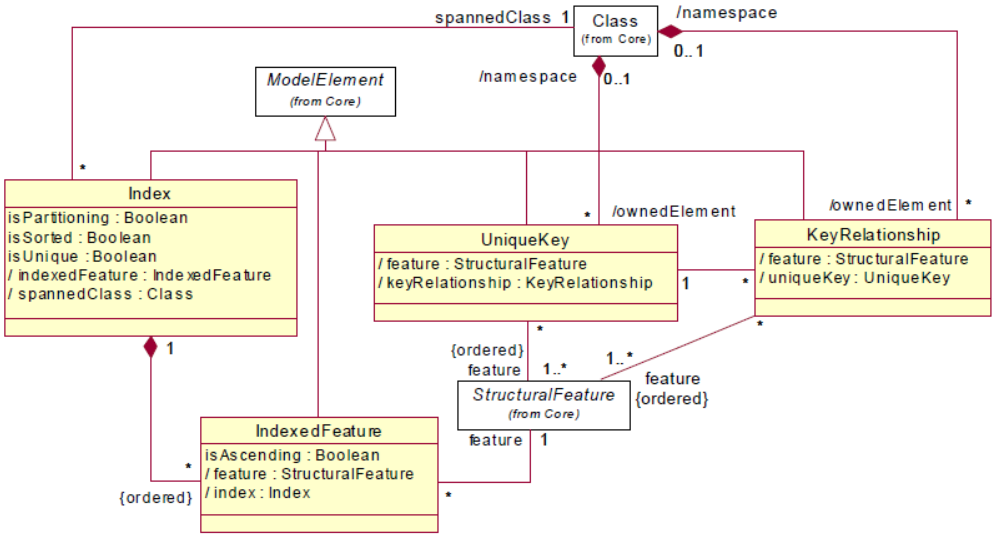


Figura 2.21. Diagrama de clases del paquete Key and Indexes CWM.

Un ejemplo de uso de instancias de Index, KeyRelationship y UniqueKey es la implementación la relación de una simple clave foránea que existe entre las estrellas y las constelaciones, representado en la Figura 2.22.

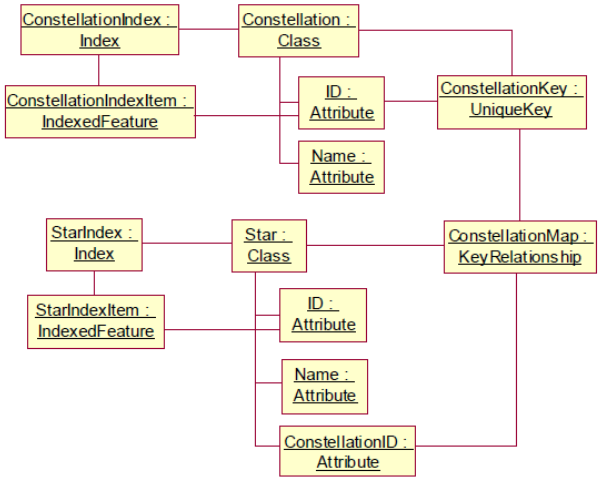


Figura 2.22. Ejemplo de uso de instancias de la clase Index en la relación entre estrella y constelación.

2.5.3. Modelo de metadatos Relacional

El modelo de metadatos Relacional definido en CWM describe el acceso a datos a través de una interface relacional como un acceso nativo RDBMS, ODBC o JDBC. El modelo Relacional está basado en el estándar SQL sección Catálogos de RDBMS.

El paquete Relacional usa elementos del modelo de Objetos para describir las extensiones de objeto añadidas a SQL por el estándar de SQL.

El paquete Relacional también se ocupa de cuestiones como la indexación, claves principales y claves externas mediante la ampliación de los conceptos correspondientes de los paquetes de la Foundation (Figura 2.23.).

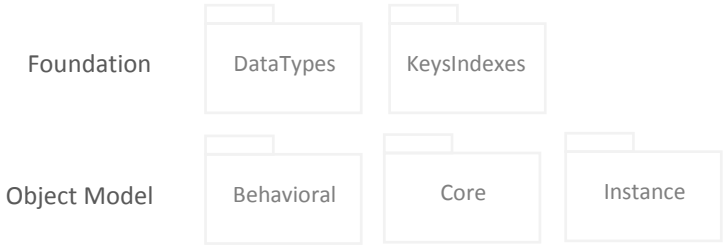


Figura 2.23. Dependencias del paquete Relacional CWM.

En la Figura 2.24. se muestran la Jerarquía de clases del modelo, así como las dependencias existentes entre otros modelos y paquetes definidos en CWM. Una breve descripción de cada una de estas clases es provista en la Tabla 2.8 para facilitar la comprensión de los siguientes apartados que desarrollan más en detalle las clases más importantes del modelo de metadatos relacional definido en CWM.

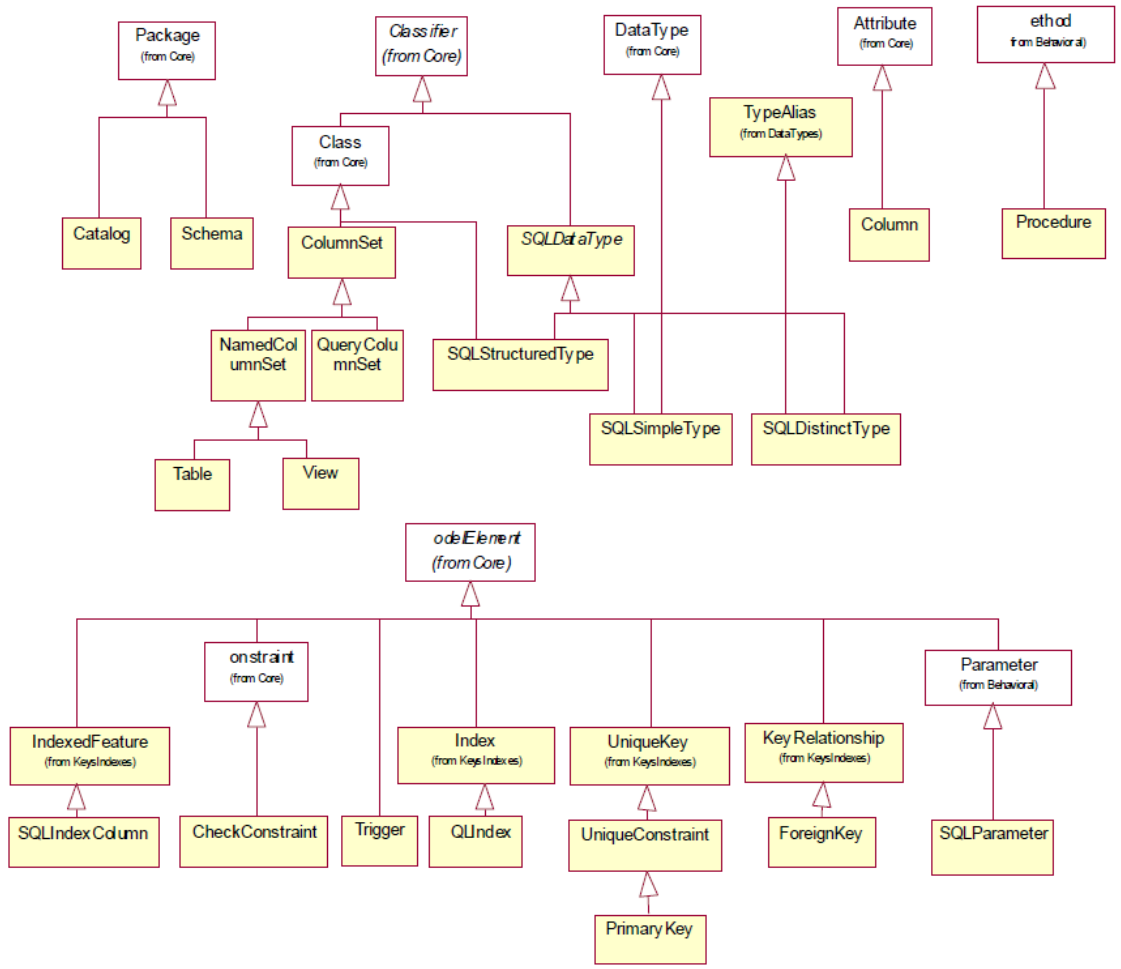


Figura 2.24. Diagrama de jerarquía de clases del paquete Relacional de CWM.

Clase	Descripción
Catalog	Es un contenedor de alto nivel que contiene todas las tablas de usuario que se pueden crear en una única sentencia.
CheckConstraint	Es una regla que especifica los valores permitidos en una o más columnas de cada fila de una tabla.
Column	Representa los valores de un conjunto de resultados de un tipo estructurado de un conjunto de resultados como son las vistas o tablas.
ColumnSet	Un conjunto de columnas representan el resultado de una consulta, vista o una tabla.
ColumnValue	El valor de una instancia Column
ForeignKey	Una clave foránea asociada a columnas de una tabla hace referencia a valores de otra tabla.
NamedColumnSet	Un conjunto de columnas catalogadas, pudiendo ser Vistas o Tablas.
PrimaryKey	Sólo existe una única restricción del tipo PrimaryKey por tabla. Cada RDBMS implementa su propia manera este tipo de restricción.
Procedure	Esta clase describe las funciones y procedimientos almacenados de una DBMS.
QueryColumnSet	El conjunto de resultados de una query.
Row	Una instancia de ColumnSet.
RowSet	Cada instancia de RowSet contiene una colección de instancias de Row.
Schema	Un esquema es una colección con nombre de tablas.
SQLDataType	SQLDataType es usado para referenciar cualquier tipo de datos asociados a una columna.
SQLDistinctType	Un tipo de dato definido como Distinct Type por el estándar SQL.

SQLIndex	Un Índice en una Tabla.
SQLIndexColumn	Asociación de un Índice con sus columnas.
SQLParameter	Parámetros de un procedimiento almacenado.
SQLSimpleType	Un tipo de datos simple usado en la definición de una columna. Unos ejemplos son Integer, Varchar, LOB, CLOB, etc.
SQLStructuredType	Un tipo de dato que se define estructurado por el estándar SQL.
Table	Es un NamedColumnSet materializado.
Trigger	Una acción ejecutada por la DBMS cuando cierto evento específico ocurre en una Tabla dueña del Trigger.
UniqueConstraint	Una condición que define la unicidad de las filas en una Tabla. Un ejemplo de UniqueConstraint es PrimaryKey.
View	Una vista es un conjunto de filas no materializadas, definidas por una query asociada.

Tabla 2.8. Descripción de las clases del paquete Relacional en CWM.

Catalogo/Esquemas

Un Catálogo es un contenedor de alto nivel que contiene todas las Tablas de usuario que se pueden crear. Un Catálogo es también la unidad que es manejada por una fuente de datos. Un Catálogo contiene esquemas que a su vez contienen Tablas, Vistas, Procedimientos y Triggers e Índices asociados a Tablas del Catálogo, como se muestra en la Figura 2.25. y la Tabla 2.9.

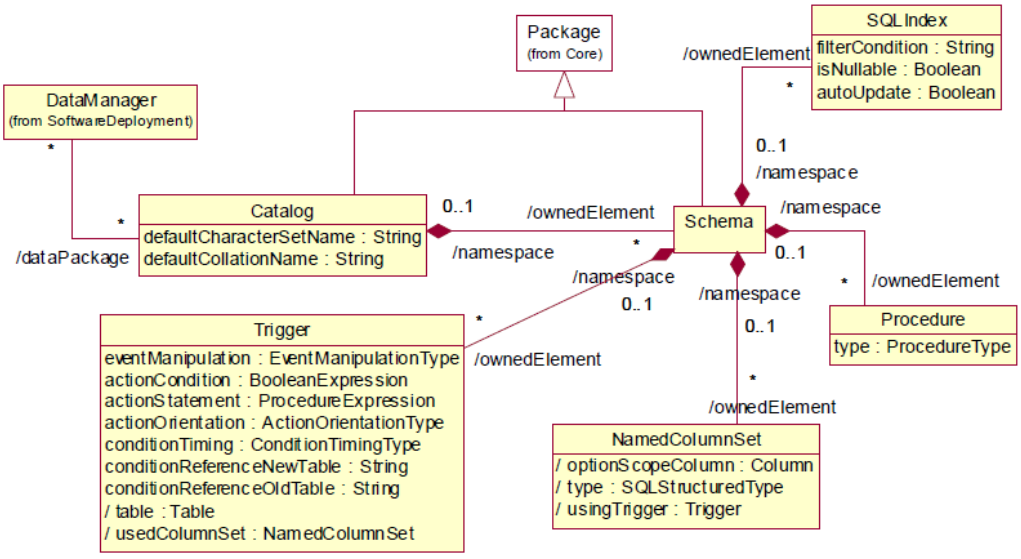


Figura 2.25. Diagrama de las principales clases y relaciones del paquete Relacional en CWM.

Clase	Descripción
Catalog	Es un contenedor de alto nivel que contiene todas las tablas de usuario que se pueden crear en una única sentencia.
NamedColumnSet	Un conjunto de columnas catalogadas, pudiendo ser Vistas o Tablas.
Procedure	Esta clase describe las funciones y procedimientos almacenados de una DBMS.
Schema	Un esquema es una colección con nombre de tablas.
SQLIndex	Un Índice en una Tabla.
Trigger	Una acción ejecutada por la DBMS cuando cierto evento específico ocurre en una Tabla dueña del Trigger.

Tabla 2.9. Descripciones de clases del modelo relacional asociadas a un esquema.

NamedColumnSet (Tablas/Vistas)

Las tablas son construidas por columnas que tienen asociadas un tipo de datos. Si el conjunto de filas asociadas a estas columnas no están materializadas son llamadas Vistas y tienen asociada una consulta (Query) que define las columnas y las condiciones que las filas cumplen (Figura 2.26.).

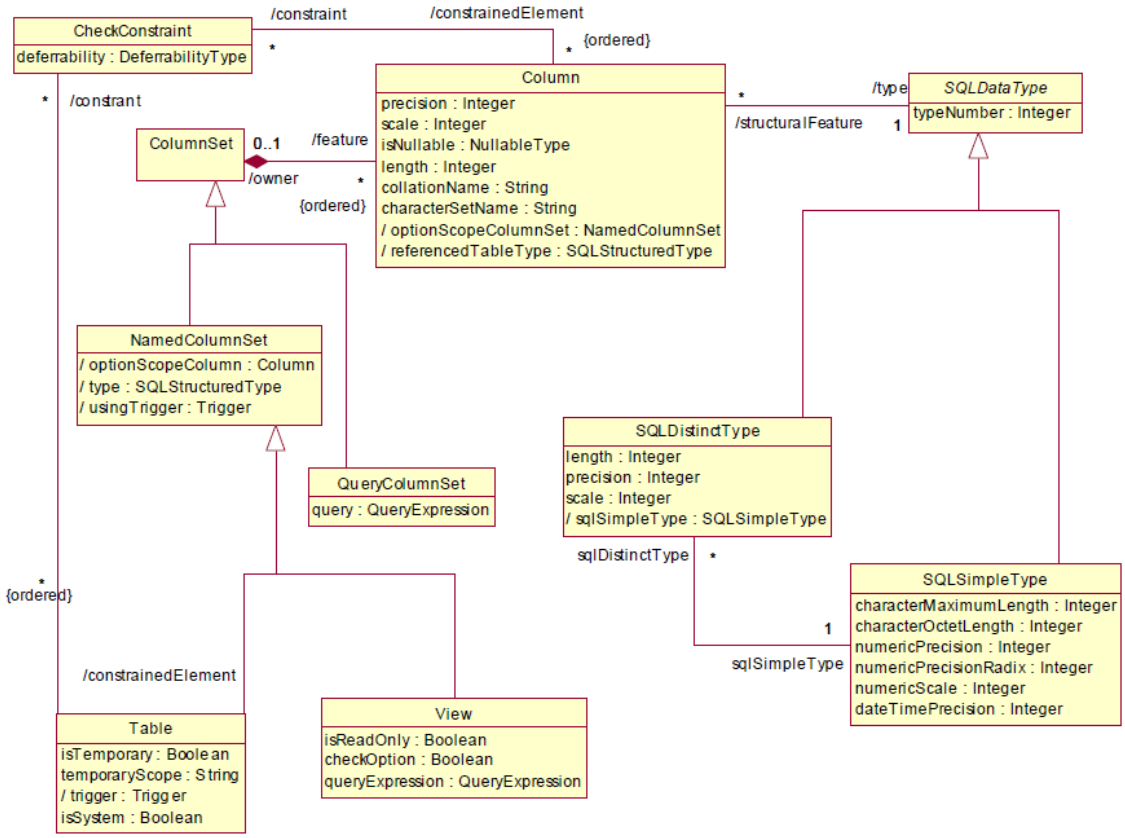


Figura 2.26. Ampliación del diagrama de clases del paquete Relacional CWM.

Por ejemplo, en la Figura 2.27. y la Tabla 2.10. se describe la modelización de una Tabla Persona definida en MySQL, que contiene diferentes columnas con tipos de datos asociadas a ellas.

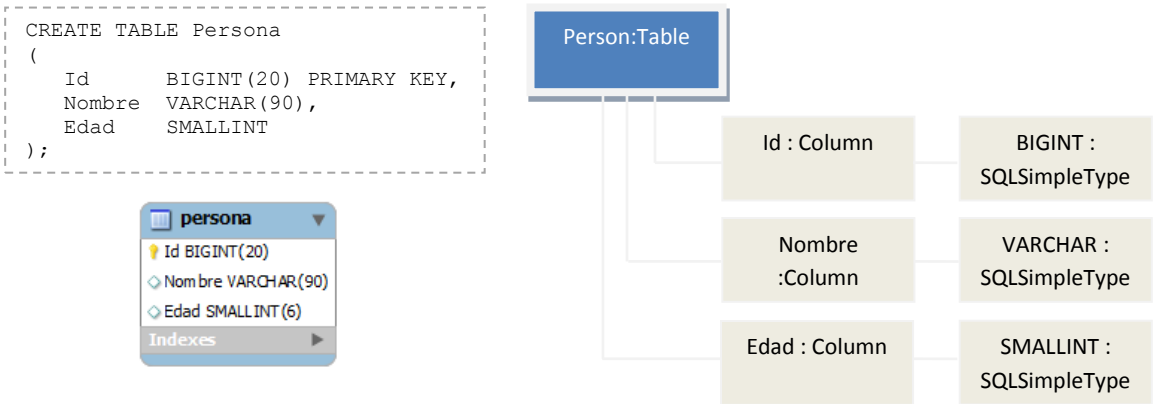


Figura 2.27. Ejemplo de representación de una tabla SQL a una instancia de Tabla CWM.

Clase	Descripción
CheckConstraint	Es una regla que especifica los valores permitidos en una o más columnas de cada fila de una tabla.

Column	Representa los valores de un conjunto de resultados de un tipo estructurado de un conjunto de resultados como son las vistas o tablas.
ColumnSet	Un conjunto de columnas representan el resultado de una consulta, vista o una tabla.
QueryColumnSet	El conjunto de resultados de una query.
SQLDataType	SQLDataType es usado para referenciar cualquier tipo de datos asociados a una columna.
SQLDistinctType	Un tipo de dato definido como Distinct Type por el estándar SQL.
SQLSimpleType	Un tipo de datos simple usado en la definición de una columna. Unos ejemplos son Integer, Varchar, LOB, CLOB, etc.
Table	Es un NamedColumnSet materializado.
View	Una vista es un conjunto de filas no materializadas, definidas por una query asociada.

Tabla 2.10. Descripciones de clases del modelo relacional asociadas a Tablas/Vistas.

Índices

Los Índices en bases de datos son estructuras utilizadas para optimizar el acceso y ordenación de los datos contenidos en Tablas y Columnas. El paquete relacional extiende el significado de la clase Index del paquete KeyIndexes para modelar los índices en bases de datos relacionales. En la Figura 2.28. y la Tabla 2.11. se muestra un diagrama de clases con las jerarquías y relaciones de los Índices modelados en CWM.

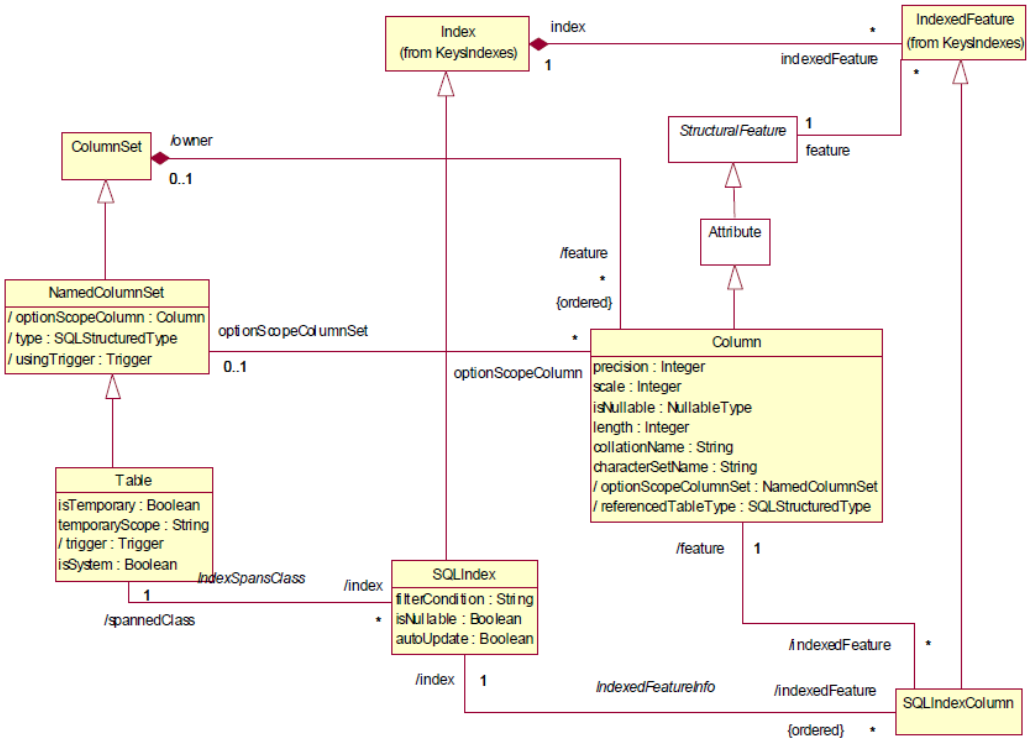


Figura 2.28. Diagrama de clases de Índices en el paquete Relacional CWM.

Clase	Descripción
SQLIndex	Un Índice en una Tabla.
SQLIndexColumn	Asociación de un Índice con sus columnas.

Tabla 2.11. Descripciones de clases del modelo relacional asociadas a Índices.

Triggers

Triggers son acciones que la base de datos realiza cuando se produce un cambio en la tabla asociada al Trigger. Un Trigger monitoriza una tabla que puede o no pertenecer al mismo esquema en el que fue creado éste. La clase Trigger es implementada en el paquete relacional para modelar este metadato, almacenando también las relaciones con las otras tablas utilizadas en la realización de las acciones definidas en el Trigger, como se muestra en la Figura 2.29. y la Tabla 2.12.

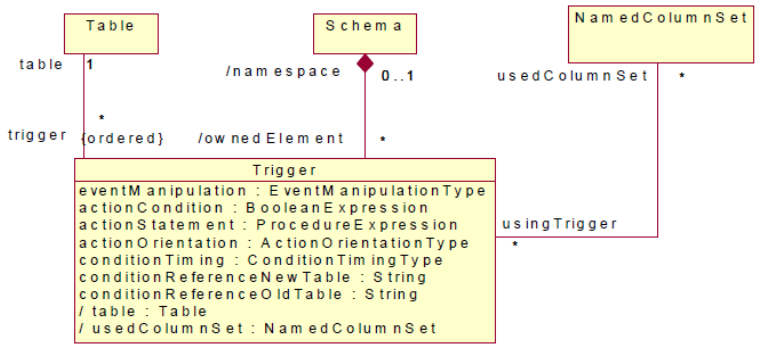


Figura 2.29. Diagrama de clases de Triggers en el paquete Relacional CWM.

Clase	Descripción
Trigger	Una acción ejecutada por la DBMS cuando cierto evento específico ocurre en una Tabla dueña del Trigger.

Tabla 2.12. Descripción de la clase Trigger del modelo relacional.

Procedimientos

Los procedimientos y funciones almacenadas de una base de datos relacional son modelados en el paquete relacional de CWM mediante la clase de Procedure. Esta clase hereda de la clase Method proveniente del modelo de metadatos de objetos y añade la propiedad type para diferenciar si el procedimiento devuelve o no un valor tras su ejecución (función o procedimiento respectivamente). En la Figura 2.30. y en la Tabla 2.13. se representa el modelo de metadatos referente a esta clase así como la relación con los parámetros de procedimientos.

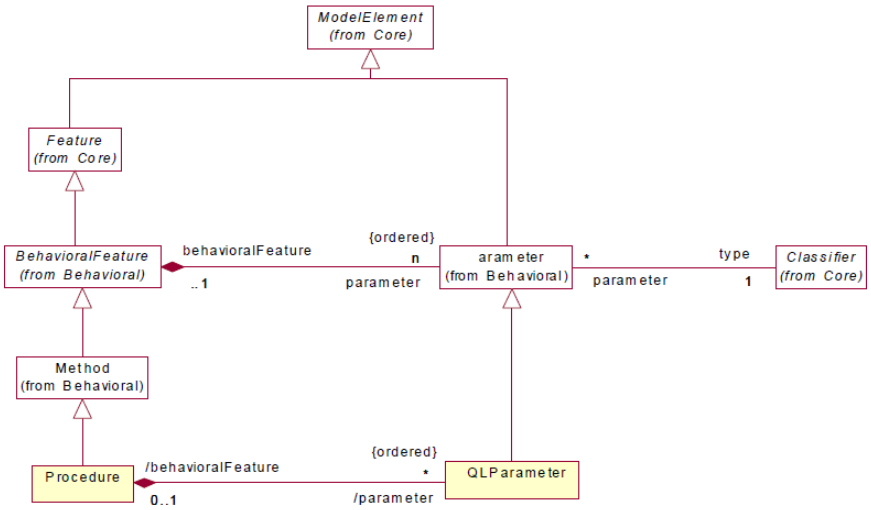


Figura 2.30. Diagrama de clases de Procedimientos del paquete Relacional CWM.

Clase	Descripción
Procedure	Esta clase describe las funciones y procedimientos almacenados de una DBMS.
SQLParameter	Parámetros de un procedimiento almacenado.

Tabla 2.13. Descripciones de clases del modelo relacional asociadas a Procedimientos.

Keys

Las claves o Keys en una base de datos relacional identifican una fila dentro de una tabla del modelo relacional, siendo implementado en el paquete Key&Indexes de CWM por la clase UniqueKey. En el paquete relacional se extiende este concepto, definiendo clases como UniqueConstraint (define una restricción a una columna de unicidad de valores), PrimaryKey (identifica de una forma única una fila dentro de una tabla) o ForeignKey (referencia a un valor que identifica una fila de otra tabla) (Figura 2.31. y Tabla 2.14.).

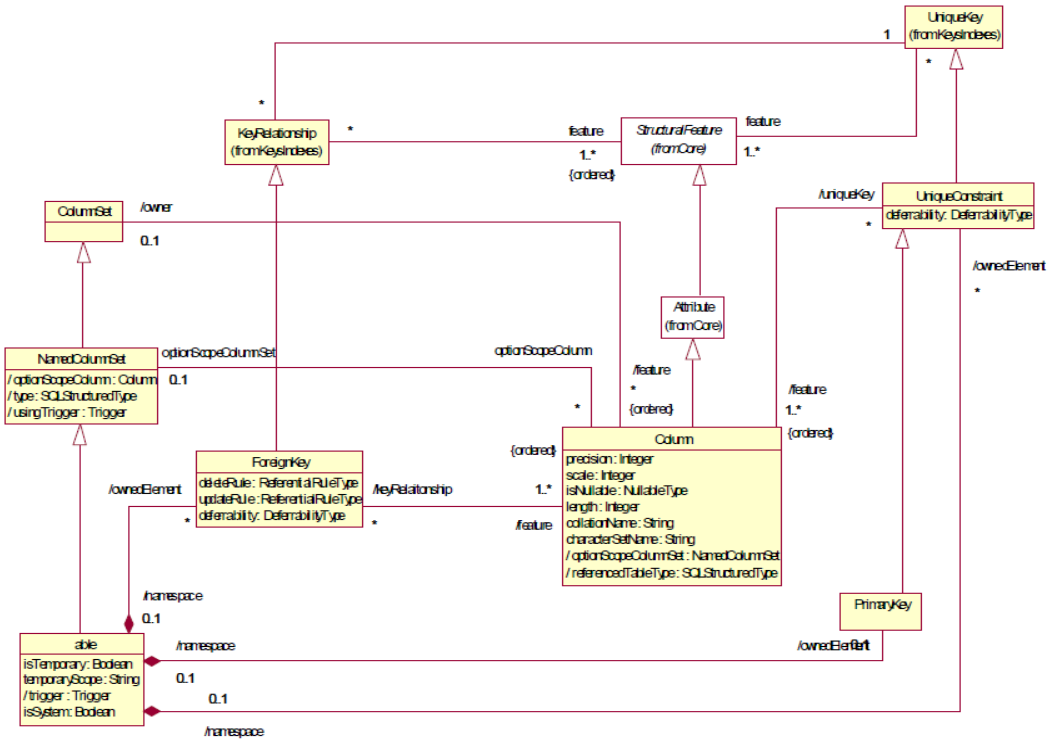


Figura 2.31. Diagrama de clases de UniqueConstraint y ForeignKey del paquete Relacional CWM.

Clase	Descripción
ForeignKey	Una clave foránea asociada a columnas de una tabla hace referencia a valores de otra tabla.
PrimaryKey	Sólo existe una única restricción del tipo PrimaryKey por tabla. Cada RDBMS implementa su propia manera este tipo de restricción.
UniqueConstraint	Una condición que define la unicidad de las filas en una Tabla. Un ejemplo de UniqueConstraint es PrimaryKey.

Tabla 2.14. Descripciones de clases del modelo relacional asociadas a Claves (Key).

Instancias

El paquete relacional de CWM también define las estructuras y clases que contienen los valores de metadatos como tablas o columnas. En la Figura 2.32. (Tabla 2.15.) se muestra la jerarquía y relaciones de las clases usadas para este modelado.

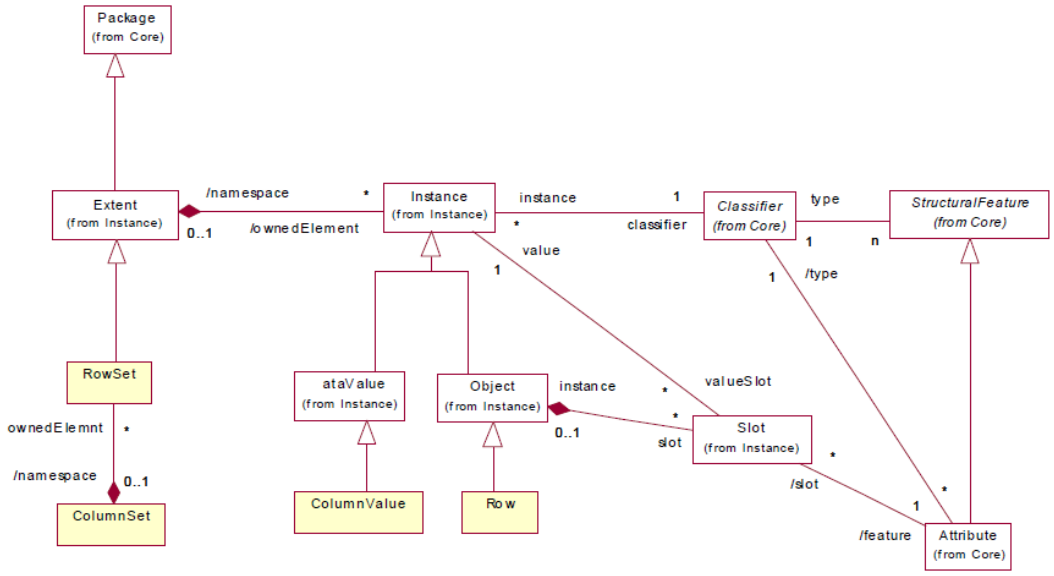


Figura 2.32. Diagrama de clases de Instancias en el paquete Relacional CWM.

Clase	Descripción
ColumnSet	Un conjunto de columnas representan el resultado de una consulta, vista o una tabla.
ColumnValue	El valor de una instancia Column
Row	Una instancia de ColumnSet.
RowSet	Cada instancia de RowSet contiene una colección de instancias de Row.

Tabla 2.15. Descripciones de clases del modelo relacional asociadas a Instancias.

2.6. Resumen

En este capítulo se presenta CWM como la propuesta del grupo OMG para tener un modelo común de metadatos sobre almacenes de datos, centrándonos especialmente en los metadatos de una base de datos relacional.

La utilización de CWM como modelo común, nos permite reducir costes y esfuerzos en el desarrollo de "puentes" o adaptadores entre los diferentes modelos existentes de los diversos fabricantes en sistemas heterogéneos. De este modo, las herramientas que hagan uso de él, pueden conseguir dar un mayor soporte a diferentes fabricantes, aumentando la competitividad de éstas.

A pesar de aumentar la integración e interoperabilidad entre sistemas heterogéneos, la gran cantidad de metadatos con significado específico en los diferentes sistemas sigue siendo una

carencia para la construcción de herramientas que integren estos metadatos de manera automatizada.

Para cubrir esta carencia, en este proyecto se propone la utilización de Ontologías para añadir significado a los metadatos modelados en CWM de manera. En el siguiente capítulo se expone de forma general la definición de ontologías y el Lenguaje de Ontologías Web (OWL).

Capítulo 3.

Web Ontology Language (OWL)

Contenido

3.1. Introducción

¿Qué es una Ontología?

A lo largo de la historia, tanto en el mundo de la Inteligencia Artificial como en el filosófico, se ha definido de diversas maneras el concepto de ontología:

> *La ciencia y filosofía primera que estudia el ente en cuanto ente.*
>
> - [9] [13] J. Clauberg, 1647

> Cristian Wolff introduce el término académico de *"ontología"* como una parte de la metafísica, y es llamada *"metafísica general"*.
>
> - [14] [13] C. Wolff, 1730

> *La ontología es una ciencia de esencias y se clasifica en formal y material. La primera estudia las esencias formales, esto es, aquellas que convienen a todas las demás esencias. La ontología material, por su parte, estudia las esencias materiales y está constituida por las ontologías regionales. La ontología formal implica las formas de todas las ontologías posibles y en ella está fundada la ontología material.*
>
> - [13] [15] E. Husserl, 1900

> *La ontología es una especificación de una conceptualización. Es decir, una ontología es una descripción (como una especificación formal de un programa) de los conceptos y relaciones que pueden existir para un agente o una comunidad de agentes.*

- [16] [17] T. R. Gruber, 1993

> *Consideremos en primer lugar la distinción entre "Ontología" (con la O mayúscula), como en la declaración "la ontología es una disciplina fascinante" y ontología (con la o minúscula), como en las expresiones "la ontología de Aristóteles" o CYC la ontología.*
>
> - [16] [18] N. Guarino, 1995

> *En la filosofía contemporánea, la ontología formal se ha desarrollado principalmente de dos maneras. El primer enfoque ha sido el estudio de la ontología formal como herramientas y métodos de ordenamiento de conocimientos, análisis, representación y transmisión de los mismos. La segunda línea de estudio nos devuelve a los orígenes de Edmund Husserl y el análisis de las categorías fundamentales del objeto.*
>
> - [19] [20] L. Albertazzi, 1996

> *Una ontología es una especificación formal de una conceptualización compartida*
>
> - [21] W. N. Borst, 1997

> *Las categorías y los modelos son entendidos como herramientas y métodos de ordenamiento de conocimientos, de su representación y su transmisión*
>
> - [22] [23] G. Negrini, 2000

> *Una ontología define un vocabulario común para investigadores que necesitan compartir información en un dominio. Ella contiene definiciones de conceptos básicos y sus relaciones que pueden ser interpretadas por una máquina.*
>
> - [24] N. F. Noy, 2005

En el marco de este proyecto, ante la gran cantidad de definiciones del término y relaciones existentes, será utilizada la siguiente definición:

> *Una ontología es una descripción explicita y formal de conceptos en un dominio de discurso (clases, a veces llamadas conceptos), propiedades de cada concepto describiendo varias características y atributos del concepto (slots, a veces llamados roles o propiedades), y restricciones sobre los slots (facetas, algunas veces llamadas restricciones de rol).*
>
> - [24] N. F. Noy, 2005

En esta definición de ontología se engloban diferentes conceptos, como son Clases, Propiedades o "Slots" y Restricciones. Las clases describen conceptos de un dominio, siendo el eje central de las ontologías.

Una clase puede tener subclases que representan conceptos que son más específicos que la superclase. Por ejemplo, una clase de vinos, puede dividirse en tres subclases según el color que estos posean, es decir, vinos tintos, blancos y rosados. A su vez, las propiedades de origen del vino o el tipo de efervescencia son otras divisiones posibles (Figura 3.1.).

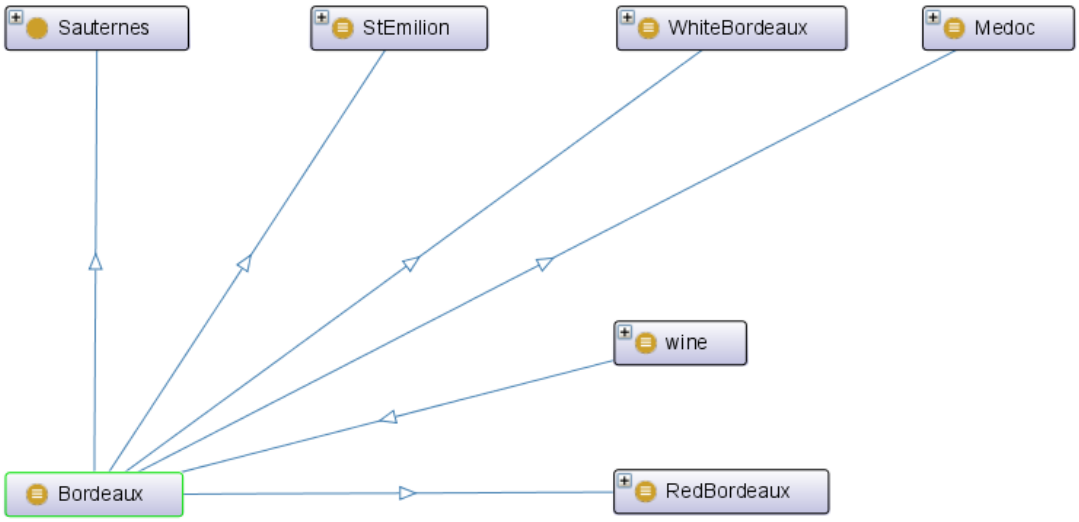


Figura 3.1. Ejemplo de clases de una ontología de vinos para la clase de vino Bordeux.

Cada clase definida tendrá a su vez una serie de propiedades que describen los individuos o instancias que pertenezcan a ellas. Los individuos que pertenezcan a una clase, deben de cumplir una serie de condiciones que se definen a través de las restricciones. Las restricciones dotan a las ontologías de una mayor expresividad a la hora de declarar dominios y rangos de las propiedades, así como nuevos términos o conceptos.

Utilización de las Ontologías

La definición de un vocabulario común así como las relaciones entre los términos utilizados en las diferentes áreas de conocimiento ha sido siempre uno de los objetivos del mundo de la Inteligencia Artificial. Categorizando los términos, se pueden establecer ciertas relaciones comunes entre ellos, extrayendo una mayor cantidad de información.

En biología, por ejemplo, ante la gran diversidad de posibles expresiones de vida y nuevas especies descubiertas, se utilizan taxonomías para clasificar a la nueva especie en una serie de categorías desde lo más general a lo más específico. Una taxonomía, puede contener una jerarquía, de forma que se pueda expresar ciertas restricciones de pertenencia y relaciones de grupos de términos.

La gran cantidad de datos e información que se produce en la actualidad, crea la necesidad de manejar esta información de manera automatizada y con un mayor contenido semántico. Las ontologías permiten definir un vocabulario de términos comunes, estableciendo relaciones entre sí (jerarquías, pertenencias, restricciones… etc.) con un significado semántico rico y entendible por una máquina. Hoy en día, en muchas y diversas disciplinas, el uso compartido de ontologías estandarizadas ha facilitado el uso e intercambio de información en estos campos. Algunos ejemplos de ontologías utilizadas son (Tabla 3.1.):

Ontología	Descripción
AIM@SHAPE	*Avanzados e innovadores modelos y herramientas para el desarrollo de sistemas semánticos para el manejo, adquisición y procesado de conocimiento embebido en objetos digitales multidimensionales.*
BIRNLex	*Una ontología creada por y para la red de investigación informática biomédica (BIRN) para proveer un framework semántico común con el que recolectar datos relacionados de BIRN, cruzar estudios de diagnósticos neurodegenerativos.*
BreastCancer	*Una ontología OWL para describir algunas características del cáncer de mamas.*
Countries.owl	*Lista de países según el código ISO 3166. Escrita por Dieter E. Jenz*
Consciousness1.owl	*Una ontología OWL que muestra el trasfondo de la conciencia, de acuerdo con la antigua literatura védica.*
DOLCE	*La ontología fundamental Dolce y sus extensiones proveen un framework independiente de dominio para construir ontologías basándose en patrones altamente reutilizables. Escrita por Aldo Gangemi.*
Ontologías de diseño de ingeniería	*Conjunto de ontologías desarrolladas por la universidad de Massachusetts Amherst para la representación de diferentes aspectos de procesos de desarrollo de productos.*
SNOMED	*Vocabulario estructurado estandarizado para describir términos en medicina. (Price and Spackman 2000).*
UNSPSC	*Dun & Bradstree unieron esfuerzos para desarrollar la ontología UNSPSC que provee terminología para productos y servicios (www.unspsc.org)*
Travel.owl	*Tutorial de una Ontología OWL para una Web Semántica de turismo. Escrita por Holger Knublauch.*
Elementos de videojuegos	*La ontología de elementos de videojuegos es usada para modelar diferentes propiedades de videojuegos como jugabilidad. Escrita por José Luis González y Francisco Luis Gutiérrez Vela. Universidad de Granada, España.*
Amino-acid.owl	*Pequeña ontología OWL de aminoácidos y sus propiedades.*

Tabla 3.1. Ejemplos de Ontologías.

3.2. Evolución de las Ontologías

Antes de llegar a definir las ontologías como las conocemos hoy en día, fueron surgiendo diferentes lenguajes y propuestas de desarrollo hasta llegar a OWL.

Simple HTML Ontology Extensions (SHOE)

El lenguaje de HTML está basado en XML, permitiendo mostrar la información en la Web de una manera legible por un ser humano. El "conocimiento" en una página web esta expresado en un lenguaje entendible por las personas, diseñado mediante tablas, gráficos y ventanas de manera que sea fácil de entender visualmente por las personas. El aumento de la información disponible en la Web, hacía cada vez más necesaria la aparición de agentes software que permita recolectar la información publicada en la Web de una manera automatizada (Ejemplo 3.1.).

```
<META HTTP-EQUIV="Instance-Key" CONTENT="http://www.cs.umd.edu/~george">
<USE-ONTOLOGY "our-ontology" VERSION="1.0" PREFIX="our" URL="http://ont.org/our-ont.html">
…
<CATEGORY "our.Person">
<RELATION "our.firstName" TO="George">
<RELATION "our.lastName" TO="Cook">
<RELATION "our.marriedTo" TO="http://www.cs.umd.edu/~helena">
<RELATION "our.employee" FROM="http://www.cs.umd.edu">
```

Ejemplo 3.1. Expresión SHOE [25].

En 1996 nace una pequeña extensión al lenguaje HTML, resolviendo algunos de estos problemas, permitiendo a los creadores de páginas web, por ejemplo, la inclusión de documentos web legibles por un agente software. SHOE hace posible la aparición de agentes software de inteligencia sobre la web, permitiendo la navegación, lectura y compresión de las páginas web por éstos [26].

Frames (Marcos)

El primer concepto de ***frames***, fue introducido por Minsky en 1975, donde en su trabajo, proponía formular una teoría global y coherente para representar el razonamiento de sentido común. Nacen por una necesidad de tener una mayor estructura en los nodos y en los arcos. Un concepto no existe en función de sus relaciones con otros conceptos (redes semánticas) sino como un todo. Su esencia ésta dada por el conjunto de atributos que lo definen.

Actualmente, el concepto de ***frames*** es definido como una colección de atributos, ranuras (slots), con valores asociados (y posibles restricciones entre los valores), que describe alguna entidad del mundo.

Existen dos tipos de ***frames***:

- **Individuales**: descripciones (aserciones) de una instancia individual de una entidad.
- **Genéricos**: descripciones (aserciones) de una categoría (clase) de entidades.

Los ***frames*** se adaptan muy bien a los problemas de clasificación, de definición de taxonomías. Las propiedades de las clases suelen ser los casos prototípicos, con lo que hay que considerar como dificultades la existencia de excepciones y las zonas grises de las taxonomías (límites difusos entre las clases) (Ejemplo 3.2.) [27].

```
SUPERCLASSES : Vehículos
SUBCLASSES: (Camiones-Volcadores, Camiones-Carrozados)
PERTENENCIA: (Clases-de-OBJETOS-Físicos)
M-SLOT: Dueño
CLASE-DE-VALORES : Personas
CRITERIO-HERENCIA: Prevalece-valor-particular
VALOR: Desconocido
M-SLOT: Peso
CLASE-DE-VALORES : Naturales
UNIDAD-DE-MEDIDA: Tonelada VALOR-MAXIMO: 1000 VALOR-MINIMO: 1 VALOR Desconocido
P-SLOT : Máximo-Peso
CLASE-DE-VALORES : Camiones VALOR-MAXIMO: 1 VALOR-MINIMO: 1 COMENTARIO: El mas pesado
conocido
```

Ejemplo 3.2. Expresiones Frames definidos utilizando el lenguaje KEE (Knowledge Engineering Enviroment).

Lógicas Descriptivas (DL)

Las ***Lógicas Descriptivas*** (Description Logics) son la evolución natural de las ***redes semánticas*** y los ***frames***, describiendo el conocimiento en términos de conceptos (clases) y restricciones de rol (relaciones) que son usados para taxonomías de clasificación, Ejemplo 3.3.

$$\text{progenitor} \doteq \text{persona} \sqcap \exists \text{hijo}.\top$$
$$\text{papa} \doteq \text{progenitor} \sqcap \text{hombre}$$
$$\text{papa}_m \doteq \text{papa} \sqcap \forall\, \text{hijo.mujer}$$
$$\text{abuelo} \doteq \text{papa} \sqcap \exists\, \text{hijo.progenitor}$$

Ejemplo 3.3. Expresiones de lógica descriptiva [28].

Las lógicas de descripción son normalmente encontradas en una base de conocimiento de un sistema de representación de conocimiento, como vemos en la siguiente Figura 3.2.:

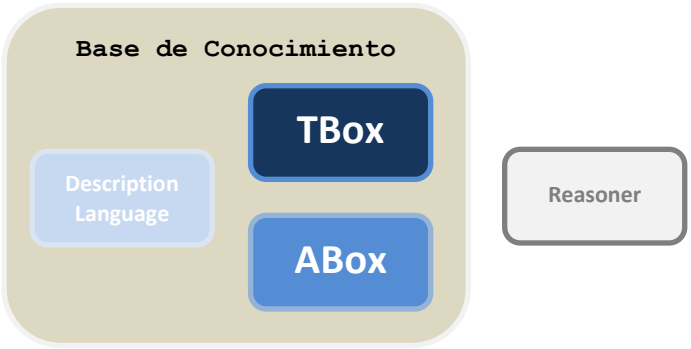


Figura 3.2. Sistema de Representación de Conocimiento.

