Año 2008

TESIS

Preparado en
Centro de Estudios de Microelectrónica y Sistemas Distribuidos

Para la Obtención del Título de
Doctor en Ciencias Aplicadas de la Universidad de Los Andes

Especialidad: Ontologías

Por ANA CELINA MUÑOZ GARCIA

MODELO ONTOLÓGICO PARA LA INTEGRACION DE BASES DE DATOS

Versión definitiva

Director de Tesis :	Dr. José Aguilar
Codirector de Tesis:	Dr. Rodrigo Martínez-Béjar
Jurados :	Dra. Isabel Bessembel Dra. Nora Montaño Dra. Flor Narciso

Agradecimientos

En el transcurso de la realización de esta tesis he recibido ayuda, apoyo y confianza de muchas personas a las que quiero expresar mi agradecimiento.

En primer lugar debo agradecer al Dr. José Aguilar, mi director de tesis, por su paciencia, comprensión y dedicación en el seguimiento de mi trabajo. Su orientación y experiencia han sido invaluables para el logro de los objetivos trazados hace ya varios años.

Agradezco también especialmente al Dr. Rodrigo Martínez-Béjar por sus preclaras indicaciones para mostrarme el camino y por la atención y apoyo en la gestión al frente del programa ALFA.

En lo institucional debo mencionar y agradecer todo el apoyo de la Universidad de Murcia a través del programa ALFA, y muy especialmente a la Universidad de Los Andes, especialmente al CEMISID en la persona del Dr. José Aguilar director del centro que me permitió trabajar y finalizar mi trabajo en sus espacios. Al IUTE la institución a la pertenezco por darme siempre el apoyo económico y profesional para poder realizar el trabajo.

Dedicatoria

Esta tesis trata sobre ontología y conocimientos ¿Qué mejor que iniciarla dedicándola a quienes mejor me enseñaron?

A Pipo y mima por enseñarme de la entrega y del trabajo duro y honesto y la importancia y la grandeza de las cosas pequeñas y sencillas.

A mis hermanos por acompañarme felizmente en el aprendizaje de mis primeros años.

A mis hijas Greshy y Grecia a quienes me debo por completo y a quienes he robado tiempo para darlo a este proyecto.

A mis amigos y amigas por su respaldo irrestricto y su consuelo en mis momentos de dudas.

Tabla de contenido

Introducción General 5

Capítulo 1 7

1. Introducción 7

1.1 Generalidades 7

1.2 Objetivos 9

1.3 Estado del Arte 10

1.4 Organización de la Tesis 12

Capítulo 2 14

2. Marco Teórico 14

2.1 Introducción 14

2.2 Integración de Base de Datos 14
2.2.1 Arquitecturas para la Integración de Bases de Datos 18

2.3 Bases de Datos Federadas 20

2.4 Bases de Datos Componentes 24
2.4.1 Bases de Datos Inteligentes 24
2.4.2 Bases de Datos Multimedia 27
2.4.3 Bases de Datos Orientadas a Objeto (BDOO) 29
2.4.4 Bases de Datos Relacionales 33

2.5 Ontología 35
2.5.1 Tipos de Ontologías 37
2.5.2 Metodologías para el diseño de Ontologías 39
2.5.3 Lenguajes para construir Ontologías 46
2.5.4 Herramientas Ontológicas 48

Capítulo 3 50

3. Modelo Ontológico para Bases de Datos Federadas 50

3.1 Introducción 50

3.2 Ontología de Integración de Bases de Datos 50
3.2.1 Conceptos para Bases de Datos Federadas 51
3.2.2 Operaciones de Integración de las Bases de Datos Federadas 53
3.2.3 Restricciones de Integración en Bases de Datos Federadas 56

3.3 Ontologías para Bases de Datos Componentes 58
3.3.1 Bases de Datos Relacionales 59
3.3.2 Bases de Datos Orientadas a Objeto 64
3.3.3 Bases de Datos Multimedia 69
3.3.4 Bases de Datos Inteligentes 74

Capítulo 4 82

4. Caso de estudio para el Modelo Ontológico de Integración de Bases de Datos Federadas 82

4.1 Introducción 82

4.2 Caso de Estudio utilizando el Modelo Ontológico de Bases de Datos Inteligentes 82
4.2.1 Descripción Conceptual de la Bases de Datos Inteligente utilizando el marco ontológico planteado 83
4.2.2 Operaciones del Modelo Ontológico para el caso de uso 85

4.3 Caso de Estudio utilizando el Modelo Ontológico de Bases de Datos Multimedia 88
4.3.1 Generalidades de los Objetos de Aprendizaje 88
4.3.2 Descripción del caso de estudio de acuerdo al modelo ontológico para bases de datos multimedia 90

4.4 Caso de Estudio utilizando el Modelo Ontológico de Integración de Bases de Datos Federadas 95
4.4.1 Proceso de Integración de la Base de Datos Federada Inscripcion_Clases. 96
4.4.2 Ejemplo de operaciones una vez creada la federación. 97

Capítulo 5 103

5. Validación de la Ontología para la Integración de Bases de Datos Federadas 103

5.1 Introducción 103

5.2 Base Teórica para validar una Ontología 103
5.2.1 Marco teórico de OWL para la validación 104

5.3 Formalización de la Ontología utilizando Protégé-OWL 111
5.3.1 Modelo de integración validado en Protégé-OWL 114
5.4 Instancias para el Modelo de Integración en Protégé-OWL 125

5.4 Validaciones adicionales 133

Conclusiones 137

Referencias 140

ANEXO 147

PROTÉGÉ-OWL 147
A.1 Ontologías OWL 147
A.2 Las tres especies de OWL 147
A.3 Componentes de las Ontologías OWL 148

Índice de Figuras

Figura 2.1 Taxonomía para las Bases de Datos Federadas 20

Figura 2.2 Arquitectura de Esquemas que componen las bases de datos federadas 22

Figura 2.3 Categorización de Lassila y McGuiness.....38

Figura 2.4 Proceso de desarrollo de una Ontología 40

Figura 2.5 Metodología de Uschold y King 41

Figura 2.6 Metodología de Grüninger y Fox.....42

Figura 2.7 Metodología Procesos de desarrollo y ciclo de vida de Methontology.....44

Figura 2.8 Tareas de la actividad de conceptualización de acuerdo a METHONTOLOGY 44

Figura 2.9. Lenguajes ontológicos tradicionales.....47

Figura 2.10. Lenguajes Ontológicos de Marca 44

Figura 3.1. Modelo Inteligente de Integración para Bases de Datos Federadas.....51

Figura 3.2. Taxonomía de los Conceptos para Bases de Datos Federadas.....52

Figura 3.3. Técnicas de comparación de esquemas 54

Figura 3.4. Taxonomía de Operaciones de Integración para Bases de Datos Federadas55

Figura 3.5. Esquema Ontológico de las Restricciones de Integración para Bases de Datos Federadas 57

Figura 3.6. Taxonomía de Conceptos para Bases de Datos Relacionales.....60

Figura 3.7. Taxonomía de Operaciones para las Bases de Datos Relacionales.....61

Figura 3.8. Taxonomía de Restricciones para Bases de Datos Relacionales 63

Figura 3.9. Taxonomía de Conceptos para Bases de Datos OO 65

Figura 3.10. Taxonomía de Operaciones para Bases de Datos Orientadas a Objetos 67

Figura 3.11. Taxonomía de las Restricciones de Bases de Datos OO 68

Figura 3.12. Taxonomía de las Bases de Datos Multimedia 69

Figura 3.13. Taxonomía de Conceptos para Bases de Datos Multimedia 71

Figura 3.14. Taxonomía de Operaciones para Bases de Datos Multimedia 72

Figura 3.15. Taxonomía de Restricciones para Bases de Datos Multimedia 74

Figura 3.16. Taxonomía de Bases de Datos Inteligentes 75

Figura 3.17. Esquema Ontológico de Conceptos para Bases de Datos Inteligentes 77

Figura 3.18. Taxonomía de las Operaciones para Bases de Datos Inteligentes 79

Figura 3.19. Esquema Ontológico de Restricciones en Bases de Datos Inteligentes 80

Figura 4.1. Módulos que conforman un CURSO 90

Figura 5.1. Vista lógica de una clase en Protégé-OWL 107

Figura 5.2. Vista de propiedades de una clase en Protégé-OWL 108

Figura 5.3. Pestaña a través de la cual se definen las propiedades objeto (ObjectProperty) del modelo 110

Figura 5.4. Propiedades datatype 111

Figura 5.5. Taxonomía de la Ontología integración de bases de datos federadas 113

Figura 5.6 Clases, subclases y restricciones de la clase Base_de_Datos_Federadas .. 115

Figura 5.7. Propiedades y Restricciones de la clase Base_de_Datos_Federadas 116

Figura 5.8. Vista lógica para las bases de datos componentes en Protégé-OWL 116

Figura 5.9. Vista de propiedades para las bases de componentes en Protégé-OWL ... 117

Figura 5.10. Vista de propiedades para las operaciones de integración en Protégé OWL118

Figura 5.11. Vista lógica para las operaciones de integración en Protégé OWL 119

Figura 5.12. Propiedades y subpropiedades de la ontología para las bases de datos componentes 120

Figura 5.13. Propiedades y subpropiedades de la ontología para las operaciones de integración 121

Figura 5.14. Propiedades y subpropiedades de la ontología para las restricciones de integración 122

Figura 5.15. Vista lógica para la clase Conceptos_BDI en Protégé OWL..... 123

Figura 5.16. Vista lógica para la clase Operaciones_BDI en Protégé OWL 123

Figura 5.17. Vista lógica para clase Restricciones_BDI en Protégé OWL 124

Figura 5.18. La ontología se encuentra en OWL DL 125

Figura 5.19. Instancia de la Base de Datos Federada Inscripción_Clase 126

Figura 5.20. Individuo Inscripción_Clase de la clase Base_de_Datos_Federada Inscripcion_Clase..... 127

Figura 5.21 Instancias de las propiedades tiene_BDComponente y tiene_Restricciones_de_Integridad de la base de datos federada Inscripcion_Clase .. 128

Figura 5.22. Instancias de la Clase BD_Componentes para la BD Federada Inscripcion_Clase..... 129

Figura 5.23. Instancia de la Base de Datos Inteligente Inscripción 130

Figura 5.24. Instancia de la Base de Conocimientos de la Base de Datos Inteligente SInscripcionInteligente..... 131

Figura 5.25. Instancia de la Base de Datos Multimedia llamada Control_de_Cursos.. 132

Figura 5.26. Instancia del Objeto de Aprendizaje Gestión de Conocimiento (OAGC) y las instancias de los metadatos que contiene 133

Figura 5.27. Test de validación de las propiedades de la ontología..... 134

Figura 5.28. Test de ontología para el individuo Inscripcion_Clase..... 135

Figura 5.29. Código OWL generado por el editor Protégé-OWL para la ontología..... 136

Figura 5.30. Ontología generada como una página Web 136

Figura A1. Representación de Individuos..... 148

Figura A2. Representación de Propiedades 149

Figura A3. Propiedades Object, Datatype y Annotation 149

Figura A4. Un ejemplo de propiedad inversa: tieneHijo es la propiedad inversa de tienePadre 150

Figura A5. Un ejemplo de propiedad Funcional: **tieneMadre**..... 150

Figura A6. Ejemplo de propiedad inversa Funcional: **esMadrede** 150

Figura A7. Ejemplo de propiedad transitiva: **tieneAncestro** 151

Figura A8. Un ejemplo de Propiedad Simétrica: **tieneHermano** 151

Figura A9. Representación de Clases (contiene individuos)..... 152

INTRODUCCIÓN GENERAL

La interoperabilidad entre diferentes sistemas de información es uno de los aspectos más críticos en la operación cotidiana de muchas organizaciones. En la última década esta preocupación se ha visto incrementada con la proliferación de diferentes bases de datos, con diferentes modelos de datos, que corren en diferentes plataformas. Los sistemas de bases de datos federadas son una respuesta a este problema, al permitir tener disponible la información desde diferentes fuentes de información, las cuales pueden ser heterogéneas, distribuidas y autónomas. Una base de datos federada actúa como una aplicación front-end[1] de múltiples componentes. La base de datos federada proporciona operaciones para el acceso a cada componente, manteniendo la consistencia de información entre las diversas fuentes, y proporcionando un método de acceso uniforme a los servicios que cada componente ofrece.