Una base de conocimiento DL es tradicionalmente dividida en tres partes principales, descritas en la Tabla 3.2. [29].

Componente	Descripción
TBox	Terminología o esquema, el vocabulario de dominio de la aplicación
ABox	Las aserciones o afirmaciones, como son llamadas individuos expresados en términos de vocabulario
Lenguaje de descripción	Define los términos y operadores con los que construir expresiones

Tabla 3.2. Componentes de un Sistema de Representación de Conocimiento.

Resource Description Framework (RDF) / RDF Schema (RDFs)

En 1999, el consorcio W3C desarrolló un framework de descripción de recursos llamado RDF [30]. El propósito principal del lenguaje era codificar conocimiento de páginas Web para hacerlas entendibles a los agentes software que buscan información. RDF define un modelo de datos basado en triples: Objeto, propiedad y valor (Figura 3.3.).

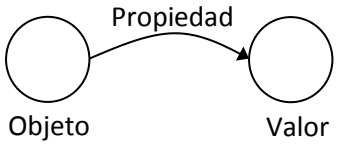
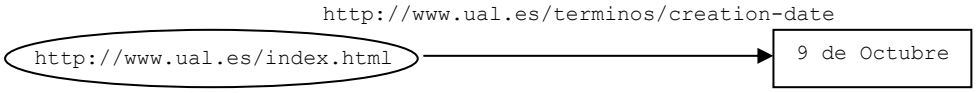


Figura 3.3. Propiedad RDF.

El lenguaje fue ampliado para conseguir un formalismo de representación mediante **RDF Schema** (**RDFS**) [30] introduciendo primitivas básicas de modelado ontológico en la web. Con RDFS, se puede hablar de clases, subclases, sub-propiedades, dominios y restricciones de rangos en propiedades,...etc en un contexto basado en la Web (Ejemplo 3.4.) (Tabla 3.3.).




```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:exterms="http://www.ual.es/terminos/">
    <rdf:Description rdf:about="http://www.ual.es/index.html">
      <exterms:creation-date>30 de febrero
      </exterms:creation-date>
   </rdf:Description>
</rdf:RDF>
```

Ejemplo 3.4. Representación RDF/XML.

La estructura de los metadatos que se definen en RDF se compone de tres paquetes:

Componente	Descripción
RDFBase	Refleja los conceptos fundamentales requeridos por todas las aplicaciones RDF
RDFSchema	El metamodelo de RDFS incluye los conceptos definidos en el paquete RDFBase, extendiéndolos para soportar el vocabulario del lenguaje definido en las especificaciones de RDF Schema
RDFWeb	El paquete RDFWeb incluye conceptos adicionales que no son específicos a RDF, pero que definen el contenido de un documento web, especificado en otros estándares de W3C y que son requeridos para la completa representación de cualquier serialización de RDF, incluyendo RDF/XML

Tabla 3.3. Estructura de paquetes de metadatos en RDFs.

Aunque RDF Schema añade un mayor contenido ontológico y semántico al lenguaje RDF, sigue dejando algunas carencias en el mismo como: i) la ausencia de cardinalidad, ii) los tipos de datos, iii) restricciones de rangos simples en las propiedades, iv) no permite la derivación de tipos por unión o intersección, v) la necesidad de relaciones entre clases o propiedades más ricas en sentido semántico. Todas estas limitaciones hacen que RDFS no sea considerado como un lenguaje ontológico completamente desarrollado.

Ontology Inference Layer / Interchange Language (OIL)

OIL toma a ***RDFS*** como punto de partida, ampliándolo y convirtiéndose en un lenguaje de ontología completamente desarrollado [31]. ***OIL*** añade los siguientes aspectos:

- Un modo más intuitivo de modelar y una extensión para enriquecer la manera de definir conceptos y atributos.
- La definición de una semántica formal para el lenguaje.
- El desarrollo de editores personalizados y motores de inferencia para trabajar con el lenguaje.

OIL unifica tres importantes aspectos provenientes de diferentes comunidades: Un modelado proveniente de la comunidad **frames**, una semántica formal y soporte de un razonamiento eficiente proveniente de la **lógica descriptiva** y una propuesta estandarizada para el intercambio sintáctico de notaciones proveniente de la comunidad **Web** (Tabla 3.4.).

```
<rdfs:Class rdf:ID="herbivore">
   <rdf:type rdf:resource="http://www.ontoknowledge.org/oil/RDFSschema/#DefinedClass"/>
   <rdfs:subClassOf rdf:resource="#animal"/>
   <rdfs:subClassOf>
       <oil:NOT>
           <oil:hasOperand rdf:resource="#carnivore"/>
       </oil:NOT>
   </rdfs:subClassOf>
</rdfs:Class>
```

Tabla 3.4. Ejemplo de expresión OIL: Animal Herbívoro como subclase de animal y disjunta de carnívoros.

OIL está organizado en una serie de capas incrementales de sub-lenguajes. Cada capa adicional añade funcionalidad y complejidad a la capa anterior (Figura 3.4.) (Tabla 3.5.).

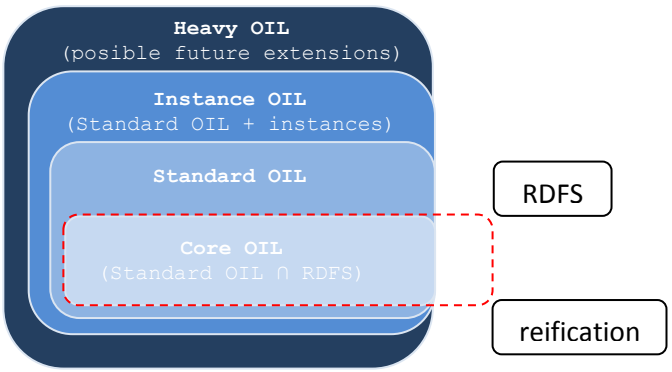


Figura 3.4. Modelo de capas del lenguaje OIL.

Capa	Descripción
Core OIL	Coincide practicamente con RDF Schema. Esto significa que incluso un simple agente de RDF Schema es capaz de procesar las ontologías OIL.
Standard OIL	Es un lenguaje destinado a capturar las principales primitivas de modelado, haciendo viable la extracción de inferencias precisas y completas.
Instance OIL	Añade una integración individual completa.
Heavy OIL	Añade más capacidades de representación y razonamiento. Especialmente un lenguaje de reglas más expresivo y facilidades de definición de metaclases. Esta extensión de OIL fue definida en cooperación con DAML en una iniciativa de un lenguaje de reglas para la web.

Tabla 3.5. Modelo de Capas del lenguaje OIL.

La arquitectura en capas de OIL tuvo tres ventajas fundamentales [32]:

- Una aplicación no está forzada a trabajar con un lenguaje que ofrece más significado, expresividad y complejidad del que realmente necesita.
- Una aplicación que sea capaz de procesar un nivel más bajo de complejidad es capaz de entender algunos aspectos de una ontología.
- Una aplicación que necesite un alto nivel de complejidad puede seguir entendiendo ontologías expresadas en un lenguaje más simple.

DARPA Agent Markup Language (DAML)

La agencia de proyectos de investigación avanzada en defensa (Defense Advanced Research Projects Agency) DARPA, conjuntamente con el consorcio W3C extendieron ***RDF*** con construcciones más expresivas, buscando facilitar la interacción de agentes en la Web [33].

DAML define una semántica de especificaciones muy débil, causando problemas en la interpretación por parte de los agentes software así como la humana.

DAML+OIL

Nace como la combinación de ambas funcionalidades del lenguaje, juntando las características de ***OIL*** y ***DAML***. Unifica las propiedades de las ***Lógicas Descriptivas*** con una sintaxis basada en ***RDFs***. ***DAML+OIL*** es un lenguaje con una mayor expresividad para realizar clasificaciones y definir propiedades de recursos que los que tiene ***RDFS***. En Marzo de 2001, la versión de ***DAML+OIL*** puede soportar la definición de tipos provenientes del lenguaje de definición de ***XML Schema*** (***XSDL***) de W3C [34] (Figura 3.5.).

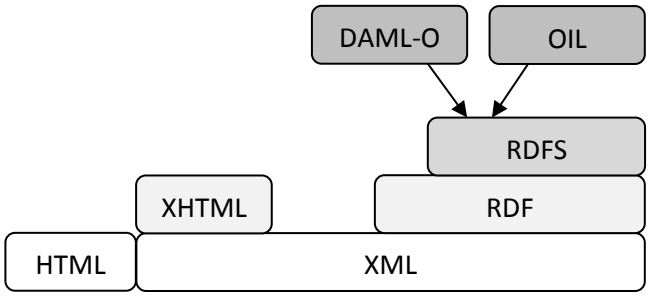


Figura 3.5. Capas de modelo de lenguaje para DAML+OIL.

A partir de esta iniciativa apareció el primer borrador de ***OWL*** (Web Ontology Language), publicado por el W3C y que está basado en DAML+OIL. Desde entonces el desarrollo de DAML parece parado, puesto que los esfuerzos se están centrando en OWL (Ejemplo 3.5.).

```
<daml:DatatypeProperty rdf:ID="productNumber">
  <rdfs:label>Product Number</rdfs:label>
  <rdfs:domain rdf:resource="#Product"/>
  <rdfs:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#nonNegativeInteger"/>
</daml:DatatypeProperty>
```

Ejemplo 3.5. Expresión de DAML.

Aun así, otras partes del proyecto DAML continúan en desarrollo. Entre las más importantes en las que se sigue trabajando podemos citar las siguientes: i) Una biblioteca de ontologías, ii) DAML-S, una ontología para servicios basados en web, iii) DAML-Time, una ontología para conceptos temporales, iv) Herramientas para recolección de ontologías, v) Mantenimiento de manuales, cursos y listas de correo [35].

Web Ontology Language (OWL)

El lenguaje web de ontologías es un lenguaje de marcado semántico para publicar y compartir ontologías en la ***World Wide Web***. Antes de la aparición de ***OWL***, los lenguajes de representación de conocimiento no habían sido diseñados para ser compatibles con la arquitectura ***WWW*** ni con la ***Web Semántica*** (Ejemplo 3.6.).

```
<owl:Class rdf:ID=”Dinero”>
   <rdfs:subClassOf rdf:resource=”http://www.w3.org/2002/07/owl#Thing”/>
</owl:Class>

<owl:Class rdf:ID=”Articulo”>
   <rdfs:subClassOf rdf:resource=http://www.w3.org/2002/07/owl#Thing”/>
</owl:Class>

<owl:ObjectProperty rdf:ID=”costar”>
   <rdfs:domain rdf:resource=”#Articulo”/>
   <rdfs:range rdf:resource=”#Dinero”/>
</owl:ObjectProperty>
```

Ejemplo 3.6. Definición OWL de la propiedad que define el coste de un artículo.

OWL utiliza ***URIs*** para los nombres y el framework de descripción para la Web provisto por ***RDF***, añadiendo las siguientes capacidades a las ontologías:

- Capacidad de ser distribuidas a través de muchos sistemas.
- Escalabilidad de las necesidades Web.

- Compatibilidad con los estándares Web de accesibilidad e internacionalización.
- Libre acceso y extensible.

OWL se construye sobre RDF y RDF Schema y aumenta el vocabulario RDFS para describir propiedades y clases, entre otros: relaciones entre clases, cardinalidad, igualdad, tipos enriquecidos de propiedades, más características de propiedades y clases enumeradas. El Meta modelo de ***OWL*** extiende del conjunto de metamodelos definidos por ***RDFBase***, ***RDFS*** y ***RDFWeb***. ***OWL*** se compone de tres sub-lenguajes diseñados para el uso de comunidades específicas de usuarios y desarrolladores, descritos en la Tabla 3.6. [4]:

Profile	Descripción
OWL Lite	Soporta principalmente una clasificaciones jerárquicas y restricciones simples
OWL DL	Soporta la máxima expresividad sin perder integridad de cómputo y la capacidad de los sistemas de razonamiento
OWL Full	Soporta la máxima expresividad y libertad sintáctica de RDF sin garantías computacionales

Tabla 3.6. Sub-lenguajes de OWL.

3.3. OWL2

A pesar de que OWL es muy conveniente para optimizar el tiempo de procesamiento, tiene algunas carencias a la hora de modelar ontologías o construirlas a partir de un lenguaje natural. En 2009, el consorcio W3C extiende OWL con un conjunto de características que permiten técnicas y semánticas mejor comprendidas y útiles para el desarrollo de nuevas herramientas.

Uno de los objetivos en el desarrollo de OWL2 fue la de mantener la compatibilidad con OWL, manteniendo válida en OWL2 cualquier ontología generada en OWL. OWL2 está basado en OWL-DL-SROIQ (D) [36], por lo que conserva una estructura muy similar a OWL.

OWL 2 añade nueva funcionalidad con respecto OWL1. Algunas de estas nuevas funcionalidades son sintácticas (por ejemplo unión de clases disjuntas) mientras otras ofrecen una nueva expresividad, incluyendo:

- Facilidades sintácticas que hacen más fácil expresar algunas afirmaciones.
- Nuevos constructores que incrementan la expresividad.
- Soporte extendido para los tipos de datos.
- Capacidades de un modelado de metadatos simple.
- Anotaciones extendidas.
- Otras innovaciones y características menores.

OWL 2 también define tres nuevos perfiles y una nueva sintaxis (Manchester Syntax). Adicionalmente, algunas de las restricciones aplicables a OWL DL han sido relajadas, como

resultado, el conjunto de RDF Graphs puede ser manejado por razonadores de lógicas descriptivas es ligeramente mayor en OWL 2.

Perfiles OWL 2 (Profiles)

OWL se compone de tres sub-lenguajes incrementalmente más expresivos diseñados para el uso por comunidades específicas de usuarios (Figura 3.6.).

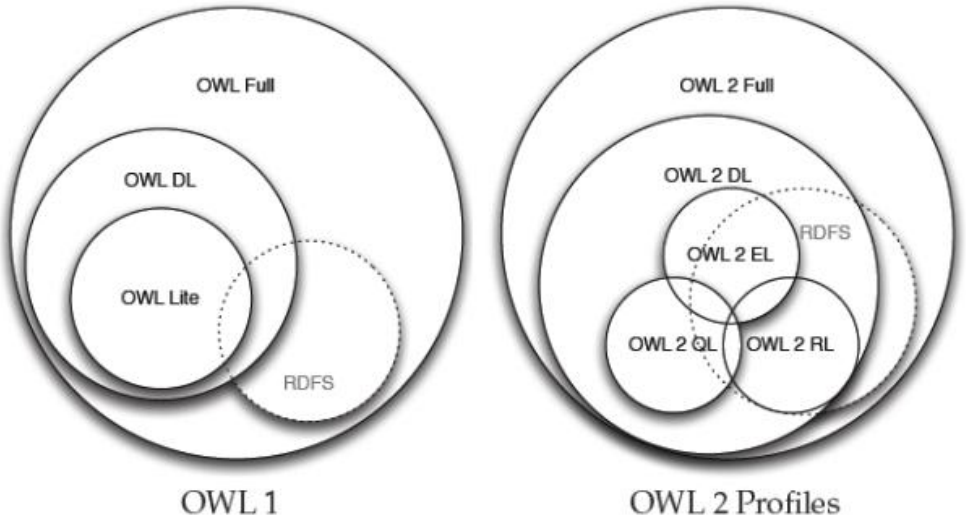


Figura 3.6. Comparativa entre Sub-lenguajes de OWL y OWL2.

OWL2 propone tres nuevos sub-lenguajes que no se incluyen el uno al otro. A su vez, un razonador (o una herramienta) basada en el lenguaje OWL2-Full o OWL2-DL podría ser utilizada para obtener los mismos resultados derivados que estos nuevos sub-lenguajes (con las diferencias evidentes en rendimiento, al soportar un conjunto más general de casos).

La segunda versión de OWL definió los tres nuevos sub-lenguajes con propiedades útiles de procesamiento [37]:

- **OWL EL**
  Optimiza la experiencia de las aplicaciones al utilizar ontologías de gran tamaño. Es capaz de realizar razonamientos sobre éstas de forma eficiente, no permitiendo la negación y disyunción de clases, cuantificación universal sobre propiedades ni propiedades inversas. El acrónimo EL refleja el origen del sub-lenguaje en la familia EL de lógica descriptiva.

- **OWL QL**
  Optimiza la interoperabilidad de las ontologías OWL con bases de datos. Un ejemplo son los Tesauros, que permiten realizar búsquedas y razonamientos sobre estructuras de bases de datos. Este sub-lenguaje es adecuado para aplicaciones con ontologías de tamaño pequeño pero con un gran número de individuos, las cuales necesitan acceder a los datos directamente vía consultas relacionales. El acrónimo QL refleja el hecho de que la respuesta a una consulta puede ser implementada por re-escribir las consultas en el Standard Relational Query Language.

- **OWL 2 RL**
  Optimiza la interoperabilidad de las ontologías OWL con máquinas de reglas. Permite definir un subconjunto sintáctico de reglas para su implementación a través de tecnologías basadas en reglas. Mediante implementaciones de reglas se puede operar directamente sobre triplas RDF y ser aplicadas arbitrariamente a grafos RDF. El acrónimo RL refleja el hecho de que el razonamiento puede ser implementado utilizando el Standard Rule Language.

Las características como la expresividad requerida por la aplicación, la prioridad dada al razonamiento sobre clases o datos, el tamaño del conjunto de datos y la importancia de la escalabilidad, entre otras, marcan la elección de un sub-lenguaje sobre otro.

Sintaxis

La principal sintaxis de intercambio para OWL2 es RDF/XML. Esta sintaxis es de hecho, la única sintaxis que debe ser soportada por todas las herramientas OWL 2 (Figura 3.7.).

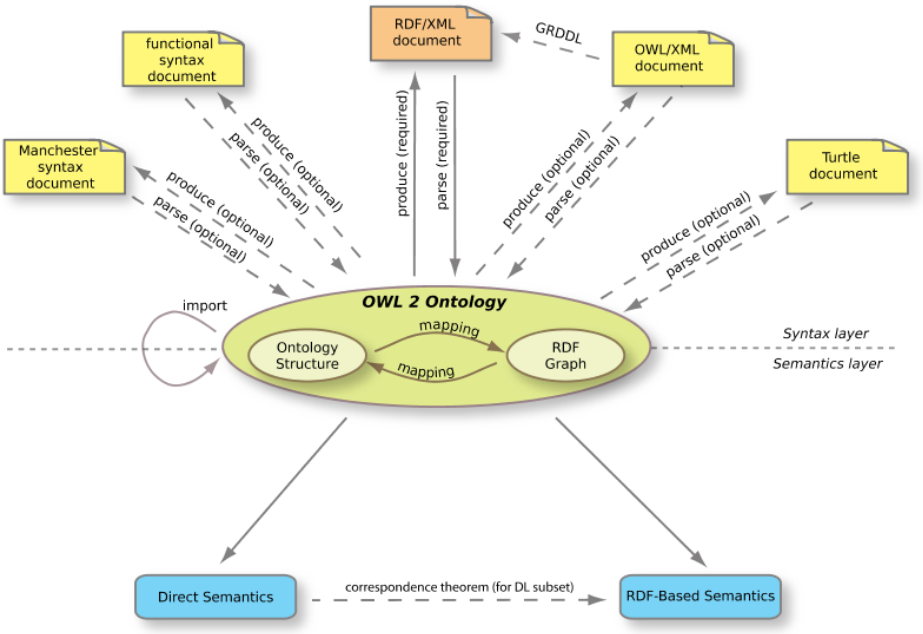


Figura 3.7. Sintaxis de OWL2 [38].

Mientras RDF/XML provee interoperabilidad para la mayoría de herramientas OWL 2, otras sintaxis concretas pueden ser también usadas. Éstas incluyen serializaciones alternativas de RDF, como son Turtle, una serialización XML y una sintaxis más legible llamada Sintaxis de Manchester que es usada en muchas herramientas de edición. Finalmente, la sintaxis de estilo funcional puede también ser usada para serialización, y cuyo principal propósito es especificar la estructura del lenguaje [38] (Tabla 3.7.).

Sintaxis	Especificación	Estado	Propósito
RDF/XML	Mapeo a Graphs RDF, RDF/XML	Obligatorio	Intercambio (puede ser escrito y leído por todas las soluciones que implementen OWL2)
OWL/XML	Serialización XML	Opcional	Fácil para procesar con herramientas de XML
Functional Syntax	Especificación Estructural	Opcional	Fácil para ver la estructura formal de las ontologías

Manchester Syntax	Sintaxis Manchester	Opcional	Fácil de leer/escribir Ontologías DL
Turtle	Mapeo a Graph RDF, Turtle	Opcional, No desde OWL-WG	Fácil de leer/escribir triples RDF

Tabla 3.7 Sintaxis de OWL.

3.4. Componentes de una ontología OWL

El diseño original de OWL era el de tener una compatibilidad completa con RDF, reutilizando así los agentes software existentes. Sin embargo, esta completa compatibilidad sólo se consigue en el perfil OWL Full, a costa de rendimiento computacional. Conceptos como Clases y Propiedades mantienen el significado inicial definido en RDF añadiéndole una mayor expresividad y capacidad de cómputo usado en un razonador [25] (Figura 3.8.).

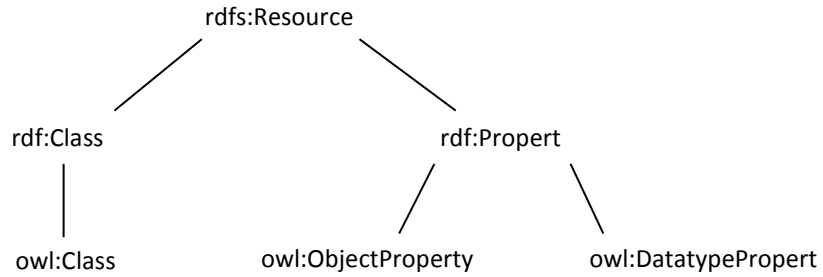


Figura 3.8. Relaciones de subclases entre OWL y RDF/RDFS.

A continuación, se definirán diferentes conceptos presentes en OWL, como son Clases, Propiedades, Restricciones, Individuos, Enumerados y Anotaciones.

3.4.1. Clases

Las clases representan conceptos en el dominio y no los términos que denotan esos conceptos (Figura 3.9.).

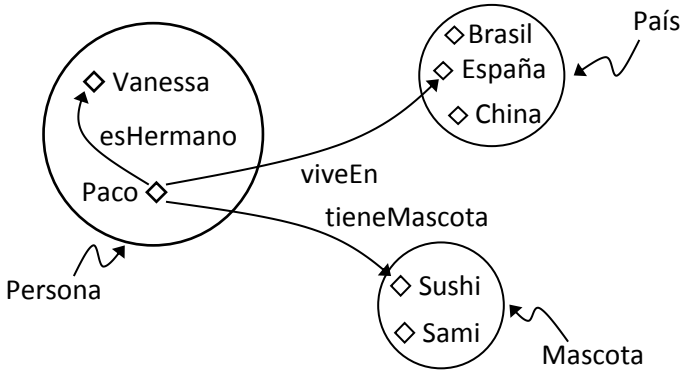


Figura 3.9. Representación de las clases de Persona, País y Mascota y sus relaciones.

Características de las clases

- ***Clases Disjuntas***
  Una Clase A es disjunta de otra B cuando ningún elemento de la Clase A sea elemento de la clase B y viceversa (Figura 3.10.).

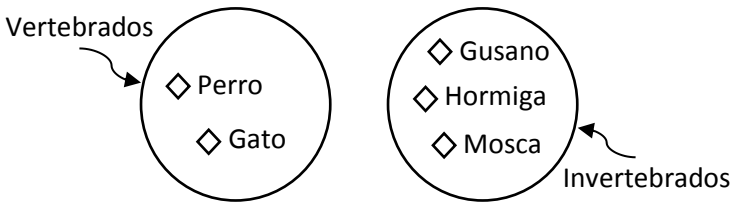


Figura 3.10. Clases disjuntas de seres vivos vertebrados e invertebrados.

- ***Jerarquías de clase***
  Una subclase de una clase representa un concepto que es un "tipo de" concepto que la superclase representa. Si B es una subclase de A y C es una subclase de B, entonces C es una subclase de A.

  Si una clase tiene solamente una subclase directa, puede existir un problema de modelamiento o sino la ontología no está completa. Si hay más de una docena de subclases para una clase dada, entonces categorías intermedias adicionales pueden ser necesarias.

- ***Condiciones Necesaria y suficientes(Clases Primitivas y Clases Definidas)***
  Condiciones necesarias pueden ser descritas como "si algo es miembro de esta clase entonces debe cumplir necesariamente estas condiciones". Con sólo condiciones necesarias, no podemos decir que "si algo cumple estas condiciones, entonces tiene que ser miembro de esta clase".

  Una clase que sólo tiene condiciones necesarias es conocida como una Clase Primitiva. Una clase que tiene al menos un conjunto de condiciones necesarias y suficientes es conocida como Clase Definida.

3.4.2. Propiedades

Las propiedades pueden utilizarse para establecer relaciones entre individuos (propiedades de objetos) o de individuos a valores de datos (propiedades de datos) (Figura 3.11.).

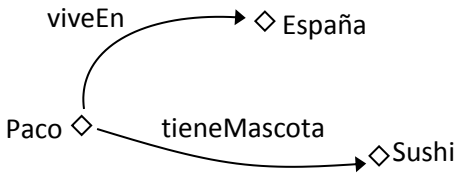


Figura 3.11. Ejemplo de propiedades de una persona.

Toda propiedad a su vez, se compone de un dominio y un rango. El dominio es el conjunto de individuos a los que se pueden aplicar esta propiedad, mientras que el rango son el conjunto de individuos o valores de datos que puede tomar esa propiedad. Es importante remarcar que el dominio y rango en OWL no deberían verse sólo como restricciones a ser comprobadas, ya que además son usados como axiomas en el razonamiento.

3.4.2.1. Propiedades de Objeto

Las propiedades de objeto son relaciones que se establecen entre individuos de dos clases (Figura 3.12.).

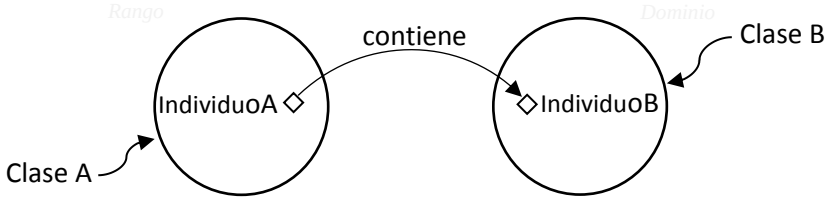


Figura 3.12. Ejemplo de Propiedad de Objeto.

Características de las propiedades

OWL permite enriquecer el significado de las propiedades mediante el uso de las características de la propiedad. Definamos las diferentes características que las propiedades pueden tener:

- ***Propiedad funcional***
  Si una propiedad es funcional, para un individuo dado, puede haber más de un individuo que esté relacionado con este individuo mediante esta propiedad (Figura 3.13.).

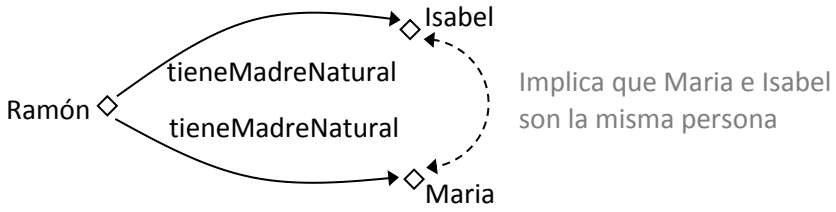


Figura 3.13. Ejemplo de propiedad funcional.

- ***Propiedad Inversa***
  Si una propiedad es funcional inversa, significa que la propiedad inversa es funcional. Dado un individuo, puede haber más de un individuo relacionado con este individuo a través de la propiedad (Figura 3.14.).

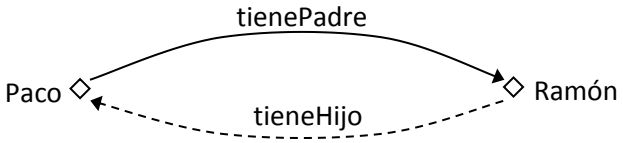


Figura 3.14. Ejemplo de propiedad inversa.

- ***Propiedad Transitiva***
  Si una propiedad p es transitiva, y la propiedad relaciona un individuo A con otro individuo B y a su vez este individuo B con otro individuo C, entonces se puede inferir que el individuo A está relacionado con el individuo C a través de la propiedad P (Figura 3.15.).

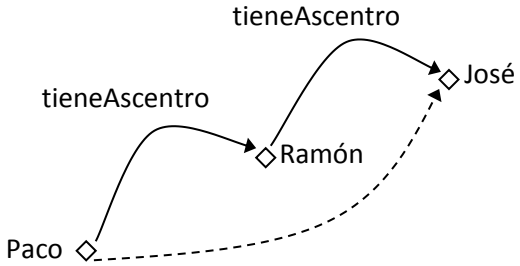


Figura 3.15. Ejemplo de la propiedad transitiva.

- ***Propiedad Simétrica***
  Si una propiedad P es simétrica y la propiedad relaciona un individuo A con un individuo B, entonces el individuo B estará relacionado con el individuo A a través de la propiedad P (Figura 3.16.).

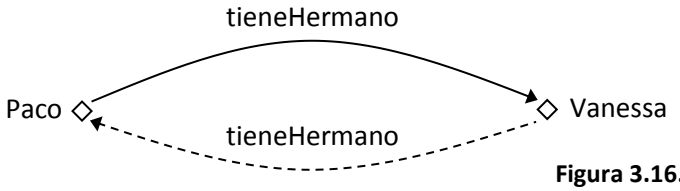


Figura 3.16. Propiedad simétrica.

- ***Propiedad Asimétrica***
  Si una propiedad P es asimétrica y la propiedad relaciona un individuo A con un individuo B entonces el individuo B no podría estar relacionado con el individuo A mediante la propiedad P (Figura 3.17.).

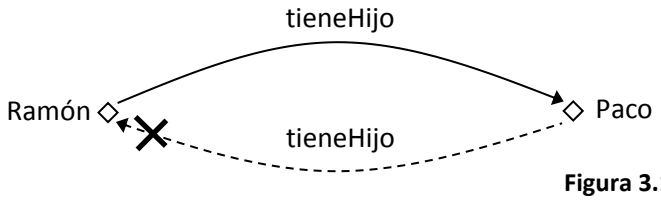


Figura 3.17. Propiedad asimétrica.

- ***Propiedad reflexiva***
  Una propiedad P es reflexiva cuando la propiedad puede relacionar un individuo A consigo mismo (Figura 3.18.).

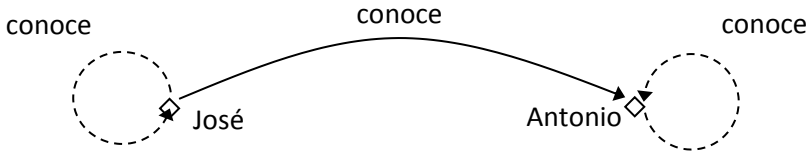


Figura 3.18. Propiedad reflexiva.

- ***Propiedad Irreflexiva***
  Si una propiedad P es irreflexiva, significa que relaciona un individuo A con un individuo B, donde el individuo A y el individuo B no pueden ser el mismo (Figura 3.19.).

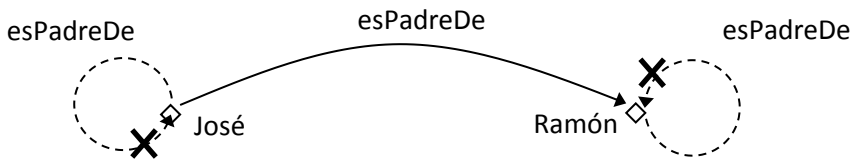


Figura 3.19. Ejemplo propiedad irreflexiva.

3.4.2.2. Propiedades de tipos de datos

Las propiedades de tipos de datos relacionan un individuo con un valor tipo de dato de XML o un literal RDF. En otras palabras, estas propiedades describen relaciones entre individuos y valores de datos (Figura 3.20.).

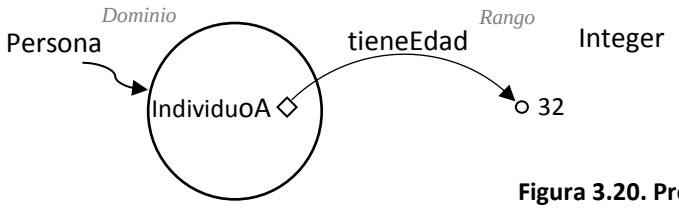


Figura 3.20. Propiedad de tipos de datos.

La clase Persona tiene una propiedad 'tieneEdad' que nos devuelve un número que expresará la edad en la unidad de tiempo definida (años, meses, días…etc.).

La mayoría de las características descritas anteriormente no pueden ser usadas con las propiedades de tipos de datos.

3.4.3. Restricciones

Una restricción define una clase de individuos basados en las relaciones entre miembros de una clase. En otras palabras, una restricción es un tipo de clase, de la misma manera que una clase con nombre es un tipo de clase.

Una restricción describe una clase anónima (clase sin nombre). La clase anónima contiene todos los individuos que satisfagan la restricción. Por ejemplo todos los individuos que tengan las relaciones requeridas para ser miembros de una clase.

OWL permite definir todas las clases de individuos usando restricciones. Las restricciones OWL se pueden englobar en tres categorías:

- Restricciones de Cuantificadores.
- Restricciones Cardinales.
- Restricciones hasValue.

Restricciones de Cuantificadores (Existencial y Universal)
Las restricciones de cuantificadores, pueden ser categorizadas en restricciones existenciales y restricciones universales.

La restricción Existencial define clases de individuos que participan al menos en una relación de una específica propiedad con individuos que son miembros de una clase específica. Las restricciones existenciales son con diferencia las restricciones más comunes en las ontologías

OWL. Las restricciones existenciales también son conocidas como restricciones **Some** y representadas con el símbolo ∃ (Figura 3.21.).

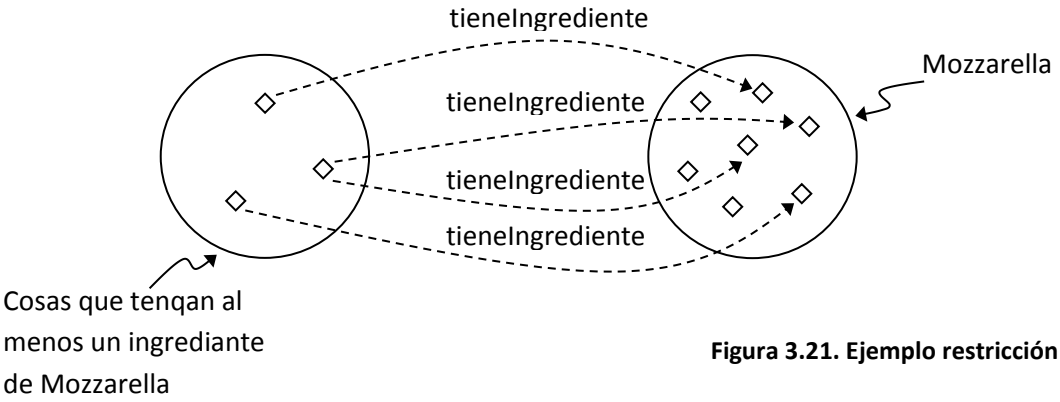


Figura 3.21. Ejemplo restricción Some.

Las restricciones Universales definen clases de individuos que dada una determinada propiedad sólo tiene relaciones mediante esta propiedad con miembros de una determinada clase. Las restricciones universales son dadas por el símbolo ∀. Las restricciones universales también son conocidas como restricciones **AllValuesFrom**.

Para una propiedad dada, las restricciones universales no especifican la existencia de una relación. Estas restricciones limitan que si una relación para una propiedad existe entonces los individuos de esta propiedad deben ser individuos miembros de una clase específica.

Restricciones de cardinalidad

Las restricciones de cardinalidad definen la clase de individuos que tienen al menos, como máximo o exactamente un número específico de relaciones con otros individuos o valores de tipo de datos.

Relaciones (por ejemplo entre dos individuos) son sólo contadas como relaciones separadas si esta puede determinar que los individuos que son rellenos por las relaciones son diferentes entre ellos (Figura 3.22.).

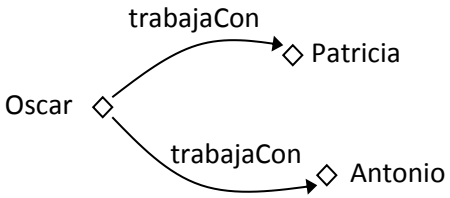


Figura 3.22. Restriccion de cardinalidad (min 2) en la propiedad 'trabajaCon'.

Restricción de cardinalidad cualificada (QCR)

Las restricciones cardinales cualificadas son más específicas que las restricciones cardinales donde el estado de clase de objetos entre las restricciones (Figura 3.23.).

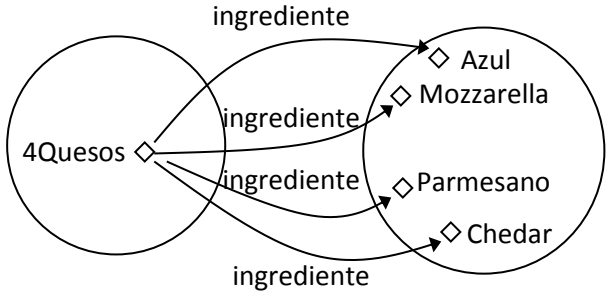


Figura 3.23. Restricción QCR sobre la pizza de 4 quesos.

Restricción hasValue

Una restricción hasValue, denotada por el símbolo ∋, define un conjunto de individuos que tienen al menos una relación con una específica propiedad con un individuo específico (Figura 3.24.).

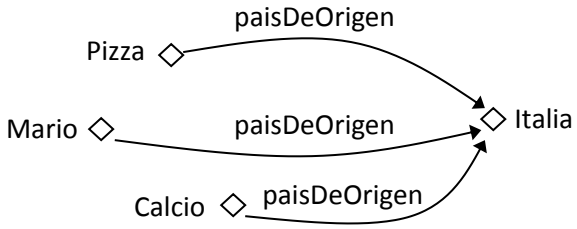


Figura 3.24. Restricción hasValue con la propiedad 'paisDeOrigen' con valor Italia.

3.4.4. Individuos

Individuos son las instancias u objetos de las Clases o dominios definidos. Las instancias individuales son los conceptos más específicos representados en una base de conocimientos (Figura 3.25.).

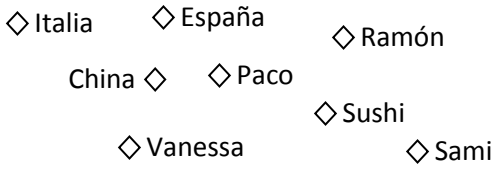


Figura 3.25. Individuos.

Una clase a la cual una instancia individual pertenece no debería cambiar a menudo.

3.4.5. Clases Enumeradas

Las clases enumeradas son clases que contienen una lista fija de individuos miembros de la clase (Figura 3.26.).

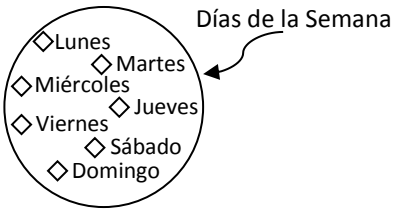


Figura 3.26. Clases Enumeradas.

3.4.6. Anotaciones

Las anotaciones son diferentes elementos de información/metadatos que OWL permite añadir a clases, propiedades, individuos y la propia ontología (técnicamente hablando la cabecera de la ontología). Estos elementos de información pueden tomar el formato de auditoría o información editorial (Tablas 3.8. y 3.9.).

Anotación	Rango	Definición
owl:versionInfo	*string*	*Indica la versión de la ontología, clase, propiedad o individuo*
rdf:label	*string*	*Añade significado y asigna nombres incluso en diferentes lenguajes*
rdfs:comment	*string*	*Añade comentarios a las clases, propiedades o individuos*
rdfs:seeAlso	*URI*	*Identifica recursos relacionados*
rdfs:isDefinedBy	*URI*	*Referencia una ontología que define elementos como clases, propiedades e individuos*

Tabla 3.8. Anotaciones OWL predefinidas para clases, propiedades e individuos.

Anotación	Rango	Definición
owl:priorVersion	*URI*	*Identifica versiones anteriores de la ontología*
owl:backwardsCompatibleWith	*URI*	*Identifica versiones compatibles anteriores*
owl:incompatibleWith	*URI*	*Identifica versiones anteriores no compatibles*

Tabla 3.9. Ejemplo de anotaciones de ontologías.

3.5. Razonador

El desarrollo y aplicación de ontologías depende de razonamiento. Un razonador es un componente clave para trabajar con ontologías OWL DL [39]. Los razonadores también son conocidos erróneamente como clasificadores, sin embargo la clasificación no es el único servicio de inferencia que ofrecen, entre otros:

- Validación de la Taxonomía de clases de la ontología
- Inferir la jerarquía de clases de la ontología
- Validación de consistencia de clases. Una clase es considerada inconsistente si esta no puede tener ninguna instancia.

Una Taxonomía define un conjunto de clases que pueden ser organizadas en una jerarquía de superclases-subclases, donde las subclases especializan las superclases. OWL-DL permite procesar este tipo de relaciones y realizar consultas e inferencias sobre ontologías utilizando un razonador que deduzca este conocimiento implícito.

OWL dota de un mayor contenido y expresividad a las ontologías del que tenía RDF y RDFS, permitiendo la aparición de razonamientos robustos mediante diferentes motores de inferencia. Para que los razonadores sean robustos, la complejidad computacional requerida por los mecanismos de razonamiento y el poder expresivo de los conceptos definidos deben

estar balanceados. A su vez, OWL2 añade a la mejora computacional introducida por OWL una mayor expresividad, basada en la experiencia de usuario vivida con la primera versión.

3.5.1. Algoritmos de razonamiento

Los principales razonadores de lenguajes semánticos se basan en la siguiente serie de algoritmos [40]:

Basados en reglas
Este algoritmo se basa en representar el conocimiento mediante un conjunto de reglas que se utilizan para inferir las acciones o conclusiones del sistema. El algoritmo utiliza un conjunto de reglas de hechos (if-then) para describir el problema, un conjunto de reglas que lo resuelven y de un intérprete que gestione la aplicación de esas reglas sobre la base de hechos.

Primer orden (First-order prover)
Basado en los lenguajes de primer orden (FOL) permite hacer cuantificación sobre los objetos de un dominio y representar propiedades a través de relaciones y funciones.

Tablas semánticas (Tableaux)
Los algoritmos basados en tablas semánticas son un método de demostración por refutación. Este método se basa más en la semántica que en la sintaxis de las fórmulas, aunque se requiere una clasificación de las fórmulas basada en la sintaxis.

Resolución
El método de resolución es una regla de inferencia que toma dos cláusulas y produce una tercera que es consecuencia de ellas. Identifica y borra la cláusula complementaria de esas dos cláusulas y combina las otras literales para formar una cláusula nueva.

Datalog
La evaluación de consultas con Datalog se basa en la lógica de primer orden y fue inicialmente diseñado para bases de datos deductivas. El lenguaje de manipulación es un lenguaje de reglas construido a partir de cláusulas de Horn.

3.5.2. Tipos de razonadores

Los razonadores pueden ser agrupados en dos categorías: razonadores de lógica descriptiva y de programación lógica.

Razonadores de lógica descriptiva
Las lógicas descriptivas permiten representar bases de conocimiento que describen un dominio particular. Una lógica descriptiva permite representar clases (conceptos), individuos, roles (propiedades) y relaciones entre éstos.

Los razonadores DL brindan los siguientes servicios de inferencia [40]:

- Validación de la consistencia de una ontología: el razonador puede comprobar si una ontología no contiene hechos contradictorios.

- Validación del cumplimiento de los conceptos de la ontología: el razonador determina si es posible que una clase tenga instancias. En el caso de que un concepto no sea satisfecho la ontología será inconsistente.

- Clasificación de la ontología: el razonador computa a partir de los axiomas declarados en el TBox, las relaciones de subclase entre todos los conceptos declarados explícitamente a fin de construir la jerarquía de clases.

- Posibilita la resolución de consultas durante la recuperación de información basada en ontologías: a partir de la jerarquía de clases se pueden formular consultas como conocer todas las subclases de un concepto, inferir nuevas subclases de un concepto, las superclases directas, etc.

- Precisiones sobre los conceptos de la jerarquía: el razonador puede inferir cuáles son las clases a las que directamente pertenece y mediante la jerarquía inferida obtener todas las clases a las cuales indirectamente pertenece una clase o individuo dentro de la ontología.

Razonadores de programación lógica

En la programación lógica, se trabaja de una forma descriptiva, estableciendo relaciones entre entidades e indicando no el cómo se deben de hacer las cosas, sino el qué hacer. La programación lógica se basa en un subconjunto de LPO denominada 'Lógica de Horn' [41]. Sin embargo, la semántica está basada en los modelos mínimos de Herbrand.

Una cláusula de Horn es una fórmula clausular en donde aparece, como mucho, un literal afirmado:

$$\neg A(x) \vee \neg D(x) \vee \neg E(x) \vee F(x,y)$$

De esta forma, un programa lógico consiste en sentencias del tipo 'Si A entonces B'.

$$progenitor(?x,?y) \wedge hermano(?x,?z) \rightarrow tío(?z,?y)$$

Los razonadores de programación lógica representan el conocimiento en forma de reglas [42].

3.5.3. Razonadores de OWL2

La Tabla 3. **10**. describe algunos de los razonadores existentes para OWL2 [43].

Nombre	Perfil Nativo	Semánticas	Carencias	Descripción
CB	EL	Directa	Falta de soporte para tipos de datos/valores (próximamente)	Un razonador basado en resolución para OWL EL. Sustituido en 2011 por ELK.
CEL	EL	Directa	Falta de soporte para nominales (ObjectHasValue y ObjectOneOf) y tipos de datos/valores.	CEL es un clasificador polimonial para el perfil OWL2-EL. Escalabilidad demostrada y probada, e indicado para ontologías biomédicas.
ELK Reasoner	EL	Directa	No tiene soporte para razonamiento de claves, tipos de datos y limitado soporte para nominales/datos	ELK es un razonador open-source en Java para OWL-EL que tiene un rendimiento excelente demostrado en varias ontologías científicas.
ELLY	EL, RL	Directa	Soporte del perfil OWL en desarrollo	Implementación centrada en los datos de razonamiento y respuesta a consultas basadas en la traducción EL/RL a datalog de una manera que preserva el contenido de las aserciones

FaCT++	DL	Directa	Completo a excepción de claves y algunos tipos de datos (próximamente)	FaCT++ es un razonador de OWL2-DL open-source basado en tableaux. Implementado en C++ ofrece un rendimiento excepcional en ontologías expresivas.
HermiT	DL	Directa	Completo	Basado en un nuevo algoritmo "hypertableau", HermiT puede determinar si la ontología de entrada es consistente o no, identificar relaciones entre clases, y mucho más.
Jena	RL	Basado RDF / Directa	Completo soporte RL, el soporte de PR1 esta planificado	Implementación experimental de OWL2 RL, basada en la traducción de las premisas de la ontología a un conjunto de reglas de inferencias Jena. Esta siendo desarrollado por HP Labs Bristol y la universidad de Aberdeen.
Owlgres	QL	Directa	Completo	Un razonador open source para OWL2 QL que esta diseñado para usar PostgreSQL
OWLIM	QL, RL	Basado RDF	1.DisjointObjectProperties y DisjointDataProperties usadas para inferir propertyDisjointWith en lugar de la restricción AllDifferent en valores para estas propiedades. 2.No hay inferencia de uniones anónimas para todas las clases. 3.Las clases disjuntas no infieren differentFrom para las instancias. 4. Razonamiento sobre Tipos de datos no esta completamente soportado	OWLIM es un repositorio semántico escalable con un conjunto de reglas personalizable e implementado como un almacén y capa de inferencia del framework Sesame OpenRDF. SwiftOWLIM es capaz de escalar a millones de sentencias en un ordenador de sobremesa. BigOWLIM consige una escala de diez millones de sentencias en un hardware mediano y soporta un amplio rango de funcionalidades como: clustering, búsquedas full-text, ranking y selección.
Oracle Database 11g OWL Reasoner	RL	Basado RDF / Directa	Soporte completo a RL esta planeado.	El motor de inferencia nativo de RDFS/ OWL en Oracle Database 11g soporta la mayoría de reglas RL/RDF. Soporta también las reglas definidas por usuarios
OWLRL	RL	Basado RDF /Directo		Implementación de Prueba de conceptos de Ivan Herman
Pellet	DL, EL	Directa	Completo	Pellet es un razonador open source de OWL2 DL codificado en Java. Provee servicios estándar para las ontologías OWL
Quill	DL, QL	Directa		Quill es un sistema de OWL2 QL de la universidad de Aberdeen que potencia su motor de búsqueda ONTOSEARCH2
QuOnto	QL	Directa		QuOnto es un sistema que es capaz de razonar sobre ontologías de OWL2 QL que contengan un gran número de individuos.
RacerPro	DL	Directa		RacerPro es un razonador y servidor comercial de inferencia OWL.
REL	DL, EL	Directa		REL es un razonador orientado por aproximación que provee un razonamiento TBox para OWL2 DL desarrollado por la Universidad de Aberdeen. El componente básico de REL es la implementación optimizada del algoritmo de CEL.
SHER	DL, EL	Directa	No soporta nominales ni composición de rol. Limitado soporte de tipos de datos. No soporta restricciones de cardinalidad cualificadas.	SHER (Scalable Highly Expressive Reasoner) es una técnología innovadora que prove análisis de ontologías sobre una gran y expressiva base de conocimiento de OWL2
snorocket	EL	Directa		Una optimizada implementación de OWL EL basada en el algoritmo CELL desarrollado por CSIRO

Tabla 3. 10. Razonadores existentes para OWL2.

3.6. Resumen

Como se ha expuesto a lo largo del capítulo, las ontologías son utilizadas para expresar relaciones semánticas entre términos de un dominio. Con ellas conseguimos extraer la lógica semántica de una aplicación, permitiendo un fácil mantenimiento, una rápida adaptación a

nuevos escenarios, análisis del contenido de un dominio, reutilización de las ontologías y la comunicación entre diferentes dominios.

La manera de representar y expresar una ontología ha ido evolucionando a lo largo de la historia, permitiendo cada vez más añadir mejores características de computacionales y expresividad semántica. En este capítulo se presenta OWL como el lenguaje de ontologías más extendido y recomendado por el consorcio W3C para el desarrollo de una ontología. OWL añade a las ontologías la propiedad de ser expresadas y utilizadas a través del mundo Web, utilizando referencias URIs e IRIs en las definiciones de sus componentes. OWL es una evolución lógica de otros lenguajes en los que está basado como son DAML+OIL, RDF, RDFs, programación descriptiva, frames o la programación lógica.

La segunda versión de OWL, permite aumentar las capacidades expresivas del lenguaje con nuevas expresiones, así como una mayor capacidad computacional en los razonadores debido a la definición de nuevos perfiles que optimizan estos procesos. OWL2 puede ser expresado mediante cinco sintaxis diferentes, como son RDF/XML, OWL/XML, funcional, Manchester y Turtle. Así pues, OWL2 ofrece una gran capacidad de razonamiento, y expresividad de términos y conocimiento semántico en relaciones entre términos de un dominio. Como vimos en el capítulo 2, uno de los grandes problemas que tienen las herramientas que trabajan con bases de datos, es la extracción de metadatos y la comunicación e interactuación entre diferentes fabricantes para la explotación de datos. OWL2 nos permite definir un modelo de metadatos común expresado en CWM mediante relaciones semánticas, expresando todo el conocimiento de conceptos, relaciones y casos particulares mediante ontologías definidas con OWL2.

CWM define el modelo de metadatos de una base de datos, y las relaciones entre las diferentes clases y restricciones. Para separar este conocimiento semántico del modelo de metadatos del conocimiento operacional, podemos usar las ontologías como una capa de razonamiento y análisis de los conceptos definidos en CWM. La utilización de ontologías permite expresar restricciones como definíamos en el capítulo 2 de una manera sencilla, fácilmente entendibles por herramientas automáticas a la hora de extracción de los metadatos de una base de datos, permitiendo la interactuación de diferentes dominios y fabricantes de bases de datos.

El siguiente capítulo define la creación de ontologías OWL para ser utilizadas en la descripción del modelo de metadatos relacional de una base de datos establecido en CWM. Se describen las herramientas que nos pueden ayudar en la construcción de las ontologías como son editores, una api de programación de OWL2 y un ejemplo de desarrollo de una aplicación capaz de extraer los metadatos de diferentes fabricantes de bases de datos, utilizando los conceptos de CWM expresándolos mediantes ontologías de OWL2.

Capítulo 4.

Creación Ontologías OWL2 sobre el modelo relacional CWM

Contenido

4.1. Objetivos

El objetivo principal de este capítulo es construir y aplicar una ontología OWL que defina y dote de significado semántico al paquete relacional expresado en CWM 1.1.

Este capítulo también cubre los siguientes objetivos secundarios:

- Expresar ontologías mediante Web Ontology Language 2 (OWL2).
- Estudiar el uso de herramientas de edición de ontologías (Protégé 4).
- Utilizar la librería de java OWL API desarrollada para gestionar las entidades, expresiones y axiomas definidos por las ontologías [6].
- Desarrollar una aplicación capaz de expresar objetos CWM 1.1 como individuos de una ontología OWL.

4.2. Introducción

En capítulos anteriores se ha presentado el modelo de metadatos propuesto por CWM para la representación de los metadatos de bases de datos relacionales y cómo las ontologías podrían cubrir las carencias de significado semántico de estos metadatos en cada modelo.

Este capítulo se muestra la materialización de una ontología OWL basada en el modelo CWM y su aplicación en una solución web como caso práctico.

Figura 4.1. Elementos del caso práctico.

La **Figura 4.1**. introduce los principales elementos de este caso práctico, donde la aplicación será capaz de proveer los metadatos de diferentes fabricantes de bases de datos y expresarlos en objetos CWM. Además también proporciona la transformación automática de estos objetos relacionales en individuos OWL de la ontología CWM creada. Como ejemplo final, se muestra la aplicación de una ontología de individuos OWL creada a partir de la ontología CWM del proyecto, un esquema relacional que almacena una lista de vinos y la ontología de vinos y comida `wine.owl`.

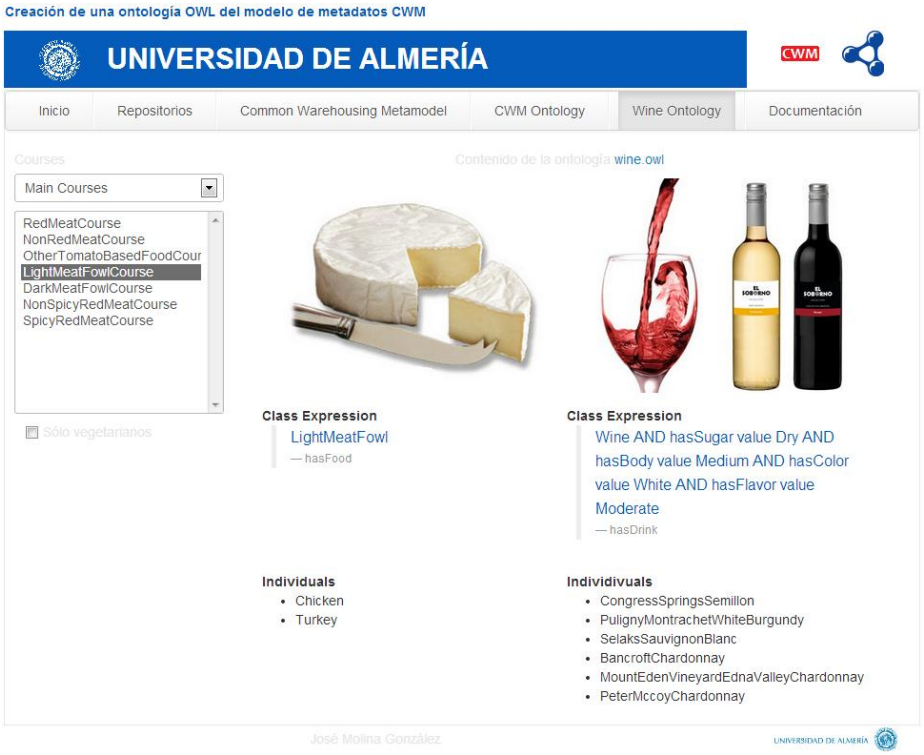


Figura 4.2 Página 'Wine Ontology' del caso práctico.

En este último ejemplo se pretende mostrar un uso cotidiano de búsquedas semánticas sobre las ontologías OWL que han sido creadas automáticamente gracias al uso combinado de CWM y OWL. La **Figura 4.2**. visualiza el cliente web construido a tal fin.

4.3. Creación de una ontología basada en el modelo de metadatos del paquete relacional

El objetivo de representar el modelo de metadatos definido en CWM usando ontologías es el de dotar de un mayor significado semántico en la construcción de las expresiones que los definen. El uso de ontologías para definir estos modelos nos permite la creación de sistemas más complejos capaces de interpretar de una forma automatizada los metadatos de diferentes bases de datos con el significado semántico que poseen en cada dominio.

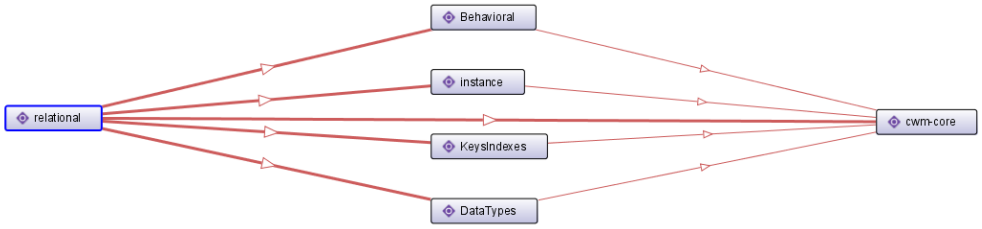


Figura 4. 3. Ontologías del proyecto que definen el paquete relacional de CWM.

La Figura 4. **3** describe la organización de ontologías que se ha seguido a la hora de la creación de los modelos, identificando cada paquete de modelos de los que dependía el paquete relacional en CWM, como una ontología OWL.

4.3.1. Herramientas de edición de ontologías

La definición y edición de las ontologías que forman los modelos de metadatos expresados en este proyecto referentes al paquete relacional han sido realizadas con la herramienta de edición Protégé v4.0.2. Protégé (**Figura 4.4**.) es una herramienta de código abierto desarrollada en Java más utilizada y da soporte, entre otros, al lenguaje de ontologías OWL, permitiendo a los usuarios [44]:

- Cargar y guardar ontologías OWL y RDF.
- Editar y visualizar clases, propiedades y reglas SWRL.
- Definir las características lógicas de clase como expresiones OWL.
- Ejecutar razonadores como clasificadores de lógica de descripción.
- Editar individuos de OWL.

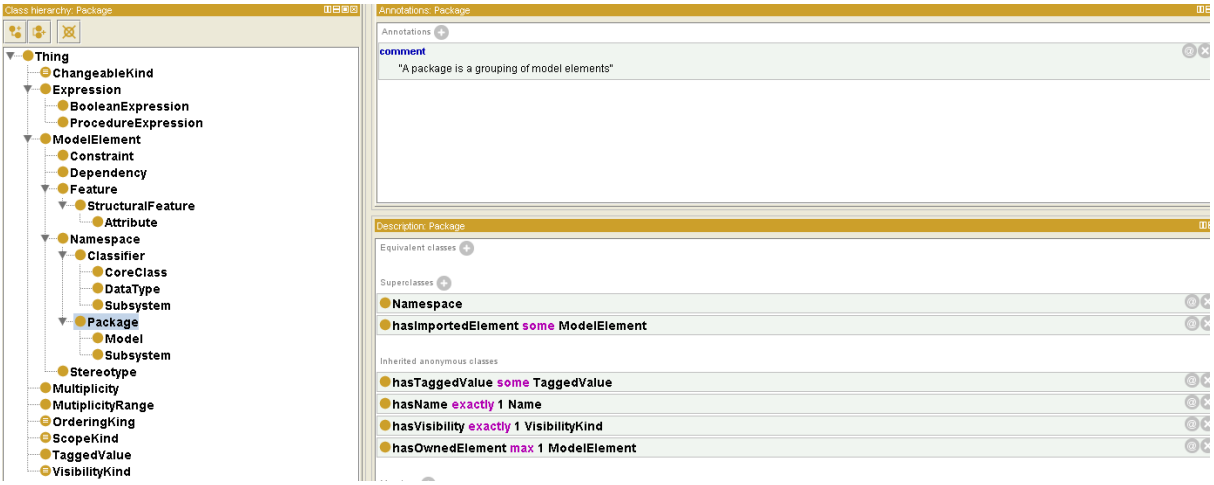


Figura 4.4. Ontología del paquete Core CWM cargada en Protégé.

La amplia funcionalidad, integración y fácil utilización de la herramienta es el motivo de su elección en este proyecto sobre el resto de herramientas relacionadas (Tabla 4.1.).

Herramienta	Descripción
CORESE	Herramienta RDF basada en grafos conceptuales
DOME	Distributed Ontology Management Environment
KAON	Gestor de ontologías en código abierto. Incluye un conjunto de herramientas para crear y gestionar ontologías.
Mindswap	Editor de ontologías hipermedia basado en OWL
OntoEdit	Herramienta para construir ontologías usando significaciones gráficas.
OntoLingua	Provee un entorno colaborativo, un buscador, generador y editor de ontologías.
Ontomat	Herramienta de código abierto que soporta marcado OWL.
Ontopia	Conjunto de herramientas de desarrollo y mantenimiento de ontologías
Protégé	Editor de ontologías de código abierto para construir ontologías sobre RDFS, OWL y XML Schema. Es uno de los editores más utilizados.
PROTON Ontology	Plataforma de construcción de ontologías, anotación semántica, indización y la recuperación de información.
SWOOP	Editor de ontologías hipermedia
WebODE	Herramienta que permite el desarrollo de ontologías sobre ingeniería.
WebOnto	Applet de java que permite navegar y editar modelos de conocimiento sobre la Web.
WSMO Studio	Editor de ontologías para el modelado de servicios de la Web Semántica.

Tabla 4.1. Listado de herramientas para la construcción de ontologías [45].

4.3.2. Ontologías del paquete relacional

En la creación de una ontología basada en el modelo de metadatos propuesto por CWM, se ha identificado una ontología como un paquete del modelo, manteniendo las dependencias de paquetes en las ontologías declaradas (**Ejemplo 4.1.**).

```
Ontology: <http://www.cwm.org/resource/relational.owl>

Import: <http://www.cwm.org/foundation/KeysIndexes.owl>
Import: <http://www.cwm.org/objectmodel/Behavioral.owl>
Import: <http://www.cwm.org/objectmodel/instance.owl>
Import: <http://www.cwm.org/foundation/DataTypes.owl>
Import: <http://www.cwm.org/ontologies/cwm-core.owl>
…
```

Ejemplo 4.1. Dependencias de la ontología Relational.owl.

La jerarquía y definición de clases en las ontologías CWM creadas, corresponden a una equivalencia con los metadatos declarados en los paquetes CWM y su jerarquía (**Ejemplo 4.2.**).

```
…
Class: <http://www.cwm.org/resource/relational.owl#Table>

    Annotations:
        rdfs:comment "A materialized NamedColumnSet"

    SubClassOf:
        <http://www.cwm.org/resource/relational.owl#NamedColumnSet>,
…
```

Ejemplo 4.2. Clase Table de la ontología Relational.owl.

Las propiedades de objetos y datos de una clase semántica hacen referencia a las propiedades y atributos de los metadatos de los paquetes CWM. Las propiedades de objetos identifican

relaciones entre metadatos y propiedades complejas, mientras que se ha definido como propiedades de datos los atributos simples de los metadatos CWM.

```
…
Class: <http://www.cwm.org/resource/relational.owl#Column>

    Annotations:
        rdfs:comment "A column in a result set, a view, a table, or an SQLStructuredType."@en

    SubClassOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#Attribute>,
        <http://www.cwm.org/resource/relational.owl#hasPrecision> max 1 xsd:integer,
        <http://www.cwm.org/resource/relational.owl#hasScale> max 1 xsd:integer,
        <http://www.cwm.org/resource/relational.owl#hasLength> max 1 xsd:integer,
        <http://www.cwm.org/resource/relational.owl#hasCollationName> exactly 1 xsd:string,
        <http://www.cwm.org/resource/relational.owl#hasOptionScopeColumnSet> max 1
<http://www.cwm.org/resource/relational.owl#NamedColumnSet>,
        <http://www.cwm.org/resource/relational.owl#hasReferencedTableType> max 1
<http://www.cwm.org/resource/relational.owl#SQLStructuredType>,
        <http://www.cwm.org/resource/relational.owl#isNullable> exactly 1
<http://www.cwm.org/resource/relational.owl#NullableType>
...
```

Ejemplo 4.3. Propiedades de objetos y de datos de la clase Column.

Los valores de las propiedades que pueden ser `null` se expresan mediante el operador de cardinalidad `max 1` y los valores obligatorios con el operador de cardinalidad `exactly 1` (**Ejemplo 4.3.**). Los valores enumerados de los paquetes CWM han sido expresados como individuos equivalentes de una clase (**Ejemplo 4.4.**).

```
…
Individual: <http://www.cwm.org/resource/relational.owl#PT_FUNCTION>
    Types:
        <http://www.cwm.org/resource/relational.owl#ProcedureType>

Individual: <http://www.cwm.org/resource/relational.owl#PT_PROCEDURE>
    Types:
        <http://www.cwm.org/resource/relational.owl#ProcedureType>

Class: <http://www.cwm.org/resource/relational.owl#ProcedureType>

    EquivalentTo:
        {<http://www.cwm.org/resource/relational.owl#PT_FUNCTION> ,
<http://www.cwm.org/resource/relational.owl#PT_PROCEDURE>}
…
```

Ejemplo 4.4. Valores de la Clase Enumerada ProcedureType.

Las propiedades o relaciones con más de un metadato en los paquetes CWM se traducen como propiedades con el operador `some` (**Ejemplo 4.5.**).

```
...
Class: <http://www.cwm.org/resource/relational.owl#NamedColumnSet>

    SubClassOf:
        <http://www.cwm.org/resource/relational.owl#SQLStructuredType>,
        <http://www.cwm.org/resource/relational.owl#hasOptionScopeColumn> some
<http://www.cwm.org/resource/relational.owl#Column>,
...
```

Ejemplo 4.5. Propiedades some de la clase NamedColumnSet.

4.4. Caso Práctico

Para el caso práctico de este proyecto, se plantea una solución web que haga uso de la ontología creada a partir del paquete relacional CWM. El hecho de poder expresar los metadatos de un esquema de una base de datos relacional como una ontología permite aplicar semántica a la recuperación de los datos y metadatos y separar la lógica de negocio de los procesos de la aplicación.

Como ejemplo de un uso básico, en este capítulo se propone extender las ontologías `wine.owl` y `food.owl` que nos proporcionan información sobre cómo maridar platos de comida y vino, añadiéndoles individuos de la clase `Wine` provenientes de ontologías de esquemas de bases de datos de diferentes fabricantes.

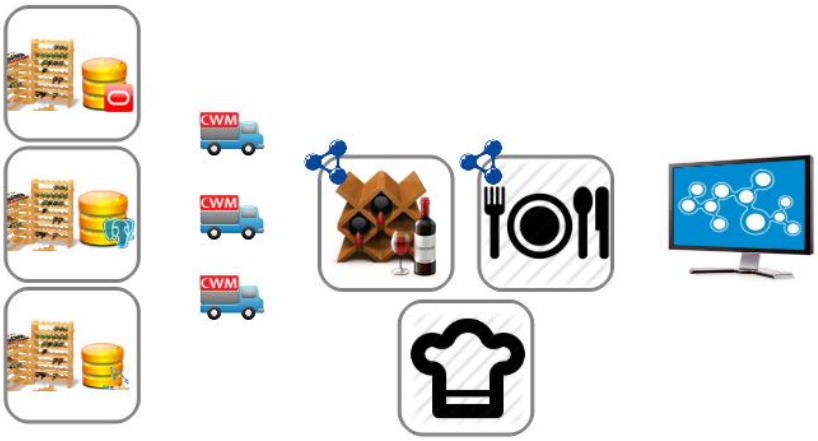


Figura 4.5. Restaurante con 3 distribuidores de vinos distintos y página web de maridaje plato-vino.

En la **Figura 4.5**. vemos el caso extremo de un restaurante quiere publicar en la Web los platos del menú y recomendar los vinos que mejor los acompañan. El restaurante trabaja con diferentes distribuidores de vinos que utilizan distintos sistemas de almacenamiento y publicación de vinos y sus características. La aplicación debería ser también capaz de añadir nuevos distribuidores de vinos en el futuro, por lo que el uso de CWM ayudaría a la aplicación a conectar fácilmente con los posibles nuevos modelos de datos de futuros proveedores.

A parte de tener la mejor elección de vinos según el plato seleccionado, la solución web planteada en el proyecto como caso práctico ofrece la siguiente funcionalidad:

- Gestionar instancias de bases de datos relacionales que la aplicación tiene registrada.
- Explorar y navegar a través de los metadatos de una base de datos relacional mediante el modelo de metadatos CWM.
- Transformar metadatos del modelo CWM en individuos de la ontología CWM.
- Explorar y navegar sobre los individuos de ontologías generadas de los metadatos relacionales CWM.
- Realizar búsquedas semánticas sobre las ontologías generadas de metadatos CWM.

Expresando los nuevos metadatos como ontologías CWM, podemos aplicar reglas específicas al dominio de estos datos, añadiendo valor semántico en la exploración de los datos y metadatos. Así pues, realizar búsquedas entre los individuos de ontologías que cumplan las reglas semánticas sobre el dominio de la aplicación será más sencillo y dinámico, permitiendo rápidamente adaptarse a futuros cambios en las estructuras de los modelos de datos utilizados.

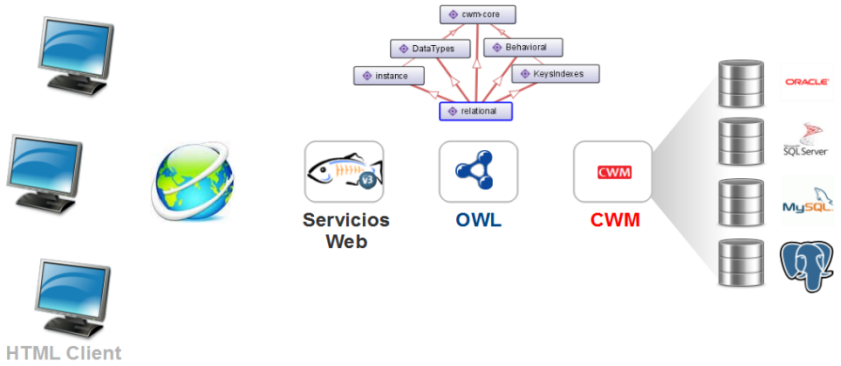


Figura 4.6. Tecnologías y librerías utilizadas en el caso práctico.

La **Figura 4.6.** representa las tecnologías y librerías necesarias para completar la funcionalidad de este proyecto. Los principales pasos seguidos son:

- Construcción de una ontología CWM que define todos los metadatos de los paquetes relacionales modelados en CWM

- Desarrollo de una librería java CWM 1.1. que permite a la aplicación la conexión y extracción de metadatos CWM de bases de datos relacionales (Postgresql, MySQL y Oracle).

- Desarrollo de una librería java OWL 2.0 que extiende la funcionalidad de la librería java `owlapi-2.0` para gestionar y transformar objetos de metadatos definidos en la librería CWM 1.1 en individuos de ontologías extendidas de la ontología relacional CWM.

- Implementación de una plataforma web que permita la visualización y manipulación remota de los servicios ofrecidos por las librerías implementadas en el caso práctico.

4.4.1. Herramientas utilizadas para el desarrollo del caso práctico

En el desarrollo del caso práctico se ha hecho uso de las siguientes herramientas:

Herramienta	Descripción
OWL2 API	API de desarrollo en Java para la manipulación de ontologías OWL2
HermiT	Razonador semántico
Netbeans	Entorno de desarrollo Java
Maven	Proyecto
Glassfish	Servidor de aplicaciones
HTML5	Cliente Web
Bootstrap twitter	Cliente Web
DataTable	Cliente Web

Tabla 4.2. Herramientas utilizadas en el desarrollo del caso práctico.

En los siguientes apartados se exponen cada una de las herramientas utilizadas en detalle.

4.4.1.2 OWL Api

OWL API es una librería de desarrollo Java para la creación, manipulación y serialización de ontologías OWL. OWL API es una librería de código libre que da soporte de lectura y escritura a las sintaxis de Ontologías OWL2, RDF/XML, OWL/XML, OWL Funcional, Turtle, lectura de KRSS y OBO en fichero de formato plano e incluye interfaces para trabajar con razonadores como FaCT++, HermiT, Pellet y Racer [46].

4.4.1.3. Hermit

Para las búsquedas semánticas, se ha utilizado el razonador HermiT. HermiT es el primer razonador disponible basado en los cálculos "hypertableau" que proporcionan un razonamiento más eficiente que cualquier otro algoritmo anterior. Hermit permite clasificar ontologías que tardaban minutos u horas en unos pocos segundos [47].

4.4.1.4. Entorno de desarrollo java

En el desarrollo java del caso práctico se han elegido las herramientas mostradas en la Figura 4.7. debido a la rápida integración entre ellas y su alto rendimiento en el desarrollo y despliegue de los proyectos. Así pues, el editor de desarrollo NetBeans puede gestionar proyectos Web de Maven y desplegarlos sobre el servidor de aplicaciones Glassfish de una forma integrada en el propio entorno sin necesidad de ampliar la herramienta.

Figura 4.7. Versiones de las herramientas de desarrollo java utilizadas en el proyecto.

A continuación se explicará más en detalle las características de cada uno de ellos.

NetBeans IDE

NetBeans IDE es un entorno de desarrollo para escribir, compilar, depurar y ejecutar programas con el lenguaje Java. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

Maven

Maven es una herramienta utilizada para crear y gestionar cualquier proyecto basado en Java. Es una manera estándar de construir los proyectos, una definición clara de lo que el proyecto consiste en una forma fácil de publicar la información del proyecto y una manera de gestionar las dependencias entre proyectos de manera independiente de la herramienta de desarrollo utilizada.

Servidor de aplicaciones Glassfish

GlassFish es un servidor de aplicaciones de código abierto compatible con Java EE, listo para funcionar en entornos de producción. GlassFish v3 proporciona una pequeña base con todas las funciones para la implementación de Java EE 6.

4.4.1.5. Herramientas de desarrollo del cliente web

El cliente presentado en este proyecto está construido utilizando el lenguaje de páginas web HTML5 (HyperText Markup Language). A diferencia de otras versiones de HTML, los cambios en HTML5 comienzan añadiendo semántica y accesibilidad implícitas, especificando cada detalle y borrando cualquier ambigüedad. Se tiene en cuenta el dinamismo de muchos sitios webs (Facebook, Twitter… etc.), donde su aspecto y funcionalidad son más semejantes a aplicaciones webs que a documentos [47].

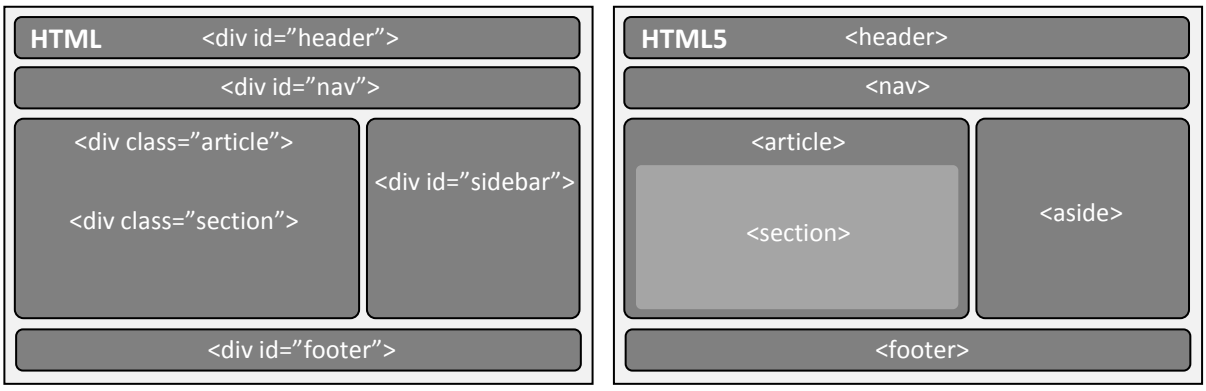


Figura 4.8. Diferencias entre HTML y HTML5.

En la Figura 4.8. se muestra algunas de las diferencias que se introducen en HTML5, eliminando el uso innecesario de las etiquetas div en gran parte del código.

Bootstrap Twitter

El cliente HTML del caso práctico hace uso de los componentes HTML, CSS y simple Javascript definidos en la librería Bootstrap de Twitter. Aparte de conseguir un aspecto más profesional y de aplicaciones más conocidas como son Facebook y Twitter, permite la adaptación e integración del cliente a los diferentes dispositivos de visualización de la página Web, adaptando los componentes de la interfaz a la resolución del dispositivo (Figura 4.9.).

Figura 4.9. Interfaz adaptativa al dispositivo de visualización.

DataTables

DataTables es un plug-in para la biblioteca jQuery Javascript. Es una herramienta muy flexible, basado en los fundamentos de la mejora progresiva, lo que añadirá controles avanzados de interacción para cualquier tabla HTML (Figura 4.10.). [48]

Views

Show 10 entries Search:

Name	Query Expression
test_view	(select 'VIEW' AS `NAME`,`test`.`test_table`.`COLUMN1` AS `COLUMN1`,`test`.`test_table`.`COLUMN2` AS `COLUMN2` from `test`.`test_table`)
test_view2	(select 'VIEW2' AS `NAME`,`test`.`test_table2`.`COLUMN3` AS `COLUMN3`,`test`.`test_table2`.`COLUMN4` AS `COLUMN4` from `test`.`test_table2`)
test_view3	(select 'VIEW3' AS `NAME`,`test`.`test_table3`.`COLUMN5` AS `COLUMN5`,`test`.`test_table3`.`COLUMN6` AS `COLUMN6` from `test`.`test_table3`)

Showing 1 to 3 of 3 entries Previous Next

Figura 4.10. Tabla de vistas relacionales creada mediante DataTable.

4.4.2. Diseño del caso práctico

El diseño del caso práctico de este proyecto ha sido implementado siguiendo el patrón de desarrollo software Modelo Vista Controlador (MVC). MVC se basa en separar la estructura del proyecto en tres capas diferenciadas: los datos y la lógica interna de acceso y gestión, la comunicación entre las peticiones de la interfaz y la respuesta y por último, la interfaz del usuario. La Figura 4.11. muestra esta división siguiendo este patrón entre los módulos desarrollados en el proyecto.

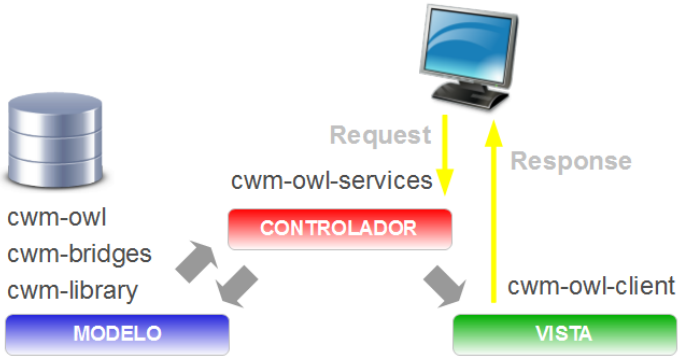


Figura 4.11. Modelo Vista Controlador.

El caso práctico es una aplicación web, donde la vista son páginas creadas con HTML5, la librería de Twitter bootstrap y las tablas de DataTable, el modelo es el Sistema de Gestión de Base de Datos y la lógica interna separada en las librerías de cwm y owl, y el controlador es el responsable de recibir las peticiones y respuestas a los servicios REST ofrecidos en la aplicación web del proyecto `cwm-owl-services` (Tabla 4.3.).

Proyecto Maven	Descripción
cwm-library	Librería del modelo relacional definido en CWM
cwm-bridges	Librería para la gestión de modelos CWM y transformaciones de modelos relacionales.
cwm-owl	Librería de transformación de objetos CWM a instancias de OWL y gestión de ontologías OWL basadas en el modelo de metadatos CWM.
cwm-owl-services	Controlador de Servicios Rest de navegación sobre las librerías de OWL y CWM.
cwm-owl-client	Cliente HTML5

Tabla 4.3. Descripción de los proyectos que componen el caso práctico.

En los siguientes apartados se verá más en profundidad cada elemento (Tabla 4.4.) [49]:

MVC	Descripción
Modelo	Es la representación de la información en el sistema. Trabaja junto a la vista para mostrar la información al usuario y es accedido por el controlador para añadir, eliminar, consultar o actualizar datos.
Vista	Es la representación del modelo en un formato adecuado para que el usuario pueda interactuar con él.
Controlador	Es el elemento más abstracto. Recibe, trata y responde los eventos enviados por el usuario o por la propia aplicación. Interactúa tanto con el modelo como con la vista.

Tabla 4.4. Descripción del patrón Modelo Vista Controlador.

4.4.2.1. Vista

La capa de presentación de la aplicación se encuentra alojada dentro del proyecto java `cwm-owl-client`, que contiene la funcionalidad requerida en el caso práctico repartida en las páginas web de la Tabla 4.5.

Página Web	Descripción
Inicio	Contiene los objetivos e introducción de la solución web. Esta página es meramente informativa.
Repositorios	Contiene la funcionalidad de añadir, eliminar o modificar repositorios declarados en la aplicación. Podemos identificar un repositorio como una instancia de base de datos que queramos usar en la aplicación.
CWM Navegation	Contiene la funcionalidad de navegar y representar los metadatos CWM de bases de datos relacionales de una manera unificada. Además permite la creación automática de estos metadatos cargados en objetos CWM a una ontología OWL.
CWM Ontology	Visualiza los individuos de las ontologías generadas a partir de esquemas relacionales de diferentes fabricantes de base de datos.
Wine Ontology	Permite realizar búsquedas semánticas sobre una ontología de vinos y comida extendida para añadir individuos de vinos cargados desde una ontología generada de CWM.
Documentación	Contiene accesos directos a toda la documentación generada en este proyecto.

Tabla 4.5. Contenido de las páginas web del caso práctico.

Todas las vistas se generan siguiendo un estilo común, incluyendo su funcionalidad dentro del contenido de una página principal que incluye una cabecera (Figura 4.12.) y un pie de página (Figura 4.13.).

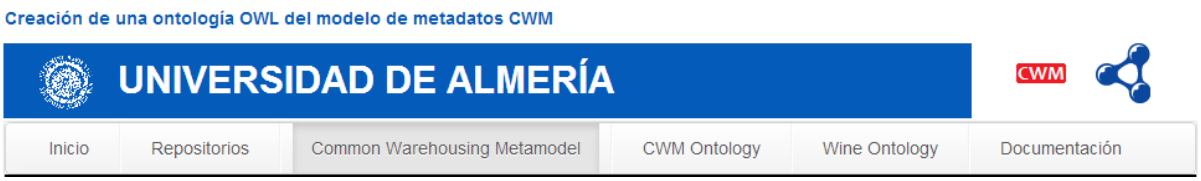


Figura 4.12. Cabecera de la solución web del caso práctico.

Figura 4.13. Pie de página de la solución web del caso práctico.

A continuación veremos en detalle las vistas principales y su funcionalidad.

Repositorios

La **Figura 4.14**. muestra la vista que contiene la funcionalidad de visualizar, registrar y eliminar repositorios en la aplicación. En el dominio de esta aplicación, la entidad repositorio es una instancia de una base de datos relacional que puede ser usada por la aplicación.

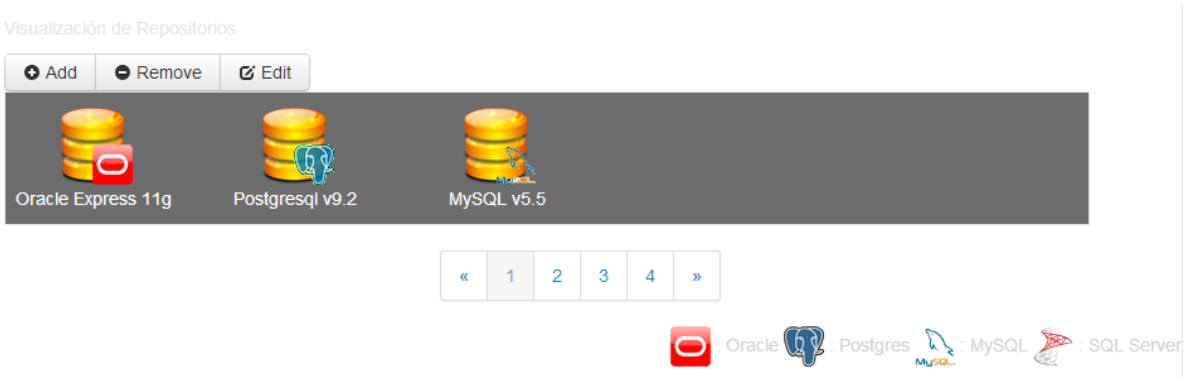


Figura 4.14. Cliente web de la gestión de repositorios.

CWM Navegation

La Figura 4.15. es la vista que permite la representación simplificada de los metadatos CWM del paquete relacional. En la página se pueden listar los cinco principales metadatos de un esquema relacional: Tablas, Vistas, Triggers, Procedimientos y funciones e índices. Cada uno de estos metadatos estará organizado dentro de una tabla paginada.

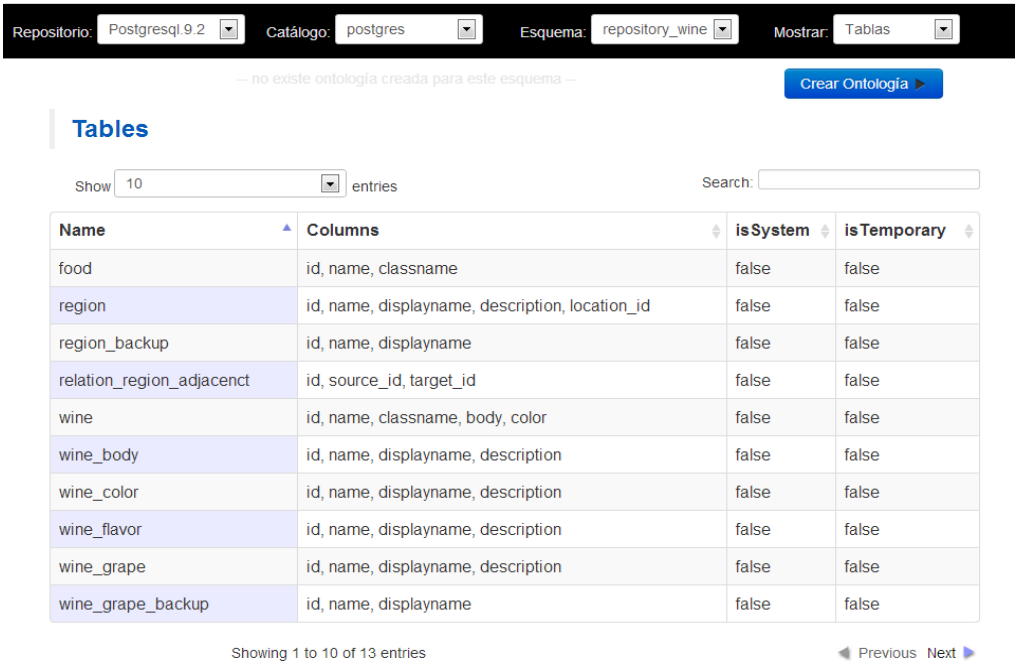


Figura 4.15. Cliente Web de los servicios CWM.

Además de facilitar la navegación a través del modelo de metadatos CWM de bases de datos relacionales de diferentes fabricantes, hace posible la materialización de este modelo de metadatos CWM en individuos de una ontología CWM (Figura 4.16.).

Figura 4.16. Creación de una ontología CWM basada en un esquema.

CWM Ontology

La vista de ontologías CWM permite la visualización de los individuos creados a partir de un esquema, funcionalidad proporcionada por la vista CWM Navegation. En la Figura 4.17. se ve el listado de individuos agrupados por las principales clases de los metadatos: Tablas, Vistas, Procedimientos/Funciones, Índices y Triggers.

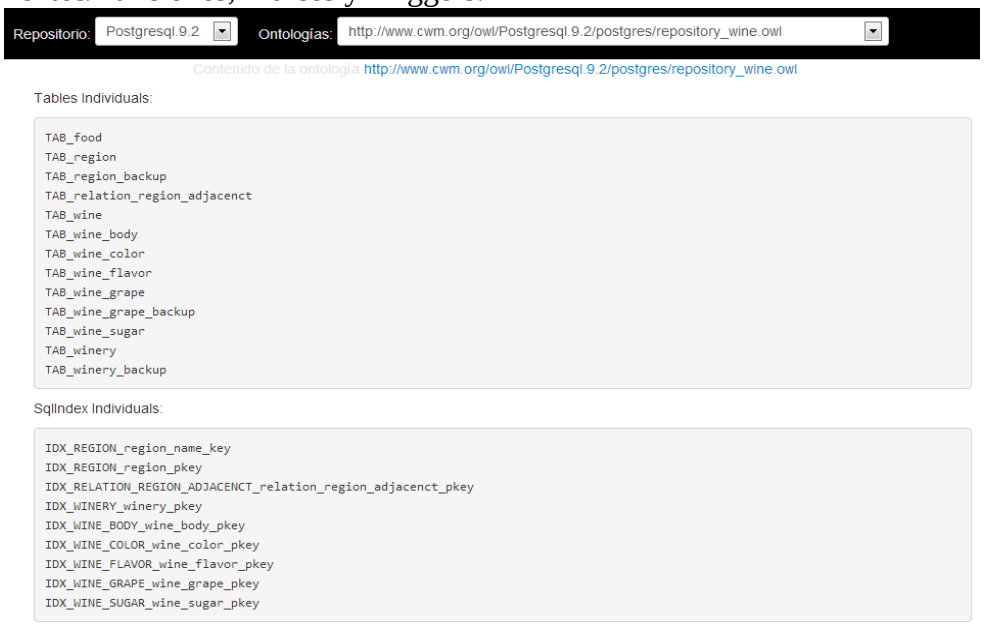


Figura 4.17. Vista simplificada de los individuos de un esquema relacional.

Wine Ontology

En la Figura 4. **18**. se muestra un caso práctico de uso de búsquedas semánticas sobre ontologías generadas extendiendo de la ontología `wine.owl` y `food.owl`. Sobre una lista de platos de comida, se sugieren los mejores vinos, individuos cargados de la ontología `wine.owl` y la generada del esquema postgresql `repository_wine.owl`.

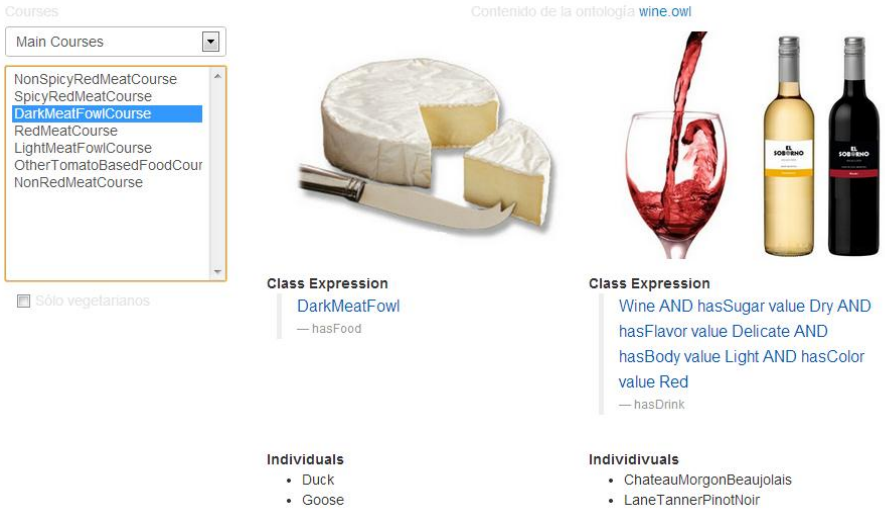


Figura 4. 18. Vista de aplicación de una ontología de vinos y búsquedas semánticas.

En esta vista se puede visualizar las expresiones de búsqueda semántica y los individuos que encajan en el criterio de búsqueda. De este modo, podríamos validar los resultados ofrecidos en el cliente web, mediante el editor de ontologías Protégé como se muestra en la Figura 4. **19.**

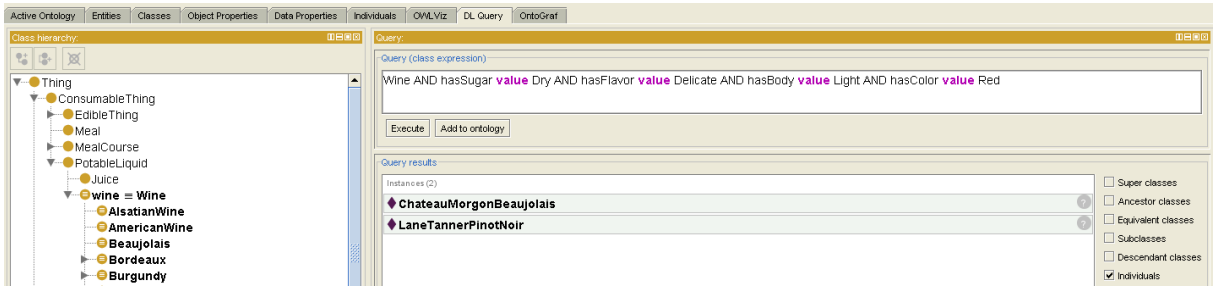


Figura 4. 19. Búsquedas DL en Protégé.

4.4.2.2 Controlador

El proyecto de Maven Web que implementa la capa de controlador del caso práctico es cwm-owl-services e implementa la comunicación entre el Modelo (transformaciones y modelo de CWM y OWL) y la Vista (el cliente Web) mediante Servicios Web (Figura 4.20.).

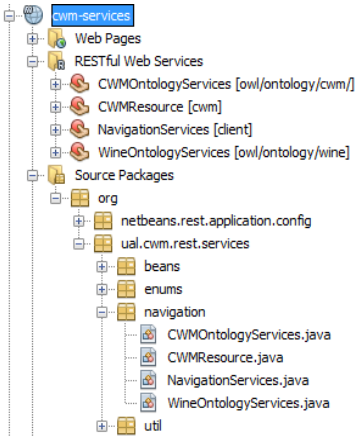


Figura 4.20. Organización del proyecto cwm-owl-services.