La diversidad de lenguajes de programación y de consulta, modelos de datos, métodos de integración, entre otros, determinan diferentes arquitecturas de una base de datos federada, que varían desde un enfoque fuertemente acoplado a uno débilmente acoplado. En general, los sistemas fuertemente acoplados integran las diversas fuentes de información a través de un esquema conceptual global, normalmente denominado *modelo canónico*, que nosotros también llamamos en este trabajo *modelo común de datos*, proporcionando una vista uniforme de los diversos componentes a un alto nivel. El uso de un modelo canónico oculta las diferencias estructurales entre los diferentes componentes, y da al usuario la ilusión de estar accesando una simple base de datos centralizada. Por otra parte, en los sistemas débilmente acoplados la integración de los componentes se basa en un lenguaje de acceso común que todos los componentes deben acordar, en el cual todas las funciones están estandarizadas. Una arquitectura clásica para la integración de bases de datos es la propuesta por Shet&Larson (Shet&Larson 1990), que realiza la integración de bases de datos federadas cuyos componentes son únicamente bases de datos relacionales. Utilizando como base la arquitectura de Shet&Larson, en este trabajo se propone un modelo de integración para bases de datos federadas, cuyos componentes pueden ser de diferentes tipos (inteligentes, multimedia, orientadas a objeto y relacionales), utilizando ontologías.

Una ontología es una descripción de conceptos y sus relaciones en un dominio dado. Las ontologías permiten la comunicación entre personas, organizaciones y aplicaciones, ya que proporcionan una comprensión común de un dominio, de modo que se eliminan confusiones conceptuales y terminológicas. Las ontologías también ayudan a que los sistemas interoperen, sin esfuerzo por parte del usuario.

[1] Interfaz de usuario

Las ontologías resultan muy útiles para facilitar el razonamiento automático. Partiendo de unas reglas de inferencia, un motor de razonamiento puede usar los datos de las ontologías para inferir conclusiones. El razonamiento automático también se usa para establecer relaciones entre ontologías, para descubrir relaciones ocultas o inesperadas entre los datos, entre otras cosas. También, el razonamiento automático puede ser usado para integrar esquemas de bases de datos. Como hemos visto antes, la integración de esquemas de bases de datos resulta imprescindible cuando se trabaja con bases de datos federadas. Nosotros usaremos esa idea en nuestra propuesta.

Así, este trabajo muestra un modelo ontológico para el proceso de integración de bases de datos federadas. Además, presenta varios ejemplos de aplicación del modelo, como su implementación en Protégé[2].

[2] Editor de ontologías y estructura para base de conocimientos open source y gratis. http://protege.stanford.edu/

Capítulo 1

1. INTRODUCCIÓN

Este capítulo presenta el marco general en el que se da el desarrollo de este trabajo. En la sección 1 se presentan generalidades sobre el tema a abordar, particularmente las diferentes necesidades que han tenido las organizaciones en la integración de bases de datos y en el uso de ontologías; en la sección 2 se presentan los objetivos del trabajo y se describen los alcances y limitaciones de la propuesta de esta tesis. A continuación, en la sección 3 se presenta el estado del arte; y finalmente, en la sección 4 se presenta la organización del documento.

1.1 Generalidades

Actualmente, todas las organizaciones, ya sean industriales, de servicios, educativas, etc., manejan información que es necesaria para su desempeño, lo que les obliga a utilizar los últimos avances en tecnologías para adaptar todos sus sistemas de información a la velocidad con que cambian sus entornos. La creciente necesidad de cooperación entre entidades independientes requiere el acceso integrado a múltiples bases de datos autónomas y heterogéneas, surgen entonces muchas incógnitas en cuanto al almacenamiento y distribución de la información. La gestión de la información distribuida se ha visto influenciada por las nuevas capacidades que aportan las bases de datos en las siguientes dimensiones: Rendimiento, Inteligencia y Distribución [Piattini M, 2000].

En la dimensión de Rendimiento, cuyo objetivo es mejorar los tiempos de acceso, destacan los siguientes tipos de tecnologías [Piattini M, 2000]: bases de datos paralelas, bases de datos en tiempo real y bases de datos en memoria principal.

En la dimensión Inteligencia, cuyo objetivo es generar y descubrir información y conocimiento necesarios en toda organización, destacan las bases de datos activas, deductivas, de conocimientos y difusas, entre otras. En general las bases de datos inteligentes son la evolución natural de los sistemas de bases de datos tradicionales, no solo porque permite la manipulación de los datos, sino elementos cognitivos en forma, por ejemplo, de hechos y reglas. Un aspecto fundamental de estas bases de datos son las técnicas para descubrir conocimiento. Para eso se requieren de técnicas del área de minería de datos [Bertino E., Catania B., Zarri Gian P. 2001], que permitan, entre otras cosas: aprender, analizar datos, realizar tareas de clasificación, reconocimiento, etc.

En la dimensión Distribución [Date C. J, H Darwen, 2007], cuyo objetivo es manejar la información dispersa geográficamente, destaca particularmente todo lo que tiene que ver con bases de datos distribuidas, por ejemplo, las bases de datos federadas, las multibases de datos, las bases de datos móviles, etc. La extensión de Internet a todos los ámbitos ha popularizado el uso de los sistemas distribuidos. En los sistemas distribuidos se han generado diversos campos de estudio. En nuestro caso, para el problema de gestión de la información, el papel de las bases de datos distribuidas es fundamental cuando es necesario acceder y almacenar los datos en lugares geográficamente dispersos. Ahora bien, un problema abierto en esta dimensión es el de la integración.

El problema de integración de componentes heterogéneos en los sistemas distribuidos debe resolver el tema de interoperabilidad. Particularmente, la heterogeneidad de los datos dentro de la comunidad de bases de datos distribuidas pueden ser del tipo (Kim y Seo, 1991): heterogeneidad estructural (heterogeneidad esquemática) y heterogeneidad semántica (heterogeneidad en el significado de los datos). La heterogeneidad estructural significa que los diferentes sistemas de información almacenan sus datos en diferentes estructuras. La heterogeneidad semántica considera el contenido de la información y su significado.

Por otro lado, los problemas de interoperabilidad fueron clasificados por Shet a nivel de [Shet, 1998]: sistema, sintáctico, estructural y semántico. A nivel de sistema incluye hardware y sistema operativo incompatible; a nivel sintáctico se refiere a diferentes lenguajes y representación de datos; a nivel estructural se refiere a los diferentes modelos de datos, y a nivel semántico se refiere al significado de los términos usados en el intercambio de información.

Específicamente, para alcanzar la interoperabilidad semántica en un sistema de información heterogéneo el significado de la información que se intercambia debe ser comprendido entre los sistemas. Los conflictos semánticos ocurren cuando dos contextos no usan la misma interpretación de la información, por ejemplo cuando se utilicen términos diferentes equivalentes entre distintas comunidades. La integración semántica es un área de investigación reciente en diferentes disciplinas como bases de datos, ontologías, etc. [Natalia F. Noy, 2004]. Específicamente, todo proceso de integración en bases de datos federadas requiere establecer correspondencias semánticas entre los esquemas componentes de la federación.

La teoría de ontologías juega un rol central en la resolución de la heterogeneidad semántica [Uschold M., Gruninger M. 2004]. Una ontología desde un punto de vista filosófico es la explicación sistemática de la existencia. En inteligencia artificial, una ontología es una especificación explícita de una conceptualización [Gruber 1993]. El uso de las ontologías para la explicación de conocimiento implícito y escondido es un posible enfoque para resolver el problema de heterogeneidad semántica. Uschold y Grüninger mencionan la interoperabilidad como la aplicación clave de las ontologías [Uschold and

Grüninger, 1996]. Las ontologías pueden ser usadas en tareas de integración para describir la semántica de las fuentes de información. Con respecto a la integración de fuentes de datos, pueden ser utilizadas para la identificación y asociación de conceptos semánticamente equivalentes.

Existen diferentes maneras de emplear las ontologías en tareas de integración de fuentes de información, para resolver el problema de heterogeneidad semántica [Wache H. et al., 2001]: como ontología simple, ontologías múltiples o híbridas. La ontología simple usa una única ontología global para proporcionar un vocabulario compartido para la especificación de la semántica. Todas las fuentes de información se relacionan a través de la ontología global. Como ontologías múltiples cada fuente de información es descrita por su propia ontología. La ontología híbrida es similar a la ontología múltiple, donde cada fuente de información se describe a través de su propia ontología, pero utiliza una ontología global para compartir vocabulario.

Las ontologías son ampliamente utilizadas en Ciencias de la Computación, en aplicaciones como procesamiento de lenguaje natural, comercio electrónico, integración inteligente de la información, recuperación de información, bio-informática, educación, ingeniería del conocimiento y nuevos campos emergentes como la Web Semántica [Gómez-Pérez et al., 2004].

1.2 Objetivos

Este trabajo de tesis propone modelar el proceso de integración de bases de datos heterogéneas utilizando ontologías. Para ello, se tomó como referencia la arquitectura de bases de datos federadas de Shet & Larson presentada en [Shet, A., and Larson, J. 1990]. Los objetivos específicos de este trabajo son:

- Extender la arquitectura de Bases de Datos Federadas de Shet&Larson para diferentes tipos de Bases de Datos (Bases de Datos Multimedia, Bases de Datos Orientadas a Objeto y Bases de Datos Inteligentes).
- Proponer un Modelo Ontológico[3] para el proceso de Integración de las Bases de Datos Federadas.
- Proponer un Modelo Ontológico para representar a cada uno de los siguientes tipos de Bases de Datos: Bases de Datos Multimedia, Bases de Datos Inteligentes, y Bases de Datos Orientados a Objeto.
- Presentar casos de uso y una posible implementación del marco de referencia de integración propuesto en la tesis.

Alcance y Limitaciones.

[3] Un modelo ontológico comprende atributos, relaciones entre atributos, axiomas y metodología de desarrollo

En esta tesis se desarrolla el modelo ontológico para el proceso de integración de bases de datos heterogéneas, así como también los modelos ontológicos de las posibles bases de datos componentes de una federación, como son las bases de datos inteligentes, las bases de datos multimedia, y las bases de datos orientadas a objeto. Además, se comprueba el uso de los mismos en diferentes casos de estudio (para diseñar bases de datos inteligentes de gestión de inscripción de alumnos en cursos, bases de datos multimedia de gestión de objetos de aprendizaje, y la integración de ambas).