Existen diferentes estilos a la hora de implementar los Servicios Web, como se muestra en la Tabla 4. **6**, siendo desarrollados en este proyecto basados en la arquitectura de Servicios REST.

Servicios	Descripción
RPC	Los Servicios Web basados en Llamadas a Procedimientos Remotos representan una interfaz de llamada a procedimientos y funciones distribuidas.
SOA	Los Servicios Web basados en Arquitectura Orientada a Servicios tienen como unidad básica de comunicación el mensaje, más que la operación.
REST	Los Servicios Web basados en REpresentation State Template intentan emular al protocolo HTTP o protocolos similares mediante la restricción de establecer la interfaz a un conjunto conocido de operaciones.

Tabla 4. 6. Tipos de estilos de Servicios Web [50].

REST es una colección de principios para el diseño de arquitecturas en red. Estos principios resumen cómo los recursos son definidos y diseccionados. Cabe destacar que REST no es un estándar, ya que es tan solo un estilo de arquitectura. La implementación del proyecto sigue los cuatro principios de diseño fundamentales de REST:

- Utiliza los métodos de HTTP de manera explícita.
- No mantiene estado.
- Expone URIs con forma de directorios.
- Transfiere XML, JavaScript Object Notation (JSON) o ambos.

Este proyecto utiliza REST debido a la flexibilidad que supone exponer los recursos (individuos/objetos de OWL/CWM) con esta API, suministrando datos al cliente Web. XML sobre HTML es una interfaz que permite el desarrollo de aplicaciones con interfaces basadas en JavaScript Asíncrono + XML (AJAX), conectándose y consumiendo los recursos del Modelo.

Para el desarrollo de los Servicios Web basados en REST se ha utilizado las notaciones que el estándar de java provee, facilitando su implementación en el código como se muestra en los siguientes ejemplos:

```
@Stateless
@Path("cwm/navigation")
public class NavigationServices {
    @Context
    private UriInfo context;
    CWMRelationalPackage relational;
```

Ejemplo 4. 6. Anotación Path de Contexto para los Servicios REST.

```
/**
 * Service getRepositories returns a list of repositories stored into the application
 * @return list of repositories stored into the application
 */
@GET
@Produces("application/xml")
public List<Repository> getRepositories()
{
```

Ejemplo 4. 7. Ejemplo de anotación endpoint de método HTTP GET con REST.

```
/**
 * Service addRepository add a new repository into application
 * @param repository Repository
 */
 @PUT
 @Consumes("application/xml")
 public void addRepository(Repository repository)
 {
```

Ejemplo 4. 8. Ejemplo de anotación endpoint de método HTTP POST con REST.

La Tabla 4. **7**. muestra los tres principales módulos de Servicios REST de los que se compone este proyecto. Los servicios declarados en este proyecto, están adaptados al caso de uso del mismo, y son mayoritariamente servicios de navegación sobre el modelo de metadatos declarados en CWM y las ontologías OWL que se han definido sobre éstos.

Servicios	Descripción
ObjectServices	Servicios de administración y configuración de repositorios y fuentes de datos de la aplicación.
CWMNavegation	Servicios de navegación sobre los objetos definidos en el modelo de metadatos desarrollados en el paquete relacional CWM.
OWLNavegation	Servicios de navegación sobre los individuos OWL que se extraen del modelo de metadatos del paquete relacional CWM.
WineOntology	Servicios específicos sobre gestión y búsquedas de individuos de la ontología `wine.owl` y una ontología extendida de CWM.

Tabla 4. 7. Servicios REST agrupados por funcionalidad del proyecto cwm-owl-services.

A continuación se exponen con más detalle cada uno de los servicios Web proporcionados en este proyecto.

Servicios de Objetos

Los Servicios Web de Objetos son servicios cuya principal funcionalidad es la de crear, borrar, actualizar y acceder sobre recursos u objetos relacionados con la administración de la configuración de la aplicación, como son los Repositorios (`Repository`) y las diferentes fuentes de datos (`DataSource`) (Tabla 4.8.).

- Path : /ual/objects/

Servicios	Descripción
createRepository	Crea un nuevo repositorio en la aplicación para la consulta de sus metadatos
createDataProvider	Crea una nueva configuración de acceso a los datos que se nutre un repositorio.
dropRepository	Borra el repositorio del identificador suministrado como parámetro
dropDataProvider	Borra la configuración del identificador suministrado como parámetro
getRepositories	Devuelve la lista de repositorios registrados en la aplicación
getRepository	Devuelve la información en detalle del repositorio correspondiente a un identificador
getDataproviders	Devuelve la lista de fuentes de datos configuradas en la aplicación
getDataprovider	Devuelve la información detallada de una configuración de un identificador dado

Tabla 4.8. Servicios de gestión de objetos.

Servicios de Navegación del modelo CWM

El ámbito de los objetivos de este proyecto se centra en la implementación de operaciones de navegación y transformación de un modelo relacional de bases de datos al modelo CWM. Los servicios definidos en la Tabla 4.9. son los encargados de permitir el acceso a los metadatos definidos en un Repositorio de la aplicación.

- Path : /ual/cwm/navegation/

Métodos	Descripción
getCatalogs	Obtiene la lista de metadatos de catálogos en CWM de un repositorio dado
getCatalog	Obtiene los metadatos del objeto Catalog CWM dado un identificador de catálogo.
getSchemas	Obtiene la lista de esquemas asociados a un repositorio dado.
getSchema	Obtiene la información detallada y sus relaciones de los metadatos objeto Schema en CWM del nombre de esquema de un repositorio dado.
getProcedures	Obtiene una lista de metadatos CWM de procedimientos almacenados en

	un esquema de base de datos.
getProcedure	Obtiene los metadatos CWM del procedimiento almacenado en el esquema indicado de base de datos
getIndexes	Obtiene la lista de metadatos CWM de índices declarados en un esquema de base de datos dado.
getIndex	Obtiene los metadatos CWM que definen el índice de un esquema dado.
getTriggers	Obtiene la lista de metadatos CWM de Triggers definidos en un esquema de un repositorio dado.
getTrigger	Obtiene los metadatos de CWM para un Trigger en un esquema de un repositorio dado.
getNamedColumnSets	Obtiene la lista de metadatos CWM de tablas y vistas definidas en un esquema y repositorio dado.
getNamedColumnSet	Obtiene los metadatos CWM de una tabla o vista solicitada de un esquema o repositorio dado.
getTables	Obtiene la lista de metadatos CWM de las tablas definidas en un esquema y repositorio dado.
getTable	Obtiene los metadatos CWM de la tabla solicitada de un esquema o repositorio dado.
getViews	Obtiene la lista de metadatos CWM de las vistas definidas en un esquema y repositorio dado.
getView	Obtiene los metadatos CWM de la vista solicitada de un esquema o repositorio dado
getNamedColumnSetConstraints	Obtiene la lista de metadatos CWM de las restricciones asociadas a una tabla o vista definidas en un esquema y repositorio dado.
getNamedColumnSetColumns	Obtiene la lista de metadatos CWM de las columnas asociadas a una tabla o vista definidas en un esquema y repositorio dado.
getNamedColumnSetTriggers	Obtiene la lista de metadatos CWM de Triggers asociados a una tabla o vista definidos en un esquema y repositorio dado.

Tabla 4.9. Métodos de navegación del proyecto CWM.

Todos los servicios de navegación sobre metadata relacional CWM definidos en este proyecto siguen el mismo esquema de llamadas de servicios entre sí. En la Figura 4.21. se muestra el diagrama de secuencias que resulta de la solicitud del servicio de navegación `getSchema()`. En él se hace uso de los métodos de acceso a la metadata de una base de datos relacional, mediante las reglas de transformación SQL (`SQLBridge`) y una conexión al pool de conexiones asociadas al repositorio definido para el esquema (`DBPoolManager`). La clase DBRelationalPackage hace uso de las sentencias y la conexión SQL a la base de datos donde está definido el esquema, para acceder a los metadatos y aplicar las reglas de transformación para obtener el elemento CWM (`Schema`).

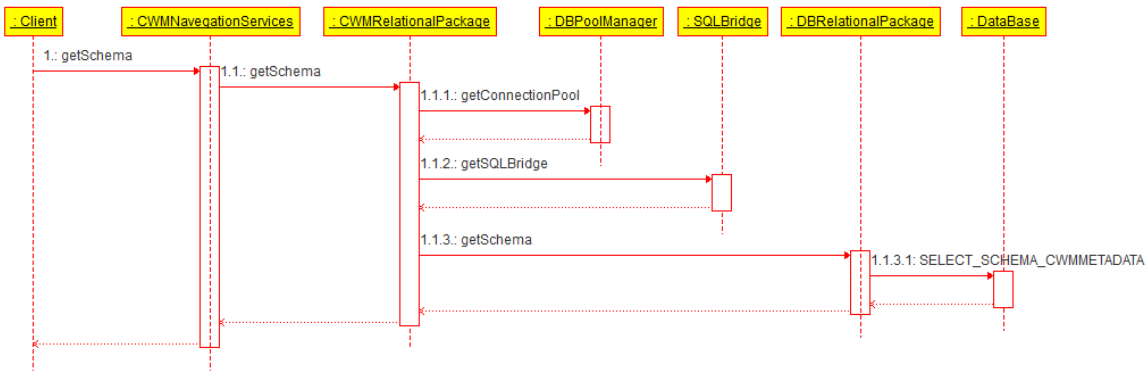


Figura 4.21. Diagrama de secuencias del servicio getSchemas de CWM.

Servicios de Navegación OWL

Los servicios definidos en la Tabla 4.10. contienen todos los accesos a Individuos OWL definidos sobre los metadatos que una base de datos relacional contenga, basados las ontologías definidas para el modelo de metadatos relacional definido en CWM.

- Path: /ual/owl/navegation/

Métodos	Descripción
getCatalogs	Obtiene la lista de individuos OWL de catálogos en CWM de un repositorio dado.
getCatalog	Obtiene el individuo OWL del Catálogo solicitado.
getSchemas	Obtiene la lista de individuos OWL de esquemas asociados a un repositorio dado.
getSchema	Obtiene la información detallada y sus relaciones de los individuos OWL de un esquema de base de datos solicitado.
getProcedures	Obtiene una lista de individuos OWL de procedimientos almacenados en un esquema de base de datos.
getProcedure	Obtiene el individuo OWL del procedimiento almacenado en el esquema indicado de base de datos
getIndexes	Obtiene la lista de individuos OWL de índices declarados en un esquema de base de datos dado.
getIndex	Obtiene el individuo OWL que definen el índice de un esquema dado.
getTriggers	Obtiene la lista de individuos OWL de Triggers definidos en un esquema de un repositorio dado.
getTrigger	Obtiene el individuo OWL para un Trigger en un esquema de un repositorio dado.
getNamedColumnSets	Obtiene la lista de individuos OWL de tablas y vistas definidas en un esquema y repositorio dado.
getNamedColumnSet	Obtiene el individuo OWL de una tabla o vista solicitada de un esquema o repositorio dado.
getTables	Obtiene la lista de individuos OWL de las tablas definidas en un esquema y repositorio dado.
getTable	Obtiene el individuo OWL de la tabla solicitada de un esquema o repositorio dado.
getViews	Obtiene la lista de individuos OWL de las vistas definidas en un esquema y repositorio dado.
getView	Obtiene el individuo OWL de la vista solicitada de un esquema o repositorio dado
getNamedColumnSetConstraints	Obtiene la lista de individuos OWL de las restricciones asociadas a una tabla o vista definidas en un esquema y repositorio dado.
getNamedColumnSetColumns	Obtiene la lista de individuos OWL de las columnas asociadas a una tabla o vista definidas en un esquema y repositorio dado.
getNamedColumnSetTriggers	Obtiene la lista de individuos OWL de Triggers asociados a una tabla o vista definidos en un esquema y repositorio dado.

Tabla 4.10. Lista de servicios de navegación sobre individuos OWL.

Los servicios de navegación para individuos OWL hacen uso de la librería de acceso a metadatos relacionales de CWM (`CWMRelationalPackage`) y añaden una transformación a los individuos OWL (`OWLSchema`) utilizando las ontologías OWL basadas en el modelo de metadatos CWM (Figura 4.22.).

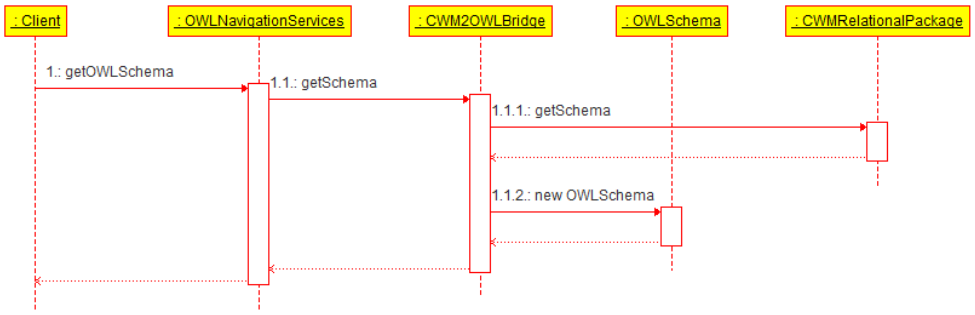


Figura 4.22. Diagrama de Secuencia del servicio de navegación OWL getSchema.

Servicios de búsqueda sobre la ontología Wine.owl

La Tabla 4.11. define los servicios ofrecidos en la clase `WineOntologyServices`. Además de estos servicios, la clase carga la ontología `wine.owl` y la procesa con el razonador HermiT para poder realizar búsquedas DL sobre ella.

Métodos	Descripción
getPlatesByCategory	Obtiene la lista de individuos de la Clase `Course`(plato) dada una categoría (Starters, MainCourse, Dessert).
getNamedIndividuals	Obtiene la lista directa u indirecta de individuos que cumplan las condiciones de la consulta DL que recibe como entrada.

Tabla 4.11. Lista de servicios web de la ontología wine.owl.

4.4.2.3. Modelo

El modelo de la aplicación contiene librerías java para la gestión del modelo de metadatos propuesto en CWM y la gestión de ontologías OWL. La Tabla 4.12. lista los proyectos java que componen esta capa.

Proyecto Maven	Descripción
cwm-library	Librería del modelo relacional definido en CWM
cwm-bridges	Librería para la gestión de modelos CWM y transformaciones de modelos relacionales.
cwm-owl	Librería de transformación de objetos CWM a instancias de OWL y gestión de ontologías OWL basadas en el modelo de metadatos CWM.

Tabla 4.12. Proyectos maven que forman parte del Modelo.

A continuación se expondrán con más detalle cada uno de ellos.

Proyecto cwm-relational

El proyecto **cwm-relational** es una librería Java que contiene las clases y relaciones del paquete relacional definido en CWM así como de los paquetes de los cuales dependa o haga referencia. La implementación de este proyecto se basa en las interfaces definidas para Java en la especificación Java Metadata Interface (JMI/JSR-40), Versión 1.0 [51] y las especificaciones realizadas en CWM 1.1 [2].

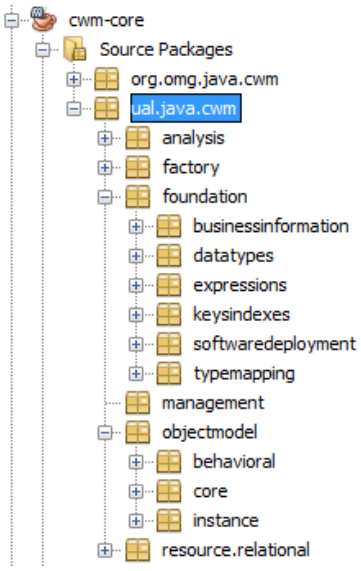


Figura 4.23. Paquetes CWM implementados en el proyecto maven cwm-relational.

La Figura 4.23. muestra los paquetes que se han implementado para la construcción de este proyecto que será usado por los demás proyectos como librería de paquetes CWM. La interface de un metadato, se descompone en dos clases que serán implementadas en este proyecto como muestra el Ejemplo 4.9, una interface define los métodos de acceso a sus propiedades y la otra los métodos específicos de la clase, como pueden ser los constructores de clases en una Factoría Java.

```
/**
 * Class Catalog is the unit of logon and identification. It also identifies the
 * scope of SQL statements: the tables contained in a catalog can be used in a
 * single SQL statement.
 * Contained Elements : Schema
 * @author jmolina
 */
public class Catalog extends ual.java.cwm.objectmodel.core.Package
                     implements  org.omg.java.cwm.resource.relational.Catalog,
                                 org.omg.java.cwm.resource.relational.CatalogClass
{
```

Ejemplo 4.9. Ejemplo de implementación de metadatos CWM usando definiciones Java Metadata Interface.

Una utilidad añadida en la implementación de estos metadatos es la inclusión de anotaciones estándar Java para la serialización Marshall de estas clases en XML. El Ejemplo 4.10. muestra la inclusión de las notaciones y de las importaciones de clases necesarias para poder realizar marshalling sobre esta clase.

```
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

/**
 * Class Catalog is the unit of logon and identification. It also identifies the
 * scope of SQL statements: the tables contained in a catalog can be used in a
 * single SQL statement.
 * Contained Elements : Schema
 * @author jmolina
 */
@XmlRootElement(name ="Catalog")
@XmlAccessorType(XmlAccessType.FIELD)
public class Catalog extends ual.java.cwm.objectmodel.core.Package
                     implements org.omg.java.cwm.resource.relational.Catalog,
                                org.omg.java.cwm.resource.relational.CatalogClass
{
```

```
    /**
     * The name of the default character set used for the values in the column.
     */
    @XmlElement
    private String defaultCharacterSetName;
```

Ejemplo 4.10. Anotaciones Java para la serialización XML marshalling de la clase Catalog.

Un ejemplo de salida de esta serialización Marshall sería la mostrada en el Ejemplo 4. **11.**

```
<Catalog>
    <defaultCharacterSetName>utf-8</defaultCharacterSetName>
</Catalog>
```

Ejemplo 4. 11. XML de salida en el proceso de serialización de la clase Catalog.

Proyecto cwm-bridges

El proyecto cwm-brigdes es un proyecto Maven de Java que hace uso de la librería CWM creada en el proyecto cwm-relational, implementando las transformaciones necesarias de modelos de diferentes fabricantes de bases de datos relacionales al modelo propuesto en CWM.

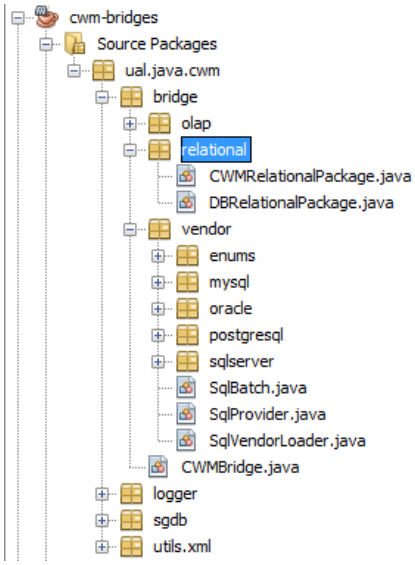


Figura 4.24. Proyecto cwm-bridges implementa el acceso CWM de diferentes fabricantes de bases de datos.

El objetivo de este proyecto es permitir la integración e interoperabilidad entre CWM y diferentes sistemas de bases de datos de una manera rápida y flexible, que nos permita hacer modificaciones o ampliaciones en el caso que fuese necesario. La Figura 4.24. muestra la organización en paquetes establecida por el diseño enfocado a conseguir los objetivos.

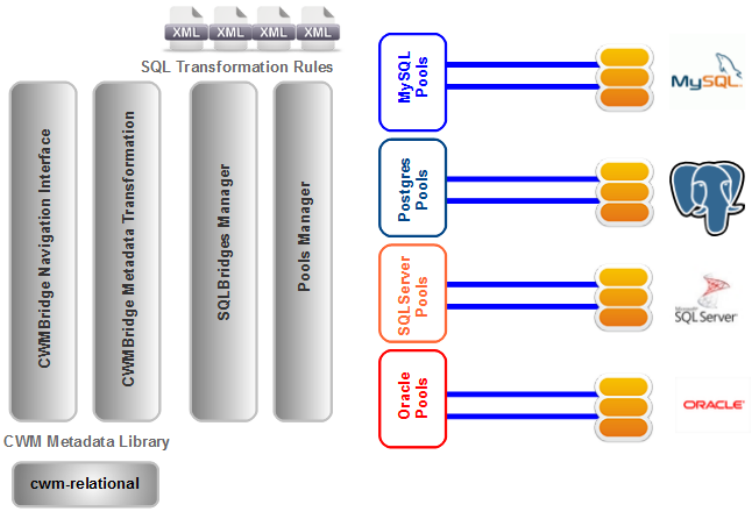


Figura 4.25. Diseño funcional del proyecto maven cwm-bridges.

Como muestra la Figura 4.25, el proyecto se compone principalmente de cuatro clases de funcionalidad:

- **Interfaz de navegación**
  Interface que agrupa los métodos cuya funcionalidad permite la navegación de objetos del paquete relacional de metadatos CWM independientemente del modelo de bases de datos al que acceda.

- **Transformación de los metadatos**
  Clase que aplica las reglas de transformación SQL correspondientes al fabricante de bases de datos relacional, cargadas desde ficheros configurables de la aplicación (SQLBridge) sobre el pool de conexiones correspondiente.

- **Administración de las reglas de transformación SQL**
  Clase encargada de manejar las reglas de transformación a aplicar desde ficheros que contienen sentencias SQL de transformación de los metadatos

- **Administración de las conexiones a las Bases de Datos**
  Clase administradora de los pools de conexiones que se crean en la aplicación para la conexión y navegación de datos entre los diferentes fabricantes de bases de datos. Esta clase utiliza el api **common-pool** que proporciona Apache para la creación de pool de manera universal, indicándole el driver jdbc específico en cada fabricante/versión y su configuración correspondiente.

Cómo añadir un nuevo fabricante al proyecto

El diseño de este proyecto ha sido elaborado para facilitar la ampliación y mantenimiento de fabricantes de bases de datos soportados de una forma modular. La inclusión de un nuevo fabricante al modelo de metadatos definido en CWM, requiere añadir sigue los siguientes componentes:

- **Driver JDBC**
  El establecimiento de una conexión a una base de datos determinada requiere la utilización del driver JDBC de esa base de datos. El pool de conexiones permite abstraer a la aplicación de la gestión individual de cada conexión, así como la interacción con los driver de cada fabricante, limitándose únicamente a la inclusión de la dependencia del driver JDBC al proyecto y la configuración de sus parámetros.

- **Reglas de Transformación**
  Cada nuevo modelo de un fabricante tendrá unas reglas de transformación al modelo CWM particulares de ese fabricante/versión. `SQLBridge` es la clase encargada de facilitar la lectura e inclusión de éstas, mediante la lectura de consultas de transformación en SQL serializadas mediante marshalling XML.

- **Registrar el nuevo fabricante/versión**
  Para que la aplicación pueda hacer uso de ese nuevo driver de base de datos y de las reglas de transformación que tiene asociadas, es necesario registrar y asociar éstos componentes a la versión y el fabricante del modelo de datos al que queremos acceder en los ficheros de configuración de la aplicación.

Proyecto CWM-OWL

El módulo **cwm-owl** es un proyecto Maven de Java cuya principal función es la gestión de las ontologías que definen los modelos de metadatos establecidos en CWM. Este proyecto proporciona los métodos necesarios para la carga, almacenamiento y uso de ontologías usando el API OWL2.

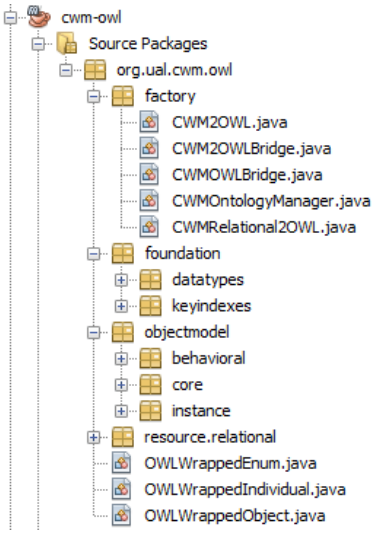


Figura 4.26. Estructura de paquetes del proyecto cwm-owl.

Como se muestra en la Figura 4.26, este proyecto se compone de paquetes que contienen las clases OWL que representan los metadatos definidos en CWM y el paquete **factory** que contiene las clases que proveen los servicios de navegación y transformación sobre estas entidades.

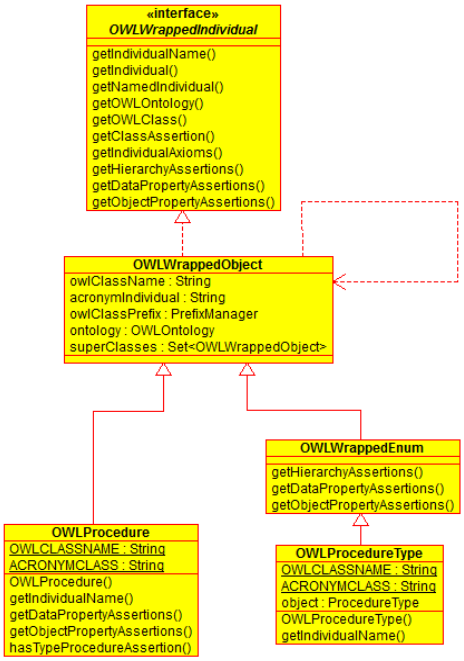


Figura 4.27. Diagrama de las clases principales del proyecto cwm-owl.

Toda clase definida en el proyecto **cwm-library** tiene una clase OWL en este paquete que la contendrá y aportará la funcionalidad heredadas de la clase `OWLWrappedObject` (Figura 4.27.). Esta clase a su vez es la encargada de implementar todos los métodos definidos en la interfaz `OWLWrappedIndividual` necesarios para la lectura de propiedades y definiciones correspondientes de la ontología donde está definida y facilitar la transformación de objetos CWM a Individuos OWL. En la Tabla 4.13. se detallan los métodos definidos en la interfaz a implementar por cada clase OWL definida.

Métodos	Descripción
getIndividualName	Devuelve el nombre del individuo en la ontología. Los objetos CWM que contengan la propiedad nombre implementarán este método utilizando el acrónimo de la clase más el valor del nombre.
getIndividual	Devuelve el individuo de la ontología que corresponde al objeto CWM que contiene en la clase.
getNamedIndividual	Devuelve el individuo con nombre de la ontología que corresponde al objeto CWM que contiene en la clase.
getOWLOntology	Devuelve la ontología a la cuál pertenece el Individuo
getOWLClass	Devuelve la clase OWL definida en la ontología de la que pertenece el Individuo.
getClassAssertion	Devuelve la definición de la clase OWL para ser introducida en una ontología diferente.
getIndividualAxioms	Devuelve el conjunto de definiciones que componen el individuo OWL con los valores definidos en sus propiedades.
getHierarchyAssertions	Devuelve el conjunto de definiciones de las clases de las que hereda.
getDataPropertyAssertions	Devuelve el conjunto de definiciones y valores de las propiedades de datos definidas en la ontología y el objeto CWM que contiene.
getObjectPropertyAssertions	Devuelve el conjunto de definiciones y valores de las propiedades de objetos definidas en la ontología y el objeto CWM que contiene. En estas definiciones se obtienen todos los individuos definidos en la ontología por las propiedades de objetos de la clase.

Tabla 4.13. Descripción de los métodos de la clase Interface OWLWrappedIndividual.

4.5. Resumen

En el desarrollo de este ejemplo práctico del proyecto, se ha hecho especial hincapié en la separación de la lógica de negocio de la implementación del acceso a los datos y la representación, siguiendo el patrón de desarrollo Modelo-Vista-Controlador.

El uso del modelo de metadatos CWM en este caso práctico permite un rápido desarrollo de un nuevo acceso a los metadatos de una base de datos no soportada. La aplicación trabaja directamente con el modelo de metadatos definido en CWM y permite la inclusión de una nueva base de datos añadiendo únicamente unas nuevas reglas de transformación del modelo de metadatos específico de ese fabricante/versión al modelo relacional de CWM sin necesidad de cambiar ni una sola línea de código de la aplicación. Así pues, el uso de CWM aporta a cualquier aplicación de un entorno que tenga unas condiciones muy cambiantes, la facilidad de adaptarse a un nuevo modelo sin necesidad de realizar muchos cambios en el sistema.

Al trabajar con diferentes modelos de bases de datos, los metadatos de cada una de ellas pueden tener diferentes significados o relaciones entre sí, siendo más difícil el manejo de éstos a la hora de realizar operaciones automatizadas. OWL aporta al desarrollo de cualquier aplicación un mayor contenido semántico, separando este significado y razonamiento del desarrollo del sistema, permitiendo un fácil mantenimiento y rápida adaptación a condiciones cambiantes de la lógica de negocio. El desarrollo de una ontología basada en el modelo de metadatos de CWM añade este significado, y permite una mayor expresividad semántica sobre los metadatos. En el caso práctico que en este capítulo se expone, se realiza de manera automatizada la conversión de objetos CWM a individuos OWL de las ontologías definidas, permitiendo el uso de razonadores y reglas semánticas utilizadas sobre los individuos para obtener el significado de cada uno de los metadatos adaptados al dominio de trabajo que convenga en cada situación.

El desarrollo de este caso práctico es únicamente un pequeño ejemplo del uso y posibilidades que ofrece la combinación del modelo de metadatos definido en CWM con el uso de ontologías expresadas mediante OWL. En el siguiente capítulo se exponen el estado de estas tecnologías así como futuros trabajos para continuar por esta línea de investigación y desarrollo.

Capítulo 5.

Conclusiones y Líneas Futuras

Contenido

5.1. Conclusiones

En el desarrollo de este proyecto se expuso los beneficios de la construcción de ontologías OWL para representar el conocimiento definido en el modelo de metadatos del estándar Common Warehousing Metamodel. Para ilustrar estos beneficios, se diseñó e implementó una solución web permitiendo el acceso a los metadatos de diferentes instancias de bases de datos de diferentes fabricantes. En el desarrollo del proyecto se incluyeron los siguientes puntos:

- Se analizó la especificación Common Warehouse Metadata (CWM).
  El desarrollo de la solución implementa el modelo de metadatos relacional propuesto en CWM, permitiendo la construcción de 'puentes' sobre los diferentes modelos de bases de datos existentes en la industria.

- Se investigó el uso de ontologías para añadir significado semántico.
  La utilización de ontologías mantiene la lógica y razonamiento semántico separado de otras capas de desarrollo, permitiendo una rápida adaptación a cambios de dominio y significado de los metadatos. Las ontologías facilitan la automatización de las herramientas que manejan estos metadatos.

- Se expresaron las ontologías utilizadas en el lenguaje OWL2.
  Se aumentó la capacidad computacional y de expresividad semántica utilizando el lenguaje OWL2, así como el acceso web a estos términos. OWL2 permite la construcción de herramientas sobre los metadatos de CWM de una manera escalable y con una mayor expresividad semántica.

- Se estudió y utilizó herramientas de edición de ontologías.
  La utilización de herramientas de edición de ontologías reduce los tiempos de desarrollo de las ontologías y aumentan la consistencia de las mismas. Protégé es la herramienta que se utilizó, ya que además de la edición, permite el uso de razonadores, búsquedas sobre las ontologías, importación y chequeos de consistencia que dan robustez a las ontologías construidas.

- Se integró el uso de la librería de java OWL API para gestionar las entidades, expresiones y axiomas definidos por las ontologías.

- Se construyeron ontologías OWL que definen el paquete relacional de CWM 1.1.
  Se expresaron los metadatos del modelo de CWM mediante ontologías OWL, añadiendo un mayor significado semántico al modelo e interoperabilidad entre diferentes dominios. Funciones inversas, transitividad y restricciones, entre otras expresiones semánticas, de los metadatos y sus relaciones están definidas en estas ontologías.

- Se accedió a metadatos de bases de datos relacionales de diferentes fabricantes, mediante objetos CWM e individuos de la ontología definidos en OWL2.
  Se automatizó la transformación entre objetos de CWM y los individuos OWL y el acceso a los mismos mediante aplicaciones web, abriendo el sistema a nuevos modelos de bases de datos, nuevas reglas de transformación, nuevas ontologías sobre dominios de trabajo y en general a un sistema fácil de administrar y adaptado a los cambios de una manera sencilla y localizada.

Así pues, comprobamos que el desarrollo de soluciones basadas en el modelo CWM y el razonamiento semántico de sus términos y relaciones expresadas en ontologías OWL permiten una rápida integración en sistemas de condiciones cambiantes y heterogéneas, reduciendo costes y aumentando la interoperabilidad de las soluciones.

5.2. Trabajos y Líneas Futuras

El contenido de este proyecto se centra en la creación de una ontología OWL basada en el modelo de metadatos que se definen en el paquete relacional de CWM. Este proyecto ofrece un primer acercamiento a la utilización de herramientas semánticas en la extracción y uso de los metadatos de bases de datos. Durante los capítulos centrales de este proyecto se introducen algunos conceptos, consolidando la base de la línea de investigación futura. Algunos de estos trabajos futuros son:

- Ampliar el estudio de los modelos de metadatos definidos en CWM.

- Integrar librerías OLAP Java (Java Metadata Interface JMI/JSR-40, J2EE, JOLAP o JDMAPI) basadas en Model Driven Data Warehousing (MDA, UML, MOF, XMI, CWM).

- Desarrollar ontologías de todos los paquetes de metadatos definidos en CWM.

- Integrar un razonador de ontologías OWL2 existente en el mercado, ampliando la herramienta que se desarrolla en el proyecto.

- Usar reglas SWRL y buscadores semánticos sobre las ontologías OWL2 basadas en el estándar CWM.

- Adaptar las ontologías de diferentes dominios de conocimientos (Biología, Medicina, Audiovisuales, Telecomunicaciones… etc.) sobre las ontologías CWM para automatizar la extracción y administración de metadatos.

El estudio y desarrollo de estas líneas de investigación cubre las necesidades actuales de las herramientas Warehousing, permitiendo una mejor integración entre ellas y los metadatos que manejan de diferentes dominios y fabricantes.

Un ejemplo de aplicación se produce en el mundo de las Telecomunicaciones, donde el amplio número de fabricantes (Ericsson, Huawei, ZTE, Nokia, Alcatel…etc.), las diferentes tecnologías (GSM, UMTS, LTE) y los términos y relaciones que manejan cada uno de ellos hacen que el uso de herramientas Warehousing que utilicen estas el significado de cada uno de estos términos, integren los modelos de datos de cada uno de los diferentes fabricantes, así como los diferentes metadatos de bases de datos (Oracle, Sybase, DB2...etc.) den un alto rendimiento y reduzcan costes a las operadoras que se hagan con los servicios.

Las herramientas ETL son soluciones que tienen un alto coste de implantación en una empresa, debido a la integración de diferentes fuentes de datos y su fuerte análisis funcional de los datos de origen para transformarlos en los de destino. Por ejemplo, las televisiones manejan diferentes fuentes de datos, de distintos sistemas que los proveen, con un conjunto de metadatos que son necesarios definirlos en la forma que sea utilizable por diferentes tipos de usuarios como son periodistas, editores, presentadores y directores. Herramientas ETL que utilicen el estándar CWM y ontologías, permitirían una mayor integración, análisis y automatización de los metadatos de cada una de las fuentes de origen [52].

ANEXO A.

Uso del editor Protégé v4.3 para la creación de una ontología

Este anexo es una guía introductoria para la creación de una ontología mediante el editor Protégé 4.3v. Este editor ha sido utilizado para la generación y edición de todas las ontologías mostradas en este proyecto.

Se mostraran los siguientes pasos fundamentales para realizar para la creación de una ontología en esta herramienta:

1. Crear un fichero para la definición de nuestra ontología.
2. Definición de elementos de una ontología (Anotaciones, Clases, Propiedades… etc).
3. Validar y consultar la creación de una Ontología.

Creación de una nueva ontología

Protégé es una herramienta de edición de ontologías que nos permite crear una ontología ahorrándonos gran parte de la escritura de sintaxis en el formato que se desee y representando los elementos de una ontología de una manera visual.

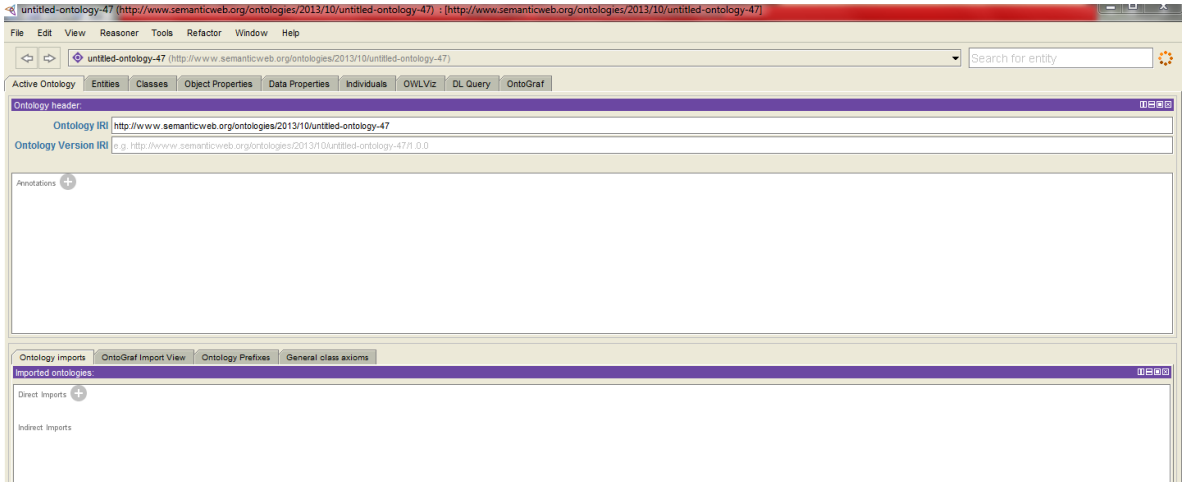


Figura AI.1. Pantalla inicial del editor Protégé.

En la siguiente pantalla del asistente (**Figura AI.2**.) tendremos que seleccionar la sintaxis deseada para nuestra ontología entre los diferentes lenguajes de ontologías que dispone. Protégé permite realizar un cambio de sintaxis entre los lenguajes que dispone manteniendo la ontología de forma transparente al usuario.

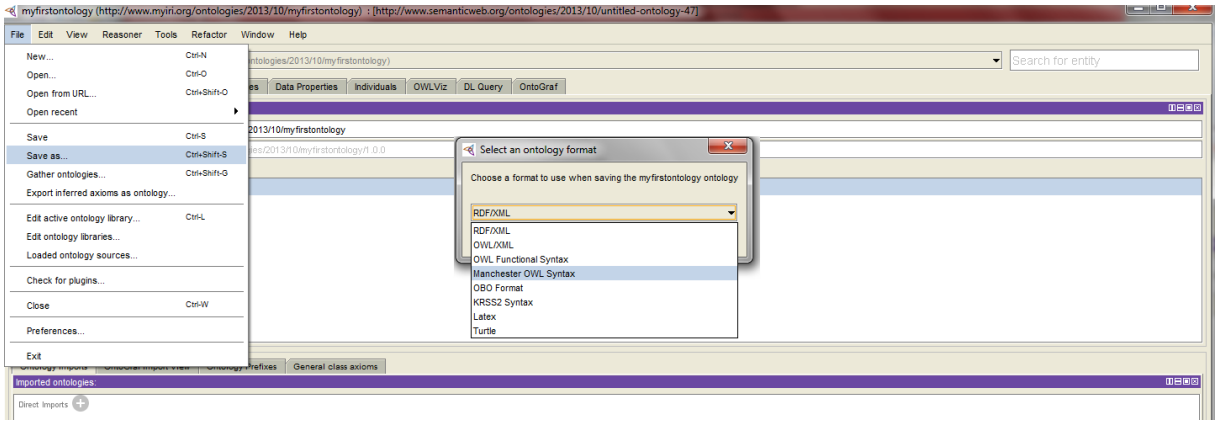


Figura AI.2. Pantalla de selección de formato en Protégé.

En el caso que nuestra ontología necesite importar otra ontología, Protégé facilita esta acción mediante un asistente de importación localizado dentro de la pestaña "Ontology Active" en la sección "Imported Ontologies" como se muestra en la **Figura AI.3.**

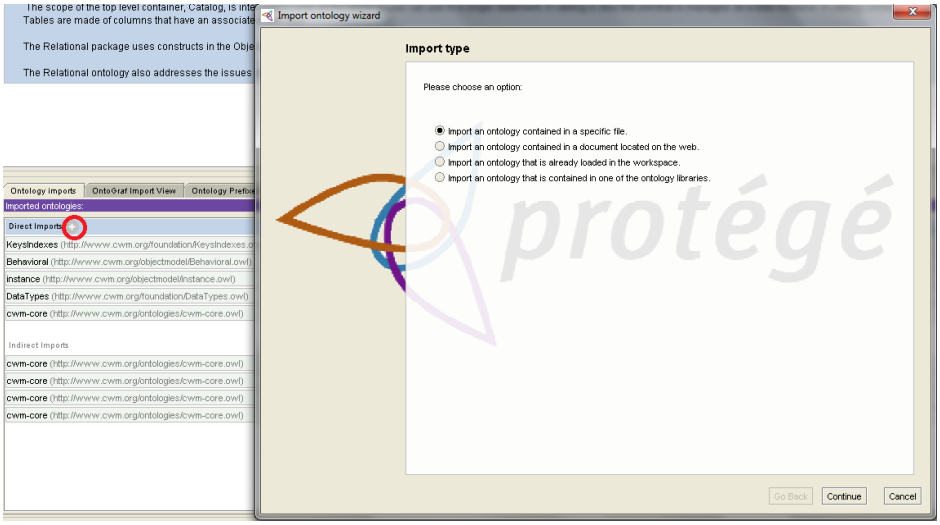


Figura AI.3. Importar una ontología en Protégé.

Definición de las entidades de una ontología

La pantalla principal del editor tiene organizada la funcionalidad por pestañas, donde se pueden realizar acciones sobre diferentes entidades de la ontología como son las Clases, Propiedades e Individuos. Tanto en la ontología como en cada una de las entidades mencionadas, es posible añadir anotaciones para añadir significado semántico a la ontología como se muestra en la **Figura AI.4.**

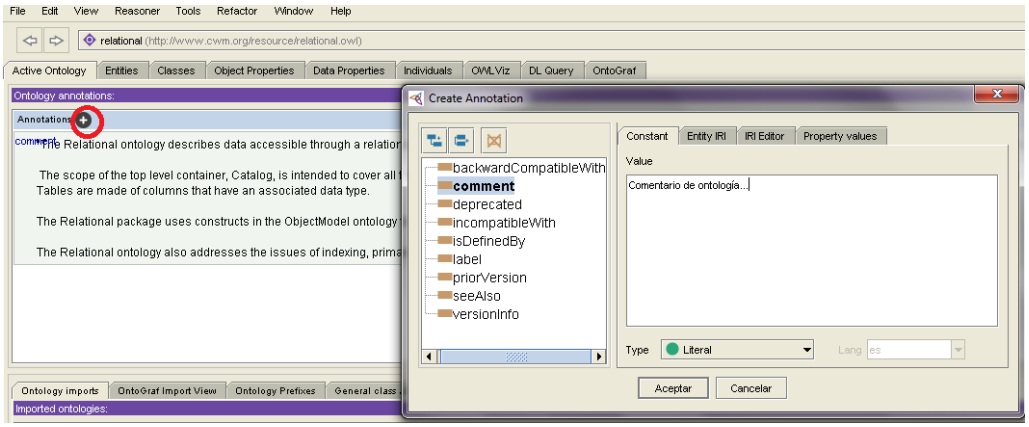


Figura AI.4. Añadir anotaciones de comentarios a una ontología en Protégé.

La pestaña **Classes** (**Figura AI.5.**) contiene la funcionalidad relacionada con la edición y representación de clases de la ontología. Se representa la jerarquía de clases mediante un árbol a la izquierda de la pantalla y una serie de acciones a realizar sobre esta (Creación de una subclase, de una clase hermana o borrado de la clase seleccionada). A la derecha de la pantalla, muestra para su edición, los detalles de la Clase seleccionada.

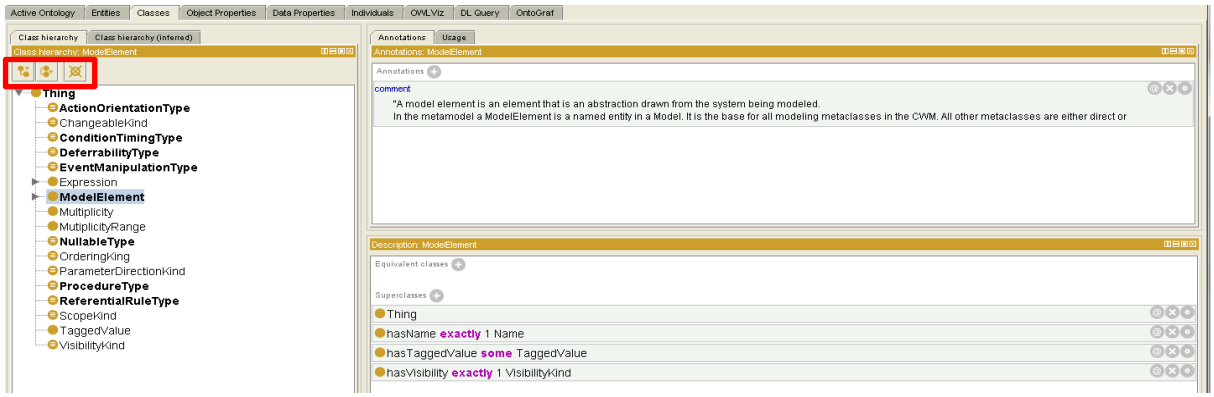


Figura AI.5. Pestaña de edición de clases en Protégé.

La edición de propiedades se realiza en las pestañas “Object Properties” y “Data Properties” (**Figura AI.6.** y **Figura AI.7.** respectivamente). Al igual que en la edición de clases, a la izquierda de la pantalla se representa la jerarquía de propiedades y a su derecha los detalles de la propiedad seleccionada.

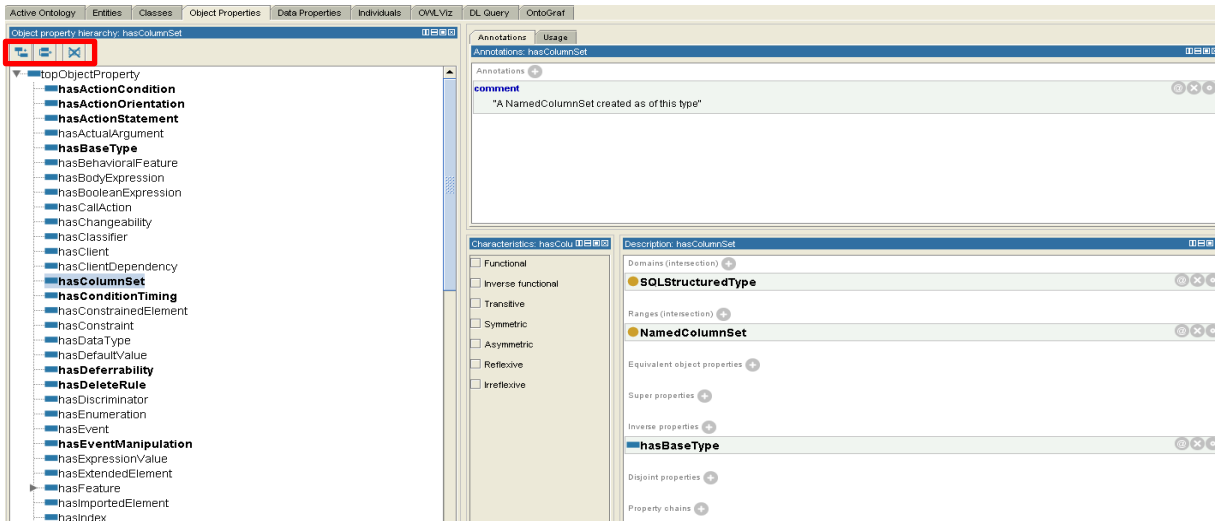


Figura AI.6. Pestaña de edición de propiedades de Objetos en Protégé.

En los detalles de las propiedades de objetos y datos nos permite editar dominios, rangos, propiedades equivalentes y disjuntas así como las características especiales de las propiedades (funcional, inversa, transitiva, simétrica, asimétrica, reflexiva, irreflexiva).

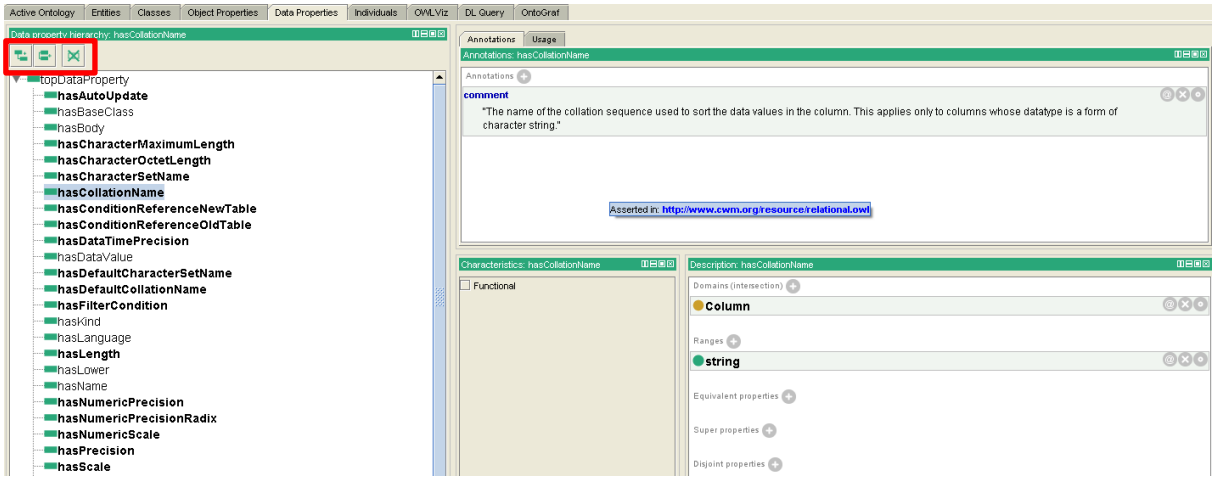


Figura AI.7. Pestaña de edición de propiedades de datos en Protégé v4.3.

La creación de individuos de una clase es representada en la pestaña "Individuals" (**Figura AI.8.**), permitiendo añadir/quitar individuos a la clase seleccionada en la jerarquía.

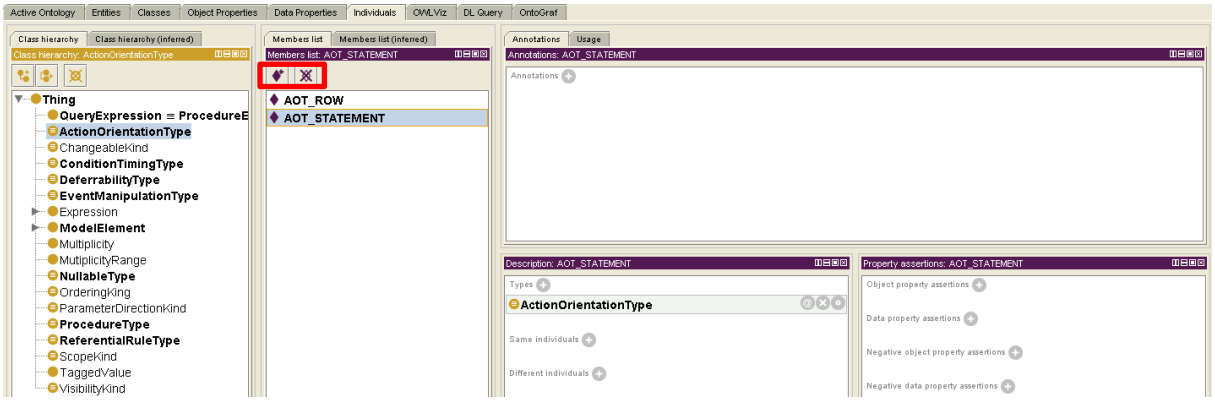


Figura AI.8. Pestaña de Individuos en Protégé v4.3.

El editor Protégé permite añadir Restricciones en sus definiciones a Clases, Individuos y Propiedades para darle un mayor contenido semántico mediante un asistente de restricciones como se muestra en la **Figura AI.9**.

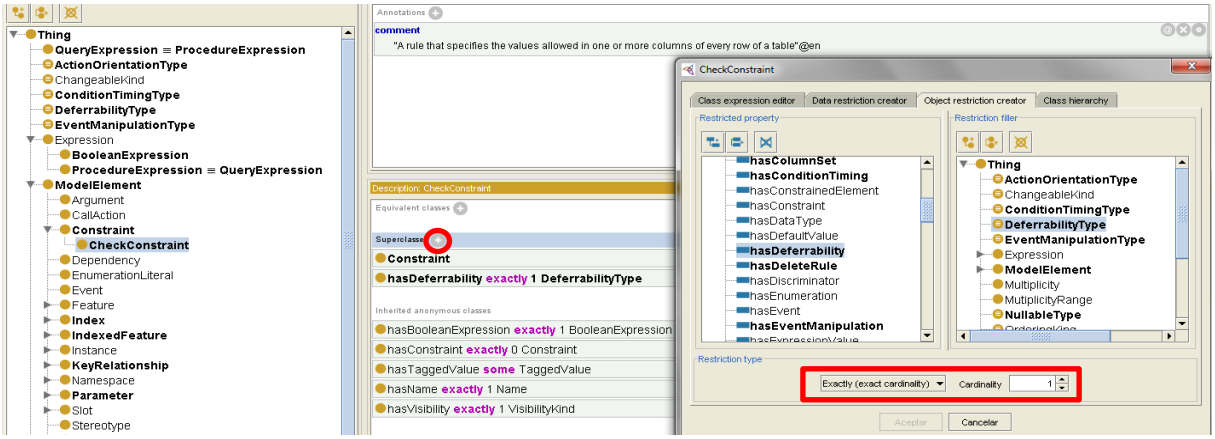


Figura AI.9. Asistente de creación de Restricciones en Protégé.

Validar y consultar la creación de una ontología

Además de la creación y edición de ontologías, el editor Protégé incluye herramientas útiles para la representación, validación y consultas semánticas mediante razonadores sobre la ontología creada.

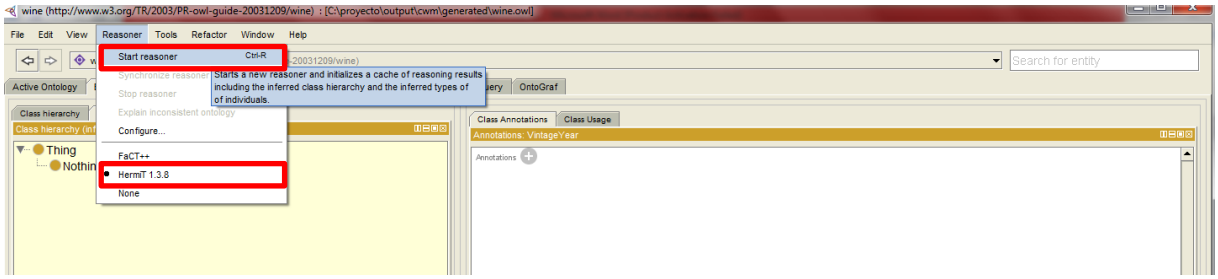


Figura AI.10. Iniciar razonador sobre la ontología en Protégé.

Desde el menú Reasoner (**Figura AI.10.**) se puede lanzar el análisis de la ontología creada, analizando la consistencia estructural y semántica de la ontología. En la pestaña Class Hierarchy (inferred) de la **Figura AI.11.** podemos visualizar la jerarquía inferida de las clases. Este análisis permite detectar, en caso de que existan, las inconsistencias de la ontología creada, marcando la clase en rojo.

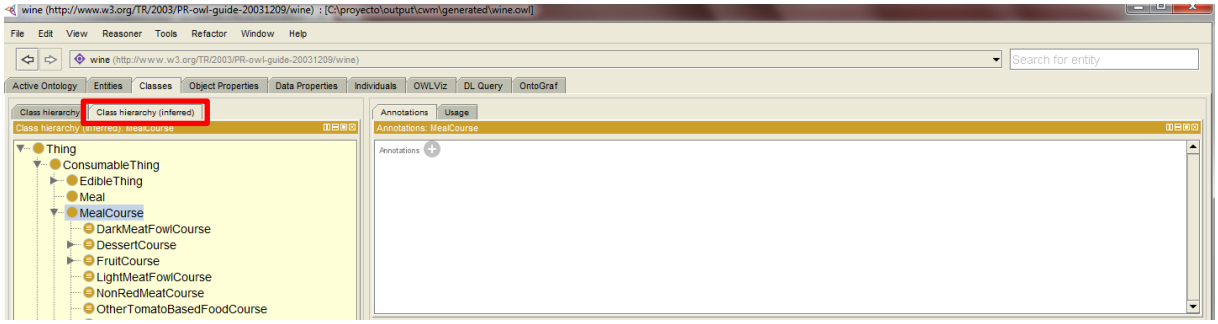


Figura AI.11. Pestaña que muestra como salida la jerarquía inferída.

Al igual que se infiere la jerarquía de clases, se realiza sobre individuos y propiedades permitiendo realizar consultas DL en la pestaña "DL Query" (**Figura AI.12.**).

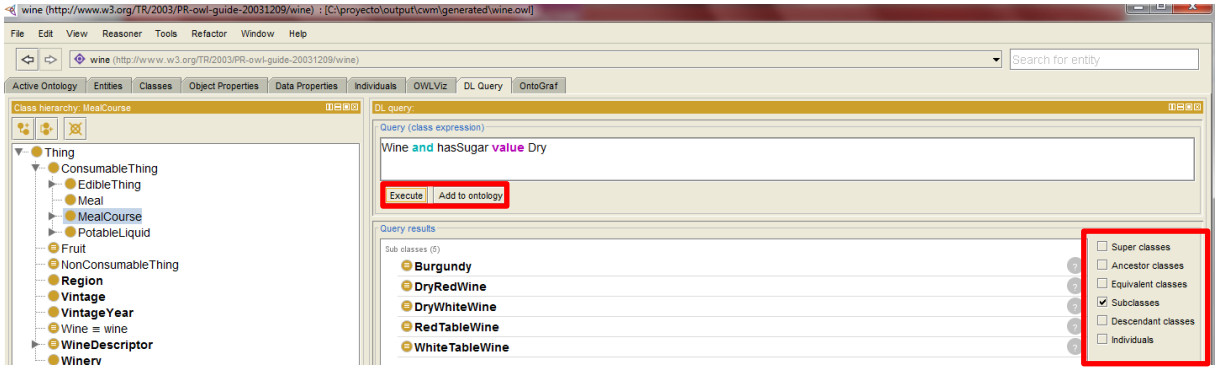


Figura AI.12. Pestaña de consultas DL sobre la ontología en Protégé.

Protégé permite seguir ampliando su funcionalidad mediante la instalación de plugins. Estos plugins pueden añadir nuevos razonadores, exportadores, nuevas pestañas… etc. Un plugin utilizado en la consecución de este proyecto ha sido Ontograf, realizando la representación de la ontología en forma de grafos y permitiendo búsquedas sobre sus componentes y relaciones (**Figura AI.13.**).

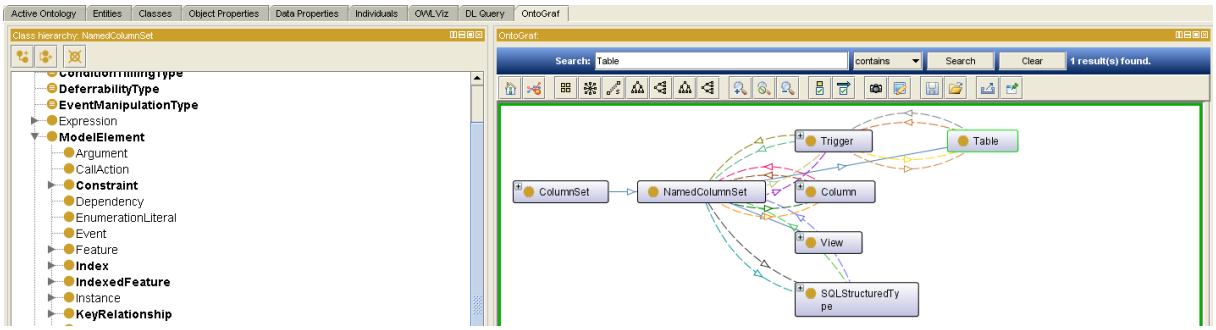


Figura AI.13. Plugin de visualización gráfica OntoGraf en Protégé v4.3.

ANEXO B.

Modelo de datos y Puentes CWM utilizados en el caso práctico

Este anexo contiene los Modelos de datos y "puentes" CWM de los diferentes fabricantes de bases de datos utilizados en el desarrollo de este proyecto (Figura A2. 1.).

Figura A2. 1. Bases de datos utilizadas en el proyecto.

MySQL Bridge

A partir de la versión 5.0.2 de MySQL los metadatos están almacenados en el esquema `INFORMATION_SCHEMA`. Las tablas de este esquema almacenan los metadatos de Vistas, Tablas, Procedimientos y otros metadatos de la base de datos siguiendo el estándar ANSI/ISO SQL:2003 Parte 11 Schemata [53]. En la Figura A2.2. se representan las tablas (en realidad estas tablas son vistas) utilizadas para la lectura de los principales metadatos del modelo relacional CWM.

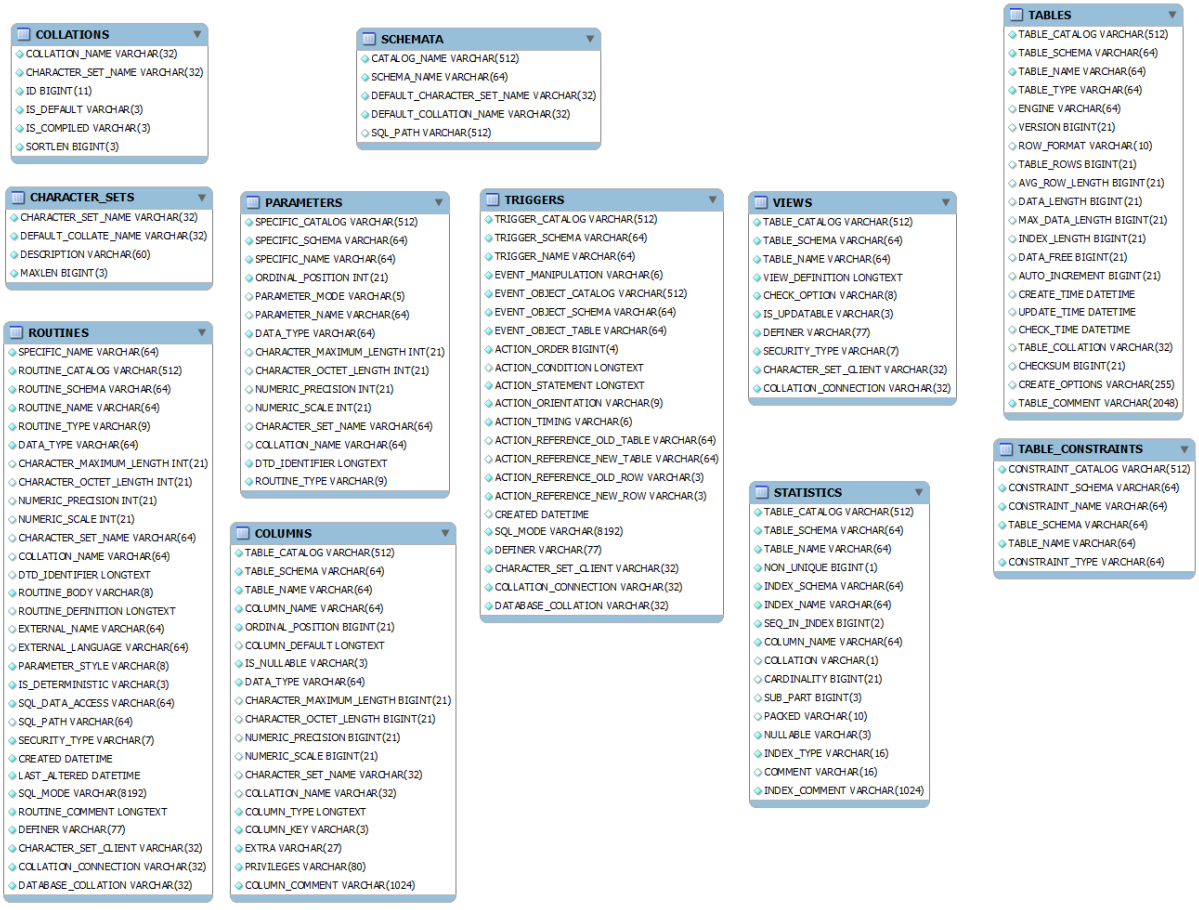


Figura A2.2. Modelo de datos de MySQL (information_schema).

La solución utilizada en el caso práctico, lee la información de los metadatos de MySQL haciendo uso de la lista de transformaciones (consultas SQL del Archivo A2. 1) sobre las tablas

del esquema `INFORMATION_SCHEMA` para la carga de objetos java del paquete CWM Relational.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<bridge name="MySQL.5.5" displayName="MySQL Bridge(Default)">
    <vendor>MySQL</vendor>
    <rules>
        <rule name="GET_CATALOGS">
            <transformation>SELECT DISTINCT IFNULL(a.CATALOG_NAME,'def') NAME,'vk_public' VISIBILITY,
                                DEFAULT_CHARACTER_SET_NAME DEFAULTCHARACTERSETNAME,DEFAULT_COLLATION_NAME
DEFAULTCOLLATIONNAME FROM INFORMATION_SCHEMA.SCHEMATA a </transformation>
        </rule>
        <rule name="GET_CATALOG">
            <transformation>SELECT 'vk_public' VISIBILITY,DEFAULT_CHARACTER_SET_NAME DEFAULTCHARACTERSETNAME,
                                DEFAULT_COLLATION_NAME DEFAULTCOLLATIONNAME
                             FROM INFORMATION_SCHEMA.SCHEMATA a
                            WHERE a.CATALOG_NAME=?
                            LIMIT 1
                        </transformation>
                             <parameter name="catalogName" order="1" type="STRING"/>
        </rule>
        <rule name="GET_SCHEMAS">
            <transformation>SELECT SCHEMA_NAME SCHEMANAME,'vk_public' VISIBILITY,
                                DEFAULT_CHARACTER_SET_NAME DEFAULTCHARACTERSETNAME,
                                DEFAULT_COLLATION_NAME DEFAULTCOLLATIONNAME
                             FROM INFORMATION_SCHEMA.SCHEMATA a
                            WHERE a.CATALOG_NAME=?
                 </transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
        </rule>
        <rule name="GET_SCHEMA">
            <transformation>SELECT 'vk_public' VISIBILITY,DEFAULT_CHARACTER_SET_NAME DEFAULTCHARACTERSETNAME,
                                DEFAULT_COLLATION_NAME DEFAULTCOLLATIONNAME
                             FROM INFORMATION_SCHEMA.SCHEMATA a
                            WHERE a.CATALOG_NAME=? AND a.SCHEMA_NAME=?</transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
            <parameter name="schemaName" order="2" type="STRING"/>
        </rule>
        <rule name="GET_PROCEDURES">
            <transformation>SELECT ROUTINE_NAME NAME,'vk_public' VISIBILITY,'sk_classifier' OWNERSCOPE,
                                IF(ROUTINE_BODY='SQL',0,1) ISQUERY, ROUTINE_DEFINITION BODYEXPRESSION,
                                DATABASE_COLLATION LANGUAGE,ROUTINE_TYPE PROCEDURETYPE
                             FROM INFORMATION_SCHEMA.ROUTINES a
                            WHERE a.ROUTINE_CATALOG=? AND a.ROUTINE_SCHEMA=?
                         ORDER BY a.CREATED DESC
                            </transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
            <parameter name="schemaName" order="2" type="STRING"/>
        </rule>
        <rule name="GET_PROCEDURE">
            <transformation>SELECT 'vk_public' VISIBILITY,'sk_classifier' OWNERSCOPE,IF(ROUTINE_BODY='SQL',0,1) ISQUERY,
                                ROUTINE_DEFINITION BODYEXPRESSION,DATABASE_COLLATION LANGUAGE,ROUTINE_TYPE PROCEDURETYPE
                             FROM INFORMATION_SCHEMA.ROUTINES a
                            WHERE a.ROUTINE_CATALOG=? AND a.ROUTINE_SCHEMA=?  AND a.ROUTINE_NAME=?
                         ORDER BY a.CREATED DESC
                            </transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
            <parameter name="schemaName" order="2" type="STRING"/>
            <parameter name="procedureName" order="3" type="STRING"/>
        </rule>
        <rule name="GET_SQLPARAMETERS">
            <transformation>SELECT IFNULL(PARAMETER_NAME,'RETURNPARAM') NAME,'vk_public' VISIBILITY,
                                ORDINAL_POSITION,DTD_IDENTIFIER BODY,COLLATION_NAME LANGUAGE,
                                CONCAT('PDK_',IFNULL(PARAMETER_MODE,'RETURN')) DIRECTION
                             FROM INFORMATION_SCHEMA.PARAMETERS a
                            WHERE a.SPECIFIC_CATALOG=? AND a.SPECIFIC_SCHEMA=? AND a.SPECIFIC_NAME=?
                         ORDER BY a.ORDINAL_POSITION
                            </transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
            <parameter name="schemaName" order="2" type="STRING"/>
            <parameter name="procedureName" order="3" type="STRING"/>
        </rule>
        <rule name="GET_INDEXES">
            <transformation>SELECT a.TABLE_SCHEMA,a.TABLE_NAME,INDEX_NAME NAME,'vk_notapplicable' VISIBILITY,
                                (SUB_PART IS NOT NULL) ISPARTITIONING,TRUE ISSORTED,NON_UNIQUE ISUNIQUE,
                                 GROUP_CONCAT(COLUMN_NAME) FILTER_CONDITION,
                                MIN(IF(NULLABLE IS NOT NULL AND NULLABLE = 'YES',0,1)) ISNULLABLE, TRUE AUTOUPDATE
                             FROM INFORMATION_SCHEMA.STATISTICS a
                            WHERE a.TABLE_CATALOG=? AND a.INDEX_SCHEMA=?
                         GROUP BY a.TABLE_CATALOG,a.TABLE_SCHEMA,a.TABLE_NAME,a.INDEX_SCHEMA, a.INDEX_NAME
                         ORDER BY a.SEQ_IN_INDEX, a.SEQ_IN_INDEX
                            </transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
            <parameter name="schemaName" order="2" type="STRING"/>
        </rule>
        <rule name="GET_INDEX">
            <transformation>SELECT a.TABLE_SCHEMA,a.TABLE_NAME,INDEX_NAME NAME,'vk_notapplicable' VISIBILITY,
                                (SUB_PART IS NOT NULL) ISPARTITIONING,TRUE ISSORTED,NON_UNIQUE ISUNIQUE,
                                 GROUP_CONCAT(COLUMN_NAME) FILTER_CONDITION,
                                 MIN(IF(NULLABLE IS NOT NULL AND NULLABLE = 'YES',0,1)) ISNULLABLE,TRUE AUTOUPDATE
                             FROM INFORMATION_SCHEMA.STATISTICS a
                            WHERE a.TABLE_CATALOG=? AND a.INDEX_SCHEMA=? AND a.INDEX_NAME=?
                         GROUP BY a.TABLE_CATALOG,a.TABLE_SCHEMA,a.TABLE_NAME,a.INDEX_SCHEMA, a.INDEX_NAME
                         ORDER BY a.SEQ_IN_INDEX, a.SEQ_IN_INDEX
                            </transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
            <parameter name="schemaName" order="2" type="STRING"/>
            <parameter name="indexName" order="3" type="STRING"/>
        </rule>
```

```
        <rule name="GET_TRIGGERS">
            <transformation>SELECT a.EVENT_OBJECT_CATALOG TABLE_CATALOG,a.EVENT_OBJECT_SCHEMA TABLE_SCHEMA,
                                a.EVENT_OBJECT_TABLE TABLE_NAME,a.TRIGGER_NAME NAME,'vk_public' VISIBILITY,
                                a.EVENT_MANIPULATION, a.ACTION_CONDITION, a.ACTION_STATEMENT, a.ACTION_ORIENTATION,
                                a.ACTION_TIMING CONDITION_TIMING,
                                a.ACTION_REFERENCE_OLD_TABLE CONDITION_REFERENCEOLDTABLE,
                                a.ACTION_REFERENCE_NEW_TABLE CONDITION_REFERENCENEWTABLE, DATABASE_COLLATION LANGUAGE
                            FROM INFORMATION_SCHEMA.TRIGGERS a
                           WHERE a.TRIGGER_CATALOG=? AND a.TRIGGER_SCHEMA=?
                        ORDER BY a.TRIGGER_NAME, a.ACTION_ORDER
                      </transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
            <parameter name="schemaName" order="2" type="STRING"/>
        </rule>
        <rule name="GET_TRIGGER">
            <transformation>SELECT a.EVENT_OBJECT_CATALOG TABLE_CATALOG,a.EVENT_OBJECT_SCHEMA TABLE_SCHEMA,
                                a.EVENT_OBJECT_TABLE TABLE_NAME,a.TRIGGER_NAME NAME,'vk_public' VISIBILITY,
                                a.EVENT_MANIPULATION, a.ACTION_CONDITION, a.ACTION_STATEMENT, a.ACTION_ORIENTATION,
                            a.ACTION_TIMING CONDITION_TIMING, a.ACTION_REFERENCE_OLD_TABLE CONDITION_REFERENCEOLDTABLE,
                                a.ACTION_REFERENCE_NEW_TABLE CONDITION_REFERENCENEWTABLE, DATABASE_COLLATION LANGUAGE
                             FROM INFORMATION_SCHEMA.TRIGGERS a
                            WHERE a.TRIGGER_CATALOG=? AND a.TRIGGER_SCHEMA=? AND a.TRIGGER_NAME=?
                         ORDER BY a.TRIGGER_NAME, a.ACTION_ORDER</transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
            <parameter name="schemaName" order="2" type="STRING"/>
            <parameter name="triggerName" order="3" type="STRING"/>
        </rule>
        <rule name="GET_TABLES">
            <transformation>SELECT TABLE_NAME NAME,'vk_notapplicable' VISIBILITY,FALSE ISABSTRACT, FALSE ISTEMPORARY,
                                NULL TEMPORARYSCOPE, IF(TABLE_TYPE='SYSTEM TABLE',TRUE,FALSE) ISSYSTEM
                             FROM INFORMATION_SCHEMA.TABLES a
                            WHERE a.TABLE_CATALOG = ? AND a.TABLE_SCHEMA= ? AND a.TABLE_TYPE LIKE '% TABLE'
                         ORDER BY a.TABLE_NAME DESC
                            </transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
            <parameter name="schemaName" order="2" type="STRING"/>
        </rule>
        <rule name="GET_TABLE">
            <transformation>SELECT 'vk_public' VISIBILITY,FALSE ISABSTRACT, FALSE ISTEMPORARY,NULL TEMPORARYSCOPE,
                                IF(TABLE_TYPE='SYSTEM TABLE',TRUE,FALSE) ISSYSTEM
                             FROM INFORMATION_SCHEMA.TABLES a
                            WHERE a.TABLE_CATALOG = ? AND a.TABLE_SCHEMA= ? AND a.TABLE_NAME=? AND a.TABLE_TYPE LIKE '% TABLE'
                         ORDER BY a.CREATE_TIME DESC
                            LIMIT 1
                             </transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
            <parameter name="schemaName" order="2" type="STRING"/>
            <parameter name="tableName" order="3" type="STRING"/>
        </rule>
        <rule name="GET_VIEWS">
            <transformation>SELECT TABLE_NAME NAME,'vk_public' VISIBILITY, VIEW_DEFINITION QUERYEXPRESSIONBODY,
                                COLLATION_CONNECTION QUERYEXPRESSIONLANGUAGE, FALSE ISABSTRACT,
                              IF(IS_UPDATABLE='YES',FALSE,TRUE) ISREADONLY, IF(CHECK_OPTION='NONE',FALSE,TRUE) CHECKOPTION
                             FROM INFORMATION_SCHEMA.VIEWS a
                            WHERE a.TABLE_CATALOG = ? AND a.TABLE_SCHEMA= ?
                         ORDER BY TABLE_NAME
                        </transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
            <parameter name="schemaName" order="2" type="STRING"/>
        </rule>
        <rule name="GET_VIEW">
            <transformation>SELECT 'vk_public' VISIBILITY,VIEW_DEFINITION QUERYEXPRESSIONBODY,
                               COLLATION_CONNECTION QUERYEXPRESSIONLANGUAGE, FALSE ISABSTRACT,
                             IF(IS_UPDATABLE='YES',FALSE,TRUE) ISREADONLY, IF(CHECK_OPTION='NONE',FALSE,TRUE) CHECKOPTION
                             FROM INFORMATION_SCHEMA.VIEWS a
                            WHERE a.TABLE_CATALOG = ? AND a.TABLE_SCHEMA= ? AND a.TABLE_NAME=?
                             LIMIT 1
                             </transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
            <parameter name="schemaName" order="2" type="STRING"/>
            <parameter name="viewName" order="3" type="STRING"/>
        </rule>
        <rule name="GET_NAMEDCOLUMNSETCONSTRAINTS">
            <transformation>SELECT a.CONSTRAINT_NAME NAME, 'vk_notapplicable' VISIBILITY,
                                a.CONSTRAINT_TYPE BOOLEANEXPRESSIONBODY,b.TABLE_COLLATION BOOLEANEXPRESSIONLANGUAGE
                             FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS a ,INFORMATION_SCHEMA.TABLES b
                             WHERE b.TABLE_CATALOG=? AND b.TABLE_SCHEMA=? AND b.TABLE_NAME=?
                               AND b.TABLE_CATALOG=a.CONSTRAINT_CATALOG AND b.TABLE_SCHEMA=a.TABLE_SCHEMA
                               AND b.TABLE_NAME=a.TABLE_NAME
                             </transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
            <parameter name="schemaName" order="2" type="STRING"/>
            <parameter name="ownerName" order="3" type="STRING"/>
        </rule>
                <rule name="GET_NAMEDCOLUMNSETSCOLUMNS">
            <transformation>SELECT TABLE_NAME OWNER,COLUMN_NAME NAME, 'vk_notapplicable' VISIBILITY,DATA_TYPE DATATYPE,
                                'sk_classifier' OWNERSCOPE,'ck_changeable' CHANGEABILITY,NULL MULTIPLICITY,
                                'ok_unordered' ORDERING,'sk_classifier' TARGETSCOPE, COLUMN_DEFAULT INITIALVALUE,
                                NUMERIC_PRECISION, NUMERIC_SCALE SCALE,IF(IS_NULLABLE IS   NULL,
                             'columnNullableUnknown',IF(IS_NULLABLE='YES','columnNullable','columnNoNulls')) ISNULLABLE,
                CHARACTER_MAXIMUM_LENGTH COLUMN_LENGTH, COLLATION_NAME COLLATIONNAME, CHARACTER_SET_NAME CHARACTERSETNAME
                             FROM INFORMATION_SCHEMA.COLUMNS a
                            WHERE TABLE_CATALOG=? AND TABLE_SCHEMA=?
                         ORDER BY a.TABLE_NAME,a.ORDINAL_POSITION
                             </transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
            <parameter name="schemaName" order="2" type="STRING"/>
        </rule>
        <rule name="GET_NAMEDCOLUMNSETCOLUMNS">
            <transformation>SELECT COLUMN_NAME NAME, 'vk_notapplicable' VISIBILITY,DATA_TYPE DATATYPE, 'sk_classifier'
OWNERSCOPE,'ck_changeable' CHANGEABILITY,NULL MULTIPLICITY,'ok_unordered' ORDERING,'sk_classifier' TARGETSCOPE,
COLUMN_DEFAULT INITIALVALUE, NUMERIC_PRECISION, NUMERIC_SCALE SCALE,IF(IS_NULLABLE IS
NULL,'columnNullableUnknown',IF(IS_NULLABLE='YES','columnNullable','columnNoNulls')) ISNULLABLE, CHARACTER_MAXIMUM_LENGTH
```

```
COLUMN_LENGTH, COLLATION_NAME COLLATIONNAME, CHARACTER_SET_NAME CHARACTERSETNAME
                    FROM INFORMATION_SCHEMA.COLUMNS a
                    WHERE TABLE_CATALOG=? AND TABLE_SCHEMA=? AND TABLE_NAME=?
                ORDER BY a.ORDINAL_POSITION
                            </transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
            <parameter name="schemaName" order="2" type="STRING"/>
            <parameter name="ownerName" order="3" type="STRING"/>
        </rule>
        <rule name="GET_NAMEDCOLUMNSETTRIGGERS">
            <transformation>SELECT a.TRIGGER_NAME NAME,'vk_public' VISIBILITY, a.EVENT_MANIPULATION, a.ACTION_CONDITION,
                                    a.ACTION_STATEMENT, a.ACTION_ORIENTATION, a.ACTION_TIMING CONDITION_TIMING,
         a.ACTION_REFERENCE_OLD_TABLE CONDITION_REFERENCEOLDTABLE, a.ACTION_REFERENCE_NEW_TABLE CONDITION_REFERENCENEWTABLE
                            FROM INFORMATION_SCHEMA.TRIGGERS a
                           WHERE a.TRIGGER_CATALOG=? AND a.TRIGGER_SCHEMA=? AND a.ACTION_REFERENCE_OLD_TABLE=?
                       ORDER BY a.TRIGGER_NAME
                    </transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
            <parameter name="schemaName" order="2" type="STRING"/>
            <parameter name="ownerName" order="3" type="STRING"/>
        </rule>
        <rule name="GET_TRIGGERUSEDCOLUMNSET">
        </rule>
                  <rule name="GET_ROWSTABLE">
                  SELECT :COLUMNS FROM :SCHEMANAME.:TABLENAME
                  </rule>
          </rules>
</bridge>
```

Archivo A2. 1. Bridge CWM de MySQL 5.5.

Postgresql Bridge

La base de datos PostgreSQL además de las vistas sobre objetos definidos en el estándar `INFORMATION_SCHEMA`, dispone de un catálogo de metadatos específico de PostgreSQL. En la **Figura A2.3** se muestran las tablas utilizadas en la carga de objetos CWM.

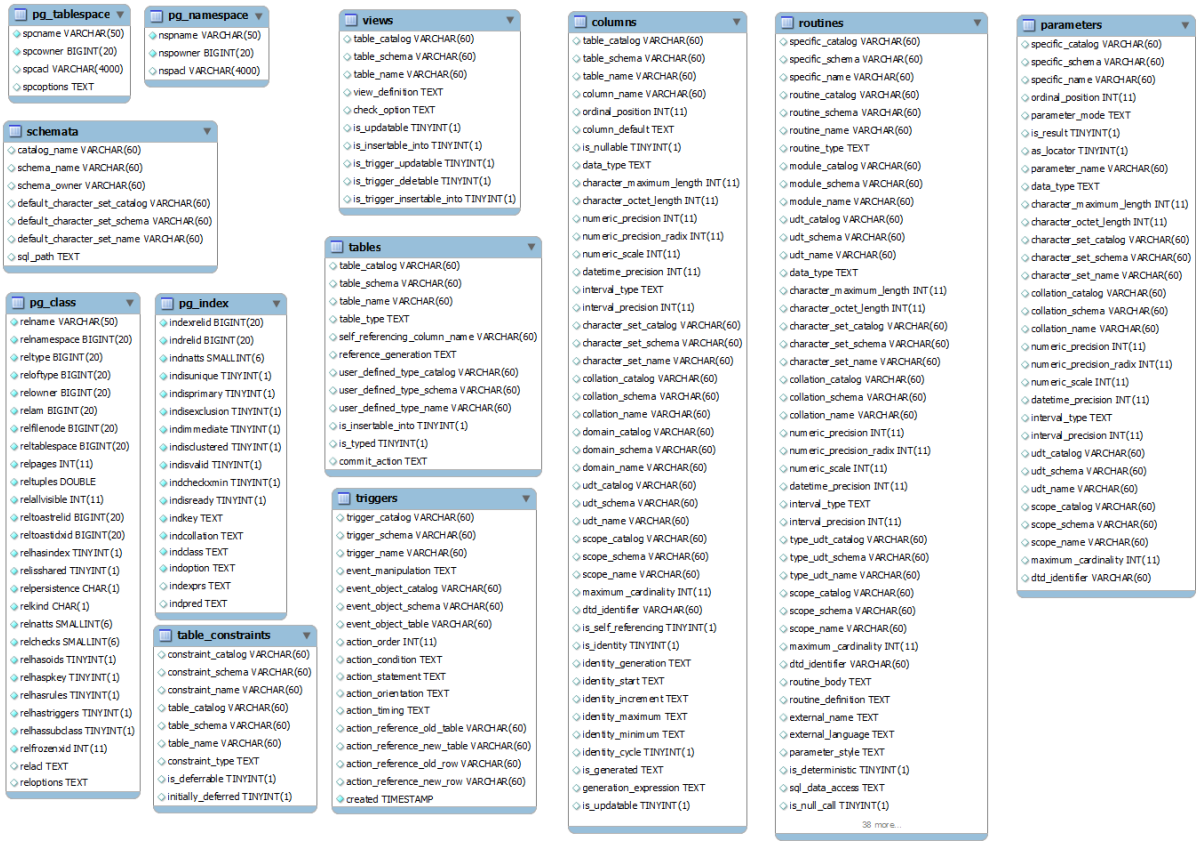


Figura A2.3. Modelo de datos en Postgresql.

Los objetos del paquete relacional CWM son cargados mediante las consultas SQL establecidas en el Archivo A2.2.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<bridge name="Postgresql.9.2" displayName="Postgresql Bridge(Default)">
    <vendor>Postgresql</vendor>
    <rules>
        <rule name="GET_CATALOGS">
            <transformation>SELECT DATNAME as NAME,
                                (CASE WHEN (DATALLOWCONN = TRUE) THEN 'vk_public' ELSE 'vk_private' END) as VISIBILITY,
                                 DATCTYPE DEFAULTCHARACTERSETNAME,
                                 DATCOLLATE DEFAULTCOLLATIONNAME
                               FROM PG_DATABASE a
                              WHERE DATISTEMPLATE = FALSE
                           ORDER BY DATNAME </transformation>
        </rule>
        <rule name="GET_CATALOG">
            <transformation>SELECT (CASE WHEN (DATALLOWCONN = TRUE) THEN 'vk_public' ELSE 'vk_private' END) as VISIBILITY,
                                 DATCTYPE DEFAULTCHARACTERSETNAME, DATCOLLATE DEFAULTCOLLATIONNAME
                               FROM PG_DATABASE a
                              WHERE DATISTEMPLATE = FALSE AND a.DATNAME=?
                     LIMIT 1
                          </transformation>
                                <parameter name="catalogName" order="1" type="STRING"/>
        </rule>
        <rule name="GET_SCHEMAS">
            <transformation>SELECT SCHEMA_NAME SCHEMANAME,'vk_public' VISIBILITY,
                                 DEFAULT_CHARACTER_SET_NAME DEFAULTCHARACTERSETNAME,
                                 default_character_set_catalog DEFAULTCOLLATIONNAME
                               FROM INFORMATION_SCHEMA.SCHEMATA a
                              WHERE a.CATALOG_NAME=?
                    </transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
        </rule>
        <rule name="GET_SCHEMA">
            <transformation>SELECT 'vk_public' VISIBILITY,
                                 DEFAULT_CHARACTER_SET_NAME DEFAULTCHARACTERSETNAME,
                                 default_character_set_catalog DEFAULTCOLLATIONNAME
                               FROM INFORMATION_SCHEMA.SCHEMATA a
                              WHERE a.CATALOG_NAME=? AND a.SCHEMA_NAME=?</transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
            <parameter name="schemaName" order="2" type="STRING"/>
        </rule>
        <rule name="GET_PROCEDURES">
            <transformation>   SELECT ROUTINE_NAME as name, 'vk_public' VISIBILITY,'sk_classifier' OWNERSCOPE,
                                      (CASE WHEN (routine_body='SQL') THEN 0 ELSE 1 END) ISQUERY,
                                       ROUTINE_DEFINITION BODYEXPRESSION,EXTERNAL_LANGUAGE AS LANGUAGE,
                                       ROUTINE_TYPE PROCEDURETYPE
                                  FROM INFORMATION_SCHEMA.ROUTINES a
                                 WHERE a.ROUTINE_CATALOG=? AND a.ROUTINE_SCHEMA=?
                               ORDER BY a.ROUTINE_NAME
                               </transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
            <parameter name="schemaName" order="2" type="STRING"/>
        </rule>
        <rule name="GET_PROCEDURE">
            <transformation>SELECT 'vk_public' as VISIBILITY,'sk_classifier' as OWNERSCOPE,
                                 (CASE WHEN (routine_body='SQL') THEN 0 ELSE 1 END) as ISQUERY,
                                  ROUTINE_DEFINITION as BODYEXPRESSION,EXTERNAL_LANGUAGE AS LANGUAGE,
                                  ROUTINE_TYPE PROCEDURETYPE
                               FROM INFORMATION_SCHEMA.ROUTINES a
                               WHERE a.ROUTINE_CATALOG=? AND a.ROUTINE_SCHEMA=?  AND a.ROUTINE_NAME=?
                           ORDER BY a.ROUTINE_NAME
                               </transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
            <parameter name="schemaName" order="2" type="STRING"/>
            <parameter name="procedureName" order="3" type="STRING"/>
        </rule>
        <rule name="GET_SQLPARAMETERS">
            <transformation>SELECT COALESCE(PARAMETER_NAME,'RETURNPARAM') as NAME,
                                 'vk_public' as VISIBILITY,
                                 ORDINAL_POSITION,
                                 DTD_IDENTIFIER BODY,
                                 COLLATION_NAME as LANGUAGE,
                                 CONCAT('PDK_',COALESCE(PARAMETER_MODE,'RETURN')) as DIRECTION
                               FROM INFORMATION_SCHEMA.PARAMETERS a
                              WHERE a.SPECIFIC_CATALOG=? AND a.SPECIFIC_SCHEMA=? AND a.SPECIFIC_NAME=?
                           ORDER BY a.ORDINAL_POSITION
                               </transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
            <parameter name="schemaName" order="2" type="STRING"/>
            <parameter name="procedureName" order="3" type="STRING"/>
        </rule>
        <rule name="GET_INDEXES">
            <transformation>SELECT n.nspname AS TABLE_SCHEMA,
                                 c.relname AS TABLE_NAME,
                                 i.relname AS NAME,
                                 'vk_notapplicable' as VISIBILITY,
                                 x.indisclustered as ISPARTITIONING,
                                 TRUE as ISSORTED,
                                 x.indisunique AS ISUNIQUE,
                                 pg_get_indexdef(i.oid) AS FILTER_CONDITION,
                                 NULL as ISNULLABLE,
                                 TRUE as AUTOUPDATE
                               FROM pg_index x
                               JOIN pg_class c ON c.oid = x.indrelid
                               JOIN pg_class i ON i.oid = x.indexrelid
                               LEFT JOIN pg_namespace n ON n.oid = c.relnamespace
                               LEFT JOIN pg_tablespace t ON t.oid = i.reltablespace
                              WHERE (n.NSPNAME=? OR n.NSPNAME=?) AND c.relkind = 'r'::"char" AND i.relkind = 'i'::"char"
                              ORDER BY n.NSPNAME,c.relname
                              </transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
            <parameter name="schemaName" order="2" type="STRING"/>
        </rule>
        <rule name="GET_INDEX">
```

```
        <transformation>SELECT n.nspname AS TABLE_SCHEMA,
                               c.relname AS TABLE_NAME,
                               i.relname AS NAME,
                               'vk_notapplicable' as VISIBILITY,
                               x.indisclustered as ISPARTITIONING,
                               TRUE as ISSORTED,
                               x.indisunique AS ISUNIQUE,
                               pg_get_indexdef(i.oid) AS FILTER_CONDITION,
                               NULL as ISNULLABLE,
                               TRUE as AUTOUPDATE
                          FROM pg_index x
                          JOIN pg_class c ON c.oid = x.indrelid
                          JOIN pg_class i ON i.oid = x.indexrelid
                          LEFT JOIN pg_namespace n ON n.oid = c.relnamespace
                          LEFT JOIN pg_tablespace t ON t.oid = i.reltablespace
                WHERE (n.NSPNAME=? OR n.NSPNAME=?) AND i.relname=? AND c.relkind = 'r'::"char" AND i.relkind = 'i'::"char"
                         ORDER BY n.NSPNAME,c.relname
                        </transformation>
        <parameter name="catalogName" order="1" type="STRING"/>
        <parameter name="schemaName" order="2" type="STRING"/>
        <parameter name="indexName" order="3" type="STRING"/>
    </rule>
    <rule name="GET_TRIGGERS">
        <transformation>SELECT a.EVENT_OBJECT_CATALOG as TABLE_CATALOG,
                               a.EVENT_OBJECT_SCHEMA as TABLE_SCHEMA,
                               a.EVENT_OBJECT_TABLE  as TABLE_NAME,
                               a.TRIGGER_NAME as NAME,
                               'vk_public' as VISIBILITY,
                               a.EVENT_MANIPULATION,
                               a.ACTION_CONDITION,
                               a.ACTION_STATEMENT,
                               a.ACTION_ORIENTATION,
                               a.ACTION_TIMING CONDITION_TIMING,
                               a.ACTION_REFERENCE_OLD_TABLE CONDITION_REFERENCEOLDTABLE,
                               a.ACTION_REFERENCE_NEW_TABLE CONDITION_REFERENCENEWTABLE,
                               NULL as LANGUAGE
                          FROM INFORMATION_SCHEMA.TRIGGERS a
                         WHERE a.TRIGGER_CATALOG=? AND a.TRIGGER_SCHEMA=?
                      ORDER BY a.TRIGGER_NAME, a.ACTION_ORDER
                    </transformation>
        <parameter name="catalogName" order="1" type="STRING"/>
        <parameter name="schemaName" order="2" type="STRING"/>
    </rule>
    <rule name="GET_TRIGGER">
        <transformation>SELECT a.EVENT_OBJECT_CATALOG as TABLE_CATALOG,
                               a.EVENT_OBJECT_SCHEMA as TABLE_SCHEMA,
                               a.EVENT_OBJECT_TABLE as TABLE_NAME,
                               a.TRIGGER_NAME as NAME,
                               'vk_public' as VISIBILITY,
                               a.EVENT_MANIPULATION,
                               a.ACTION_CONDITION,
                               a.ACTION_STATEMENT,
                               a.ACTION_ORIENTATION,
                               a.ACTION_TIMING CONDITION_TIMING,
                               a.ACTION_REFERENCE_OLD_TABLE CONDITION_REFERENCEOLDTABLE,
                               a.ACTION_REFERENCE_NEW_TABLE CONDITION_REFERENCENEWTABLE,
                               NULL as LANGUAGE
                          FROM INFORMATION_SCHEMA.TRIGGERS a
                         WHERE a.TRIGGER_CATALOG=? AND a.TRIGGER_SCHEMA=? AND a.TRIGGER_NAME=?
                      ORDER BY a.TRIGGER_NAME, a.ACTION_ORDER</transformation>
        <parameter name="catalogName" order="1" type="STRING"/>
        <parameter name="schemaName" order="2" type="STRING"/>
        <parameter name="triggerName" order="3" type="STRING"/>
    </rule>
    <rule name="GET_TABLES">
        <transformation>SELECT TABLE_NAME as NAME,
                              'vk_notapplicable' as VISIBILITY,
                               FALSE ISABSTRACT,
                               FALSE ISTEMPORARY,
                               NULL TEMPORARYSCOPE,
                               (CASE WHEN(TABLE_TYPE='SYSTEM TABLE') THEN TRUE ELSE FALSE END) as ISSYSTEM
                          FROM INFORMATION_SCHEMA.TABLES a
                         WHERE a.TABLE_CATALOG = ? AND a.TABLE_SCHEMA= ? AND a.TABLE_TYPE LIKE '% TABLE'
                      ORDER BY a.TABLE_NAME DESC
                        </transformation>
        <parameter name="catalogName" order="1" type="STRING"/>
        <parameter name="schemaName" order="2" type="STRING"/>
    </rule>
    <rule name="GET_TABLE">
        <transformation>SELECT 'vk_notapplicable' as VISIBILITY,
                               FALSE as ISABSTRACT,
                               FALSE as ISTEMPORARY,
                               NULL as TEMPORARYSCOPE,
                               (CASE WHEN(TABLE_TYPE='SYSTEM TABLE') THEN TRUE ELSE FALSE END) as ISSYSTEM
                          FROM INFORMATION_SCHEMA.TABLES a
                        WHERE a.TABLE_CATALOG = ? AND a.TABLE_SCHEMA= ? AND a.TABLE_NAME=? AND a.TABLE_TYPE LIKE '% TABLE'
                ORDER BY a.TABLE_NAME DESC
                LIMIT 1
                        </transformation>
        <parameter name="catalogName" order="1" type="STRING"/>
        <parameter name="schemaName" order="2" type="STRING"/>
        <parameter name="tableName" order="3" type="STRING"/>
    </rule>
    <rule name="GET_VIEWS">
        <transformation>  SELECT TABLE_NAME as NAME,
                               'vk_public' as VISIBILITY,
                               VIEW_DEFINITION as QUERYEXPRESSIONBODY,
                               NULL as QUERYEXPRESSIONLANGUAGE,
                               FALSE as ISABSTRACT,
                               (CASE WHEN (IS_UPDATABLE='YES') THEN FALSE ELSE TRUE END) as ISREADONLY,
                               (CASE WHEN (CHECK_OPTION='NONE') THEN FALSE ELSE TRUE END) as CHECKOPTION
                          FROM INFORMATION_SCHEMA.VIEWS a
                         WHERE a.TABLE_CATALOG = ? AND a.TABLE_SCHEMA= ?
```

```
                        ORDER BY TABLE_NAME
                      </transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
            <parameter name="schemaName" order="2" type="STRING"/>
        </rule>
        <rule name="GET_VIEW">
            <transformation>  SELECT 'vk_public' as VISIBILITY,
                                VIEW_DEFINITION as QUERYEXPRESSIONBODY,
                                NULL as QUERYEXPRESSIONLANGUAGE,
                                FALSE as ISABSTRACT,
                                (CASE WHEN (IS_UPDATABLE='YES') THEN FALSE ELSE TRUE END) as ISREADONLY,
                                (CASE WHEN (CHECK_OPTION='NONE') THEN FALSE ELSE TRUE END) as CHECKOPTION
                              FROM INFORMATION_SCHEMA.VIEWS a
                             WHERE a.TABLE_CATALOG = ? AND a.TABLE_SCHEMA= ? AND a.TABLE_NAME=?
                             LIMIT 1
                              </transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
            <parameter name="schemaName" order="2" type="STRING"/>
            <parameter name="viewName" order="3" type="STRING"/>
        </rule>
        <rule name="GET_NAMEDCOLUMNSETCONSTRAINTS">
            <transformation>SELECT a.CONSTRAINT_NAME as NAME,
                                'vk_notapplicable' as VISIBILITY,
                                a.CONSTRAINT_TYPE as BOOLEANEXPRESSIONBODY,
                               NULL as BOOLEANEXPRESSIONLANGUAGE
                             FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS a ,INFORMATION_SCHEMA.TABLES b
                            WHERE b.TABLE_CATALOG=? AND b.TABLE_SCHEMA=? AND b.TABLE_NAME=?
                              AND b.TABLE_CATALOG=a.CONSTRAINT_CATALOG AND b.TABLE_SCHEMA=a.TABLE_SCHEMA
                              AND b.TABLE_NAME=a.TABLE_NAME
                              </transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
            <parameter name="schemaName" order="2" type="STRING"/>
            <parameter name="ownerName" order="3" type="STRING"/>
        </rule>
        <rule name="GET_NAMEDCOLUMNSETSCOLUMNS">
            <transformation>    SELECT TABLE_NAME as OWNER,
                                     COLUMN_NAME as NAME,
                                   'vk_notapplicable' as VISIBILITY,
                                   DATA_TYPE DATATYPE,
                                   'sk_classifier' as OWNERSCOPE,
                                   'ck_changeable' as CHANGEABILITY,
                                   NULL as MULTIPLICITY,
                                   'ok_unordered' as ORDERING,
                                   'sk_classifier' as TARGETSCOPE,
                                   COLUMN_DEFAULT as INITIALVALUE,
                                   NUMERIC_PRECISION,
                                   NUMERIC_SCALE SCALE,
                                   (CASE WHEN (IS_NULLABLE IS NULL) THEN 'columnNullableUnknown' ELSE (CASE
WHEN(IS_NULLABLE='YES') THEN 'columnNullable' ELSE 'columnNoNulls' END) END) as ISNULLABLE,
                                   CHARACTER_MAXIMUM_LENGTH COLUMN_LENGTH,
                                   COLLATION_NAME COLLATIONNAME,
                                   CHARACTER_SET_NAME CHARACTERSETNAME
                                FROM INFORMATION_SCHEMA.COLUMNS a
                               WHERE TABLE_CATALOG=? AND TABLE_SCHEMA=?
                              ORDER BY a.TABLE_SCHEMA,a.TABLE_NAME,a.ORDINAL_POSITION
                               </transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
            <parameter name="schemaName" order="2" type="STRING"/>
        </rule>
        <rule name="GET_NAMEDCOLUMNSETCOLUMNS">
            <transformation>    SELECT COLUMN_NAME as NAME,
                                   'vk_notapplicable' as VISIBILITY,
                                   DATA_TYPE DATATYPE,
                                   'sk_classifier' as OWNERSCOPE,
                                   'ck_changeable' as CHANGEABILITY,  NULL as MULTIPLICITY,
                                   'ok_unordered' as ORDERING,
                                   'sk_classifier' as TARGETSCOPE,
                                   COLUMN_DEFAULT as INITIALVALUE,
                                   NUMERIC_PRECISION, NUMERIC_SCALE SCALE,
                                   (CASE WHEN (IS_NULLABLE IS NULL) THEN 'columnNullableUnknown' ELSE (CASE
WHEN(IS_NULLABLE='YES') THEN 'columnNullable' ELSE 'columnNoNulls' END) END) as ISNULLABLE,
                                   CHARACTER_MAXIMUM_LENGTH COLUMN_LENGTH,
                                   COLLATION_NAME COLLATIONNAME,
                                   CHARACTER_SET_NAME CHARACTERSETNAME
                                FROM INFORMATION_SCHEMA.COLUMNS a
                               WHERE TABLE_CATALOG=? AND TABLE_SCHEMA=? AND TABLE_NAME=?
                              ORDER BY a.ORDINAL_POSITION
                               </transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
            <parameter name="schemaName" order="2" type="STRING"/>
            <parameter name="ownerName" order="3" type="STRING"/>
        </rule>
        <rule name="GET_NAMEDCOLUMNSETTRIGGERS">
            <transformation>SELECT a.TRIGGER_NAME as NAME,'vk_public' as VISIBILITY, a.EVENT_MANIPULATION,
                                a.ACTION_CONDITION, a.ACTION_STATEMENT, a.ACTION_ORIENTATION,
                                a.ACTION_TIMING CONDITION_TIMING, a.ACTION_REFERENCE_OLD_TABLE
CONDITION_REFERENCEOLDTABLE, a.ACTION_REFERENCE_NEW_TABLE CONDITION_REFERENCENEWTABLE
                          FROM INFORMATION_SCHEMA.TRIGGERS a
                         WHERE a.TRIGGER_CATALOG=? AND a.TRIGGER_SCHEMA=? AND a.ACTION_REFERENCE_OLD_TABLE=?
                      ORDER BY a.TRIGGER_NAME
                           </transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
            <parameter name="schemaName" order="2" type="STRING"/>
            <parameter name="ownerName" order="3" type="STRING"/>
        </rule>
        <rule name="GET_TRIGGERUSEDCOLUMNSET">        </rule>
        <rule name="GET_ROWSTABLE">
                    SELECT :COLUMNS FROM :SCHEMANAME.:TABLENAME
                    </rule>
      </rules>
</bridge>
```