A nivel de implementación, el alcance abarca incorporarlo en un sistema editor de ontologías (Protégé), para verificar la correctitud teórica del mismo. También, se instancia la misma en la herramienta gestora de ontologías, para los casos de estudio previos, para comprobar su uso desde ella (capacidad de razonar en diferentes situaciones, según los casos de estudio).

Una de las limitaciones que se presentan para la validación del modelo de integración es la poca posibilidad de obtener instancias reales de bases de datos federadas. También, otra limitación es la no disponibilidad de un software en el que se puedan incorporar los casos estudios elaborados en la tesis, para verificar si realmente el modelo ontológico funciona. Otra limitación tiene que ver con la no disponibilidad de herramientas gestoras de ontologías que incorporen todas las relaciones entre conceptos establecidas en los modelos ontológicos desarrollados en la tesis. Las herramientas gestoras de ontologías solamente permiten relaciones "es un" o "tiene", y en nuestro caso podríamos tener otros tipos de relaciones (por ejemplo: "forma", "es del tipo", "incorpora", etc.).

1.3 Estado del Arte

A continuación se describen algunos trabajos recientes en el área de integración de datos y ontologías.

En el trabajo Ontology-based integration for relational databases (Dejing Dou, Paea LePendu 2006) se realiza una integración de bases de datos relacionales con lógica de primer orden, basados en ontologías. Esto lo llamaron ONTOGrate, que combina representaciones de esquemas de bases de datos relacionales basados en ontología, inferencia en lógica de primer orden (utilizan el lenguaje Web-PDDL), y algunos wrappers en SQL, para integrar bases de datos relacionales. Utilizan reglas de transformación para llevar los esquemas a ontologías, como: Relation <-> Type; Attribute <-> Predicate; Integrity Constraint <-> Axiom; Primary Key <-> Fact. La integración de datos se realiza uniendo los esquemas expresados como ontologías en lógica de primer orden.

En [Collins S, Shamkant N., Mark Leo, 2002] presentan una estructura en XML Schema[4] como modelo de datos canónico para consultar e integrar datos desde fuentes heterogéneas. Así, utilizan XML Schema como modelo de datos unificado para la integración de datos. Ellos presentan un algoritmo para proyectar esquemas relacionales a XML Schema, lo que permite generar el modelo canónico de datos automáticamente.

En [Huimin Zhao, Sudha Ram, 2005] describen un sistema que detecta la correspondencia semántica de registros a través de un clasificador de identificación múltiple[5], para determinar cuando dos registros de diferentes bases de datos representan la misma entidad en el mundo real. Aplican diferentes técnicas para implementar al clasificador de identificación múltiple, tales como las máquinas de aprendizaje y las redes neuronales.

SEMEDA (Semantic MetaDatabase) es un sistema para integración y consulta semántica de bases de datos relacionales biológicos, la cual posee tres componentes [Kholer et al., 2003]: un componente que habilita el acceso SQL a las bases de datos que integran la federación (MARGBench), una ontología basada en la metabase de datos semántica[6] (SEMEDA), y una ontología de interfaz de consultas (SEMEDA-query). La idea principal de este trabajo es definir atributos de base de datos a través de referencias a conceptos ontológicos. Los proveedores de aplicaciones de bases de datos Web pueden introducir toda la semántica relevante y la información técnica acerca de sus bases de datos por sí mismos a través de la Web. En SEMEDA, las ontologías son consideradas como un conjunto de conceptos que están conectados por relaciones binarias (is-a, is-part etc). Los conceptos son entidades bien definidas: tienen un único significado, y propiedades como un nombre (etiqueta), una descripción y un identificador.

TAMBIS (Transparent Access to Multiple Biological Information) permite el acceso transparente a diferentes bases de datos y herramientas de análisis biológicas (Stevens R. et al. 2001). El sistema tiene una base de conocimientos de conceptos biológicos (el modelo biológico), un modelo de las fuentes de datos subyacentes[7], y una interfaz usuario "conocimiento-guiado". Los conceptos biológicos están representados en la base de conocimiento usando un lenguaje lógico. El modelo de conceptos proporciona al usuario los conceptos necesarios para construir múltiples formas de consulta, y la interfaz de usuario proporciona medios flexibles para manipular ésas consultas. El modelo fuente proporciona una descripción de las fuentes de datos y hace la relación entre los términos usados en las fuentes y los términos del modelo de conceptos biológico de TAMBIS. Esta aplicación

[4] **XML Schema** es un lenguaje de esquema (lenguajes que proporcionan las mismas funciones que un archivo DTD (Definición de Tipo de Documento) más los tipos de datos, herencia y normas de presentación) utilizado para describir la estructura y las restricciones de los contenidos de los documentos XML de una forma muy precisa, más allá de las normas sintácticas impuestas por el propio lenguaje XML

[5] Un clasificador de identificación múltiple consiste de un conjunto de clasificadores (sistema de producción de reglas) y una función de decisión combinatoria.

[6] Base de datos que almacena metadatos de un dominio específico

[7] Almacena los conceptos, relaciones e instancias de la ontología

permite responder a preguntas en el área de bioinformática para biólogos, y las bases de datos que accesa son base de datos relacionales.

Un trabajo clásico de integración de bases de datos de principios de los 90 es el de Shet & Larson, quienes crearon un modelo de integración por capas basados en la arquitectura ANSI/SPARC de tres capas, ampliándola después a cinco capas, llamada arquitectura de bases de datos federadas [Shet, A. & Larson, J., 1990]. Cada capa realiza un proceso de traducción y conversión a un modelo común (llamado Modelo Canónico de Datos), a través del cual se entenderán las bases de datos que conforman la federación. Así, en este modelo se transforman todas las bases de datos que componen la federación a un modelo canónico de datos por medio del cual se realizan todas las operaciones de actualización y consulta de las bases de datos federadas.

Así, el problema de integración de bases de datos ha sido estudiado desde los años 80, principalmente el problema de unión de un conjunto de esquemas dados en un esquema global (llamado integración de esquemas) (Batini, Lenzerini, y Navathe 1986; Elmagarmid y PU 1990; Shet y Larson 1990; Parent y Spaccapietra 1998). Recientemente se le ha prestado atención a la gestión de modelos de datos[8], donde la integración semántica desempeña un papel central en la unión y comparación de modelos (Bernstein 2003; Pottinger y Bernstein 2003; Rahm & Bernstein 2001).

Todos estos proyectos realizan integración de bases de datos relacionales heterogéneas, donde la heterogeneidad consiste en el tipo de datos que utilizan las diferentes bases de datos relacionales y el significado de los mismos. Este trabajo propone una ontología como modelo de integración que permita integrar a diferentes tipos de bases de datos (bases de datos inteligentes, bases de datos multimedia, bases de datos orientadas a objeto y bases de datos relacionales), sin importar el modelo en que estén expresadas (modelo entidad relación, orientado a objetos, relacional, etc.), resolviendo los conflictos que surjan durante el proceso de integración.

1.4 Organización de la Tesis

La organización de este documento es la siguiente: En el capítulo 1, "Introducción General", se presenta una descripción general del trabajo de tesis, y el estado del arte en materia de integración.

En el capítulo 2, "Marco Teórico", se presentan los basamentos teóricos necesarios para desarrollar la propuesta de este trabajo, como son: las bases de datos federadas y su

[8] Un modelo de datos es un sistema formal y abstracto que permite describir los datos de acuerdo con reglas y convenios predefinidos (Ullman, 1988). Gestión de modelo de datos: tareas que permiten la representación y/o manipulación de los modelos de datos

proceso de integración, y las ontologías con algunas metodologías para construirlas y lenguajes para implementarlas.

En el capítulo 3 se muestra, en una primera parte, la ontología para el proceso de integración de bases de datos federadas, y luego, los modelos ontológicos de las bases de datos que podrían conformar una federación, como son: las bases de datos inteligentes, las bases de datos multimedia, las bases de datos orientadas a objeto y las bases de datos relacionales.

En el capítulo 4 se muestran dos casos de estudio: el uso de los modelos ontológicos de las bases de datos inteligentes y de las bases de datos multimedia en un problema dado, y un ejemplo de integración de dichas bases de datos utilizando el modelo ontológico de integración mostrados en el capítulo 3.

En el capítulo 5 se presenta la implementación del ejemplo de la ontología de integración en OWL Protégé [Holger Knublauch et al., 2004]. Además, se muestra una instanciación en OWL Protégé de una operación que muestra el proceso de integración de una base de datos multimedia con una inteligente. Finalmente, se presentan las conclusiones donde se describen los resultados obtenidos y futuros trabajos a realizar. En el anexo 1 se describe de manera breve el funcionamiento y utilización de Protégé-OWL.

Capítulo 2

2. MARCO TEÓRICO

2.1 Introducción

En esta sección se presenta el marco teórico en el que se desenvuelve este trabajo, comenzando con el tema de integración de bases de datos, luego con el tema de bases de datos federadas, seguidamente se presentan las bases de datos componentes que serán objeto de descripción ontológica en nuestro trabajo (relacionales, inteligentes, orientadas a objetos y multimedia), y finalmente el tema de Ontologías.

2.2 Integración de Base de Datos

Hoy en día existen en las organizaciones bases de datos aisladas que requieren ser interoperables, lo cual significa que dichas bases de datos deben tener capacidades como compartir, interpretar y manipular información de manera uniforme [Lakshmanan et. al., 2001]. Para lograr la interoperabilidad de las bases de datos, se han dado algunas soluciones, algunas de las cuales son las siguientes:

- *Esquema global lógico (GS).* Este esquema es una vista virtual del conjunto de esquemas de las bases de datos a integrar, lo cual significa que no almacena datos físicamente pero permite el acceso a estos [Batini et. al. 1986]. Las ventajas de integrar por medio de un esquema global lógico son las siguientes:
  1. *Proporciona transparencia semántica a los usuarios:* lo cual significa que dichos usuarios no tienen la necesidad de conocer las estructuras de los esquemas de las bases de datos locales.
  2. *Los conflictos entre n-esquemas son reducidos a conflictos entre 2-esquemas:* el esquema global y el esquema de la base de datos local.
  3. *Permite el acceso a los datos:* lo cual significa que es posible que un usuario global pueda hacer actualizaciones en las fuentes de datos por medio de este esquema.
  4. *Respeta la autonomía de las fuentes de datos:* ya que no es necesario que las fuentes de datos tengan que ser cambiadas en cuanto a su estructura original.

  Las desventajas en el uso de este esquema son las siguientes:
  1. *El mantenimiento del esquema global es una actividad difícil de realizar:* ya que los cambios en las estructuras de los esquemas de las fuentes de datos locales deben ser reflejados en este esquema, desarrollar el mecanismo para esto es una tarea compleja.
  2. *El esquema global es un medio para establecer las correspondencias semántica entre los esquemas locales heterogéneos*: Sin embargo, frecuentemente no sólo

existen diferencias entre estos esquemas, sino también existe heterogeneidad entre los datos de las bases de datos locales. No es posible establecer esta heterogeneidad sólo con un esquema global. Para dar solución a la heterogeneidad de los datos, es necesario agregar componentes al esquema global, por ejemplo: procedimientos para hacer conversiones de datos, tipos de datos o expresiones, etc.