Archivo A2.2. Bridge CWM de Postgresql.

Oracle

Oracle facilita el acceso a los metadatos mediante los objectos y vistas definidas con el prefijo `DBA_`. En la **Figura A2.4** se muestran las vistas utilizadas para la carga de los metadatos CWM.

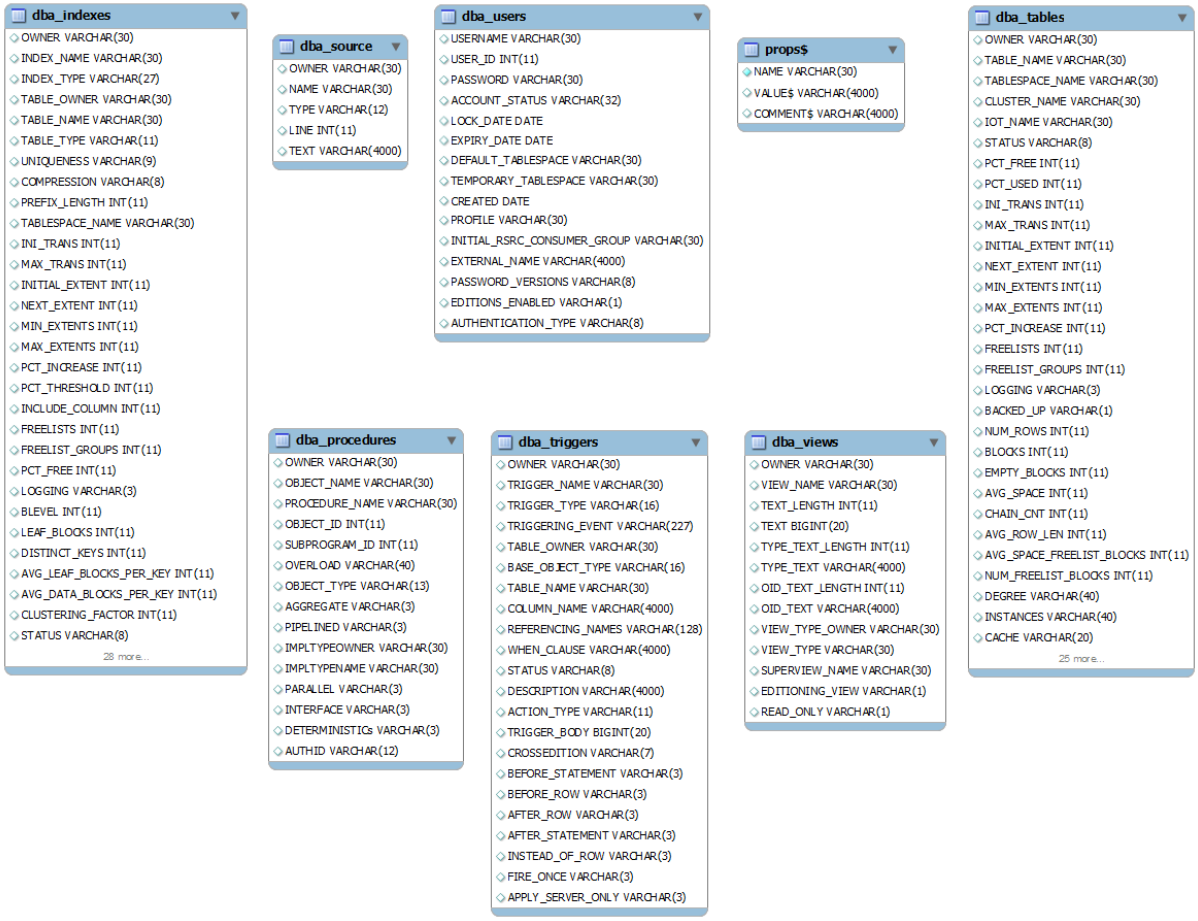


Figura A2.4. Modelo de datos Oracle 11.2.

El **Archivo A2.3** contiene las transformaciones necesarias de los metadatos de la base de atos Oracle para la carga de objetos del paquete relacional CWM.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<bridge name="Oracle.11.2" displayName="Oracle Bridge(Default)">
    <vendor>ORACLE</vendor>
    <rules>
        <rule name="GET_CATALOGS">
            <transformation>SELECT 'default'   NAME,'vk_public' VISIBILITY, value$ DEFAULTCHARACTERSETNAME,
                                null DEFAULTCOLLATIONNAME
                             FROM SYS.PROPS$ WHERE NAME ='NLS_CHARACTERSET'</transformation>
        </rule>
        <rule name="GET_CATALOG">
            <transformation>SELECT 'vk_public' VISIBILITY, value$ DEFAULTCHARACTERSETNAME, null DEFAULTCOLLATIONNAME
                             FROM SYS.PROPS$ WHERE NAME ='NLS_CHARACTERSET' AND 'default'=?  </transformation>
                            <parameter name="catalogName" order="1" type="STRING"/>
        </rule>
        <rule name="GET_SCHEMAS">
            <transformation>SELECT USERNAME SCHEMANAME,'vk_public' VISIBILITY,
                               (SELECT value$ FROM SYS.PROPS$ WHERE NAME ='NLS_CHARACTERSET') DEFAULTCHARACTERSETNAME,
                                NULL DEFAULTCOLLATIONNAME
                            FROM DBA_USERS a
                           WHERE 'default'=?
                 </transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
        </rule>
        <rule name="GET_SCHEMA">
            <transformation>SELECT 'vk_public' VISIBILITY,
                               (SELECT value$ FROM SYS.PROPS$ WHERE NAME ='NLS_CHARACTERSET') DEFAULTCHARACTERSETNAME,
                                NULL DEFAULTCOLLATIONNAME
                            FROM DBA_USERS a
                           WHERE 'default'=? AND a.USERNAME=?</transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
            <parameter name="schemaName" order="2" type="STRING"/>
        </rule>
        <rule name="GET_PROCEDURES">
```

```
    <transformation>  SELECT b.NAME,
                           'vk_public'     VISIBILITY, 'sk_classifier' OWNERSCOPE, NULL            ISQUERY,
                            b.BODY            BODYEXPRESSION, b.TYPE            PROCEDURETYPE
                       FROM DBA_PROCEDURES a,
                      (SELECT c.OWNER,c.TYPE,c.NAME,(listagg(c.text, ' \n ' ) WITHIN GROUP(ORDER BY c.LINE)) AS body
                         FROM DBA_SOURCE c
                        WHERE 'default'=? AND c.OWNER=? AND c.TYPE IN ('FUNCTION','PROCEDURE')
                       GROUP BY c.OWNER,c.TYPE,c.NAME) b
                       WHERE a.OBJECT_TYPE IN ('FUNCTION','PROCEDURE') AND a.OWNER=b.OWNER AND a.OBJECT_NAME=b.NAME
                        AND a.OBJECT_TYPE=b.TYPE
                    ORDER BY a.OWNER,a.OBJECT_TYPE,a.PROCEDURE_NAME,a.OVERLOAD,a.SUBPROGRAM_ID
                     </transformation>
    <parameter name="catalogName" order="1" type="STRING"/>
    <parameter name="schemaName" order="2" type="STRING"/>
</rule>
<rule name="GET_PROCEDURE">
    <transformation>  SELECT 'vk_public'     VISIBILITY, 'sk_classifier' OWNERSCOPE,  NULL            ISQUERY,
                            b.BODY            BODYEXPRESSION, b.TYPE            PROCEDURETYPE
                       FROM DBA_PROCEDURES a,
                      (SELECT c.OWNER,c.TYPE,c.NAME,(listagg(c.text, ' \n ' ) WITHIN GROUP(ORDER BY c.LINE)) AS body
                               FROM DBA_SOURCE c
                              WHERE 'default'=? AND c.OWNER=? AND c.NAME=? AND c.TYPE IN ('FUNCTION','PROCEDURE')
                             GROUP BY c.OWNER,c.TYPE,c.NAME) b
                      WHERE a.OBJECT_TYPE IN ('FUNCTION','PROCEDURE') AND a.OWNER=b.OWNER AND a.OBJECT_NAME=b.NAME
                AND a.OBJECT_TYPE=b.TYPE ORDER BY a.OWNER,a.OBJECT_TYPE,a.PROCEDURE_NAME,a.OVERLOAD,a.SUBPROGRAM_ID
                     </transformation>
    <parameter name="catalogName" order="1" type="STRING"/>
    <parameter name="schemaName" order="2" type="STRING"/>
    <parameter name="procedureName" order="3" type="STRING"/>
</rule>
<rule name="GET_INDEXES">
    <transformation>SELECT a.TABLE_OWNER        TABLE_SCHEMA,
                           a.TABLE_NAME,
                           a.INDEX_NAME NAME,
                           'vk_notapplicable' VISIBILITY,
                           a.PARTITIONED ISPARTITIONING,
                           0 ISSORTED,
                           (CASE WHEN(a.UNIQUENESS='UNIQUE') THEN 0 ELSE 1 END) ISUNIQUE,
                           a.PARAMETERS FILTER_CONDITION,
                           1 ISNULLABLE,
                           0 AUTOUPDATE
                      FROM DBA_INDEXES a
                     WHERE 'default'=? AND a.OWNER=?
                  ORDER BY a.TABLE_NAME, a.INDEX_NAME
                     </transformation>
    <parameter name="catalogName" order="1" type="STRING"/>
    <parameter name="schemaName" order="2" type="STRING"/>
</rule>
<rule name="GET_INDEX">
    <transformation>SELECT a.TABLE_OWNER        TABLE_SCHEMA,
                           a.TABLE_NAME,
                           a.INDEX_NAME NAME,
                           'vk_notapplicable' VISIBILITY,
                           a.PARTITIONED ISPARTITIONING,
                           0 ISSORTED,
                           (CASE WHEN(a.UNIQUENESS='UNIQUE') THEN 0 ELSE 1 END) ISUNIQUE,
                           a.PARAMETERS FILTER_CONDITION,
                           1 ISNULLABLE,
                           0 AUTOUPDATE
                      FROM DBA_INDEXES a
                     WHERE 'default'=? AND a.OWNER=? AND a.INDEX_NAME=?
                  ORDER BY a.TABLE_NAME, a.INDEX_NAME
                     </transformation>
    <parameter name="catalogName" order="1" type="STRING"/>
    <parameter name="schemaName" order="2" type="STRING"/>
    <parameter name="indexName" order="3" type="STRING"/>
</rule>
<rule name="GET_TRIGGERS">
    <transformation>SELECT 'default'             TABLE_CATALOG, a.TABLE_OWNER          TABLE_SCHEMA,
                            a.TABLE_NAME, a.TRIGGER_NAME         NAME, 'vk_public'            VISIBILITY,
                           NULL EVENT_MANIPULATION, NULL ACTION_CONDITION, NULL ACTION_STATEMENT,
                           NULL ACTION_ORIENTATION, NULL CONDITION_TIMING, NULL CONDITION_REFERENCEOLDTABLE,
                           NULL CONDITION_REFERENCENEWTABLE, NULL LANGUAGE
                      FROM DBA_TRIGGERS a
                     WHERE 'default'=? AND a.OWNER=?
                  ORDER BY a.TABLE_NAME,a.TRIGGER_NAME
                </transformation>
    <parameter name="catalogName" order="1" type="STRING"/>
    <parameter name="schemaName" order="2" type="STRING"/>
</rule>
<rule name="GET_TRIGGER">
    <transformation>SELECT 'default'             TABLE_CATALOG, a.TABLE_OWNER          TABLE_SCHEMA,
                            a.TABLE_NAME, a.TRIGGER_NAME         NAME, 'vk_public'            VISIBILITY,
                           NULL EVENT_MANIPULATION, NULL ACTION_CONDITION, NULL ACTION_STATEMENT,
                           NULL ACTION_ORIENTATION, NULL CONDITION_TIMING, NULL CONDITION_REFERENCEOLDTABLE,
                           NULL CONDITION_REFERENCENEWTABLE, NULL LANGUAGE
                      FROM DBA_TRIGGERS a
                     WHERE 'default'=? AND a.OWNER=? AND a.TRIGGER_NAME=?
                  ORDER BY a.TABLE_NAME,a.TRIGGER_NAME</transformation>
    <parameter name="catalogName" order="1" type="STRING"/>
    <parameter name="schemaName" order="2" type="STRING"/>
    <parameter name="triggerName" order="3" type="STRING"/>
</rule>
<rule name="GET_TABLES">
    <transformation>   SELECT TABLE_NAME, 'vk_notapplicable'    VISIBILITY, 1 ISABSTRACT, TEMPORARY ISTEMPORARY,
                           NULL TEMPORARYSCOPE, 1 ISSYSTEM
                      FROM DBA_TABLES
                     WHERE 'default'=? AND OWNER=?
                    ORDER BY TABLE_NAME
                     </transformation>
    <parameter name="catalogName" order="1" type="STRING"/>
    <parameter name="schemaName" order="2" type="STRING"/>
</rule>
```

```
        <rule name="GET_TABLE">
            <transformation>SELECT 'vk_notapplicable'    VISIBILITY, 1 ISABSTRACT, TEMPORARY ISTEMPORARY,
                                NULL TEMPORARYSCOPE, 1 ISSYSTEM
                            FROM DBA_TABLES
                           WHERE 'default'=? AND OWNER=? AND TABLE_NAME=?
                            </transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
            <parameter name="schemaName" order="2" type="STRING"/>
            <parameter name="tableName" order="3" type="STRING"/>
        </rule>
        <rule name="GET_VIEWS">
            <transformation>   SELECT VIEW_NAME NAME, 'vk_public'   VISIBILITY, TEXT          QUERYEXPRESSIONBODY,
                                    'SQL' QUERYEXPRESSIONLANGUAGE, CASE READ_ONLY WHEN 'N' THEN 1 ELSE 0 END  ISREADONLY,
                                     1 CHECKOPTION
                               FROM DBA_VIEWS a
                              WHERE 'default'=? AND a.OWNER=?
                           ORDER BY NAME
                        </transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
            <parameter name="schemaName" order="2" type="STRING"/>
        </rule>
        <rule name="GET_VIEW">
            <transformation>   SELECT VIEW_NAME NAME, 'vk_public'   VISIBILITY, TEXT          QUERYEXPRESSIONBODY,
                         'SQL' QUERYEXPRESSIONLANGUAGE, CASE READ_ONLY WHEN 'N' THEN 1 ELSE 0 END  ISREADONLY, 1 CHECKOPTION
                                 FROM DBA_VIEWS a
                                WHERE 'default'=? AND a.OWNER=? AND a.VIEW_NAME=?
                               </transformation>
            <parameter name="catalogName" order="1" type="STRING"/>
            <parameter name="schemaName" order="2" type="STRING"/>
            <parameter name="viewName" order="3" type="STRING"/>
        </rule>
        <rule name="GET_TRIGGERUSEDCOLUMNSET">
        </rule>
                    <rule name="GET_ROWSTABLE">
                    SELECT :COLUMNS FROM :SCHEMANAME.:TABLENAME
                    </rule>
          </rules>
</bridge>
```

Archivo A2.3. Bridge CWM para Oracle 11.2.

ANEXO C.

Ontologías CWM

El anexo contiene la lista de ontologías utilizadas para definir los metadatos modelados en el estándar CWM. Las ontologías definidas son:

- **Core** <http://www.cwm.org/ontologies/cwm-core.owl>
- **Behavioral** <http://www.cwm.org/objectmodel/Behavioral.owl>
- **Instance** <http://www.cwm.org/objectmodel/instance.owl>
- **KeyIndexes** <http://www.cwm.org/foundation/KeysIndexes.owl>
- **DataTypes** <http://www.cwm.org/foundation/DataTypes.owl>
- **Relational** <http://www.cwm.org/resource/relational.owl>

Core

El **Archivo A3.1** contiene las entidades de la ontología que define los metadatos del paquete CWM Core.

```
Prefix: rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
Prefix: dc: <http://purl.org/dc/elements/1.1/>
Prefix: xsd: <http://www.w3.org/2001/XMLSchema#>
Prefix: xml: <http://www.w3.org/XML/1998/namespace>
Prefix: rdfs: <http://www.w3.org/2000/01/rdf-schema#>
Prefix: owl: <http://www.w3.org/2002/07/owl#>


Ontology: <http://www.cwm.org/ontologies/cwm-core.owl>


Annotations:
    rdfs:comment "The Common Warehouse Metamodel Ontology that describes the relational metadata package"@en

AnnotationProperty: rdfs:comment


Datatype: xsd:integer


Datatype: xsd:Name


Datatype: xsd:string


Datatype: xsd:boolean


Datatype: rdf:PlainLiteral


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasConstrainedElement>

    Annotations:
        rdfs:comment "A ModelElement or list of ModelElements affected by the Constraint"

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#Constraint>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>

    InverseOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#hasConstraint>


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasSupplier>

    Annotations:
        rdfs:comment "Inverse of client. Designates the element that is unaffected by a change. In a two-way relationship
this would be the more general element. In an undirected situation the choice of client and supplier may be irrelevant."

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#Dependency>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>
```

```
    InverseOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#hasSupplierDependency>


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasMultiplicy>

    Annotations:
        rdfs:comment "References the Multiplicity instance that owns the MultiplicityRange",
        rdfs:comment "The possible number of data values for the feature that may be held by an instance. The cardianality
of the set of values is an implicit part of the feature. In the common case in which the multiplicity is 1..1, then the
feature is a scalar; that is, it holds exactly one value"

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#StructuralFeature>,
        <http://www.cwm.org/ontologies/cwm-core.owl#MutiplicityRange>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#Multiplicity>

    InverseOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#hasRange>


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasFeature>

    Annotations:
        rdfs:comment "And ordered list of Features owned by the Classifier"

    Characteristics:
        Functional

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#Classifier>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#Feature>

    InverseOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#hasOwner>


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasTaggedValue>

    Annotations:
        rdfs:comment "References the set of TaggedValue instances that extend the ModelElement"

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#TaggedValue>

    InverseOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#hasModelElement>


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasClientDependency>

    Annotations:
        rdfs:comment "Inverse of hasClient. Designates a set of Dependency in which the ModelElement is a client."

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#Dependency>

    InverseOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#hasClient>


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasOrdering>

    Annotations:
        rdfs:comment "Specifies whether the set of instances is ordered. The ordering must be determinated and maintained by
Operations that add values to the feature. This property is only relevant if the multiplicity is greater than one"

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#StructuralFeature>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#OrderingKing>


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasExtendedElement>

    Annotations:
        rdfs:comment "Designates the model elements affected by the stereotype. Each one must be a model element of the kind
specified by the baseClass attribute."

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#Stereotype>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasRequiredTag>

    Annotations:
        rdfs:comment "Specifies a set of TaggedValues, each of which specifies a tag that an element classified by the
Sterotype is required to have. The value part indicates the default value for the tagged value, that is, the tagged value
```

```
that is, the tagged value that an element will be presumed to have if it is not overridden by an explicit tagged value on
the element bearing the sterotype. If the value is unspecified, then the element must explicit tagged value on the element
bearing the stereotype. If the value is unspecified, then the element must explicity specify a tagged value with the given
tag."

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#Stereotype>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#TaggedValue>


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasSupplierDependency>

    Annotations:
        rdfs:comment "Inverse of hasSupplier. Designates a set of Dependency in which the ModelElement is a client"

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#Dependency>

    InverseOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#hasSupplier>


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasVisibility>

    Annotations:
        rdfs:comment "Specifies extent of the visiblity of the ModelElement within its owning Namespace"

    Characteristics:
        Functional

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#VisibilityKind>


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasOwnedElement>

    Annotations:
        rdfs:comment "A set of ModelElements owned by the Namespace. The ModelElement's visiblity attribute states whether
the element is visible outside the namespace"

    Characteristics:
        InverseFunctional

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#Namespace>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>

    InverseOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#hasNamespace>


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasStereotype>

    Annotations:
        rdfs:comment "References a Stereotype that uses the TaggedValue"

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#TaggedValue>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#Stereotype>


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasNamespace>

    Annotations:
        rdfs:comment "Designates the Namespace that containts the ModelElement. Every ModelElement except a root element
must belong to exactly one Namespace or else be a composite part of another ModelElement (which is a kind of virtual
namespace). The pathname of Namespace or ModelElement names starting from the root package provides a unique designation for
every ModelElement. The association attribute visibility specifies the visibility specifies the visibility of the element
outside its namespace."

    Characteristics:
        Functional

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#Namespace>

    InverseOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#hasOwnedElement>


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasConstraint>

    Annotations:
        rdfs:comment "A set of Constraints affecting the element. A constraint that must be satisfied by the model element.
A model element may have a set of constraints. The constraint is to be evaluated when the system is stable; that is, not in
the middle of an atomic operation"

    Domain:
```

```
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#Constraint>

    InverseOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#hasConstrainedElement>


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasImportedElement>

    Annotations:
        rdfs:comment "The namespace defined by the package is extended by model elements imported from other packages"

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#Package>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>

    InverseOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#isImporter>


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasTargetScope>

    Annotations:
        rdfs:comment "Specifies whether the targets are ordinary Instances or are Classifiers"

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#StructuralFeature>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#ScopeKind>


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasOwnerScope>

    Annotations:
        rdfs:comment "Specifies whether the Feature appears in every instance of the Classifier or whether it appears only
once for the entire Classifier"

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#Feature>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#ScopeKind>


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasBooleanExpression>

    Annotations:
        rdfs:comment "A BooleanExpression that must be true when evaluated for an instance of a system to be well formed. A
boolean expression defining the constraint. Expressions are written as strings in a designated language. For the model to be
well formed, the expression must always yield a true"

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#Constraint>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#BooleanExpression>


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasModelElement>

    Annotations:
        rdfs:comment "References the ModelElement to which the TaggedValue pertains"

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#TaggedValue>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>

    InverseOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#hasTaggedValue>


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasChangeability>

    Annotations:
        rdfs:comment "Specifieswhether the value may be modified after the object is created"

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#StructuralFeature>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#ChangeableKind>


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasClient>

    Annotations:
        rdfs:comment "The element that is affected by the supplier element. In some cases the direction is unimportant and
serves only to distinguish the two elements."

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#Dependency>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>

    InverseOf:
```

```
        <http://www.cwm.org/ontologies/cwm-core.owl#hasClientDependency>


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasStereotypeConstraint>

    Annotations:
        rdfs:comment "Designates constraints that apply to all model elements branded by this stereotype. These constraints
are defined in the scope of the full metamodel."

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#Stereotype>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#Constraint>


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasOwner>

    Annotations:
        rdfs:comment "A feature is a property, like attribute or operation that is encapsulated within a Classifier.
In the metamodel a Feature declares a structural or behavioral characteristic of an instance of a Classifier or of the
Classifier itself. Feature is an abstract metaclass"

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#Feature>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#Classifier>

    InverseOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#hasFeature>


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#isImporter>

    Annotations:
        rdfs:comment "References the set of Package instances that import the ModelElement"

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#Package>

    InverseOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#hasImportedElement>


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasType>

    Annotations:
        rdfs:comment "Designates the Classifier whose instances are values of the feature. It must be a Class, DataType, or
Interface."

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#StructuralFeature>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#Classifier>


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasInitialValue>

    Annotations:
        rdfs:comment "An Expression specifying the value of the attribute upon initialization. It is meant to be evaluated
at the time the  object is initialized."

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#Attribute>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#Expression>


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasRange>

    Annotations:
        rdfs:comment "References the set of MultiplicityRange instances that describe the cardinality of the Multiplicity
instance."

    Characteristics:
        Functional

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#Multiplicity>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#MutiplicityRange>

    InverseOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#hasMultiplicy>


DataProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasName>

    Annotations:
        rdfs:comment "An identifier for the ModelElement within its containing Namespace"

    Characteristics:
        Functional

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>
```

```
    Range:
        xsd:Name


DataProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasLower>

    Annotations:
        rdfs:comment "Specifies the positive integer lower bound of the range"

    Characteristics:
        Functional

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#MutiplicityRange>

    Range:
        xsd:integer


DataProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasLanguage>

    Annotations:
        rdfs:comment "Names the language in which the expression body is represented. The interpretation of the expression
depends on the language. If the langugage name is omitted, no interpretation for the expression can be assumed. In general,
a language name should be spelled and capitalized exactly as it appears in the document defining the language."

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#Expression>

    Range:
        xsd:Name


DataProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasKind>

    Annotations:
        rdfs:comment "Contains a description of the nature of the dependency relationship between the client and supplier.
The list of possible values is open-ended. However, CWM predefines the values \"Abstraction\" and \"Usage\"."

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#Dependency>

    Range:
        xsd:string


DataProperty: <http://www.cwm.org/ontologies/cwm-core.owl#isAbstract>

    Annotations:
        rdfs:comment "An abstract Classifier is not instantiable"

    Characteristics:
        Functional

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#Classifier>

    Range:
        xsd:boolean


DataProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasBody>

    Annotations:
        rdfs:comment "The text of the expression expressed in the given language"

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#Expression>

    Range:
        xsd:string


DataProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasBaseClass>

    Annotations:
        rdfs:comment "Specifies the name of a modeling element to which the stereotype applies, such as Class, Association,
Constraint, etc. This is the name of a metaclass; that is, a class from the metamodel itself rather than a user model
class."

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#Stereotype>

    Range:
        xsd:Name


DataProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasTag>

    Annotations:
        rdfs:comment "Contains the name of the TaggedValue. This name determines the semantics that are applicable to the
contents of the value attribute"

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#TaggedValue>

    Range:
        xsd:Name


DataProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasUpper>
```

```
    Characteristics:
        Functional

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#MutiplicityRange>

    Range:
        xsd:integer


DataProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasValue>

    Annotations:
        rdfs:comment "Contains the current value of the TaggedValue"

    Domain:
        <http://www.cwm.org/ontologies/cwm-core.owl#TaggedValue>

    Range:
        xsd:string


Class: <http://www.cwm.org/ontologies/cwm-core.owl#VisibilityKind>

    Annotations:
        rdfs:comment "In the metamodel VisibilityKind defines an enumeration that denotes how the element to which it seen
outside the enclosing name space. "

    EquivalentTo:
        {<http://www.cwm.org/ontologies/cwm-core.owl#VK_NOTAPPLICABLE> , <http://www.cwm.org/ontologies/cwm-
core.owl#VK_PACKAGE> , <http://www.cwm.org/ontologies/cwm-core.owl#VK_PRIVATE> , <http://www.cwm.org/ontologies/cwm-
core.owl#VK_PROTECTED> , <http://www.cwm.org/ontologies/cwm-core.owl#VK_PUBLIC>}

    SubClassOf:
        owl:Thing


Class: <http://www.cwm.org/ontologies/cwm-core.owl#Feature>

    Annotations:
        rdfs:comment "A feature is a property, like attribute or operation that is encapsulated within a Classifier.
In the metamodel a Feature declares a structural or behavioral characteristic of an instance of a Classifier or of the
Classifier itself. Feature is an abstract metaclass."

    SubClassOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#hasOwner> some <http://www.cwm.org/ontologies/cwm-core.owl#Classifier>,
        <http://www.cwm.org/ontologies/cwm-core.owl#hasOwnerScope> max 1 <http://www.cwm.org/ontologies/cwm-
core.owl#ScopeKind>,
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#Classifier>

    SubClassOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#hasFeature> some <http://www.cwm.org/ontologies/cwm-core.owl#Feature>,
        <http://www.cwm.org/ontologies/cwm-core.owl#Namespace>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#ProcedureExpression>

    Annotations:
        rdfs:comment "In the metamodel ProcedureExpression defines a statement that will result in a change to the values of
its environment when it is evaluated."

    SubClassOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#Expression>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#Attribute>

    Annotations:
        rdfs:comment "An attribute describes a named slot within a Classifier that may hold a value"

    SubClassOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#hasInitialValue> max 1 <http://www.cwm.org/ontologies/cwm-
core.owl#Expression>,
        <http://www.cwm.org/ontologies/cwm-core.owl#StructuralFeature>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#Namespace>

    Annotations:
        rdfs:comment "In the metamodel, a Namespace is a ModelElement that can own other ModelElements, such as Classifiers.
The name of each owned ModelElement must be unique within the Namespace. Moreover, each contained ModelElement is owned by
at most one Namespace. The concrete subclasses of Namespace may have additional constraints on which kind of elements may be
contained.
Namesapace is an abstract metaclasss.
Note htat explicit parts of a model element, such as the features of a Classifier, are not modeled as owned elements in a
namespace. A namespace is used for unstructured contents such as the contents of a package, or a class declared inside the
scope of another class",
        rdfs:comment "A namespace is a part of a model that contains a set of ModelElements each of whose names designates a
unique element within the namespace"

    SubClassOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#hasOwnedElement> max 1 <http://www.cwm.org/ontologies/cwm-
core.owl#ModelElement>,
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#OrderingKing>

    Annotations:
        rdfs:comment "In the metadamodel OrderingKind defines an enumeration that specifies how the elements of a set are
```

```
arranged. Used in conjunction with elements that have a multiplicity in cases when the multiplicity value is greater than
one. The ordering must be determined and maintained by operations that modify the set."

    EquivalentTo:
        {<http://www.cwm.org/ontologies/cwm-core.owl#OK_ORDERED> , <http://www.cwm.org/ontologies/cwm-
core.owl#OK_UNORDERED>}

    SubClassOf:
        owl:Thing


Class: <http://www.cwm.org/ontologies/cwm-core.owl#Package>

    Annotations:
        rdfs:comment "A package is a grouping of model elements"

    SubClassOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#hasImportedElement> some <http://www.cwm.org/ontologies/cwm-
core.owl#ModelElement>,
        <http://www.cwm.org/ontologies/cwm-core.owl#Namespace>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#Model>

    SubClassOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#Package>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#Constraint>

    Annotations:
        rdfs:comment "A constraint is a semantic condition or restriction expressed in text."

    SubClassOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#hasConstraint> exactly 0 <http://www.cwm.org/ontologies/cwm-
core.owl#Constraint>,
        <http://www.cwm.org/ontologies/cwm-core.owl#hasBooleanExpression> exactly 1 <http://www.cwm.org/ontologies/cwm-
core.owl#BooleanExpression>,
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#Dependency>

    Annotations:
        rdfs:comment "A dependency states that the implementation or functioning of one or more elements requires the
presence of one or more other elements."

    SubClassOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#hasKind> max 1 xsd:string,
        <http://www.cwm.org/ontologies/cwm-core.owl#hasSupplier> min 1 <http://www.cwm.org/ontologies/cwm-
core.owl#ModelElement>,
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>,
        <http://www.cwm.org/ontologies/cwm-core.owl#hasClient> min 1 <http://www.cwm.org/ontologies/cwm-
core.owl#ModelElement>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#MutiplicityRange>

    Annotations:
        rdfs:comment "In the metamodel a MultiplicityRange defines a range of integers. The upper bound of the range cannot
be below the lower bound. The lower bound must be a nonnegative integer. The upper bound must be a nonnegative integer or
the special value unlimited, which indicates ther is no upper bound on the range"

    SubClassOf:
        owl:Thing,
        <http://www.cwm.org/ontologies/cwm-core.owl#hasUpper> exactly 1 xsd:integer,
        <http://www.cwm.org/ontologies/cwm-core.owl#hasMultiplicy> exactly 1 <http://www.cwm.org/ontologies/cwm-
core.owl#Multiplicity>,
        <http://www.cwm.org/ontologies/cwm-core.owl#hasLower> exactly 1 xsd:integer


Class: <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>

    Annotations:
        rdfs:comment "A model element is an element that is an abstraction drawn from the system being modeled.
In the metamodel a ModelElement is a named entity in a Model. It is the base for all modeling metaclasses in the CWM. All
other metaclasses are either direct or indirect subclasses of ModelElement."

    SubClassOf:
        owl:Thing,
        <http://www.cwm.org/ontologies/cwm-core.owl#hasVisibility> exactly 1 <http://www.cwm.org/ontologies/cwm-
core.owl#VisibilityKind>,
        <http://www.cwm.org/ontologies/cwm-core.owl#hasName> exactly 1 xsd:Name,
        <http://www.cwm.org/ontologies/cwm-core.owl#hasTaggedValue> some <http://www.cwm.org/ontologies/cwm-
core.owl#TaggedValue>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#ChangeableKind>

    Annotations:
        rdfs:comment "In the metamodel ChangeableKind defines an enumeration that denotes how an attribute link may be
modified"

    EquivalentTo:
        {<http://www.cwm.org/ontologies/cwm-core.owl#CK_ADDONLY> , <http://www.cwm.org/ontologies/cwm-
core.owl#CK_CHANGEABLE> , <http://www.cwm.org/ontologies/cwm-core.owl#CK_FROZEN>}


Class: <http://www.cwm.org/ontologies/cwm-core.owl#DataType>

    Annotations:
        rdfs:comment "A data type is a type whose values have no identity; that is, they are pure values. Data types include
primitive built-in types (such as integer and string) as well as definable enumeration types."
```

```
    SubClassOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#Classifier>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#Subsystem>

    Annotations:
        rdfs:comment "A subsystem is a grouping of model elements that represents a behavioral unit in a physical system. A
subsystem offers interfaces and has operations."

    SubClassOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#Package>,
        <http://www.cwm.org/ontologies/cwm-core.owl#Classifier>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#ScopeKind>

    Annotations:
        rdfs:comment "In the metamodel ScopeKind defines an enumeration that denotes whether a feature belongs to individual
instances or an entire classifier."

    EquivalentTo:
        {<http://www.cwm.org/ontologies/cwm-core.owl#SK_CLASSIFIER> , <http://www.cwm.org/ontologies/cwm-
core.owl#SK_INSTANCE>}

    SubClassOf:
        owl:Thing


Class: <http://www.cwm.org/ontologies/cwm-core.owl#Multiplicity>

    Annotations:
        rdfs:comment "In the metamodel a Multiplicity defines a non-empty set of non-negative integers. A set that only
contains zero ({0}) is not considered a valid Multiplicity. Every Multiplicity has at least one corresponding String
representation."

    SubClassOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#hasRange> min 1 <http://www.cwm.org/ontologies/cwm-
core.owl#MutiplicityRange>,
        owl:Thing


Class: owl:Thing

    Annotations:
        rdfs:comment "An element is an atomic constituent of a model. In the metamodel, an Element is the top metaclass in
the metaclass hierarchy."


Class: <http://www.cwm.org/ontologies/cwm-core.owl#BooleanExpression>

    Annotations:
        rdfs:comment "In the metamodel BooleanExpression defines a statement that will evaluate to an instance of Boolean
when it is evaluated"

    SubClassOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#Expression>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#Expression>

    Annotations:
        rdfs:comment "In the metamodel an Expression defines a statement that will evaluate to a (possibly empty) set of
instances when executed in a context. An expression does not modify the environment in which it is evaluated. An expression
contains an expression string and the name of an interpretation language with which to evaluate the string"

    SubClassOf:
        owl:Thing,
        <http://www.cwm.org/ontologies/cwm-core.owl#hasBody> exactly 1 xsd:string,
        <http://www.cwm.org/ontologies/cwm-core.owl#hasLanguage> max 1 xsd:Name


Class: <http://www.cwm.org/ontologies/cwm-core.owl#Stereotype>

    Annotations:
        rdfs:comment "The stereotype concept provides a way of branding (classifying) model elements so that they behave as
if they were instances of new virtual metamodel constructs. These model elements have the same structure (attributes,
associations, operations )  as similar non-stereotyped model elements of the same kind. The stereotype may specify
additional constraints and required tagged values that apply to model elements. In addition, a stereotype may be used to
indicate a difference in meaning or usage between two model elements whit identical structure."

    SubClassOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#hasStereotypeConstraint> some <http://www.cwm.org/ontologies/cwm-
core.owl#Constraint>,
        <http://www.cwm.org/ontologies/cwm-core.owl#hasRequiredTag> some <http://www.cwm.org/ontologies/cwm-
core.owl#TaggedValue>,
        <http://www.cwm.org/ontologies/cwm-core.owl#hasBaseClass> exactly 1 xsd:Name,
        <http://www.cwm.org/ontologies/cwm-core.owl#hasExtendedElement> some <http://www.cwm.org/ontologies/cwm-
core.owl#ModelElement>,
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#StructuralFeature>

    Annotations:
        rdfs:comment "A structural feature refers to a static feature of a model element.
In the metamodel, a StructuralFeature declares a structural aspect of a Classifier that is typed, such as an attribute. For
example, it specifies the multiplicity and changeability of the StructuralFeature. StructuralFeature is an abstract
metaclass"

    SubClassOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#hasMultiplicy> max 1 <http://www.cwm.org/ontologies/cwm-
```

```
core.owl#Multiplicity>,
        <http://www.cwm.org/ontologies/cwm-core.owl#hasOrdering> max 1 <http://www.cwm.org/ontologies/cwm-
core.owl#OrderingKing>,
        <http://www.cwm.org/ontologies/cwm-core.owl#Feature>,
        <http://www.cwm.org/ontologies/cwm-core.owl#hasTargetScope> max 1 <http://www.cwm.org/ontologies/cwm-
core.owl#ScopeKind>,
        <http://www.cwm.org/ontologies/cwm-core.owl#hasChangeability> exactly 1 <http://www.cwm.org/ontologies/cwm-
core.owl#ChangeableKind>,
        <http://www.cwm.org/ontologies/cwm-core.owl#hasType> exactly 1 <http://www.cwm.org/ontologies/cwm-
core.owl#Classifier>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#CoreClass>

    Annotations:
        rdfs:comment "A class is a description of a set of objects that share the same attributes, operations, methods ,
relationships and semantics. A class may use a set of interfaces to specify collections of operations it provides to its
environment. "

    SubClassOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#Classifier>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#TaggedValue>

    Annotations:
        rdfs:comment "A tagged value allows information to be attached to any model element in the form of a \"tagged
value\" pair; that is, name=value. The interpreation of tagged value semantics is intentionally beyond the scope of CWM. It
must be determined by user or tool conventions. It is expected that tools will define tags to supply information needed for
their operations beyond the basic semantics of CWM. Such information could include code generation options, model management
information, or user-specified semantics."

    SubClassOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#hasTag> exactly 1 xsd:Name,
        owl:Thing,
        <http://www.cwm.org/ontologies/cwm-core.owl#hasModelElement> max 1 <http://www.cwm.org/ontologies/cwm-
core.owl#ModelElement>,
        <http://www.cwm.org/ontologies/cwm-core.owl#hasStereotype> max 1 <http://www.cwm.org/ontologies/cwm-
core.owl#Stereotype>,
        <http://www.cwm.org/ontologies/cwm-core.owl#hasValue> exactly 1 xsd:string


Individual: <http://www.cwm.org/ontologies/cwm-core.owl#OK_UNORDERED>

    Annotations:
        rdfs:comment "The elements of the set have no inherent ordering"

    Types:
        <http://www.cwm.org/ontologies/cwm-core.owl#OrderingKing>


Individual: <http://www.cwm.org/ontologies/cwm-core.owl#VK_PACKAGE>

    Annotations:
        rdfs:comment "Elements declared in the same package as the target element may see and use the target element"

    Types:
        <http://www.cwm.org/ontologies/cwm-core.owl#VisibilityKind>


Individual: <http://www.cwm.org/ontologies/cwm-core.owl#SK_CLASSIFIER>

    Annotations:
        rdfs:comment "A classifier is an element that describes structural and behavioral features; it comes in several
specific forms, including class, idata type, interface, component, and others that are defined in other metamodel packages.
Classifier is often used as type."

    Types:
        <http://www.cwm.org/ontologies/cwm-core.owl#ScopeKind>


Individual: <http://www.cwm.org/ontologies/cwm-core.owl#CK_ADDONLY>

    Annotations:
        rdfs:comment "If the multiplicity is not fixed, values may be added at any time from the source object, but once
created a value may not be removed from the source end. Operations on the other end may change a value"

    Types:
        <http://www.cwm.org/ontologies/cwm-core.owl#ChangeableKind>


Individual: <http://www.cwm.org/ontologies/cwm-core.owl#VK_NOTAPPLICABLE>

    Annotations:
        rdfs:comment "May be used where namespaces do not support the concept of visiblity"

    Types:
        <http://www.cwm.org/ontologies/cwm-core.owl#VisibilityKind>


Individual: <http://www.cwm.org/ontologies/cwm-core.owl#CK_CHANGEABLE>

    Annotations:
        rdfs:comment "No restrictions on modification"

    Types:
        <http://www.cwm.org/ontologies/cwm-core.owl#ChangeableKind>


Individual: <http://www.cwm.org/ontologies/cwm-core.owl#SK_INSTANCE>

    Annotations:
        rdfs:comment "The feature pertains to instances of a Classifier. For example, it is a distinct attribute in each
```

```
instance or an operation that works on an instance."

    Types:
        <http://www.cwm.org/ontologies/cwm-core.owl#ScopeKind>


Individual: <http://www.cwm.org/ontologies/cwm-core.owl#VK_PUBLIC>

    Annotations:
        rdfs:comment "Other elements may see and use the target element"

    Types:
        <http://www.cwm.org/ontologies/cwm-core.owl#VisibilityKind>


Individual: <http://www.cwm.org/ontologies/cwm-core.owl#VK_PRIVATE>

    Annotations:
        rdfs:comment "Only the source element may see and use the target element"

    Types:
        <http://www.cwm.org/ontologies/cwm-core.owl#VisibilityKind>


Individual: <http://www.cwm.org/ontologies/cwm-core.owl#OK_ORDERED>

    Annotations:
        rdfs:comment "The elements of the set have a sequential ordering"

    Types:
        <http://www.cwm.org/ontologies/cwm-core.owl#OrderingKing>


Individual: <http://www.cwm.org/ontologies/cwm-core.owl#VK_PROTECTED>

    Annotations:
        rdfs:comment "Descendants of the source element may see and use the target element"

    Types:
        <http://www.cwm.org/ontologies/cwm-core.owl#VisibilityKind>


Individual: <http://www.cwm.org/ontologies/cwm-core.owl#CK_FROZEN>

    Annotations:
        rdfs:comment "The value may not be changed from the source end after the creation and initialization of the source
object. Operations on the other end may change a value"

    Types:
        <http://www.cwm.org/ontologies/cwm-core.owl#ChangeableKind>
```

Archivo A3.1. Ontología del paquete Core CWM.

Behavioral

El **Archivo A3.2** contiene las entidades de la ontología que define los metadatos del paquete CWM Behavioral.

```
Prefix: rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
Prefix: xsd: <http://www.w3.org/2001/XMLSchema#>
Prefix: xml: <http://www.w3.org/XML/1998/namespace>
Prefix: rdfs: <http://www.w3.org/2000/01/rdf-schema#>
Prefix: owl: <http://www.w3.org/2002/07/owl#>



Ontology: <http://www.cwm.org/objectmodel/Behavioral.owl>

Import: <http://www.cwm.org/ontologies/cwm-core.owl>

AnnotationProperty: rdfs:comment


Datatype: xsd:boolean


Datatype: rdf:PlainLiteral


ObjectProperty: <http://www.cwm.org/objectmodel/Behavioral.owl#hasEvent>

    Annotations:
        rdfs:comment "References the Event instance for which the Parameter instance describes a parameter"

    Domain:
        <http://www.cwm.org/objectmodel/Behavioral.owl#Parameter>

    Range:
        <http://www.cwm.org/objectmodel/Behavioral.owl#Event>

    InverseOf:
        <http://www.cwm.org/objectmodel/Behavioral.owl#hasParameter>


ObjectProperty: <http://www.cwm.org/objectmodel/Behavioral.owl#hasActualArgument>

    Annotations:
        rdfs:comment "The Argument(s) supplied to the CallAction"
```

```
    Domain:
        <http://www.cwm.org/objectmodel/Behavioral.owl#CallAction>

    Range:
        <http://www.cwm.org/objectmodel/Behavioral.owl#Argument>

    InverseOf:
        <http://www.cwm.org/objectmodel/Behavioral.owl#hasCallAction>


ObjectProperty: <http://www.cwm.org/objectmodel/Behavioral.owl#hasBodyExpression>

    Annotations:
        rdfs:comment "A specification of the Method in some appropriate form (sush as a programming language). The exact
form of a Method's specification an knowledge of the language in which it is described is outside the scope of the CWM"

    Domain:
        <http://www.cwm.org/objectmodel/Behavioral.owl#Method>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#ProcedureExpression>


ObjectProperty: <http://www.cwm.org/objectmodel/Behavioral.owl#hasDefaultValue>

    Annotations:
        rdfs:comment "An Expression whose evaluation yields a value to be used when no argument is supplied for the
Parameter"

    Domain:
        <http://www.cwm.org/objectmodel/Behavioral.owl#Parameter>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#Expression>


ObjectProperty: <http://www.cwm.org/objectmodel/Behavioral.owl#hasSpecification>

    Annotations:
        rdfs:comment "References the Operation that the Method implements"

    Domain:
        <http://www.cwm.org/objectmodel/Behavioral.owl#Method>

    Range:
        <http://www.cwm.org/objectmodel/Behavioral.owl#Operation>

    InverseOf:
        <http://www.cwm.org/objectmodel/Behavioral.owl#hasMethod>


ObjectProperty: <http://www.cwm.org/objectmodel/Behavioral.owl#hasParameterDirectionKind>

    Annotations:
        rdfs:comment "Specifieds what kind of a Parameter is required"

    Domain:
        <http://www.cwm.org/objectmodel/Behavioral.owl#Parameter>

    Range:
        <http://www.cwm.org/objectmodel/Behavioral.owl#ParameterDirectionKind>


ObjectProperty: <http://www.cwm.org/objectmodel/Behavioral.owl#hasValueExpression>

    Annotations:
        rdfs:comment "An expression determining the actual Argument instance when executed"

    Domain:
        <http://www.cwm.org/objectmodel/Behavioral.owl#Argument>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#Expression>


ObjectProperty: <http://www.cwm.org/objectmodel/Behavioral.owl#hasCallAction>

    Annotations:
        rdfs:comment "Indentifies the CallAction that uses the Argument"

    Domain:
        <http://www.cwm.org/objectmodel/Behavioral.owl#Argument>

    Range:
        <http://www.cwm.org/objectmodel/Behavioral.owl#CallAction>

    InverseOf:
        <http://www.cwm.org/objectmodel/Behavioral.owl#hasActualArgument>


ObjectProperty: <http://www.cwm.org/objectmodel/Behavioral.owl#hasParameter>

    Annotations:
        rdfs:comment "An ordered list of Parameters for the BehavioralFeature. To call the BehavioralFeature, the caller
must supply a list of values comnpatible with the types of the Parameters."

    Domain:
        <http://www.cwm.org/objectmodel/Behavioral.owl#BehavioralFeature>,
        <http://www.cwm.org/objectmodel/Behavioral.owl#Event>

    Range:
        <http://www.cwm.org/objectmodel/Behavioral.owl#Parameter>
```

```
    InverseOf:
        <http://www.cwm.org/objectmodel/Behavioral.owl#hasBehavioralFeature>,
        <http://www.cwm.org/objectmodel/Behavioral.owl#hasEvent>


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasType>

    Domain:
        <http://www.cwm.org/objectmodel/Behavioral.owl#Parameter>


ObjectProperty: <http://www.cwm.org/objectmodel/Behavioral.owl#hasBehavioralFeature>

    Annotations:
        rdfs:comment "References the BehavioralFeature instance for which the Parameter instance describes a parameter"

    Domain:
        <http://www.cwm.org/objectmodel/Behavioral.owl#Parameter>

    Range:
        <http://www.cwm.org/objectmodel/Behavioral.owl#BehavioralFeature>

    InverseOf:
        <http://www.cwm.org/objectmodel/Behavioral.owl#hasParameter>


ObjectProperty: <http://www.cwm.org/objectmodel/Behavioral.owl#hasOperation>

    Annotations:
        rdfs:comment "The operation that will be invoked when the CallAction is executed"

    Domain:
        <http://www.cwm.org/objectmodel/Behavioral.owl#CallAction>

    Range:
        <http://www.cwm.org/objectmodel/Behavioral.owl#Operation>


ObjectProperty: <http://www.cwm.org/objectmodel/Behavioral.owl#hasMethod>

    Annotations:
        rdfs:comment "References the set of Method instances defined for the Operation"

    Domain:
        <http://www.cwm.org/objectmodel/Behavioral.owl#Operation>

    Range:
        <http://www.cwm.org/objectmodel/Behavioral.owl#Method>

    InverseOf:
        <http://www.cwm.org/objectmodel/Behavioral.owl#hasSpecification>


DataProperty: <http://www.cwm.org/objectmodel/Behavioral.owl#isQuery>

    Annotations:
        rdfs:comment "Specifies whether an execution of the BehavioralFeature leaves the state of the system unchanged. True
indicates that the state is unchanged; false indicates that side-effects may occur"

    Domain:
        <http://www.cwm.org/objectmodel/Behavioral.owl#BehavioralFeature>

    Range:
        xsd:boolean


DataProperty: <http://www.cwm.org/ontologies/cwm-core.owl#isAbstract>

    Annotations:
        rdfs:comment "If true, then the Operation does not have an implementation, and one must be supplied by a descendant.
If false, the Operation must have an implementation in the class or inherited from an ancestor"

    Domain:
        <http://www.cwm.org/objectmodel/Behavioral.owl#Operation>


Class: <http://www.cwm.org/objectmodel/Behavioral.owl#ParameterDirectionKind>

    Annotations:
        rdfs:comment "In the metamodel PrameterDirectionKind defines an enmuerations that denotes if a Parameter is used for
supplying an argument and/or for returning a value."

    EquivalentTo:
        {<http://www.cwm.org/objectmodel/Behavioral.owl#PDK_IN> , <http://www.cwm.org/objectmodel/Behavioral.owl#PDK_INOUT>
, <http://www.cwm.org/objectmodel/Behavioral.owl#PDK_OUT> , <http://www.cwm.org/objectmodel/Behavioral.owl#PDK_RETURN>}


Class: <http://www.cwm.org/objectmodel/Behavioral.owl#BehavioralFeature>

    Annotations:
        rdfs:comment "A behavioral feature refers to a dynamic feature of a model element, such as an operation or method.
In the metamodel BehavioralFeature specifies a behavioral aspect of a Classifier. All different kinds of behavioral aspects
of a Classifier, such as Operation and Method, are subclasses of BehavioralFeature."

    SubClassOf:
        <http://www.cwm.org/objectmodel/Behavioral.owl#hasParameter> some
<http://www.cwm.org/objectmodel/Behavioral.owl#Parameter>,
        <http://www.cwm.org/ontologies/cwm-core.owl#Feature>,
        <http://www.cwm.org/objectmodel/Behavioral.owl#isQuery> exactly 1 xsd:boolean


Class: <http://www.cwm.org/objectmodel/Behavioral.owl#Parameter>
```

```
    Annotations:
        rdfs:comment "Parameters are used in the specification of operations, methods, and events. A Parameter may include a
name, type and direction of communication"

    SubClassOf:
        <http://www.cwm.org/objectmodel/Behavioral.owl#hasParameterDirectionKind> max 1
<http://www.cwm.org/objectmodel/Behavioral.owl#ParameterDirectionKind>,
        <http://www.cwm.org/objectmodel/Behavioral.owl#hasDefaultValue> max 1 <http://www.cwm.org/ontologies/cwm-
core.owl#Expression>,
        <http://www.cwm.org/objectmodel/Behavioral.owl#hasBehavioralFeature> max 1
<http://www.cwm.org/objectmodel/Behavioral.owl#BehavioralFeature>,
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>,
        <http://www.cwm.org/ontologies/cwm-core.owl#hasType> exactly 1 <http://www.cwm.org/ontologies/cwm-
core.owl#Classifier>,
        <http://www.cwm.org/objectmodel/Behavioral.owl#hasEvent> max 1 <http://www.cwm.org/objectmodel/Behavioral.owl#Event>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#Feature>


Class: <http://www.cwm.org/objectmodel/Behavioral.owl#CallAction>

    Annotations:
        rdfs:comment "A call action is an action resulting in an invocation of an operation.
The purpose of a CallAction is to identify the actual Arguments used in a specific invocation of an Operation"

    SubClassOf:
        <http://www.cwm.org/objectmodel/Behavioral.owl#hasOperation> exactly 1
<http://www.cwm.org/objectmodel/Behavioral.owl#Operation>,
        <http://www.cwm.org/objectmodel/Behavioral.owl#hasActualArgument> some
<http://www.cwm.org/objectmodel/Behavioral.owl#Argument>,
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#ProcedureExpression>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#Classifier>


Class: <http://www.cwm.org/objectmodel/Behavioral.owl#Argument>

    Annotations:
        rdfs:comment "Argument is an expression describing how to determine an actual value passed in a CallAction"

    SubClassOf:
        <http://www.cwm.org/objectmodel/Behavioral.owl#hasValueExpression> exactly 1 <http://www.cwm.org/ontologies/cwm-
core.owl#Expression>,
        <http://www.cwm.org/objectmodel/Behavioral.owl#hasCallAction> max 1
<http://www.cwm.org/objectmodel/Behavioral.owl#CallAction>,
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>


Class: <http://www.cwm.org/objectmodel/Behavioral.owl#Event>

    Annotations:
        rdfs:comment "Event is a specification of an observable occurrence. The occurrence that generates an event instance
is assumed to take place at an instant in time."

    SubClassOf:
        <http://www.cwm.org/objectmodel/Behavioral.owl#hasParameter> some
<http://www.cwm.org/objectmodel/Behavioral.owl#Parameter>,
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#Expression>


Class: <http://www.cwm.org/objectmodel/Behavioral.owl#Operation>

    Annotations:
        rdfs:comment "Operation is a service that can be requested from an object to effect behavior. An Operation has a
signature, which describes the parameters that are possible (including possible return values)."

    SubClassOf:
        <http://www.cwm.org/objectmodel/Behavioral.owl#BehavioralFeature>,
        <http://www.cwm.org/ontologies/cwm-core.owl#isAbstract> exactly 1 xsd:boolean,
        <http://www.cwm.org/objectmodel/Behavioral.owl#hasMethod> some
<http://www.cwm.org/objectmodel/Behavioral.owl#Operation>


Class: <http://www.cwm.org/objectmodel/Behavioral.owl#Interface>

    Annotations:
        rdfs:comment "Interface is a named set of operations that specify the behavior of an element."

    SubClassOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#Classifier>


Class: <http://www.cwm.org/objectmodel/Behavioral.owl#Method>

    Annotations:
        rdfs:comment "Method is the implementation of an Operation. It specifies the algorithm or procedure that effects the
results of an Operation"

    SubClassOf:
        <http://www.cwm.org/objectmodel/Behavioral.owl#BehavioralFeature>,
        <http://www.cwm.org/objectmodel/Behavioral.owl#hasBodyExpression> exactly 1 <http://www.cwm.org/ontologies/cwm-
core.owl#ProcedureExpression>,
        <http://www.cwm.org/objectmodel/Behavioral.owl#hasSpecification> exactly 1
<http://www.cwm.org/objectmodel/Behavioral.owl#Operation>
```

```
Class: <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>


Individual: <http://www.cwm.org/objectmodel/Behavioral.owl#PDK_RETURN>

    Annotations:
        rdfs:comment "A return value of a call"

    Types:
        <http://www.cwm.org/objectmodel/Behavioral.owl#ParameterDirectionKind>


Individual: <http://www.cwm.org/objectmodel/Behavioral.owl#PDK_OUT>

    Annotations:
        rdfs:comment "An output Parameter (may be modified to communicate information to the caller)"

    Types:
        <http://www.cwm.org/objectmodel/Behavioral.owl#ParameterDirectionKind>


Individual: <http://www.cwm.org/objectmodel/Behavioral.owl#PDK_INOUT>

    Annotations:
        rdfs:comment "An input Parameter that may be modified"

    Types:
        <http://www.cwm.org/objectmodel/Behavioral.owl#ParameterDirectionKind>


Individual: <http://www.cwm.org/objectmodel/Behavioral.owl#PDK_IN>

    Annotations:
        rdfs:comment "An input Parameter (may not be modified)"

    Types:
        <http://www.cwm.org/objectmodel/Behavioral.owl#ParameterDirectionKind>
```

Archivo A3.2. Ontología del paquete Behavioral CWM.

Instance

El **Archivo A3.3** contiene las entidades de la ontología que define los metadatos del paquete CWM Instance.

```
Prefix: rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
Prefix: dc: <http://purl.org/dc/elements/1.1/>
Prefix: xsd: <http://www.w3.org/2001/XMLSchema#>
Prefix: xml: <http://www.w3.org/XML/1998/namespace>
Prefix: rdfs: <http://www.w3.org/2000/01/rdf-schema#>
Prefix: owl: <http://www.w3.org/2002/07/owl#>



Ontology: <http://www.cwm.org/objectmodel/instance.owl>

Import: <http://www.cwm.org/ontologies/cwm-core.owl>

AnnotationProperty: rdfs:comment


Datatype: xsd:string


Datatype: rdf:PlainLiteral


ObjectProperty: <http://www.cwm.org/objectmodel/instance.owl#hasSlot>

    Annotations:
        rdfs:comment "The set of Slot instances owned by the Object"

    Domain:
        <http://www.cwm.org/objectmodel/instance.owl#Object>

    Range:
        <http://www.cwm.org/objectmodel/instance.owl#Slot>

    InverseOf:
        <http://www.cwm.org/objectmodel/instance.owl#hasObject>


ObjectProperty: <http://www.cwm.org/objectmodel/instance.owl#hasStructuralFeature>

    Annotations:
        rdfs:comment "References the StructuralFeature instance that describes the value held by the Slot instance"

    SubPropertyOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#hasFeature>
```

```
    Domain:
        <http://www.cwm.org/objectmodel/instance.owl#Slot>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#StructuralFeature>


ObjectProperty: <http://www.cwm.org/objectmodel/instance.owl#hasInstanceValue>

    Annotations:
        rdfs:comment "References the DataValue or Object instance that contains the current value held by the Slot"

    Domain:
        <http://www.cwm.org/objectmodel/instance.owl#Slot>

    Range:
        <http://www.cwm.org/objectmodel/instance.owl#Instance>


ObjectProperty: <http://www.cwm.org/objectmodel/instance.owl#hasObject>

    Annotations:
        rdfs:comment "References the Object instance that owns the Slot"

    Domain:
        <http://www.cwm.org/objectmodel/instance.owl#Slot>

    Range:
        <http://www.cwm.org/objectmodel/instance.owl#Object>

    InverseOf:
        <http://www.cwm.org/objectmodel/instance.owl#hasSlot>


ObjectProperty: <http://www.cwm.org/objectmodel/instance.owl#hasClassifier>

    Annotations:
        rdfs:comment "The Classifier that declares the structure of the Instance"

    Domain:
        <http://www.cwm.org/objectmodel/instance.owl#Instance>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#Classifier>


ObjectProperty: <http://www.cwm.org/objectmodel/instance.owl#hasDataType>

    Annotations:
        rdfs:comment "The type of the dataValue. If not set, the type is taken as the type of the Attribute
(StructuralFeature) that is the feature for the DataSlot"

    Domain:
        <http://www.cwm.org/objectmodel/instance.owl#DataSlot>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#DataType>


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasFeature>


DataProperty: <http://www.cwm.org/objectmodel/instance.owl#hasDataValue>

    Annotations:
        rdfs:comment "The value for the slot expressed as a string"

    Domain:
        <http://www.cwm.org/objectmodel/instance.owl#DataSlot>

    Range:
        xsd:string


Class: <http://www.cwm.org/ontologies/cwm-core.owl#Package>


Class: <http://www.cwm.org/objectmodel/instance.owl#Extent>

    Annotations:
        rdfs:comment "Each instance of Extent owns a collection of instances an is used to link such collections to their
structural and behavioral definitions in CWM Resource packages. Because Extent is a subclass of package, it owns member
instances via the ElementOwnership association."

    SubClassOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#Package>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#DataType>


Class: <http://www.cwm.org/objectmodel/instance.owl#Object>

    Annotations:
        rdfs:comment "An object is an instance that originates from a class"

    SubClassOf:
        <http://www.cwm.org/objectmodel/instance.owl#hasSlot> some <http://www.cwm.org/objectmodel/instance.owl#Slot>,
        <http://www.cwm.org/objectmodel/instance.owl#Instance>


Class: <http://www.cwm.org/objectmodel/instance.owl#Slot>
```

```
    Annotations:
        rdfs:comment "A slot is a named location in an Object instance that holds the current value of the StructuralFeature
associated with the Slot instance. Normally, the StructuralFeature associated with the slot will be either an Attribute
instance or an AssociationEnd instance. Slots are owned by Objects; DataValues do not have slots"

    SubClassOf:
        <http://www.cwm.org/objectmodel/instance.owl#hasInstanceValue> max 1
<http://www.cwm.org/objectmodel/instance.owl#Instance>,
        <http://www.cwm.org/objectmodel/instance.owl#hasObject> max 1 <http://www.cwm.org/objectmodel/instance.owl#Object>,
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>,
        <http://www.cwm.org/objectmodel/instance.owl#hasStructuralFeature> exactly 1 <http://www.cwm.org/ontologies/cwm-
core.owl#StructuralFeature>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#Classifier>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#StructuralFeature>


Class: <http://www.cwm.org/objectmodel/instance.owl#Instance>

    Annotations:
        rdfs:comment "The instance construct defines an entity to which a set of operations casn be applied and which has a
state that stores the effects of the operations.In the metamodel Instance is connected to a Classifier that declares its
structure and behavior. It has a set of attribute values matching the definition of its Classifier. The set of attribute
values implements the current state of the Instance."

    SubClassOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>


Class: <http://www.cwm.org/objectmodel/instance.owl#DataSlot>

    Annotations:
        rdfs:comment "A Slot that is used to hold a data value where there is no need to manage the value as an element in
its own right (in which case a DataValue would be used) - for example it is a one-off string value or a number. The data
Value (and dataType where set) must be consistent with the type of the DataSlot's feature (Attribute) and must obey any
constraints on the full descriptor of the Attribute's DataType (including both explicit constraints and built-in constraints
such as multiplicity)"

    SubClassOf:
        <http://www.cwm.org/objectmodel/instance.owl#hasDataType> max 1 <http://www.cwm.org/ontologies/cwm-
core.owl#DataType>,
        <http://www.cwm.org/objectmodel/instance.owl#hasDataValue> exactly 1 xsd:string,
        <http://www.cwm.org/objectmodel/instance.owl#Slot>
```

Archivo A3.3. Ontología del paquete Instance CWM.

KeyIndexes

El **Archivo A3.4** contiene las entidades de la ontología que define los metadatos del paquete CWM KeyIndexes.

```
Prefix: rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
Prefix: dc: <http://purl.org/dc/elements/1.1/>
Prefix: xsd: <http://www.w3.org/2001/XMLSchema#>
Prefix: xml: <http://www.w3.org/XML/1998/namespace>
Prefix: rdfs: <http://www.w3.org/2000/01/rdf-schema#>
Prefix: owl: <http://www.w3.org/2002/07/owl#>



Ontology: <http://www.cwm.org/foundation/KeysIndexes.owl>

Import: <http://www.cwm.org/ontologies/cwm-core.owl>

Annotations:
    rdfs:comment "Kehys and indexes as means for specifying instances and for identifying alternate sortings of instances
are represented in the CWMFoundation so that they can be shared among the various data models that employ them. The CWM
Foundation defines the base concepts (uniqueness and relationships implemented as keys) upon which more specific key
structures can be built by other CWM and tool-specific packages"

AnnotationProperty: rdfs:comment


Datatype: xsd:boolean


Datatype: rdf:PlainLiteral


ObjectProperty: <http://www.cwm.org/foundation/KeysIndexes.owl#hasKeyRelationship>

    Annotations:
        rdfs:comment "Identifies the KeyRelationship instances that reference this UniqueKey instance"

    Domain:
        <http://www.cwm.org/foundation/KeysIndexes.owl#UniqueKey>
```

```
    Range:
        <http://www.cwm.org/foundation/KeysIndexes.owl#KeyRelationship>

    InverseOf:
        <http://www.cwm.org/foundation/KeysIndexes.owl#hasUniqueKey>


ObjectProperty: <http://www.cwm.org/foundation/KeysIndexes.owl#hasIndex>

    Annotations:
        rdfs:comment "Identifies the Index instance for which this IndexedFeature instance is relevant"

    Domain:
        <http://www.cwm.org/foundation/KeysIndexes.owl#IndexedFeature>

    Range:
        <http://www.cwm.org/foundation/KeysIndexes.owl#Index>

    InverseOf:
        <http://www.cwm.org/foundation/KeysIndexes.owl#hasIndexedFeature>


ObjectProperty: <http://www.cwm.org/foundation/KeysIndexes.owl#hasSpannedClass>

    Annotations:
        rdfs:comment "Indentifies the Class instance spanned by the Index instance"

    Domain:
        <http://www.cwm.org/foundation/KeysIndexes.owl#Index>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#CoreClass>


ObjectProperty: <http://www.cwm.org/foundation/KeysIndexes.owl#hasUniqueKey>

    Annotations:
        rdfs:comment "Identifies the UniqueKey instance that serves as the \"primary key\" for this KeyRelationship
instance."

    Domain:
        <http://www.cwm.org/foundation/KeysIndexes.owl#KeyRelationship>

    Range:
        <http://www.cwm.org/foundation/KeysIndexes.owl#UniqueKey>

    InverseOf:
        <http://www.cwm.org/foundation/KeysIndexes.owl#hasKeyRelationship>


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasFeature>

    Annotations:
        rdfs:comment "Identifies the StructuralFeature instance for which this IndexedFeature instance is relevant",
        rdfs:comment "Identifies StructuralFeature instances that participate as (part of) the key of this KeyRelationship
instance.",
        rdfs:comment "Identifies the StructuralFeature instances thata make up the unique key. The ordered constaint is used
to represent the sequence of StructuralFeature instances thata make up the UniqueKey instance's key"

    Domain:
        <http://www.cwm.org/foundation/KeysIndexes.owl#KeyRelationship>,
        <http://www.cwm.org/foundation/KeysIndexes.owl#UniqueKey>,
        <http://www.cwm.org/foundation/KeysIndexes.owl#IndexedFeature>


ObjectProperty: <http://www.cwm.org/foundation/KeysIndexes.owl#hasIndexedFeature>

    Annotations:
        rdfs:comment "Indentifies the IndexedFeature instance htat associateds this Index with one of the StructuralFeature
elements of the Index's key. The ordered constraint on this reference can be used to represent the sequential order of
elements of the Index's key."

    Domain:
        <http://www.cwm.org/foundation/KeysIndexes.owl#Index>

    Range:
        <http://www.cwm.org/foundation/KeysIndexes.owl#IndexedFeature>

    InverseOf:
        <http://www.cwm.org/foundation/KeysIndexes.owl#hasIndex>


DataProperty: <http://www.cwm.org/foundation/KeysIndexes.owl#isAscending>

    Annotations:
        rdfs:comment "The isAscending attribute is true if the feature is sorted in ascending order and false, if descending
order"

    Domain:
        <http://www.cwm.org/foundation/KeysIndexes.owl#IndexedFeature>

    Range:
        xsd:boolean


DataProperty: <http://www.cwm.org/foundation/KeysIndexes.owl#isPartitioning>

    Annotations:
        rdfs:comment "If True, this Index instance is used as a partitioning index"

    Domain:
        <http://www.cwm.org/foundation/KeysIndexes.owl#Index>
```

```
    Range:
        xsd:boolean


DataProperty: <http://www.cwm.org/foundation/KeysIndexes.owl#isUnique>

    Annotations:
        rdfs:comment "The isUnique attribute is True if the Index instance guarantees all of tits instances have a unique
key value"

    Domain:
        <http://www.cwm.org/foundation/KeysIndexes.owl#Index>

    Range:
        xsd:boolean


DataProperty: <http://www.cwm.org/foundation/KeysIndexes.owl#isSorted>

    Annotations:
        rdfs:comment "If True, the Index instance is maintained in a sorted order"

    Domain:
        <http://www.cwm.org/foundation/KeysIndexes.owl#Index>

    Range:
        xsd:boolean


Class: <http://www.cwm.org/foundation/KeysIndexes.owl#KeyRelationship>

    Annotations:
        rdfs:comment "KeyRelationship instances represent relationships between UniqueKey instances and the Class(es) that
reference them. This class is intended as a starting point for the creation of \"foreign key\" and other associative
relationships."

    SubClassOf:
        <http://www.cwm.org/foundation/KeysIndexes.owl#hasUniqueKey> exactly 1
<http://www.cwm.org/foundation/KeysIndexes.owl#UniqueKey>,
        <http://www.cwm.org/ontologies/cwm-core.owl#hasFeature> min 1 <http://www.cwm.org/ontologies/cwm-
core.owl#StructuralFeature>,
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>


Class: <http://www.cwm.org/foundation/KeysIndexes.owl#UniqueKey>

    Annotations:
        rdfs:comment "A UniqueKey represents a collection of features of some Class that, taken together, uniquely identify
instances of the Class. Instances of UniqueKey for which all features are required to have non-null values are candidates
for use as primary keys such as those defined by relational DBMSs"

    SubClassOf:
        <http://www.cwm.org/foundation/KeysIndexes.owl#hasKeyRelationship> some
<http://www.cwm.org/foundation/KeysIndexes.owl#KeyRelationship>,
        <http://www.cwm.org/ontologies/cwm-core.owl#hasFeature> min 1 <http://www.cwm.org/ontologies/cwm-
core.owl#StructuralFeature>,
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>


Class: <http://www.cwm.org/foundation/KeysIndexes.owl#Index>

    Annotations:
        rdfs:comment "Instances of the Index class represent the ordering of the instances of some other Class, and the
Index is said to \"span\" the Class. Indexes normally have an ordered set of attributes of the Class instance they span that
make up the \"key\" of the index; this set of relationships is represented by the IndexedFeature class that indicates how
the attributes are used by the Index instance.
The index class is intended primarily as a starting point for tools that require the notion of an index."

    SubClassOf:
        <http://www.cwm.org/foundation/KeysIndexes.owl#isPartitioning> exactly 1 xsd:boolean,
        <http://www.cwm.org/foundation/KeysIndexes.owl#isSorted> exactly 1 xsd:boolean,
        <http://www.cwm.org/foundation/KeysIndexes.owl#hasSpannedClass> exactly 1 <http://www.cwm.org/ontologies/cwm-
core.owl#CoreClass>,
        <http://www.cwm.org/foundation/KeysIndexes.owl#hasIndexedFeature> min 1
<http://www.cwm.org/foundation/KeysIndexes.owl#IndexedFeature>,
        <http://www.cwm.org/foundation/KeysIndexes.owl#isUnique> exactly 1 xsd:boolean,
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>


Class: <http://www.cwm.org/foundation/KeysIndexes.owl#IndexedFeature>

    Annotations:
        rdfs:comment "Instances of the IndexedFeature class map StructuralFeature instances of the spanned Class instance to
the Index instances that employ them as (part of) their key. Attributes of IndexedFeature instances indicate how specific
StructuralFeature instances are used in Index keys."

    SubClassOf:
        <http://www.cwm.org/foundation/KeysIndexes.owl#isAscending> max 1 xsd:boolean,
        <http://www.cwm.org/foundation/KeysIndexes.owl#hasIndex> exactly 1
<http://www.cwm.org/foundation/KeysIndexes.owl#Index>,
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>,
        <http://www.cwm.org/ontologies/cwm-core.owl#hasFeature> exactly 1 <http://www.cwm.org/ontologies/cwm-
core.owl#StructuralFeature>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#StructuralFeature>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#CoreClass>
```

Archivo A3.4. Ontología del paquete Key Indexes CWM.

DataTypes

El **Archivo A3.5** contiene las entidades de la ontología que define los metadatos del paquete CWM DataTypes.

```
Prefix: rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
Prefix: dc: <http://purl.org/dc/elements/1.1/>
Prefix: xsd: <http://www.w3.org/2001/XMLSchema#>
Prefix: xml: <http://www.w3.org/XML/1998/namespace>
Prefix: rdfs: <http://www.w3.org/2000/01/rdf-schema#>
Prefix: owl: <http://www.w3.org/2002/07/owl#>



Ontology: <http://www.cwm.org/foundation/DataTypes.owl>

Import: <http://www.cwm.org/ontologies/cwm-core.owl>

Annotations:
    rdfs:comment "The CWM DataTypes ontology supports definition of metamodel constructs that modelers can use to create the
specific data types they need. Although the CWM Foundation itself does not contain specific data type definitions, a number
of data type definitions for widely used environments are provided as examples of the appropiate usage of CWM Foundation
classes for creating data type definitions"

AnnotationProperty: rdfs:comment


Datatype: xsd:boolean


Datatype: rdf:PlainLiteral


ObjectProperty: <http://www.cwm.org/ontologies/cwm-core.owl#hasType>

    Annotations:
        rdfs:comment "Identifies the Classifier instance for which this TypeAlias instance acts as an alias"

    Domain:
        <http://www.cwm.org/foundation/DataTypes.owl#TypeAlias>


ObjectProperty: <http://www.cwm.org/foundation/DataTypes.owl#hasExpressionValue>

    Annotations:
        rdfs:comment "The value associated with an enumeration identifier can be stored here. The attribute is optional
because enumeration literals are not required to have a specific, displayable value. This is indicated by either an empty
value attribute or a value attribute value whose expression body attribute is a zero-length string"

    Domain:
        <http://www.cwm.org/foundation/DataTypes.owl#Enumeration>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#Expression>


ObjectProperty: <http://www.cwm.org/foundation/DataTypes.owl#hasMemberCase>

    Annotations:
        rdfs:comment "Contains the value of the Union's discriminator for this UnionMember"

    Domain:
        <http://www.cwm.org/foundation/DataTypes.owl#UnionMember>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#Expression>


ObjectProperty: <http://www.cwm.org/foundation/DataTypes.owl#hasDiscriminator>

    Annotations:
        rdfs:comment "Identifies the StructuralFeature instance that servers as the discriminator for the Union"

    Domain:
        <http://www.cwm.org/foundation/DataTypes.owl#Union>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#StructuralFeature>


ObjectProperty: <http://www.cwm.org/foundation/DataTypes.owl#hasEnumeration>

    Annotations:
        rdfs:comment "Identifies the Enumeration instance for which this enumeration literal is relevant"

    Domain:
        <http://www.cwm.org/foundation/DataTypes.owl#EnumerationLiteral>

    Range:
```

```
        <http://www.cwm.org/foundation/DataTypes.owl#Enumeration>

    InverseOf:
        <http://www.cwm.org/foundation/DataTypes.owl#hasLiteral>


ObjectProperty: <http://www.cwm.org/foundation/DataTypes.owl#hasLiteral>

    Annotations:
        rdfs:comment "Identifies the EnumerationLiteral instances relevant for a particular Enumeration instance. If the
Enumeration's isOrdered attribute is True, the ordering constraint on this reference end can be used to determine a logical
ordering for the EnumerationLiteral instances. Otherwise, ordering is ignored."

    Domain:
        <http://www.cwm.org/foundation/DataTypes.owl#Enumeration>

    Range:
        <http://www.cwm.org/foundation/DataTypes.owl#EnumerationLiteral>

    InverseOf:
        <http://www.cwm.org/foundation/DataTypes.owl#hasEnumeration>


DataProperty: <http://www.cwm.org/foundation/DataTypes.owl#isOrdered>

    Annotations:
        rdfs:comment "If True, the ordered constraint on the EnumerationLiterals association is relevant. Otherwise, the
ordering of EnumerationLiteral is considered unspecified."

    Domain:
        <http://www.cwm.org/foundation/DataTypes.owl#Enumeration>

    Range:
        xsd:boolean


DataProperty: <http://www.cwm.org/foundation/DataTypes.owl#isDefault>

    Annotations:
        rdfs:comment "Indicates if this UnionMember is the default member of the Union"

    Domain:
        <http://www.cwm.org/foundation/DataTypes.owl#UnionMember>

    Range:
        xsd:boolean


Class: <http://www.cwm.org/foundation/DataTypes.owl#TypeAlias>

    Annotations:
        rdfs:comment "The TypeAlias class is intended to provide a renaming capability for Classifier instances. This class
is required to support situations in which creation for an alias for a class effectively creates a new class. For example,
CORBA IDL type aliases have different typeCodes than their base types and are therefore treated as distinct types"

    SubClassOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#DataType>,
        <http://www.cwm.org/ontologies/cwm-core.owl#hasType> exactly 1 <http://www.cwm.org/ontologies/cwm-
core.owl#Classifier>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#DataType>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#Classifier>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#ProcedureExpression>


Class: <http://www.cwm.org/foundation/DataTypes.owl#UnionMember>

    Annotations:
        rdfs:comment "UnionMembers are described as features of a Union and each represents one of the members of a Union.
Note, however, thata multiple case values can map to a single UnionMember. If isDefault is true, the union member is the
default member. UnionMember instances are allowed to have a memberCase and be the default case."

    SubClassOf:
        <http://www.cwm.org/foundation/DataTypes.owl#hasMemberCase> exactly 1 <http://www.cwm.org/ontologies/cwm-
core.owl#Expression>,
        <http://www.cwm.org/foundation/DataTypes.owl#isDefault> exactly 1 xsd:boolean,
        <http://www.cwm.org/ontologies/cwm-core.owl#Attribute>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#Attribute>


Class: <http://www.cwm.org/foundation/DataTypes.owl#EnumerationLiteral>

    Annotations:
        rdfs:comment "Enumeration instances describe the enumeration identifiers, and possibly the values, associated with
an enumerated data type. Enumeration identifiers are contained in the name attribute derived from the EnuemrationLiteral
instance's ModelElement superclass."

    SubClassOf:
        <http://www.cwm.org/foundation/DataTypes.owl#hasEnumeration> exactly 1
<http://www.cwm.org/foundation/DataTypes.owl#Enumeration>,
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>,
        <http://www.cwm.org/foundation/DataTypes.owl#hasExpressionValue> some <http://www.cwm.org/ontologies/cwm-
core.owl#Expression>


Class: <http://www.cwm.org/foundation/DataTypes.owl#Enumeration>
```

```
    Annotations:
        rdfs:comment "The enumeration class is intended as a starting point from which enumerated data types can be created.
An enumerated data type is a collection of identifiers often used as the permitted states that some other attribute or
property of the enumerated type may take"

    SubClassOf:
        <http://www.cwm.org/foundation/DataTypes.owl#isOrdered> exactly 1 xsd:boolean,
        <http://www.cwm.org/ontologies/cwm-core.owl#DataType>,
        <http://www.cwm.org/foundation/DataTypes.owl#hasLiteral> min 1
<http://www.cwm.org/foundation/DataTypes.owl#EnumerationLiteral>


Class: <http://www.cwm.org/foundation/DataTypes.owl#QueryExpression>

    Annotations:
        rdfs:comment "QueryExpression instances contain query statements in language-dependent form"

    SubClassOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#ProcedureExpression>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#Expression>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#StructuralFeature>


Class: <http://www.cwm.org/foundation/DataTypes.owl#Union>

    Annotations:
        rdfs:comment "The Union class represents programming language unions and similarly structured data types. Because of
the diversity of union semantics found across software systems, the Union an UnionMember classes are likely candidates for
specialization to better capture union semantics in specific language extension packages"

    SubClassOf:
        <http://www.cwm.org/foundation/DataTypes.owl#hasDiscriminator> some <http://www.cwm.org/ontologies/cwm-
core.owl#StructuralFeature>,
        <http://www.cwm.org/ontologies/cwm-core.owl#Classifier>
```

Archivo A3.5. Ontología del paquete Data Types CWM.

Relational

El **Archivo A3.6** contiene las entidades de la ontología que define los metadatos del paquete CWM Relational.

```
Prefix: rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
Prefix: dc: <http://purl.org/dc/elements/1.1/>
Prefix: xsd: <http://www.w3.org/2001/XMLSchema#>
Prefix: xml: <http://www.w3.org/XML/1998/namespace>
Prefix: rdfs: <http://www.w3.org/2000/01/rdf-schema#>
Prefix: owl: <http://www.w3.org/2002/07/owl#>



Ontology: <http://www.cwm.org/resource/relational.owl>

Import: <http://www.cwm.org/foundation/KeysIndexes.owl>
Import: <http://www.cwm.org/objectmodel/Behavioral.owl>
Import: <http://www.cwm.org/objectmodel/instance.owl>
Import: <http://www.cwm.org/foundation/DataTypes.owl>
Import: <http://www.cwm.org/ontologies/cwm-core.owl>

Annotations:
    rdfs:comment "The Relational ontology describes data accessible through a relational interface such as a native RDBMS,
ODBC, or JDBC. The Relational ontology is based on the [SQL] standard section concerning RDBMS catalogs.

 The scope of the top level container, Catalog, is intended to cover all the tables a user can use in a single statement. A
catalog is also the unit that is managed  by a data resource. A catalog contains schemas which themselves contain tables.
Tables are made of columns that have an associated data type.

The Relational package uses constructs in the ObjectModel ontology to describe the object extensions added to SQL by the
[SQL] standards.

The Relational ontology also addresses the issues of indexing, primary keys, and foreign keys by extending the corresponding
concepts from the Foundation ontology"

AnnotationProperty: rdfs:comment


Datatype: xsd:integer


Datatype: xsd:string


Datatype: xsd:boolean
```

```
Datatype: rdf:PlainLiteral


ObjectProperty: <http://www.cwm.org/resource/relational.owl#hasColumnSet>

    Annotations:
        rdfs:comment "A NamedColumnSet created as of this type"

    Domain:
        <http://www.cwm.org/resource/relational.owl#SQLStructuredType>

    Range:
        <http://www.cwm.org/resource/relational.owl#NamedColumnSet>

    InverseOf:
        <http://www.cwm.org/resource/relational.owl#hasBaseType>


ObjectProperty: <http://www.cwm.org/resource/relational.owl#isNullable>

    Annotations:
        rdfs:comment "Indicates if null values are valid in this column"

    Domain:
        <http://www.cwm.org/resource/relational.owl#Column>

    Range:
        <http://www.cwm.org/resource/relational.owl#NullableType>


ObjectProperty: <http://www.cwm.org/resource/relational.owl#hasActionStatement>

    Annotations:
        rdfs:comment "The Trigger action itself"

    Domain:
        <http://www.cwm.org/resource/relational.owl#Trigger>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#ProcedureExpression>


ObjectProperty: <http://www.cwm.org/resource/relational.owl#hasConditionTiming>

    Annotations:
        rdfs:comment "Indicates if the trigger activity is run before or after the statement or row is modified."

    Domain:
        <http://www.cwm.org/resource/relational.owl#Trigger>

    Range:
        <http://www.cwm.org/resource/relational.owl#ConditionTimingType>


ObjectProperty: <http://www.cwm.org/resource/relational.owl#hasUsingTrigger>

    Annotations:
        rdfs:comment "A Trigger that references this NamedColumnSet in tis expression"

    Domain:
        <http://www.cwm.org/resource/relational.owl#NamedColumnSet>

    Range:
        <http://www.cwm.org/resource/relational.owl#Trigger>

    InverseOf:
        <http://www.cwm.org/resource/relational.owl#hasUsedColumnSet>


ObjectProperty: <http://www.cwm.org/resource/relational.owl#hasBaseType>

    Annotations:
        rdfs:comment "For typed Tables and Views, reference the base SQLStructuredType"

    Domain:
        <http://www.cwm.org/resource/relational.owl#NamedColumnSet>

    Range:
        <http://www.cwm.org/resource/relational.owl#SQLStructuredType>

    InverseOf:
        <http://www.cwm.org/resource/relational.owl#hasColumnSet>


ObjectProperty: <http://www.cwm.org/resource/relational.owl#hasOptionScopeColumnSet>

    Annotations:
        rdfs:comment "Reference to the NamedColumnSet (Table or View) indicated in the SCOPE clause of the Column
definition."

    Domain:
        <http://www.cwm.org/resource/relational.owl#Column>

    Range:
        <http://www.cwm.org/resource/relational.owl#NamedColumnSet>

    InverseOf:
        <http://www.cwm.org/resource/relational.owl#hasOptionScopeColumn>


ObjectProperty: <http://www.cwm.org/resource/relational.owl#hasReferencingColumn>
```

```
    Annotations:
        rdfs:comment "Reference a column of an SQLStructuredType (otherType) that is created with a REF clause referencing
this SQLStructuredType (this type). Note that in general, otherType and thisType are two different instances of
SQLStructuredType."

    Domain:
        <http://www.cwm.org/resource/relational.owl#SQLStructuredType>

    Range:
        <http://www.cwm.org/resource/relational.owl#Column>

    InverseOf:
        <http://www.cwm.org/resource/relational.owl#hasReferencedTableType>


ObjectProperty: <http://www.cwm.org/resource/relational.owl#hasUsedColumnSet>

    Annotations:
        rdfs:comment "Tables referenced by the actionStatement or the actionCondition"

    Domain:
        <http://www.cwm.org/resource/relational.owl#Trigger>

    Range:
        <http://www.cwm.org/resource/relational.owl#NamedColumnSet>

    InverseOf:
        <http://www.cwm.org/resource/relational.owl#hasUsingTrigger>


ObjectProperty: <http://www.cwm.org/resource/relational.owl#hasUpdateRule>

    Annotations:
        rdfs:comment "Same as deleteRule for updates of the primary key data record"

    Domain:
        <http://www.cwm.org/resource/relational.owl#ForeignKey>

    Range:
        <http://www.cwm.org/resource/relational.owl#ReferentialRuleType>


ObjectProperty: <http://www.cwm.org/resource/relational.owl#hasActionCondition>

    Annotations:
        rdfs:comment "A boolean expression that defines when the trigger has to be executed"

    Domain:
        <http://www.cwm.org/resource/relational.owl#Trigger>

    Range:
        <http://www.cwm.org/ontologies/cwm-core.owl#BooleanExpression>


ObjectProperty: <http://www.cwm.org/resource/relational.owl#hasDeleteRule>

    Annotations:
        rdfs:comment "An enumerated type. Indicates the disposition of the data records containing the foreign key value
when the record of the matching primary key is deleted."