- *Esquema global físico*. Este esquema, además de representar el conjunto de esquemas de la base de datos a integrar, tiene la capacidad de almacenar sus datos. Las principales ventajas de integrar con un esquema global físico son las siguientes:
  1. *El esquema global físico representa una vista integrada del conjunto de bases de datos*.
  2. *Contiene a los datos de las bases de datos:* Significa que cada registro de datos de las bases de datos locales es almacenado redundantemente en este esquema. Esto es una ventaja importante, ya que proporciona el acceso directo a los datos.
  3. *El acceso a los datos* es más rápido debido a que los datos residen en dicho esquema.
  4. *Los datos se almacenan en este esquema:* No es necesario desarrollar componentes adicionales para el acceso a los datos. Por ejemplo: si un usuario realiza una operación SELECT, la operación se ejecuta directamente en el esquema físico. La operación no tiene que ser trasformada a una llamada SQL en la base de datos fuente.

  Las desventajas son las siguientes:
  1. *El costo de actualizar los datos de las bases de datos:* Se requiere que los datos sean replicados o enviados desde las fuentes de datos a este esquema.
  2. *Se requiere un mecanismo de actualización:* Como las operaciones se realizan directamente en el esquema físico, es probable que los usuarios no obtengan resultados vigentes porque los datos podrían no estar actualizados.
  3. *El mantenimiento de dicho esquema también es un problema*: Los cambios en las estructuras de los esquemas de las bases de datos locales deben ser reflejados en este esquema.
- Sin *esquema global*. Algunos autores [Litwin 1984, 1990, Litwin et. al. 1982, Litwin y Abdellatif, 1986] sostienen que el esquema global es difícil de mantener debido a la heterogeneidad de las bases de datos locales. Por lo tanto, se resuelve la heterogeneidad semántica mediante un lenguaje manejador de bases de datos múltiples, dicho lenguaje es el sustituto del esquema global. La integración mediante un lenguaje consiste en crear un *esquema conceptual* utilizando las sentencias de dicho lenguaje (por ejemplo, Multibase, Ingres, SYBASE, Pegasus). Por ejemplo: para integrar las bases de datos A, B y C la sentencia sería de esta manera: CREATE MULTIDATABASE (A,B,C). De igual forma, el lenguaje para manejar bases de datos múltiples debe dar soporte para ejecutar operaciones SQL en este esquema, tales operaciones pueden ser tanto de definición del esquema conceptual como manipulación de datos. Las ventajas de esta opción de integración son las siguientes:
  1. *No requiere el mantenimiento de un esquema global*.
  2. *Respeta la autonomía de las bases de datos locales*.

3. *Flexibilidad para trabajar con un número variable de bases de datos:* Debido a que no existe un esquema global, en cualquier momento es posible agregar o quitar bases de datos utilizando el lenguaje para manejar bases de datos múltiples.
4. *Esta opción es apta para integrar bases de datos en Internet*: ya que la información que se encuentra en esta red cambia constantemente, y trabajar con un esquema integrado se vuelve una actividad demasiado complicada.

Las desventajas son las siguientes:

1. *Los lenguajes para trabajar con bases de datos múltiples comerciales que se han creado son muy limitados*.
2. *La creación de un lenguaje para manejar bases de datos múltiples es una actividad más compleja que la creación de un esquema global*.

- Integrar las bases de datos por medio de un *data warehouse*[9] [Johnson 1999, Anahory y Murray, 1997]. El data warehouse permite solamente el almacenamiento y la extracción de un número determinado de información de las Bases de Datos Locales. Una de las características del data warehouse es que los datos que existen en su repositorio pueden ser consultados, pero no modificados.

El esquema global es el medio ideal para lograr la integración de un conjunto de bases de datos, sin embargo, las herramientas existentes son limitadas, es decir, sólo logran una vista global[10] de las bases de datos locales, sin hacer integración de tablas, atributos y datos de las bases de datos. Ahora bien, para la definición de un esquema global se requiere la solución de la heterogeneidad de las bases de datos locales. La heterogeneidad se puede presentar en diferentes grados o niveles:

- A *nivel SGBD*[11]*.* Se presenta cuando las bases de datos locales son manejadas por diferentes SGBD. Las diferencias en SGBD se pueden presentar a nivel del lenguaje del SGBD o por el modelo de las bases de datos que manejan (relacionales, objeto-relacional, orientadas a objetos, red, jerárquicas, deductivas, etc.). Las herramientas middleware, como ODBC/JDBC, CORBA, RMI y herramientas para crear sistemas de bases de datos federados, como IBM DB2 Information Integrator o Sybase ECDA, muestran que es posible solucionar la heterogeneidad a nivel del SGBD.
- A *nivel de semánticas.* Aun cuando las bases de datos locales son manejadas por el mismo SGBD, pueden tener diferencias en semánticas, las cuales son las diferencias que existen entre las bases de datos locales en la interpretación de los datos. Para la solución de la heterogeneidad semántica no existen herramientas que permitan realizar esta actividad. Nuestro modelo ontológico de integración, entre otras cosas, apunta a solucionar la heterogeneidad semántica.

[9] **almacén de datos** (del inglés *data warehouse*), es una colección de datos orientadas a un dominio, integrado, no volátil, y variable en el tiempo, que ayuda a la toma de decisiones de la empresa u organización
[10] Una **vista** de base de datos es un resultado de una consulta SQL de una o varias tablas; también se le puede considerar una tabla virtual. Las vistas tienen la misma estructura que una tabla: filas y columnas. La única diferencia es que sólo se almacena de ellas la definición, no los datos
[11] SGBD Sistema Gestor de Base de Datos (DBMS Database Management System)

El principal reto de la solución de la heterogeneidad semántica es hacer que el conjunto de esquemas heterogéneos se vea como un esquema global con una única semántica para el conjunto de bases de datos locales. Para la creación de este esquema global, se requiere la solución de conflictos entre los esquemas locales a integrar. Los conflictos entre las bases de datos son el resultado de las diferentes representaciones simbólicas en las bases de datos locales a nivel de datos, atributos y esquemas [Kim y Seo 1991]. De acuerdo a la clasificación de conflictos de Kim y Seo [Kim y Seo 1991], los conflictos se pueden clasificar en conflictos de esquema y de datos. Los conflictos de esquema son las diferencias entre tablas y atributos, y los conflictos de datos son las diferentes formas de almacenar el mismo valor. Nosotros, en nuestro trabajo, ampliamos esa noción de conflictos considerando los conflictos que se presentan en las Bases de Datos Inteligentes. A continuación se describen los conflictos propuestos por Kim y Seo [Kim y Seo 1991]:

I.- Conflictos de Esquema

A. Conflictos en tablas

a. Conflictos en nombrado de tablas

1) Nombres diferentes para tablas equivalentes. Ocurren cuando se asignan nombres diferentes a tablas semánticamente equivalentes (sinónimos)

2) Nombres iguales para tablas diferentes. Cuando se asigna el mismo nombre a tablas semánticamente diferentes (homónimos).

b. Conflictos en restricciones de Integridad. En esta categoría se incluyen las diferencias que pueden surgir con respecto a la selección de claves primarias, secundarias, ajenas[12] y conflictos de integridad referencial[13].

c. Conflictos en la organización de la información. El número de tablas requeridas para modelar la base de datos en cada componente puede diferir de acuerdo a la conceptualización de cada diseñador.

d. Conflictos entre las estructuras de las tablas

1) Atributos faltantes

2) Atributos faltantes pero implícitos

B. Conflictos en atributos

a. Conflictos de los nombres entre los atributos. Similar a los conflictos para el nombrado de tablas de sinónimos y homónimos pero entre atributos.

1) Diferentes nombres para atributos equivalentes.

2) Mismo nombre para diferentes atributos.

b. Conflictos en valores por ausencia. La definición implícita de algunos valores por ausencia asignados por el SGBD, podría llevar a contradicciones en la semántica de los datos.

c. Conflictos por restricción de asignación de valores entre los atributos: Conflictos en los tipos de datos. Los tipos de datos pueden diferir.

[12] Una *clave ajena* es un atributo o un conjunto de atributos de una relación, cuyos valores coinciden con los valores de la clave primaria de alguna otra relación. Las claves ajenas representan *relaciones entre datos*

[13] Estos tipos de conflictos son definidos en la sección 2.4.4, cuando se hable de las Restricción de Integridad Referencial para las Bases de Datos Relacionales.

d. Conflictos grado de atomicidad. El grado de atomicidad viene dado por el grado de detalle de cada atributo que puede ser distinto en cada aplicación.
e. Conflictos en la representación de la información. Puede presentarse el caso en el que el mismo concepto se presente como una entidad en una aplicación y como un atributo en otra aplicación.

II. Conflictos de datos

A. Conflictos entre valores. Cuando se espera que instancias equivalentes tengan los mismos valores, pero muestran inconsistencia debido a que los datos son capturados incorrectamente o los datos son obsoletos.

B. Diferencias en la representación de los datos. Situaciones de contexto o cultura organizacional puede llevar a que cada base de datos componente seleccione una representación distinta en la información. Estas inconsistencias se pueden presentar por:

1. Diferentes notaciones, cuando existen diferentes formas para representar un mismo dato (ej. calificaciones numéricas con escala de letras).
2. Diferentes unidades (Ej. la diversidad de unidades, sobre todo para valores numéricos).
3. Diferentes expresiones, cuando existen diferentes formas para representar el valor de un atributo (p. e. Para el atributo Estado se puede tener: Mérida, MRD, Mrd).

2.2.1 Arquitecturas para la Integración de Bases de Datos

García F. M. [García F. M. 2005] clasifica las herramientas para la integración de bases de datos en middleware[14], data warehouse y manejadores para Bases de Datos Múltiples.

- El middleware es un tipo de software que facilita la comunicación entre dos o más sistemas de software [Linthicum 2000], y proporciona un método sencillo que permite a una entidad, ya sea una base de datos o aplicación, comunicarse con otras entidades. Estos middleware se pueden implementar como objetos distribuidos, web services, o middleware orientado a bases de datos.
    - Los objetos distribuidos son pequeños programas de aplicación que usan interfaces estándar y protocolos para la comunicación [Linthicum 2000]. Por ejemplo, los desarrolladores pueden crear objetos distribuidos de CORBA (*Common Request Object Broker Architecture*) que trabajan sobre un servidor UNIX y otros que trabajen sobre un servidor NT. Ya que ambos objetos son creados usando un estándar (CORBA), y ambos objetos utilizan el mismo protocolo de comunicaciones (Inter-ORB),

[14] Capa de software entre las redes y las aplicaciones

entonces los objetos son capaces de intercambiar información y utilizar funciones que se encuentran en máquinas remotas.

- Para Linthicum [Linthicum 2000], el *middleware orientado a bases de datos* es un middleware que facilita la comunicación con una base de datos desde una aplicación, o entre bases de datos. Los mecanismos para el acceso a datos se basan en estándares como ODBC (*Open Database Connectivity*) o JDBC (*Java Database Connectivity*). Este tipo de middleware ofrece una solución de integración dentro del nivel de datos, ya que permite acceder directamente a diferentes fuentes de datos, o a las bases de datos.
- Los *web services* son protocolos y estándares que sirven para intercambiar datos en Internet [W3C 2004]. Actualmente, los SGBD pueden hacer uso de web services [Gray 2004], por lo tanto, se puede lograr integración de bases de datos por medio de estos. Sin embargo, ellos no resuelven el problema de heterogeneidad semántica.