    Domain:
        <http://www.cwm.org/resource/relational.owl#ForeignKey>

    Range:
        <http://www.cwm.org/resource/relational.owl#ReferentialRuleType>


ObjectProperty: <http://www.cwm.org/resource/relational.owl#hasOptionScopeColumn>

    Annotations:
        rdfs:comment "This NamedColumnSet is referenced in a SCOPE clause of the referenced Column"

    Domain:
        <http://www.cwm.org/resource/relational.owl#NamedColumnSet>

    Range:
        <http://www.cwm.org/resource/relational.owl#Column>

    InverseOf:
        <http://www.cwm.org/resource/relational.owl#hasOptionScopeColumnSet>


ObjectProperty: <http://www.cwm.org/resource/relational.owl#hasDeferrability>

    Annotations:
        rdfs:comment "Indicates the timing of the constraint enforcement during multiple-user updates",
        rdfs:comment "Indicates if the validity of the ForeignKey is to be tested at each statement or at the end of a
transaction",
        rdfs:comment "Indicates if the validity of the UniqueConstraint is to be tested at each statement or at the end of a
transaction"

    Domain:
        <http://www.cwm.org/resource/relational.owl#ForeignKey>,
        <http://www.cwm.org/resource/relational.owl#UniqueConstraint>,
        <http://www.cwm.org/resource/relational.owl#CheckConstraint>

    Range:
        <http://www.cwm.org/resource/relational.owl#DeferrabilityType>


ObjectProperty: <http://www.cwm.org/resource/relational.owl#hasTypeProcedure>

    Annotations:
```

```
        rdfs:comment "A procedure can be either a Function or a true Procedure.This indicates whether this object returns a
value or not"

    Domain:
        <http://www.cwm.org/resource/relational.owl#Procedure>

    Range:
        <http://www.cwm.org/resource/relational.owl#ProcedureType>


ObjectProperty: <http://www.cwm.org/resource/relational.owl#hasActionOrientation>

    Annotations:
        rdfs:comment "Indicates if the trigger is called once per statement execution or before or after each row of the
table is modified"

    Domain:
        <http://www.cwm.org/resource/relational.owl#Trigger>

    Range:
        <http://www.cwm.org/resource/relational.owl#ActionOrientationType>


ObjectProperty: <http://www.cwm.org/resource/relational.owl#hasQuery>

    Annotations:
        rdfs:comment "The query expression generating this result. The language attribute of the expression should generally
begin with \"SQL\""

    Domain:
        <http://www.cwm.org/resource/relational.owl#QueryColumnSet>

    Range:
        <http://www.cwm.org/foundation/DataTypes.owl#QueryExpression>


ObjectProperty: <http://www.cwm.org/resource/relational.owl#hasQueryExpression>

    Annotations:
        rdfs:comment "The query associated with the View. The query result must match the set of Columns associated with the
View (in parent class ColumnSet)."

    Domain:
        <http://www.cwm.org/resource/relational.owl#View>

    Range:
        <http://www.cwm.org/foundation/DataTypes.owl#QueryExpression>


ObjectProperty: <http://www.cwm.org/resource/relational.owl#hasSQLSimpleType>

    Annotations:
        rdfs:comment "The SQLSimpleType used to define the SQLDistinctType"

    Domain:
        <http://www.cwm.org/resource/relational.owl#SQLDistinctType>

    Range:
        <http://www.cwm.org/resource/relational.owl#SQLSimpleType>


ObjectProperty: <http://www.cwm.org/resource/relational.owl#hasTable>

    Annotations:
        rdfs:comment "The table that owns the Trigger"

    Domain:
        <http://www.cwm.org/resource/relational.owl#Trigger>

    Range:
        <http://www.cwm.org/resource/relational.owl#Table>

    InverseOf:
        <http://www.cwm.org/resource/relational.owl#hasTrigger>


ObjectProperty: <http://www.cwm.org/resource/relational.owl#hasReferencedTableType>

    Annotations:
        rdfs:comment "The column, used in an SQLStructuredType is a REF to a type. This references the REF'ed
SQLStructuredType."

    Domain:
        <http://www.cwm.org/resource/relational.owl#Column>

    Range:
        <http://www.cwm.org/resource/relational.owl#SQLStructuredType>

    InverseOf:
        <http://www.cwm.org/resource/relational.owl#hasReferencingColumn>


ObjectProperty: <http://www.cwm.org/resource/relational.owl#hasEventManipulation>

    Annotations:
        rdfs:comment "Indicates what types of events are using the current Trigger"

    Domain:
        <http://www.cwm.org/resource/relational.owl#Trigger>

    Range:
        <http://www.cwm.org/resource/relational.owl#EventManipulationType>
```

```
ObjectProperty: <http://www.cwm.org/resource/relational.owl#hasTrigger>

    Annotations:
        rdfs:comment "Associates triggers executed during changes to the table"

    Domain:
        <http://www.cwm.org/resource/relational.owl#Table>

    Range:
        <http://www.cwm.org/resource/relational.owl#Trigger>

    InverseOf:
        <http://www.cwm.org/resource/relational.owl#hasTable>


DataProperty: <http://www.cwm.org/resource/relational.owl#hasAutoUpdate>

    Annotations:
        rdfs:comment "The index is updated automatically"

    Domain:
        <http://www.cwm.org/resource/relational.owl#SQLIndex>

    Range:
        xsd:boolean


DataProperty: <http://www.cwm.org/resource/relational.owl#hasConditionReferenceOldTable>

    Annotations:
        rdfs:comment "The alias for the name of the owning table, used in the actionStatement, to represent the state of the
table before the update/delete/insert"

    Domain:
        <http://www.cwm.org/resource/relational.owl#Trigger>

    Range:
        xsd:string


DataProperty: <http://www.cwm.org/resource/relational.owl#hasCharacterSetName>

    Annotations:
        rdfs:comment "The name of the character set used for the values in the column. This filed applies only to columns
whose datatype is a character string."

    Domain:
        <http://www.cwm.org/resource/relational.owl#Column>

    Range:
        xsd:string


DataProperty: <http://www.cwm.org/resource/relational.owl#hasConditionReferenceNewTable>

    Annotations:
        rdfs:comment "The alias for the owning table name, used in the actionStatement, to represent the state of the table
after the insert/delete/update"

    Domain:
        <http://www.cwm.org/resource/relational.owl#Trigger>

    Range:
        xsd:string


DataProperty: <http://www.cwm.org/resource/relational.owl#hasNumericScale>

    Domain:
        <http://www.cwm.org/resource/relational.owl#SQLSimpleType>

    Range:
        xsd:integer


DataProperty: <http://www.cwm.org/resource/relational.owl#hasNumericPrecisionRadix>

    Annotations:
        rdfs:comment "corresponding field in DATA_TYPE_DESCRIPTOR"

    Domain:
        <http://www.cwm.org/resource/relational.owl#SQLSimpleType>

    Range:
        xsd:integer


DataProperty: <http://www.cwm.org/resource/relational.owl#hasDefaultCollationName>

    Annotations:
        rdfs:comment "The name of the default collation sequence used to sort the data values in the column. This applies
only to columns whose datatype is a form of character string."

    Domain:
        <http://www.cwm.org/resource/relational.owl#Catalog>

    Range:
        xsd:string


DataProperty: <http://www.cwm.org/resource/relational.owl#isCheckOption>
```

```
    Annotations:
        rdfs:comment "This field is meaningful only if the view is not ReadOnly. CheckOption indicates that the RDBMS will
validate that changes made to the data verify the view filtering condition and belong to the view result set."

    Domain:
        <http://www.cwm.org/resource/relational.owl#View>

    Range:
        xsd:boolean


DataProperty: <http://www.cwm.org/resource/relational.owl#hasScale>

    Annotations:
        rdfs:comment "The number of digits on the right of the decimal separator"

    Domain:
        <http://www.cwm.org/resource/relational.owl#Column>,
        <http://www.cwm.org/resource/relational.owl#SQLDistinctType>

    Range:
        xsd:integer


DataProperty: <http://www.cwm.org/resource/relational.owl#isReadOnly>

    Annotations:
        rdfs:comment "Indicates whether the underlying tables can be updated through an update to this View"

    Domain:
        <http://www.cwm.org/resource/relational.owl#View>

    Range:
        xsd:boolean


DataProperty: <http://www.cwm.org/resource/relational.owl#hasDataTimePrecision>

    Annotations:
        rdfs:comment "corresponding field in DATA_TYPE_DESCRIPTOR"

    Domain:
        <http://www.cwm.org/resource/relational.owl#SQLSimpleType>

    Range:
        xsd:integer


DataProperty: <http://www.cwm.org/resource/relational.owl#hasTypeNumber>

    Annotations:
        rdfs:comment "The number assignated to the datatype by the owning RDBMS"

    Domain:
        <http://www.cwm.org/resource/relational.owl#SQLDataType>

    Range:
        xsd:integer


DataProperty: <http://www.cwm.org/resource/relational.owl#hasLength>

    Annotations:
        rdfs:comment "The length of fixed length character or byte strings. Maximum length if length is variable."

    Domain:
        <http://www.cwm.org/resource/relational.owl#Column>,
        <http://www.cwm.org/resource/relational.owl#SQLDistinctType>

    Range:
        xsd:integer


DataProperty: <http://www.cwm.org/resource/relational.owl#hasCharacterMaximumLength>

    Annotations:
        rdfs:comment "Corresponding field in DATA_TYPE_DESCRIPTOR"

    Domain:
        <http://www.cwm.org/resource/relational.owl#SQLSimpleType>

    Range:
        xsd:integer


DataProperty: <http://www.cwm.org/resource/relational.owl#hasCollationName>

    Annotations:
        rdfs:comment "The name of the collation sequence used to sort the data values in the column. This applies only to
columns whose datatype is a form of character string."

    Domain:
        <http://www.cwm.org/resource/relational.owl#Column>

    Range:
        xsd:string


DataProperty: <http://www.cwm.org/resource/relational.owl#isTemporary>

    Annotations:
        rdfs:comment "Indicates that the table content is temporary. SQL92 standards provide two types of temporary tables
(local Temporary and Global Temporary). However, RDBMS products have implemented variations on this theme. It is recommended
```

```
that the product manufacturers provide specific temporary inofmration (besides the temporaryScope attribute) in their
extensions."

    Domain:
        <http://www.cwm.org/resource/relational.owl#Table>

    Range:
        xsd:boolean


DataProperty: <http://www.cwm.org/resource/relational.owl#hasDefaultCharacterSetName>

    Annotations:
        rdfs:comment "The name of the default character set used for the values in the column.
This field applies only to columns whose datatype is a character string."

    Domain:
        <http://www.cwm.org/resource/relational.owl#Catalog>

    Range:
        xsd:string


DataProperty: <http://www.cwm.org/resource/relational.owl#hasFilterCondition>

    Annotations:
        rdfs:comment "Which subset of the table is indexed"

    Domain:
        <http://www.cwm.org/resource/relational.owl#SQLIndex>

    Range:
        xsd:string


DataProperty: <http://www.cwm.org/resource/relational.owl#isSystem>

    Annotations:
        rdfs:comment "Indicates that the Table is a System Table (generally part of or view on the system catalog)."

    Domain:
        <http://www.cwm.org/resource/relational.owl#Table>

    Range:
        xsd:boolean


DataProperty: <http://www.cwm.org/resource/relational.owl#hasCharacterOctetLength>

    Annotations:
        rdfs:comment "corresponding field in DATA_TYPE_DESCRIPTOR"

    Domain:
        <http://www.cwm.org/resource/relational.owl#SQLSimpleType>

    Range:
        xsd:integer


DataProperty: <http://www.cwm.org/resource/relational.owl#isNullableEntry>

    Annotations:
        rdfs:comment "Entries in this index can be null."

    Domain:
        <http://www.cwm.org/resource/relational.owl#SQLIndex>

    Range:
        xsd:boolean


DataProperty: <http://www.cwm.org/resource/relational.owl#hasTemporaryScope>

    Annotations:
        rdfs:comment "This attribute is meaningful only when the isTemporary flag is True. The scope indicates when the data
of this table are available. \"SESSION\", \"APPLICATION\" are examples of possible values. Look at the Scope attribute for
Global Temporary tables in the SQL standards for more details."

    Domain:
        <http://www.cwm.org/resource/relational.owl#Table>

    Range:
        xsd:string


DataProperty: <http://www.cwm.org/resource/relational.owl#hasPrecision>

    Annotations:
        rdfs:comment "The total number of digits in the fileld"

    Domain:
        <http://www.cwm.org/resource/relational.owl#Column>,
        <http://www.cwm.org/resource/relational.owl#SQLDistinctType>

    Range:
        xsd:integer


DataProperty: <http://www.cwm.org/resource/relational.owl#hasNumericPrecision>

    Annotations:
        rdfs:comment "corresponding field in DATA_TYPE_DESCRIPTOR"
```

```
    Domain:
        <http://www.cwm.org/resource/relational.owl#SQLSimpleType>

    Range:
        xsd:integer


Class: <http://www.cwm.org/resource/relational.owl#DeferrabilityType>

    Annotations:
        rdfs:comment "Indicates the timing of the constraint enforcement during multiple-user updates"@en

    EquivalentTo:
        {<http://www.cwm.org/resource/relational.owl#DT_INITIALLYDEFERRED> ,
<http://www.cwm.org/resource/relational.owl#DT_INITIALLYIMMEDIATE> ,
<http://www.cwm.org/resource/relational.owl#DT_NOTDEFERRABLE>}


Class: <http://www.cwm.org/foundation/KeysIndexes.owl#KeyRelationship>


Class: <http://www.cwm.org/resource/relational.owl#SQLDataType>

    Annotations:
        rdfs:comment "An SQSLDataType is used to reference any datatype associated with a column."

    SubClassOf:
        <http://www.cwm.org/resource/relational.owl#hasTypeNumber> max 1 xsd:integer,
        <http://www.cwm.org/ontologies/cwm-core.owl#Classifier>


Class: <http://www.cwm.org/objectmodel/Behavioral.owl#Parameter>


Class: <http://www.cwm.org/resource/relational.owl#SQLIndex>

    Annotations:
        rdfs:comment "An Index on a table"

    SubClassOf:
        <http://www.cwm.org/foundation/KeysIndexes.owl#Index>,
        <http://www.cwm.org/resource/relational.owl#hasFilterCondition> exactly 1 xsd:string,
        <http://www.cwm.org/resource/relational.owl#isNullableEntry> exactly 1 xsd:boolean,
        <http://www.cwm.org/resource/relational.owl#hasAutoUpdate> exactly 1 xsd:boolean


Class: <http://www.cwm.org/objectmodel/instance.owl#Object>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#ProcedureExpression>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#Attribute>


Class: <http://www.cwm.org/resource/relational.owl#SQLParameter>

    Annotations:
        rdfs:comment "Parameters of stored procedures"

    SubClassOf:
        <http://www.cwm.org/objectmodel/Behavioral.owl#Parameter>


Class: <http://www.cwm.org/resource/relational.owl#Row>

    Annotations:
        rdfs:comment "An instance of a ColumnSet"

    SubClassOf:
        <http://www.cwm.org/objectmodel/instance.owl#Object>


Class: <http://www.cwm.org/objectmodel/instance.owl#DataValue>


Class: <http://www.cwm.org/resource/relational.owl#UniqueConstraint>

    Annotations:
        rdfs:comment "A condition to define uniqueness of rows in a table. An example of UniqueConstraint is a primary key."

    SubClassOf:
        <http://www.cwm.org/foundation/KeysIndexes.owl#UniqueKey>,
        <http://www.cwm.org/resource/relational.owl#hasDeferrability> exactly 1
<http://www.cwm.org/resource/relational.owl#DeferrabilityType>


Class: <http://www.cwm.org/resource/relational.owl#ActionOrientationType>

    Annotations:
        rdfs:comment "Enumeration that indicates if the trigger is called once per statement execution or before or after
each row of the table is modified."

    EquivalentTo:
        {<http://www.cwm.org/resource/relational.owl#AOT_ROW> , <http://www.cwm.org/resource/relational.owl#AOT_STATEMENT>}

    SubClassOf:
        owl:Thing


Class: <http://www.cwm.org/objectmodel/Behavioral.owl#Method>
```

```
Class: <http://www.cwm.org/resource/relational.owl#Column>

    Annotations:
        rdfs:comment "A column in a result set, a view, a table, or an SQLStructuredType."@en

    SubClassOf:
        <http://www.cwm.org/resource/relational.owl#hasPrecision> max 1 xsd:integer,
        <http://www.cwm.org/resource/relational.owl#hasScale> max 1 xsd:integer,
        <http://www.cwm.org/resource/relational.owl#hasOptionScopeColumnSet> max 1
<http://www.cwm.org/resource/relational.owl#NamedColumnSet>,
        <http://www.cwm.org/resource/relational.owl#hasReferencedTableType> max 1
<http://www.cwm.org/resource/relational.owl#SQLStructuredType>,
        <http://www.cwm.org/resource/relational.owl#hasLength> max 1 xsd:integer,
        <http://www.cwm.org/resource/relational.owl#hasCollationName> exactly 1 xsd:string,
        <http://www.cwm.org/resource/relational.owl#hasCharacterSetName> exactly 1 xsd:string,
        <http://www.cwm.org/ontologies/cwm-core.owl#Attribute>,
        <http://www.cwm.org/resource/relational.owl#isNullable> exactly 1
<http://www.cwm.org/resource/relational.owl#NullableType>


Class: <http://www.cwm.org/foundation/KeysIndexes.owl#UniqueKey>


Class: <http://www.cwm.org/resource/relational.owl#View>

    Annotations:
        rdfs:comment "A view is a non-materialized set of rows, defined by the associated query."

    SubClassOf:
        <http://www.cwm.org/resource/relational.owl#hasQueryExpression> exactly 1
<http://www.cwm.org/foundation/DataTypes.owl#QueryExpression>,
        <http://www.cwm.org/resource/relational.owl#isReadOnly> exactly 1 xsd:boolean,
        <http://www.cwm.org/resource/relational.owl#NamedColumnSet>,
        <http://www.cwm.org/resource/relational.owl#isCheckOption> exactly 1 xsd:boolean


Class: <http://www.cwm.org/resource/relational.owl#Schema>

    Annotations:
        rdfs:comment "A schema is a named collection of tables"

    SubClassOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#Package>


Class: <http://www.cwm.org/resource/relational.owl#QueryColumnSet>

    Annotations:
        rdfs:comment "The result set of a query"

    SubClassOf:
        <http://www.cwm.org/resource/relational.owl#ColumnSet>,
        <http://www.cwm.org/resource/relational.owl#hasQuery> exactly 1
<http://www.cwm.org/foundation/DataTypes.owl#QueryExpression>


Class: <http://www.cwm.org/resource/relational.owl#CheckConstraint>

    Annotations:
        rdfs:comment "A rule that specifies the values allowed in one or more columns of every row of a table"@en

    SubClassOf:
        <http://www.cwm.org/resource/relational.owl#hasDeferrability> exactly 1
<http://www.cwm.org/resource/relational.owl#DeferrabilityType>,
        <http://www.cwm.org/ontologies/cwm-core.owl#Constraint>


Class: <http://www.cwm.org/resource/relational.owl#Table>

    Annotations:
        rdfs:comment "A materialized NamedColumnSet"

    SubClassOf:
        <http://www.cwm.org/resource/relational.owl#hasTrigger> some <http://www.cwm.org/resource/relational.owl#Trigger>,
        <http://www.cwm.org/resource/relational.owl#NamedColumnSet>,
        <http://www.cwm.org/resource/relational.owl#isTemporary> exactly 1 xsd:boolean,
        <http://www.cwm.org/resource/relational.owl#isSystem> exactly 1 xsd:boolean,
        <http://www.cwm.org/resource/relational.owl#hasTemporaryScope> max 1 xsd:string


Class: <http://www.cwm.org/resource/relational.owl#PrimaryKey>

    Annotations:
        rdfs:comment "There is only one UniqueConstraint of type PrimaryKey per Table. It is implemented specifically by
each RDBMS."

    SubClassOf:
        <http://www.cwm.org/resource/relational.owl#UniqueConstraint>


Class: <http://www.cwm.org/resource/relational.owl#ConditionTimingType>

    Annotations:
        rdfs:comment "Enumerations that indicates if the trigger activity is run before or after the statement or row is
modified"

    EquivalentTo:
        {<http://www.cwm.org/resource/relational.owl#CTT_AFTER> , <http://www.cwm.org/resource/relational.owl#CTT_BEFORE>}

    SubClassOf:
        owl:Thing
```

```
Class: <http://www.cwm.org/ontologies/cwm-core.owl#CoreClass>


Class: <http://www.cwm.org/resource/relational.owl#ColumnValue>

    Annotations:
        rdfs:comment "The value in a column instance"@en

    SubClassOf:
        <http://www.cwm.org/objectmodel/instance.owl#DataValue>


Class: <http://www.cwm.org/objectmodel/instance.owl#Extent>


Class: <http://www.cwm.org/foundation/DataTypes.owl#TypeAlias>


Class: <http://www.cwm.org/resource/relational.owl#SQLSimpleType>

    Annotations:
        rdfs:comment "A simple datatype used with an SQL column. Examples are Integer, Varchar, LOB, CLOB, etc."

    SubClassOf:
        <http://www.cwm.org/resource/relational.owl#hasDataTimePrecision> max 1 xsd:integer,
        <http://www.cwm.org/resource/relational.owl#SQLDataType>,
        <http://www.cwm.org/resource/relational.owl#hasNumericScale> max 1 xsd:integer,
        <http://www.cwm.org/resource/relational.owl#hasNumericPrecision> max 1 xsd:integer,
        <http://www.cwm.org/ontologies/cwm-core.owl#DataType>,
        <http://www.cwm.org/resource/relational.owl#hasCharacterOctetLength> max 1 xsd:integer,
        <http://www.cwm.org/resource/relational.owl#hasCharacterMaximumLength> max 1 xsd:integer,
        <http://www.cwm.org/resource/relational.owl#hasNumericPrecisionRadix> max 1 xsd:integer


Class: <http://www.cwm.org/foundation/KeysIndexes.owl#Index>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#Classifier>


Class: <http://www.cwm.org/resource/relational.owl#ReferentialRuleType>

    EquivalentTo:
        {<http://www.cwm.org/resource/relational.owl#RRT_IMPORTEDKEYCASCADE> ,
<http://www.cwm.org/resource/relational.owl#RRT_IMPORTEDKEYNOACTION> ,
<http://www.cwm.org/resource/relational.owl#RRT_IMPORTEDKEYRESTRICT> ,
<http://www.cwm.org/resource/relational.owl#RRT_IMPORTEDKEYSETDEFAULT> ,
<http://www.cwm.org/resource/relational.owl#RRT_IMPORTEDKEYSETNULL>}


Class: <http://www.cwm.org/resource/relational.owl#EventManipulationType>

    Annotations:
        rdfs:comment "Indicates what type of events are using"

    EquivalentTo:
        {<http://www.cwm.org/resource/relational.owl#EMT_DELETE> , <http://www.cwm.org/resource/relational.owl#EMT_INSERT> ,
<http://www.cwm.org/resource/relational.owl#EMT_UPDATE>}


Class: <http://www.cwm.org/ontologies/cwm-core.owl#Package>


Class: <http://www.cwm.org/resource/relational.owl#Catalog>

    Annotations:
        rdfs:comment "A Catalog is the unit of logon and identification. It also identifies the scope of SQL statements: the
tables contained in a catalog can be used in a single SQL statement."@en

    SubClassOf:
        <http://www.cwm.org/resource/relational.owl#hasDefaultCollationName> exactly 1 xsd:string,
        <http://www.cwm.org/ontologies/cwm-core.owl#Package>,
        <http://www.cwm.org/resource/relational.owl#hasDefaultCharacterSetName> exactly 1 xsd:string


Class: <http://www.cwm.org/foundation/DataTypes.owl#QueryExpression>


Class: <http://www.cwm.org/resource/relational.owl#ColumnSet>

    Annotations:
        rdfs:comment "A set of columns, representing either the result of a query, a view, or a physical table."@en

    SubClassOf:
        <http://www.cwm.org/ontologies/cwm-core.owl#CoreClass>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#Constraint>


Class: <http://www.cwm.org/resource/relational.owl#Procedure>

    Annotations:
        rdfs:comment "This class describes Relational DBMS Stored procedures and functions"

    SubClassOf:
        <http://www.cwm.org/resource/relational.owl#hasTypeProcedure> exactly 1
<http://www.cwm.org/resource/relational.owl#ProcedureType>,
        <http://www.cwm.org/objectmodel/Behavioral.owl#Method>


Class: <http://www.cwm.org/resource/relational.owl#SQLIndexColumn>
```

```
    Annotations:
        rdfs:comment "Associates an index with its columns.
This is really an association class. It is assocated with one index and one column"

    SubClassOf:
        <http://www.cwm.org/foundation/KeysIndexes.owl#IndexedFeature>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>


Class: <http://www.cwm.org/resource/relational.owl#ProcedureType>

    Annotations:
        rdfs:comment "A procedure can be either a Function or a true Procedure. This indicates whether this obejct returns a
value or not."

    EquivalentTo:
        {<http://www.cwm.org/resource/relational.owl#PT_FUNCTION> ,
<http://www.cwm.org/resource/relational.owl#PT_PROCEDURE>}


Class: <http://www.cwm.org/resource/relational.owl#Trigger>

    Annotations:
        rdfs:comment "An action run by the DBMS when specified events occur on the table owning the Trigger"

    SubClassOf:
        <http://www.cwm.org/resource/relational.owl#hasTable> exactly 1 <http://www.cwm.org/resource/relational.owl#Table>,
        <http://www.cwm.org/resource/relational.owl#hasConditionReferenceOldTable> exactly 1 xsd:string,
        <http://www.cwm.org/resource/relational.owl#hasActionStatement> exactly 1 <http://www.cwm.org/ontologies/cwm-
core.owl#ProcedureExpression>,
        <http://www.cwm.org/resource/relational.owl#hasEventManipulation> exactly 1
<http://www.cwm.org/resource/relational.owl#EventManipulationType>,
        <http://www.cwm.org/resource/relational.owl#hasConditionTiming> exactly 1
<http://www.cwm.org/resource/relational.owl#ConditionTimingType>,
        <http://www.cwm.org/resource/relational.owl#hasConditionReferenceNewTable> exactly 1 xsd:string,
        <http://www.cwm.org/resource/relational.owl#hasUsedColumnSet> some
<http://www.cwm.org/resource/relational.owl#NamedColumnSet>,
        <http://www.cwm.org/ontologies/cwm-core.owl#ModelElement>,
        <http://www.cwm.org/resource/relational.owl#hasActionOrientation> exactly 1
<http://www.cwm.org/resource/relational.owl#ActionOrientationType>,
        <http://www.cwm.org/resource/relational.owl#hasActionCondition> exactly 1 <http://www.cwm.org/ontologies/cwm-
core.owl#BooleanExpression>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#DataType>


Class: <http://www.cwm.org/resource/relational.owl#NullableType>

    EquivalentTo:
        {<http://www.cwm.org/resource/relational.owl#NT_COLUMNNONULLS> ,
<http://www.cwm.org/resource/relational.owl#NT_COLUMNNULLABLE> ,
<http://www.cwm.org/resource/relational.owl#NT_COLUMNNULLABLEUNKNOWN>}


Class: <http://www.cwm.org/resource/relational.owl#SQLDistinctType>

    Annotations:
        rdfs:comment "A datatype defined as a Distinct Type, per SQL standard"

    SubClassOf:
        <http://www.cwm.org/resource/relational.owl#hasPrecision> max 1 xsd:integer,
        <http://www.cwm.org/resource/relational.owl#hasScale> max 1 xsd:integer,
        <http://www.cwm.org/resource/relational.owl#SQLDataType>,
        <http://www.cwm.org/foundation/DataTypes.owl#TypeAlias>,
        <http://www.cwm.org/resource/relational.owl#hasSQLSimpleType> exactly 1
<http://www.cwm.org/resource/relational.owl#SQLSimpleType>,
        <http://www.cwm.org/resource/relational.owl#hasLength> max 1 xsd:integer


Class: <http://www.cwm.org/resource/relational.owl#ForeignKey>

    Annotations:
        rdfs:comment "A Foreign Key associates columns from one table with columns of another table."@en

    SubClassOf:
        <http://www.cwm.org/resource/relational.owl#hasDeleteRule> exactly 1
<http://www.cwm.org/resource/relational.owl#ReferentialRuleType>,
        <http://www.cwm.org/foundation/KeysIndexes.owl#KeyRelationship>,
        <http://www.cwm.org/resource/relational.owl#hasDeferrability> exactly 1
<http://www.cwm.org/resource/relational.owl#DeferrabilityType>,
        <http://www.cwm.org/resource/relational.owl#hasUpdateRule> exactly 1
<http://www.cwm.org/resource/relational.owl#ReferentialRuleType>


Class: <http://www.cwm.org/resource/relational.owl#RowSet>

    Annotations:
        rdfs:comment "Each instance of RowSet owns a collection of Rows instances. The inherited association between
Namespace (a superclass of Package) and ModelElement is used to contain Instances."

    SubClassOf:
        <http://www.cwm.org/objectmodel/instance.owl#Extent>


Class: <http://www.cwm.org/ontologies/cwm-core.owl#BooleanExpression>


Class: owl:Thing
```

```
Class: <http://www.cwm.org/resource/relational.owl#NamedColumnSet>

    Annotations:
        rdfs:comment "A catalogued set of columns, which may be Table or View.
Note for typed tables: It is assumed that the typed table will own a set of columns conforming to the type they are OF. This
set of columns allows the manipulation of the table by products tahta ignore this SQL extension. It also allows the columns
of type REF, to be copied to a column with a SCOPE reference."

    SubClassOf:
        <http://www.cwm.org/resource/relational.owl#hasBaseType> max 1
<http://www.cwm.org/resource/relational.owl#SQLStructuredType>,
        <http://www.cwm.org/resource/relational.owl#hasOptionScopeColumn> some
<http://www.cwm.org/resource/relational.owl#Column>,
        <http://www.cwm.org/resource/relational.owl#hasUsingTrigger> some
<http://www.cwm.org/resource/relational.owl#Trigger>,
        <http://www.cwm.org/resource/relational.owl#ColumnSet>


Class: <http://www.cwm.org/foundation/KeysIndexes.owl#IndexedFeature>


Class: <http://www.cwm.org/resource/relational.owl#SQLStructuredType>

    Annotations:
        rdfs:comment "A Datatype defined as Structured Type, per SQL standard"

    SubClassOf:
        <http://www.cwm.org/resource/relational.owl#hasColumnSet> some
<http://www.cwm.org/resource/relational.owl#NamedColumnSet>,
        <http://www.cwm.org/resource/relational.owl#SQLDataType>,
        <http://www.cwm.org/resource/relational.owl#hasReferencingColumn> some
<http://www.cwm.org/resource/relational.owl#Column>,
        <http://www.cwm.org/ontologies/cwm-core.owl#CoreClass>


Individual: <http://www.cwm.org/resource/relational.owl#RRT_IMPORTEDKEYRESTRICT>

    Types:
        <http://www.cwm.org/resource/relational.owl#ReferentialRuleType>


Individual: <http://www.cwm.org/resource/relational.owl#RRT_IMPORTEDKEYSETNULL>

    Types:
        <http://www.cwm.org/resource/relational.owl#ReferentialRuleType>


Individual: <http://www.cwm.org/resource/relational.owl#DT_NOTDEFERRABLE>

    Types:
        <http://www.cwm.org/resource/relational.owl#DeferrabilityType>


Individual: <http://www.cwm.org/resource/relational.owl#RRT_IMPORTEDKEYCASCADE>

    Types:
        <http://www.cwm.org/resource/relational.owl#ReferentialRuleType>


Individual: <http://www.cwm.org/resource/relational.owl#NT_COLUMNNULLABLEUNKNOWN>

    Types:
        <http://www.cwm.org/resource/relational.owl#NullableType>


Individual: <http://www.cwm.org/resource/relational.owl#CTT_AFTER>

    Types:
        <http://www.cwm.org/resource/relational.owl#ConditionTimingType>


Individual: <http://www.cwm.org/resource/relational.owl#EMT_DELETE>

    Types:
        <http://www.cwm.org/resource/relational.owl#EventManipulationType>


Individual: <http://www.cwm.org/resource/relational.owl#NT_COLUMNNONULLS>

    Types:
        <http://www.cwm.org/resource/relational.owl#NullableType>


Individual: <http://www.cwm.org/resource/relational.owl#EMT_INSERT>

    Types:
        <http://www.cwm.org/resource/relational.owl#EventManipulationType>


Individual: <http://www.cwm.org/resource/relational.owl#PT_PROCEDURE>

    Types:
        <http://www.cwm.org/resource/relational.owl#ProcedureType>


Individual: <http://www.cwm.org/resource/relational.owl#RRT_IMPORTEDKEYNOACTION>

    Types:
        <http://www.cwm.org/resource/relational.owl#ReferentialRuleType>


Individual: <http://www.cwm.org/resource/relational.owl#AOT_ROW>
```

```
    Types:
        <http://www.cwm.org/resource/relational.owl#ActionOrientationType>


Individual: <http://www.cwm.org/resource/relational.owl#PT_FUNCTION>

    Types:
        <http://www.cwm.org/resource/relational.owl#ProcedureType>


Individual: <http://www.cwm.org/resource/relational.owl#RRT_IMPORTEDKEYSETDEFAULT>

    Types:
        <http://www.cwm.org/resource/relational.owl#ReferentialRuleType>


Individual: <http://www.cwm.org/resource/relational.owl#DT_INITIALLYIMMEDIATE>

    Types:
        <http://www.cwm.org/resource/relational.owl#DeferrabilityType>


Individual: <http://www.cwm.org/resource/relational.owl#AOT_STATEMENT>

    Types:
        <http://www.cwm.org/resource/relational.owl#ActionOrientationType>


Individual: <http://www.cwm.org/resource/relational.owl#DT_INITIALLYDEFERRED>

    Types:
        <http://www.cwm.org/resource/relational.owl#DeferrabilityType>


Individual: <http://www.cwm.org/resource/relational.owl#NT_COLUMNNULLABLE>

    Types:
        <http://www.cwm.org/resource/relational.owl#NullableType>


Individual: <http://www.cwm.org/resource/relational.owl#CTT_BEFORE>

    Types:
        <http://www.cwm.org/resource/relational.owl#ConditionTimingType>


Individual: <http://www.cwm.org/resource/relational.owl#EMT_UPDATE>

    Types:
        <http://www.cwm.org/resource/relational.owl#EventManipulationType>
```

Archivo A3.6. Ontología del paquete Relational CWM.

ANEXO D.

Guía de instalación y uso del caso práctico

El anexo contiene los pasos de inicialización de

1. Instalar la máquina virtual VirtualBox de la siguiente dirección:
   https://www.virtualbox.org/wiki/Downloads

2. Importar máquina virtual

Esta máquina virtual es una modificación de la imagen de un Oracle Linux 6.4, descargada de la página de Zend (http://www.zend.com/en/products/server/virtualbox) añadiéndole la instalación de una base de datos Postgresql 9.2, Jdk 1.7, Glassfish 4.0 y Netbeans 7.4.

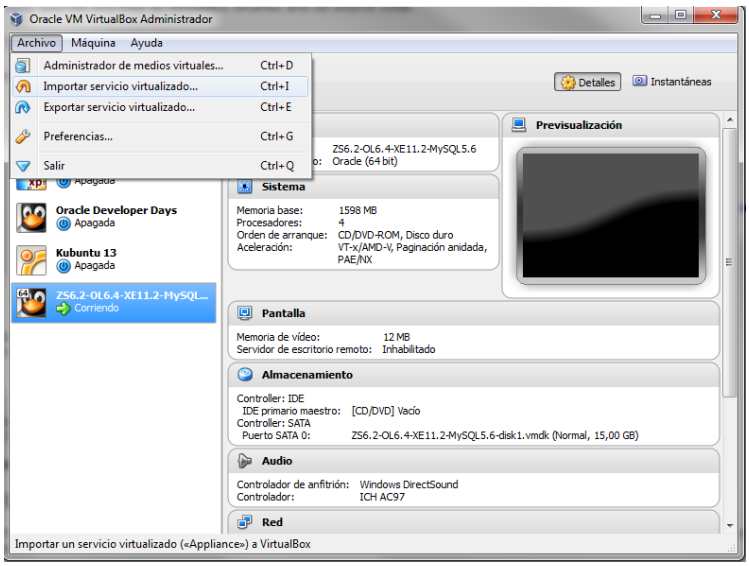


Figura A4. 1. Importación de imagenes virtuales en Virtual Box

3. Iniciar máquina virtual con el usuario Linux ‘guest’ (Figura A4. 2).

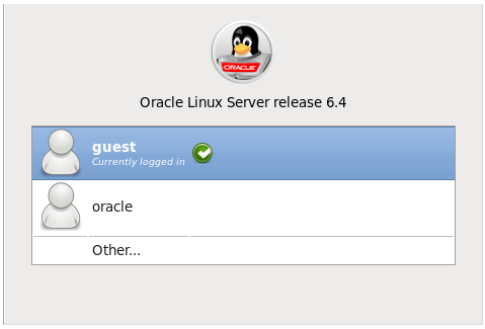


Figura A4. 2. Pantalla inicial de Oracle Linux 6.4.

La Tabla A4. 1 contiene la lista de usuarios/aplicaciones que contiene la imagen del caso práctico. Iniciamos con el usuario Linux ‘guest’.

Aplicación	Usuario	Passwords
Oracle Linux 6.4	root	zend1234
Oracle Linux 6.4	guest	guest1234
MySQL 5.6	cwm	cwm
Postgresql 9.2	cwm	cwm
Oracle 11g r2	system	oracle

Tabla A4. 1. Usuarios y aplicaciones de la máquina virtual

4. Abrir consola de comandos

5. Cambiar al usuario root

```
[guest@localhost Desktop]$ su
```

6. Iniciar la aplicación web del caso práctico:

```
[root@localhost Desktop]$ /home/guest/Desktop/startCasoPractico.sh
```

El comando del ejemplo X inicia la base de datos Oracle 11g R2 y despliega los servicios y cliente del caso práctico en el servidor web Glassfish.

7. Abrir el navegador Firefox con la dirección http://localhost:8080/ual-services/ para acceder a la interface Web (Figura A4. 3).

Figura A4. 3. Web del caso práctico.

Nota: La primera consulta de cada servicio web suele llevar más tiempo ya que Glassfish carga los módulos bajo demanda, inicializando la creación del pool de conexiones a las diferentes fuentes de datos y el análisis del razonador semántico.

Tabla de Figuras

Figuras

Tablas

Ejemplos

Archivos

Bibliografía

[1] J. Poole, D. Chang, D. Tolbert and D. Mellor, Common Warehouse Metamodel: An Introduction to the Standard for Data Warehouse Integration, John Wiley & Sons, 2002.

[2] OMG, "Common Warehouse Metamodel (CWM) Specification," in *Object Management Group*, March 2003.

[3] OMG, «CWM Metadata Interchange Patterns Specification,» March 2004.

[4] OMG, «Ontology Definition Metamodel Version 1.0,» May 2009.

[5] D. L. McGuinness, Ontologies Come of Age, Stanford University, 2003.

[6] W. O. W. Group, «OWL2 Web Ontology Language Document Overview,» W3C, 2009.

[7] «Desarrollo dirigido por modelos,» [En línea]. Available: http://www.modelosyproyectos.com/2010/01/25/el-desarrollo-dirigido-por-modelos-mdd/.

[8] The Middleware Company, Research Team, "Model Driven Development for J2EE utilizing a Model Driven Architecture (MDA) Approach," The Middleware Company, 2004.

[9] J. Clauberg, Elementa philosophiae sive ontosophia, 1647.

[10] I. Sacevski and J. Veseli, *Introduction to Model Driven Architecture (MDA),* Salzburg: University of Salzburg, 2007.

[11] J. D. Poole, The Common Warehouse Metamodel as a Foundation for Active Object Models in the Data Warehouse Enviroment, Stamford: OMG, 2000.

[12] «UML Specifications,» [En línea]. Available: http://www.uml.org/.

[13] «US Enciclopedia: Ontologia,» [En línea]. Available: http://enciclopedia.us.es/index.php/Ontolog%C3%ADa.

[14] C. Wolff, Philosophia prima sive ontologia, 1730.

[15] E. Husserl, Investigaciones Lógicas, Alianza, 2006.

[16] T. Gruber, «What is an Ontology,» [En línea]. Available: http://www-ksl.stanford.edu/kst/what-is-an-ontology.html.

[17] T. R. Gruber, A translation approach to portable ontologies specifications, Knowledge Acquisition, 1993.

[18] N. Guarino, Ontologies and Knowledge Bases. Towards a terminological clarification, 1995.

[19] «Ontology definitions,» [En línea]. Available: http://www.ontology.co/ontology-definitions-one.htm.

[20] L. Albertazzi, Formal and material ontology, 1996: Roberto Poli & Peter Simons.

[21] W. N. Borst, «Construction of Engineering Ontologies for Knowledge Sharing and Reuse,» University of Twente, Twente, 1997.

[22] «Categorie e modelli di conoscenza,» [En línea]. Available: http://www.terminometro.info/ancien/b41/es/categorie_e_modelli.htm.

[23] G. Negrini, *Categorie e modelli di conoscenza,* Milan: Istituto di studi sulla ricerca e documentazione scientifica CNR, 2000.

[24] N. F. Noy y D. L. McGuinness, *Desarrollo de Ontologías-101: Guía para crear tu primera Ontología,* Stanford: Stanford University, 2005.

[25] G. Antoniou y F. v. Harmelen, *Web Ontology Language: OWL,* Crete: Department of Computer Science, University of Crete, Department of AI, Vrije Universiteit Amsterdam, 2003.

[26] «SHOE Simple HTML Ontology Extensions,» [En línea]. Available: http://www.cs.umd.edu/projects/plus/SHOE/.

[27] «Marcos de Objectos,» [En línea]. Available: http://148.204.211.134/polilibros/portal/Polilibros/P_terminados/Lenguajes_de_Programacion_II/Unidad%20III%20Pascal%20Orientado%20a%20Objetos/Conocimiento/Marcos%20de%20Objetos.htm.

[28] J. B. Aranda, «Lógicas Descriptivas,» [En línea]. Available: http://web.ing.puc.cl/~jabaier/iic2212/description.pdf.

[29] OMG, *Ontology Definition Metamodel versión 1.0,* 2009.

[30] D. Brickley y R. . V. Guha, «Resource Description Framework (RDF) Schema Specification 1.0,» W3C, 2000.

[31] J. Broekstra, M. Klein, S. Decker, D. Fensel y I. Horrocks, «Adding formal semantics to the Web: building on top of RDF schema.,» de *Fourth European Conference on Research and Advanced Technology for Digital Libraries*, 2000.

[32] D. Fensel, I. Horrocks, F. v. Harmelen, D. McGuinness y P. F. Patel-Schneider, «OIL: Ontology Infrastructure to Enable the Semantic Web,» *IEEE Intelligent Systems,* p. 16, 2001.

[33] J. Hendler y D. L. McGuinness, «The DARPA Agent Markup Language,» *IEEE Intelligent Systems,*

p. 6, 2000.

[34] R. Ouellet y U. Oqbuji, «Introduction to DAML: Part I,» 30 January 2002. [En línea]. Available: http://www.xml.com/pub/a/2002/01/30/daml1.html.

[35] P. B. González, *Gestión de la información multimedia en Internet, Gestión del conocimiento DAML y ontologías consensuadas,* 2003.

[36] I. Horrocks, O. Kutz y U. Sattler, «The Even More Irresistible SROIQ,» de *10th Int. Conf. on Principles of Knowledge Representation and Reasoning*, 2006.

[37] N. N. Aguirre, *Un agente basado en un razonador de ontologías,* Buenos Aires: Universidad de Buenos Aires, 2011.

[38] M. Smith, I. Horrocks, M. Krötzsch y B. Glimm, «OWL 2 Web Ontology Language Conformance,» 11 December 2012. [En línea]. Available: http://www.w3.org/TR/owl2-conformance/.

[39] M. Horridge, «A Practical Guide To Building OWL Ontologies using Protégé 4 and CO-DE Tools,» The University of Manchester, 2009.

[40] C. M. Rodríguez, W. C. Montaño y J. M. Martínez, «Razonadores semánticos: un estado del arte,» *Ingenium,* p. 11, 2010.

[41] J. W. Lloyd y R. W. Topor, Foundations of logic programming, 1987.

[42] F. García Sánchez, Sistema basado en tecnologías del conocimiento para entornos de servicios web semánticos, Murcia: Universidad de Murcia, 2007.

[43] «Razonadores existentes para OWL2,» 2007. [En línea]. Available: http://www.w3.org/2007/OWL/wiki/Implementations.

[44] «Protege, Herramienta de edición de ontologías,» [En línea]. Available: http://protege.stanford.edu/overview/protege-owl.html .

[45] «Listado de herramientas de edición de ontologías,» [En línea]. Available: http://www.hipertexto.info/documentos/ontologias.html.

[46] «Api Java para ontologías en OWL2,» [En línea]. Available: http://owlapi.sourceforge.net/.

[47] «HTML5,» [En línea]. Available: http://theproc.es/files/5321.

[48] «DataTables,» [En línea]. Available: http://www.datatables.net/.

[49] «Modelo Vista Controlador,» [En línea]. Available: http://debuenamano.wordpress.com/2012/03/05/modelo-vista-controlador/.

[50] R. Navarro Marset, *Rest vs Web Services,* 2006.

[51] «Java Metadata Interface (JMI/JSR-40) version 1.0,» [En línea]. Available: http://jcp.org/en/jsr/detail?id=40.

[52] J. Poole, «CWM Forum,» 29 January 2003. [En línea]. Available: http://www.cwmforum.org/POOLEIntegrate2003.pdf.

[53] MySQL, «INFORMATION_SCHEMA Documentation,» Oracle, [En línea]. Available: http://dev.mysql.com/doc/refman/5.0/es/information-schema.html.

[54] M. Horridge, S. Jupp, G. Moulton, A. Rector, R. Stevens y C. Wroe, «A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools,» 2011.

[55] M. Heidegger, El ser y el tiempo, 1996: Fondo de Cultura Económica.

[56] R. V. García, Un Entorno para la Extracción Incremental de Conocimiento desde Texto en Lenguaje Natural, Universidad de Murcia.

[57] N. Guarino, Formal ontology and information systems, 1998.

[58] P. G. Otero, «Bases léxicas organizadas mediante un sistema de herencia Mereológica,» *Procesamiento del Lenguaje Natural,* 2000.

[59] F. Pfenning, Logic Programming, Carnegie Mellon, 2007.

[60] R. Romero, «Especificación OWL de una ontología para teleeducación en la web semántica,» Universidad Politécnica de Valencia, Valencia, 2007.

[61] M. Paczek, Implementation and design of a simple Rule-Based System for the object-relational databases, University of Mining and Metallurgy, Cracow, 2002.

[62] D. Tolbert, «The CWM Experience Implementing a UML-Based Data Warehouse Metamodel,» [En línea]. Available: http://www.omg.org/news/meetings/workshops/presentations/uml2001_presentations/12-2_Tolbert_CWM_UML_Experience.pdf.

[63] P. Chang, «Common Warehouse Metamodel (CWM): Extending UML for Data Warehousing and Business Intelligence,» 2000. [En línea]. Available: http://www.cwmforum.org/UMLDWBI.pdf.

[64] «Netbeans, Java IDE,» [En línea]. Available: http://netbeans.org/.

[65] «Maven, Java Project,» [En línea]. Available: http://maven.apache.org/what-is-maven.html.

[66] «Glassfish, Web Services Provider,» [En línea]. Available: http://glassfish.java.net/es/.

[67] «Catálogo de Postgresql,» [En línea]. Available: http://www.postgresql.org/docs/current/static/catalogs-overview.html#CATALOG-TABLE.