- El data warehouse también se puede utilizar como un medio para integrar bases de datos, pero no proporciona herramientas para la solución de la heterogeneidad semántica.
- Los tipos de arquitecturas para la integración de bases de datos basadas en Sistemas Gestores de Bases de Datos Múltiples (SGBDM) se presentan en la figura 2. Dicha clasificación es de Sheth y Larson [Shet, A., & Larson, J. 1990], en donde se puede notar que cada Sistema de Bases de Datos (SBD) es manejado por un SGBD. Éstos se clasifican en dos tipos: federado y no federado. Los SGBD no federados son sistemas de bases de datos distribuidos no autónomos (pierden su autonomía al integrarse, pues todas sus operaciones las realizan a través del esquema global). Los *SGBD Federados* (**SGBDF**) se componen de sistemas de bases de datos que son autónomos, y permiten compartir de manera controlada una parte de sus datos. Un SGBDF puede ser *débilmente acoplado* o *fuertemente acoplado*. Un sistema es débilmente acoplado cuando el usuario es el encargado de mantener la federación, y fuertemente acoplado cuando el administrador de la base de datos es quien mantiene la federación. El proceso de integración en las bases de datos federadas fuertemente acopladas se realiza en dos fases: la fase de construcción de la federación, y la fase de operación de la federación. En la fase de construcción de la federación se realiza un análisis para decidir las estrategias de integración (selección de número de esquemas a integrar), y definir qué información se compartirá y con qué restricciones de acceso; luego se analizan y comparan los esquemas para determinar la existencia de posibles conflictos. Una vez determinada la equivalencia semántica, se define el tipo de integración y finalmente, se integran los esquemas. En la fase de operación se procesan las consultas de las Bases de Datos Componentes que interoperan, incluyendo el control de acceso, la descomposición de la consulta en sub-consultas, la

combinación de sub-resultados, entre otras cosas. En general, las Bases de Datos Federadas permiten resolver los problemas de heterogeneidad semántica.

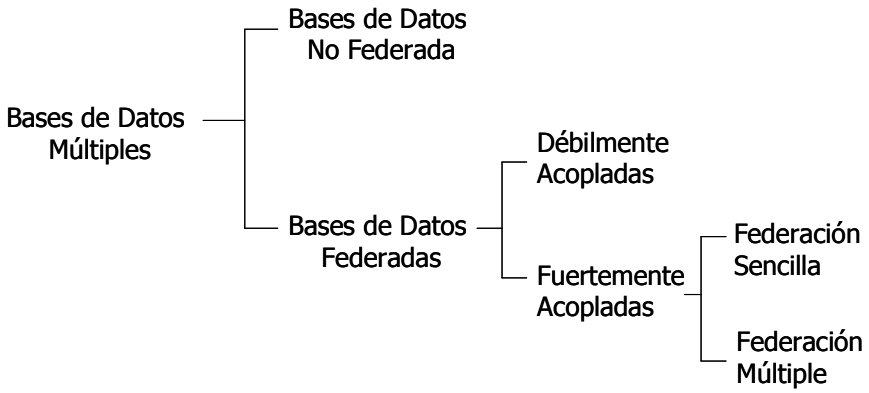

Figura 2.1 Taxonomía para las Bases de Datos Federadas

2.3 Bases de Datos Federadas

Los procesos que componen a una Bases de Datos Federada son:

- ***Proceso de transformación***, traduce comandos desde un lenguaje fuente a un lenguaje destino, o transforma datos de un formato fuente a un formato destino.
- ***Proceso de filtrado***, restringe los comandos que pueden ser usados en una federación.
- ***Proceso de construcción***, realiza tareas de integración de esquemas, negociación, optimización, descomposición de consultas, y administración de transacciones globales.
- ***Proceso de acceso***, acepta comandos y los ejecuta sobre una base de datos. Para realizar este proceso se requiere de los sistemas manejadores de las bases de datos componentes de la federación

En [Shet, A., & Larson, J. 1990] se describen los tipos de esquemas que componen las bases de datos federadas (ver figura 2.2):

- ***Esquema Local***: esquema conceptual de la base de datos componente, esta expresado en el modelo de datos nativo de la base de datos componente.
- ***Esquema Componente***: este esquema se deriva de la traducción de los esquemas locales al modelo de datos llamado "Modelo Común o Federado de Datos" (MCD) del sistema de base de datos federado. Este esquema facilita las tareas de integración que se realizan en un sistema de bases de datos federado fuertemente acoplado. También facilita la especificación de las vistas para un sistema de bases de datos federado débilmente acoplado. El proceso de transformación, transforma los comandos entendibles por el esquema componente en comandos entendibles por los esquemas locales correspondientes. El proceso de transformación y los esquemas componentes

soportan la característica de heterogeneidad de un Sistema de Base de Datos Federado.

- ***Esquema de Exportación***: representa un subconjunto de un esquema componente para ser usado durante la federación, incluye también información de control de acceso a ese esquema. El proceso de filtrado se utiliza para limitar el conjunto de operaciones que se pueden realizar sobre los esquemas componentes, y sobre qué parte de los datos se pueden hacer.
- ***Esquema Federado***: es la integración de los esquemas de exportación. El proceso de construcción transforma los comandos del esquema federado en comandos de los diferentes esquemas de exportación. El proceso de construcción y el esquema federado soportan la característica de distribución del las bases de datos federadas.
- ***Esquema Externo***: define un esquema para un usuario o para aplicaciones específicas.

La arquitectura de un sistema de base de datos federada está determinada por los esquemas que estén presentes, cómo estos esquemas se arreglan, y cómo se construyen.

La figura 2.2 muestra cómo se maneja la distribución, autonomía y heterogeneidad para integrar bases de datos componentes [Shet&Larson 1990]. Los esquemas locales representan los datos y modelos de datos de cada base de datos que compone una federación. Para comparar los datos objeto modelados en los diferentes modelos de datos, se tienen que realizar comparaciones directas y en par, o convertir los esquemas a un modelo común o modelo canónico preferiblemente con un poder de expresividad superior al modelo de las bases de datos componentes de la federación, para luego comparar los objetos. Los esquemas de exportación permiten manejar los aspectos de autonomía. La integración de los esquemas de exportación en esquemas federados permite el acceso integrado a objetos administrados por múltiples bases de datos componentes. La definición de esquemas externos permite define vistas de usuario o aplicaciones.

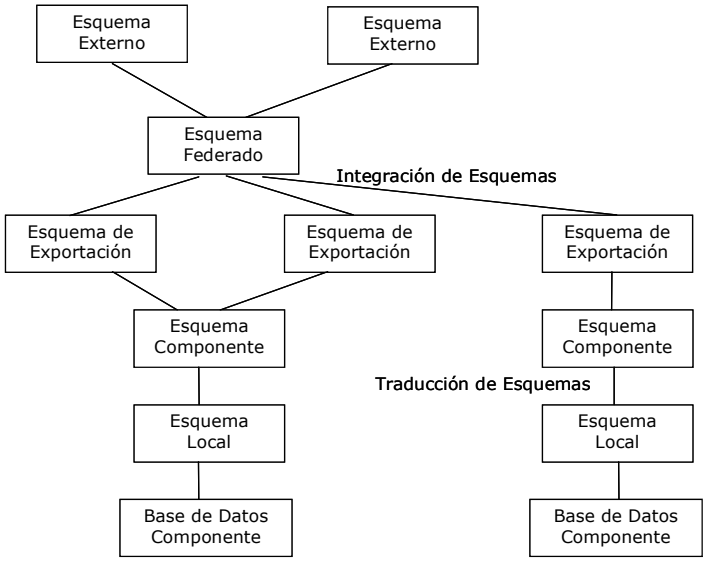

Figura 2.2 Arquitectura de Esquemas que componen las bases de datos federadas [Shet&Larson]

En [Batini, C.; Lenzerini, M.; and Navathe, S. 1986 y Shet, A., and Larson, J. 1990] se describen las diferentes formas de implementar bases de datos federadas. Ellos proponen dos maneras:, como un proceso bottom-up o como un proceso top-down. El proceso bottom-up se usa cuando se quiere agregar una base de datos componente a la federación, y se describe a continuación.

Traducción de esquemas: traduce el esquema local de una base de datos componente al esquema componente expresado en el Modelo Común de Datos (MCD). Además, desarrolla y actualiza el proceso de transformación que permite traducir los comandos expresados en el esquema componente a sus correspondientes en el esquema local.

Definir los esquemas de exportación: define los esquemas de exportación desde un esquema componente. Este paso es realizado por el administrador de la base de datos componente para autorizar la parte de la base de datos a ser incluida en el sistema de base de datos federada. Además, desarrolla o actualiza el proceso de filtrado apropiado.

Integrar esquemas: Selecciona el conjunto de esquemas de exportación a ser integrados. La integración de cada conjunto de esquemas de exportación producirá un esquema federado. Además, desarrolla o actualiza el proceso de construcción de tal manera que permite transformar los comandos expresados en el esquema federado en los comandos expresados en sus esquemas de exportación correspondientes. Este paso se repite para cada esquema de exportación y su correspondiente esquema federado.

Definir esquemas externos: Si es necesario definir esquemas externos para cada usuario o grupo de usuarios de la federación, se construyen los filtros y transformaciones necesarias.

El proceso top-down es utilizado cuando ya existe un sistema de base de datos federado y ocurren requerimientos adicionales de los usuarios. Estos requerimientos

adicionales surgen cuando se desarrolla una nueva aplicación y los datos que requiere no son soportados por los esquemas federados existentes, lo que implica eventualmente extender alguno de los esquemas federado o crear uno nuevo, o extender las bases de datos componentes existentes en la federación o crear una nueva base de datos componente. Este proceso se explica a continuación.

1. Definir o Modificar esquemas externos: recoger los requerimientos de los usuarios de la federación y analizarlos para definir nuevos esquemas externos o extensiones a los ya existentes.

2. Analizar esquemas: Compara los esquemas federados relevantes con los esquemas externos para identificar partes de los esquemas externos que existen en el esquema federado, y por lo tanto, soportado por el sistema de base de datos federado. Si alguna parte de un esquema externo no está soportado por el sistema federado, se debe extender o desarrollar el sistema federado para incluir esta parte. Esta parte no soportada es un esquema temporal. Una o más de las bases de datos componentes deben soportar este esquema temporal, para lo cual se debe hacer:

a. Si el dato requerido existe en una o más de las bases de datos componentes. En este caso, identificar el esquema componente que contiene la descripción de los datos requeridos y negociar con sus administradores para tener una descripción de estos datos colocados en un esquema de exportación con el acceso apropiado.
b. Si el dato requerido no está implementado en ninguna base de datos componente y una base de datos componente está dispuesta a colocar el dato en su base de datos. En este caso los esquemas local, componente y de exportación de las bases de datos respectiva son modificados.
c. Si el dato requerido no está implementado y ninguna base de datos componente ni administrador de base de datos componente está interesado en colocar el dato requerido en su base de datos. En este caso, el esquema temporal se implementa como una base de datos separada con su sistema manejador de base de datos componente, lo que requiere de un nuevo proceso de transformación. Así el esquema temporal se convierte en un nuevo esquema componente (pero también aparece un nuevo esquema local).

3. Integrar Esquemas: Integrar los esquemas temporales en la estructura de esquemas federados (solo es válido cuando el esquema temporal se convierte en un nuevo esquema componente).

El **modelo común de datos** es el modelo al que se convierten los esquemas de las bases de datos componentes que van a integrar la federación. Este modelo puede ser: Modelo Entidad-Relación, Modelo Entidad Relación Extendido, Orientado a Objeto, etc.

2.4 Bases de Datos Componentes

Las bases de datos componentes de una federación pueden ser bases de datos relacionales, bases de datos orientadas a objeto, bases de datos multimedia y bases de datos inteligentes (bases de datos deductivas, sistemas basados en conocimiento o sistemas expertos y bases de datos difusas).

2.4.1 Bases de Datos Inteligentes

Agrupamos dentro de las bases de datos inteligentes a las bases de datos activas, deductivas y sistemas basados en conocimiento.

2.4.1.1 Bases de Datos Activas

Una base de datos activa es aquella base de datos capaz de detectar situaciones de interés (evento) y actuar en consecuencia. Las bases de datos activas extienden la capacidad expresiva de las bases de datos relacionales incorporando reglas que modelan un comportamiento activo del sistema. Estas reglas permiten que el sistema reaccione de forma autónoma, es decir sin intervención del usuario, ante la ocurrencia de determinados eventos y se les denomina reglas activas [Mota H. Laura. (2002)]. Una regla activa consta de tres componentes:

- Evento: define el suceso o estímulo que dispara la regla.
- Condición: define las condiciones de contexto que se deben producir para que la regla sea ejecutada.
- Acción: define la acción o acciones que el sistema debe ejecutar de forma autónoma.

Debido a su estructura, Evento-Condición-Acción, otra forma de hacer referencia a una regla activa es con la expresión regla ECA. El significado de una regla ECA es el siguiente [Sharma Chakravarty 2005]:"cuando se produce el evento, si la condición se cumple entonces la acción debe ser ejecutada". Cuando el sistema detecta que ha sucedido el evento de una regla, se dice que ésta se ha disparado. Cuando su condición se evalúa a cierto y se ejecuta su acción, se dice que la regla ha sido procesada.

Mediante los sistemas de bases de datos activas se consigue un nuevo nivel de independencia de datos: la independencia de conocimiento. El conocimiento que provoca una reacción se elimina de los programas de aplicación y se codifica en forma de *reglas activas*. De este modo, al definirse las reglas como parte del esquema de la base de datos, son compartidas por todos los usuarios, en lugar de estar replicadas en todos los programas de aplicación. Cualquier cambio sobre el comportamiento reactivo se puede llevar a cabo cambiando solamente las reglas activas, sin necesidad de modificar las aplicaciones [Mota H. Laura, 2002].

Se han desarrollado numerosos sistemas activos que, aunque comparten nociones y conceptos, utilizan lenguajes diferentes y proponen modelos de ejecución distintos. Resulta pues difícil definir una base de datos activa desde un punto de vista formal que se ajuste a todos los sistemas ya existentes. Es por ello que se establece, de forma general, que una base de datos activa posee dos características fundamentales [Paton, N. W., 1998]:

1) El modelo de conocimiento, que define qué se puede expresar con una regla ECA; y

2) El modelo de ejecución, que define cómo se procesa una regla ECA en tiempo de ejecución.

El **modelo de conocimiento** de una base de datos activa, independientemente del lenguaje utilizado para su definición, considera las tres componentes de una regla ECA. Las formas más usuales que se pueden encontrar para la definición de eventos, condiciones y acciones son las siguientes:

- **EVENTO**: los eventos de una regla pueden ser operaciones de base de datos (p.e. inserción de una tupla en una relación), operaciones predefinidas por el usuario (p.e. envío de un mensaje a un determinado objeto), operaciones de control de transacciones (p.e. ejecución de la sentencia commit), excepciones (p.e. intento de acceso a ciertos datos sin la autorización necesaria); pueden ser también eventos temporales (p.e. cada lunes), o incluso eventos externos (p.e. el hecho de que una temperatura ha alcanzado un cierto valor). Se dice que un evento es primitivo si consta nada más que de una ocurrencia de alguna de las categorías descritas. Un evento es compuesto si se obtiene como combinación de varios eventos primitivos y/o compuestos. La combinación puede realizarse mediante conectivas lógicas (V, Λ, ¬), mediante la combinación secuencial, o por composición temporal.
- **CONDICIÓN**: Las condiciones de las reglas son expresiones lógicas escritas en el lenguaje de base de datos, o procedimientos de tipo lógico escritos en un lenguaje de programación. La ausencia de condición o el uso de un predicado especial (cierto o true), significa que la acción de la regla debe ejecutarse siempre que ésta se dispare.
- **ACCIÓN**: La acción de una regla ECA puede consistir en operaciones de base de datos, llamadas a procedimientos, mensajes al usuario, operaciones de control de transacciones, o la acción cambio de evento que sustituye el evento disparador por otro.

A través del **modelo de ejecución** se ejecutan el conjunto de reglas ECA, el sistema debe procesarlas cuando sea oportuno. Este procesamiento puede ser diferente en cada sistema, en función de las siguientes características:

- *Modos de acoplamiento evento-condición y condición-acción*. El modo de acoplamiento evento-condición (resp. condición-acción) define la relación existente, con respecto a la transacción activa, entre la detección de un evento relevante para una regla y la evaluación de su condición (resp. la evaluación de la condición y la ejecución de la acción). Existen tres modos de acoplamiento:

  - — Inmediato: en este modo, la condición es evaluada (resp. la acción es ejecutada) inmediatamente después de la detección del evento (resp. evaluación de la condición).
  - — Diferido: en este modo, la condición es evaluada (resp. la acción es ejecutada) justo antes de que la transacción en la que se ha detectado el evento (resp. evaluado la condición) sea confirmada.
  - — Desacoplado: en este modo, la condición es evaluada (resp. la acción es ejecutada) en una transacción distinta a aquélla en la que se detectó el evento (resp. se evaluó la condición).
- *Granularidad de las reglas*. Esta propiedad determina la relación entre la ocurrencia de un evento y la instanciación de la regla que dispara. Hay dos posibles granularidades a tener en cuenta:
  - — Orientada a la tupla: en este caso, si el evento afecta a varias tuplas, la instancia de la regla asociada a cada tupla es procesada una a una.
  - — Orientada al conjunto: en este caso, si el evento afecta a varias tuplas la regla que se dispara procesa todas las tuplas a la vez.
- *Efecto de los eventos*. Esta propiedad establece si los eventos son considerados aisladamente o si se considera su efecto global dentro de la transacción; en este último caso se dice que el efecto es neto.
- *Política de procesamiento de las reglas*. Esta propiedad establece cuándo se procesan las reglas que son disparadas por la ejecución de otras reglas. Existen dos modos:
  - — Iterativo: las reglas disparadas por cualquier motivo se van considerando una tras otra.
  - — Recursivo: cuando el procesamiento de una regla supone el disparo de otras, éstas deben ser procesadas antes de que acabe aquélla que las disparó.
- *Resolución de conflictos*. Esta propiedad determina qué sucede cuando hay varias reglas disparadas. Posibles alternativas para resolver este problema son:
  - — Aleatoriedad: la regla elegida para ser procesada se escoge arbitrariamente.
  - — Definición de prioridades: a las reglas se les asigna una prioridad absoluta o relativa a otras reglas. En este caso, al elegir una regla para ser procesada no puede haber ninguna otra disparada de prioridad mayor.

2.4.1.2 Sistemas Basados en Conocimiento y Bases de Datos Deductivas

Los sistemas basados en conocimiento (Knowledge Based Systems en ingles, KBS) y las Bases de Datos Deductivas (BDD) son aplicaciones que generan soluciones o respuestas satisfactorias a problemas que requieren un razonamiento por computadora que involucre conocimiento de algún tipo. Algunos tipos de conocimiento pueden ser *hechos* (que expresen proposiciones válidas) o *reglas* [Bertino E., Catania B., Zarri Gian P. 2001].

Los KBS y las BDD construyen su razonamiento para resolver problemas, concatenando hechos (afirmaciones) y reglas en líneas de razonamiento. Estas líneas de razonamiento muestran cómo un conjunto de suposiciones, afirmaciones y reglas, producen una conclusión particular. Algunas de las características básicas de los KBS y las BDD son: hacen una representación implícita del conocimiento, tienen capacidad de razonamiento independiente de la aplicación específica, tienen la capacidad de explicar sus conclusiones y el proceso de razonamiento [Bertino E., Catania B., Zarri Gian P. 2001].

Los elementos principales de un sistema basado en conocimiento y las bases de datos deductivas son:

- **Base de Conocimientos**: Representación del conocimiento del dominio del sistema, junto con el conjunto de información invariable (hechos). Una base de conocimientos consiste de dos componentes:
  - Un conjunto de datos, llamados hechos, que representa información explícita dada por el usuario; éstos datos son llamados base de datos extensional (EDB).
  - Un conjunto de reglas de inferencia, que codifican el conocimiento derivado de las relaciones que se dan entre los conceptos que se derivan de los hechos, estas reglas son llamadas base de datos intencionales (IDB).

  Las bases de conocimiento son la evolución lógica de los sistemas de bases de datos tradicionales, con la idea de manejar elementos de conocimiento (normalmente en forma de hechos y reglas), así como la manera en que éste ha de ser utilizado.
- **Máquina de Inferencia:** Proceso de razonamiento a partir de datos de entrada, tomando como base la base de conocimientos. Esta máquina es genérica en el sentido de que puede aplicarse a diferentes dominios con solo cambiar la base de conocimientos.

2.4.2 Bases de Datos Multimedia

En las Bases de Datos Multimedia se pueden distinguir dos tipos: las referenciales y las descriptivas. Las bases de datos referenciales son bancos de datos sobre material como películas, series de televisión o música. En la mayoría de los casos la información que se almacena hace referencia a cuestiones descriptivas (autor, título, duración, productor, etc.), o a cuestiones técnicas (formato, duración, etc.). Las bases de datos multimedia descriptivas, además de los datos técnicos y generales, aportan información específica sobre el contenido, indicando, por ejemplo, dónde se sitúan los cambios de plano en una película o la trascripción de un diálogo determinado.

En un sistema de base de datos multimedia, de acuerdo a como se realice el tratamiento de los datos (de manera integrada o no con el sistema de base de datos), se presentan diferentes arquitecturas (Prabhakaran B, 1997):

Arquitectura de Acoplamiento separado (loose coupling): Esta arquitectura está conformada por 3 módulos: el sistema manipulador de los metadatos[15] (lo hace un sistema manejador de Bases de Datos), el administrador de archivos multimedia, y el módulo integrador de ambos (Prabhakaran B, 1997).
Arquitectura de acoplamiento compacto (tight coupling): En este caso, el administrador de bases de datos maneja los archivos multimedia y los metadatos. Así, en esta arquitectura todas las funcionalidades del sistema administrador de bases de datos se aplican sobre la base de datos multimedia [Prabhakaran B, 1997].
Arquitecturas de Hipermedia: funciona básicamente como un hipertexto. El inicio del enlace o su origen está denotado por un "*link marke",* y el final del enlace o su destino es un "*link anchor"*.

Los tipos y funciones de los metadatos multimedia pueden clasificarse de acuerdo a su uso [H. Kosch and M. Döller 2005] en:

- *Metadatos para Texto.* El texto es el objeto media más simple, en términos de requerimientos de almacenamiento, representación, identificación del contenido de información, etc. Los siguientes tipos de metadatos se usan para describir objetos texto:
- *Descripción de contenidos de documentos*: proporciona información adicional del contenido del documento.
- *Representación del texto*: describe el formato, técnicas de codificación y compresión utilizadas para almacenar los datos.
- *Documento Histórico*: describe la historia de creación del documento. Este metadato es descriptivo.
- *Localización del documento*: este metadato independiente del contenido, describe la localización en donde se encuentra el documento (estación de trabajo o PC dentro de una red).
- *Metadatos para el Habla.* Se refiere al lenguaje hablado, y no es un tipo de dato independiente, sino que es considerado como parte del audio. Los metadatos generados para el habla pueden ser:
  - *Identificación de palabras habladas*. Este metadato consiste del tiempo inicial y final del discurso.
  - *Identificación del Hablante (speaker)*. Los metadatos generados como parte del reconocimiento del hablante se consideran descriptivos del contenido.
  - *Identificación de la información prosódica* que puede ser utilizada para llamar la atención de una frase o una oración, o alterar el significado de la palabra. Los metadatos que describen la información prosódica son dependientes del contenido.

[15] Un metadato describe atributos de los objetos de información, otorgándoles significado, contexto y organización

- *Metadatos para Imágenes.* Los metadatos de imágenes dependen del tipo de imágenes a ser analizadas y la aplicación que se utilizará para analizar los datos. Algunos de estos tipos de imágenes son:
- *Metadatos para Imágenes Satelitales*.
- *Metadatos para imágenes de diseño arquitectónico*.
- *Metadatos para Imágenes faciales.*
- *Metadatos para Video*.

2.4.3 Bases de Datos Orientadas a Objeto (BDOO)

Las bases de datos orientadas a objetos se crearon para tratar de satisfacer las necesidades de aplicaciones como el diseño y fabricación en ingeniería (CAD/CAM, CIM), los experimentos científicos, los sistemas de información geográfica, o los sistemas multimedia. Los requerimientos y las características de estas aplicaciones tienen una estructura de los objetos más compleja, las transacciones son de larga duración, se necesitan nuevos tipos de datos para almacenar imágenes y textos, y hace falta definir operaciones no estándar, especificas para cada aplicación. La orientación a objetos ofrece flexibilidad para manejar algunos de estos requisitos, y no está limitada por los tipos de datos y los lenguajes de consulta de los sistemas de bases de datos tradicionales [Cattell R., Barry D. 2000].

Las BDOO manejan el paradigma de la Programación Orientada a Objeto (POO), cuyos conceptos de base son los siguientes:

- Clases
- Herencia simple, múltiple y repetida.
- Polimorfismo.
- Objetos complejos

A continuación se describen estos conceptos:

- **Objeto:** es una instancia auto contenida de una entidad de interés del mundo real. Los objetos tienen un tipo de identificador único. Los tipos de objetos se descomponen en *atómicos*, *colecciones* y tipos *estructurados*. Los objetos pueden ser *transitorios* o *persistentes*. Los objetos transitorios existen mientras vive el programa de aplicación que los ha creado. Estos objetos se usan tanto como almacenamiento temporal para dar apoyo al programa de aplicación que se está ejecutando. Los objetos persistentes son aquellos que se almacenan en la base de datos [Cattell R., Barry D. 2000].
- **Clase**: Es un patrón o plantilla en la que se basan objetos que son similares. Cuando un programa crea un objeto de una clase, proporciona datos para sus variables, y el objeto puede entonces utilizar los métodos que se han escrito para la clase. Todos los objetos creados a partir de la misma clase comparten los mismos procedimientos para sus métodos, también tienen los mismos tipos para sus datos,

pero los valores pueden diferir. Una clase también es un tipo de datos. De hecho, una clase es una implementación de lo que se conoce como un tipo abstracto de datos. El que una clase sea también un tipo de datos significa que una clase se puede utilizar como tipo de datos de un atributo.

- **Herencia**. En ocasiones se necesita trabajar con clases que son similares pero no idénticas. Para ello es muy útil una de las características del paradigma orientado a objetos: la herencia. Una clase puede tener varias subclases que representan ocurrencias más específicas de la superclase. Por ejemplo, podemos tener la clase (superclase) Animal con sus atributos (nombre común, nombre científico, fecha de nacimiento y género), y las subclases Mamífero, Reptil y Pez, cada una con unos atributos específicos (Mamífero: peso, altura del hombro, raza y color; Reptil: longitud actual y longitud máxima; Pez: color). Por el hecho de ser subclases de Animal, heredan sus atributos. La relación que mantienen las subclases con la superclase es del tipo "es un": un mamífero es un animal, un reptil es un animal y un pez es un animal. No todas las clases de una jerarquía se utilizan para crear objetos. Por ejemplo, nunca se crean objetos de la clase Animal, sino que se crean objetos de las clases Mamífero, Reptil o Pez. La clase Animal sólo se utiliza para recoger los atributos y métodos que son comunes a las tres subclases. Se dice que estas clases son abstractas o virtuales. Cuando una clase hereda de más de una superclase se tiene *herencia múltiple*, y es *repetida* cuando hereda una superclase más de una vez, es un caso especial de la herencia múltiple.
- **Polimorfismo.** En general, las subclases heredan los métodos de sus superclases y los utilizan como si fueran suyos. Sin embargo, en algunas ocasiones no es posible escribir un método genérico que pueda ser usado por todas las subclases. Por ejemplo, la clase ObjetoGeométrico posee un método área que debería tener distinta implementación para sus subclases Círculo, Rectángulo y Triángulo. La superclase contendría un prototipo para el método que calcula el área, indicando sólo su interface pública. Cada subclase redefine el método, añadiendo las instrucciones necesarias para calcular su área. A esto último es lo que se le llama polimorfismo.

Otros conceptos interesantes de la POO son:

- **Tipos de clases**. En los programas orientados a objetos hay tres tipos de clases: clases de control, clases entidad y clases interface. Las clases de control gestionan el flujo de operación de un programa (por ejemplo, el programa que se ejecuta es un objeto de esta clase). Las clases entidad son las que se utilizan para crear objetos que manejan datos (por ejemplo, clases para personas, objetos tangibles o eventos). Las clases interface son las que manejan la entrada y la salida de información (por ejemplo, las ventanas gráficas y los menús utilizados por un programa).
- **Métodos:** especifican el conjunto de operaciones de una clase. Hay varios tipos de métodos que son comunes a la mayoría de las clases:

- Constructores. Un constructor es un método que tiene el mismo nombre que la clase. Se ejecuta cuando se crea un objeto de una clase. Por lo tanto, un constructor contiene instrucciones para inicializar las variables de un objeto.
- Destructores. Un destructor es un método que se utiliza para destruir un objeto. No todos los lenguajes orientados a objetos poseen destructores.
- Accesores. Un accesor es un método que devuelve el valor de un atributo privado de otro objeto. Así es como los objetos externos pueden acceder a los datos encapsulados.
- Mutadores. Un mutador es un método que almacena un nuevo valor en un atributo. De este modo es como objetos externos pueden modificar los datos encapsulados.

Además, cada clase tendría otros métodos dependiendo del comportamiento específico que deba poseer.

- **Interfaces.** Algunos lenguajes orientados a objetos no soportan la herencia múltiple. En lugar de eso permiten que una clase se derive de una sola clase, pero permiten que la clase implemente múltiples interfaces. Una interface es una especificación para una clase sin instrucciones en los métodos. Cada clase que implemente la interface proporcionaría las instrucciones para cada método de la misma. Una interface puede contener atributos y métodos, o sólo métodos.

2.4.3.1 El Modelo de Datos Orientado a Objeto

El modelo de datos orientado a objetos es una extensión del paradigma de programación orientado a objetos, con la diferencia que los objetos de la base de datos poseen persistencia, pues permanecen en la base de datos.

Existen distintas notaciones o modelos para diseñar esquemas conceptuales de bases de datos orientadas a objetos: la notación de Coad/Yourdon, la Shlaer/Mellor, la OMT (Rombaugh), o la de Booch. Cada modelo presenta distintas deficiencias, por lo que algunos de sus autores han desarrollado conjuntamente un lenguaje, denominado UML (*Unified Modeling Language*), que las evita. La notación UML (no hay que confundir con las metodologías que utilizan dicha notación), se ha convertido desde finales de los 90 en un estándar para modelar con tecnología orientada a objetos todos aquellos elementos que configuran la arquitectura de un sistema de información y, por extensión, de los procesos de negocio de una organización.

ODMG (*Object Database Management Group*) es un grupo de representantes de la industria de bases de datos que tienen como objetivo definir estándares para los SGBDOO. El modelo de objetos ODMG permite que tanto los diseños, como las implementaciones, sean portables entre los sistemas que lo soportan. La última versión del estándar, ODMG

3.0 propone los siguientes componentes principales de la arquitectura ODMG para un SGBDOO:

- Modelo de objetos.
- Lenguaje de definición de objetos (ODL, Object Definition Language).
- Lenguaje de consulta de objetos (OQL, Object Query Language).
- Conexión con los lenguajes C++, Smalltalk y Java (al menos).

ODL es un lenguaje para definir la especificación de los tipos de objetos para sistemas compatibles con ODMG. ODL es el equivalente al DDL (Data Definition Languaje o lenguaje de definición de datos) de los SGBD tradicionales. Define los atributos y las relaciones entre tipos, y especifica la signatura de las operaciones. Su principal objetivo es el de facilitar la portabilidad de los esquemas entre sistemas compatibles, al mismo tiempo que proporciona interoperabilidad entre distintos SGBD. La sintaxis de ODL extiende el lenguaje de definición de interfaces (IDL) de la arquitectura CORBA (Common Object Request Broker Architecture).

OQL es un lenguaje declarativo del tipo de SQL que permite realizar consultas sobre bases de datos orientadas a objetos. Está basado en SQL-92, proporcionando un superconjunto de la sentencia SELECT. OQL no posee primitivas para modificar el estado de los objetos, ya que éstas se deben realizar a través de los métodos que dichos objetos poseen. La sintaxis básica de OQL es una estructura SELECT...FROM...WHERE..., como en SQL.

El modelo de objetos ODMG permite que tanto los diseños como las implementaciones, sean portables entre los sistemas que lo soportan. El modelo de datos ODMG dispone de las siguientes primitivas de modelado [Cattell R., Barry D. 2000]:

- Los ***objetos* y los *literales***. Un objeto es una instancia auto contenida de una entidad de interés del mundo real. Los objetos tienen algún tipo de identificador único. Un literal es un valor específico, como "Amparo" o 36. Los literales no tienen identificadores. Un literal no tiene que ser necesariamente un solo valor, puede ser una estructura o un conjunto de valores relacionados que se guardan bajo un solo nombre.
- Los objetos y los literales se categorizan en ***tipos***. Cada tipo tiene un dominio especifico compartido por todos los objetos y literales de ese tipo. Los tipos también pueden tener comportamientos. Cuando un tipo tiene comportamientos, todos los objetos de ese tipo comparten los mismos comportamientos. En el sentido práctico, un tipo puede ser una clase de la que se crea un objeto, una interface o un tipo de datos para un literal (por ejemplo, *integer*). Un objeto se puede pensar como una *instancia* de un tipo.
- Lo que un objeto sabe hacer son sus ***operaciones***. Cada operación puede requerir datos de entrada (*parámetros de entrada*) y puede devolver algún valor de un tipo conocido.

- Los objetos tienen ***propiedades***, que incluyen sus ***atributos*** y las ***relaciones*** que tienen con otros objetos. El *estado* actual de un objeto viene dado por los valores actuales de sus propiedades. Un **atributo** se define del tipo de un objeto. Un atributo no tiene identificador, pero toma como valor un literal o el identificador de un objeto. Las **relaciones** se definen entre tipo, y son relaciones binarias con cardinalidad 1:1, 1: n y n:m.
- Una ***base de datos*** es un conjunto de objetos almacenados que se gestionan de modo que puedan ser accedidos por múltiples usuarios y aplicaciones.
- La definición de una base de datos está contenida en un ***esquema*** que se ha creado mediante el lenguaje de definición de objetos ODL (*Object Definition Language*).

Las bases de datos orientada a objetos soportan dos tipos de **herencia**: la relación "es un" y la relación "extiende". La relación "es un", que también se conoce como *generalización–especialización*, crea una jerarquía donde las subclases son tipos específicos de las superclases. Con la relación "extiende", una clase expande su superclase en lugar de estrecharla en un tipo más especifico [Cattell R., Barry D. 2000].

Una clase puede tener una o más **claves**. Una clave es un identificador único. Cuando una clave está formada por una sola propiedad, es una *clave simple*; si está formada por varias propiedades, es una *clave compuesta*.

2.4.4 Bases de Datos Relacionales

Una base de datos relacional consiste en una colección de tablas en la que todas sus entradas son datos atómicos como número, texto, fecha, etc. Las bases de datos relacionales están basadas en la teoría de conjuntos. Los elementos de una relación se denominan *tuplas,* el número de tuplas en una relación es la cardinalidad de la relación, y el número de atributos indica el grado de una relación [Date C. J. 2005]. En una base de datos relacional se encuentran los siguientes conceptos:

- ***Esquema*** que define los atributos que conforman una relación, sus claves propias o primarias[16] y ajenas. Es lo que corresponde al concepto de tabla.
- ***Interrelaciones*****:** tipo de relación que poseen los esquemas entre sí y que definen el comportamiento de la base de datos. El tipo de interrelación está dado de acuerdo a la cardinalidad en las relaciones: uno-a-uno cuando la clave primaria o propia de un esquema se convierte en la clave ajena de la otra; relación uno-a-muchos, cuando la clave de la relación del lado de uno se convierte en clave ajena de la relación del lado de muchos; y muchos-a-muchos, donde las claves propias de ambas relaciones generan una nueva relación [Date C. J. 2005].

[16] Clave primaria o propia es aquella que identifica de manera única una ocurrencia en una tabla

- El ***dominio D*** de una tabla *R* es la colección de valores posibles para un determinado atributo. Un atributo posee un valor atómico.
- La ***clave primaria*** de la relación o tabla se define como un subconjunto *C* de los atributos de *R*, cuyos valores no pueden ser repetidos. Una clave primaria puede ser simple (formada por un solo atributo) o compuesta (formada por más de un atributo) [Date C. J. 2005].
- La **clave ajena** son aquellas claves primarias de una relación o tabla que se encuentran en otras relaciones o tablas, consiguiendo así la interrelación entre las tablas o relaciones que interesan.

Las operaciones del modelo relacional se clasifican en consultas y actualizaciones.

- Las operaciones de actualización vienen dadas por operaciones para insertar, modificar y eliminar una o más tuplas de una relación dada [Date C. J, H Darwen, 2007]. La operación **Insertar** proporciona una lista de valores de los atributos para una nueva tupla t que se insertará en una relación R. La operación **Eliminar** elimina una tupla t de una relación R. La operación **Modificar** sirve para cambiar los valores de uno o más atributos en una tupla de una relación R. Surgen violaciones solo si se modifica un atributo clave, externa o primaria.
- Las **operaciones de consulta** son las operaciones estándar del álgebra relacional que permiten seleccionar tuplas y combinar tuplas a partir de varias tablas, dando como resultado una nueva tabla. Se clasifican en dos grupos: las operaciones de la teoría de conjuntos que son la **Unión**, la **Intersección**, la **Diferencia** y el **Producto Cartesiano**, y las operaciones propias del álgebra relacional como **seleccionar**, **proyectar** y **reunir** [Date C. J. 2005].

Las bases de datos relacionales poseen restricciones que deben ser consideradas al realizar operaciones en las bases de datos, y en el caso de que no se cumplan, las acciones a tomar. Particularmente, consideraremos las restricciones de integridad [Date C. J. 2005].

- **Restricciones de Dominio**. Son las restricciones sobre los tipos de valores que puede tener un atributo. Especifican que el valor de cada atributo A debe ser un valor atómico del dominio dom(A) para ese atributo.
- **Restricciones de clave**. El valor del atributo que identifica de manera única una tupla en una relación. Se especifica sobre relaciones individuales.
- **Restricción de Integridad Referencial**. Consiste en que los esquemas que se están integrando se encuentran en el mismo dominio y no tienen conflictos de claves, tanto primarias como ajenas
- **Restricciones generales o restricciones de integridad semántica**, son aquellas que se establecen respecto a las solicitudes a realizar en las consultas o vistas, por ejemplo: el número máximo de horas que un profesor trabaja en un proyecto por semana es 40. Pueden ser restricciones de integridad estáticas

(CREATE ASSERTION) o restricciones de integridad de transición (Disparadores (Trigger)).

Las operaciones de actualización: insertar, modificar y eliminar, no deben violar las restricciones de integridad especificadas anteriormente. La operación de inserción podría violar cualquiera de los cuatro tipos de restricciones: *de dominio* si proporciona un valor de atributo que no se encuentra en el dominio correspondiente; *de clave* si el valor clave de la nueva tupla t ya existe en otra tupla de la relación r(R); de *entidades* si la clave primaria de la nueva tupla t es nula, y de *integridad referencial* si el valor de cualquier clave externa de t hace referencia a una tupla que no existe en la relación referida. Si una operación de inserción viola una o más restricciones se pueden realizar las siguientes acciones: rechazar la inserción o corregir la razón por la que se rechazó la inserción. La operación Eliminar sólo podría violar la integridad referencial, si las claves externas de otras tuplas de la base de datos hacen referencia a la tupla que se va a eliminar. Existen tres acciones a tomar: 1) rechazar la eliminación; 2) tratar de propagar la eliminación, eliminando las tuplas que hacen referencia a la tupla que se desea eliminar; y por último, 3) modificar los valores del atributo de referencia que provocan la violación, todos esos valores se pondrían nulos, siempre y cuando no sean claves. En la operación Modificar surgirían violaciones solo si se realizan modificaciones a un atributo clave, externa o primaria, y se resuelven de la misma manera que para insertar o eliminar.

2.5 Ontología

El término de ontología fue tomado de la filosofía, donde significa "*una explicación sistemática del ser*". En las últimas décadas esta palabra ha tomado relevancia en el mundo de la Ingeniería del Conocimiento, la Inteligencia Artificial y las Ciencias de la Computación. A continuación se escriben algunas definiciones acerca de lo que es una ontología desde estos puntos de vista. Una primera definición fue dada por Neches y sus colegas [Neches R., et al., 1991], quienes definen una ontología de la siguiente manera:

"*Una ontología define los términos y las relaciones básicas que abarcan el vocabulario de un área específica, así como las reglas para combinar términos y relaciones para definir extensiones a dicho vocabulario".*

Esta definición identifica términos básicos y las relaciones entre términos. Además, incluye el conocimiento que puede ser inferido de los términos y sus relaciones. Gruber [Gruber, T. R. 1993] define una ontología como:

"*Una ontología es una especificación explícita de una conceptualización"*

En [Guarino N. and P. Giareta, 1995] proponen la definición de una ontología como:

"*Una teoría lógica que describe una conceptualización parcial y explícita"*

Donde una conceptualización es básicamente la idea que pueden tener del mundo una persona o grupo de personas. Weigand [Weigand, H**.** 1997] define una ontología desde el punto de vista de las bases de datos de la siguiente manera:

"*Una ontología es una base de datos describiendo los conceptos del mundo o de un dominio, sus propiedades, y cómo se relacionan esos conceptos*".

La comunidad diferencia las ontologías que son principalmente taxonomías, de las ontologías que modelan el dominio de una manera más detallada en la semántica de un dominio[17], llamándolas ontologías ligeras y ontologías pesadas, respectivamente. Las ontologías ligeras incluyen conceptos, taxonomía de conceptos, relaciones entre conceptos y propiedades que describen los conceptos. Las ontologías pesadas agregan axiomas y restricciones a las ontologías ligeras. Los axiomas y las restricciones aclaran el significado entendido de los términos contenidos en la ontología.

Desde el punto de vista de la Inteligencia Artificial las ontologías permiten definir un vocabulario con lógica [Fridman N., Hafner C. 1997]. El conocimiento en las ontologías se formaliza a través de seis componentes: clases, atributos, relaciones, funciones, axiomas e instancias [Gruber, T. R. 1993].

- Las **clases** o **conceptos** en la ontología se organizan en taxonomías.
- Los **atributos** representan la estructura interna de los conceptos. De acuerdo a su origen se clasifican en específicos y heredados. Los específicos son los propios del concepto al que pertenecen, y los heredados vienen dados por las relaciones taxonómicas en las que el concepto desempeña el rol de hijo, y por tanto, hereda los atributos del padre. Los atributos se caracterizan por el dominio en el cual pueden tomar valor.
- Las **relaciones** representan la interacción y enlace entre los conceptos de un dominio. Suelen formar la taxonomía del dominio. Ejemplos son: "subclase de", "parte-de", "conectado-a".
- Las **funciones** son un tipo concreto de relación donde se identifica un elemento mediante el cálculo de una función que considera varios elementos de la ontología. Por ejemplo pueden aparecer funciones como asignar-fecha.
- Los **axiomas** son teoremas que se declaran sobre relaciones que deben cumplir los elementos de la ontología. Los axiomas permiten inferir conocimiento que no esté indicado explícitamente en la taxonomía de conceptos.
- Las **instancias**, son las ocurrencias en el mundo real de los conceptos. En una instancia, todos los atributos del concepto tienen asignado un valor concreto [Gruber, T. R. 1993].

[17] Interpretación del significado de un determinado dominio

Inicialmente, las ontologías fueron construidas con técnicas basadas en frames[18] y en lógica de primer orden (Ej., Cyc), luego fueron utilizadas otras técnicas de representación del conocimiento para construir ontologías basadas en lenguajes de descripciones lógicas, como OIL y OWL basados en lógica de predicado de primer orden, que aparecen en el contexto de la Web Semántica[19].

Las ontologías tienen diversas aplicaciones, por ejemplo:

- Como repositorios de conocimientos e información para una organización, tanto de tipo corporativo como científico.
- Como herramienta para la adquisición de información en situaciones en las que un equipo de trabajo la utiliza como soporte común para la organización del dominio.
- Como base para la construcción de lenguajes de representación del conocimiento.

2.5.1 Tipos de Ontologías

2.5.1.1 Ontologías basadas en la Conceptualización

Ontologías de Representación del Conocimiento Una ontología de representación del conocimiento [Van Heijst et al., 1990], enlaza el modelado de primitivas utilizadas para formalizar el conocimiento bajo un paradigma de representación de conocimiento dado. Las más representativas son Frame Ontology y OKBC Ontology, disponibles en el Ontolingua Server. Proporcionan definiciones formales de la representación de primitivas utilizadas en lenguajes basados en Frames (esto es, clases, subclases, atributos, valores, relaciones y axiomas), permitiendo construir nuevas ontologías por medio de convenciones basadas en frames.

Ontologías Generales [Van Heijst et al., 1996] u **Ontologías Comunes** se usan para representar conocimiento de sentido común reutilizable entre dominios. Estas ontologías incluyen vocabulario relacionado a cosas, eventos, tiempo, espacio, causalidad, comportamiento, función, mereología[20], etc.

Ontologías de Nivel Superior (Top-Level), describen conceptos muy generales y proporcionan nociones generales bajo las cuales todos los términos raíces en ontologías existentes pueden estar enlazados.

[18] estructura de datos que contiene una descripción general de un objeto, que se deriva de conceptos básicos y de la experiencia

[19] La **Web semántica** (del inglés *semantic web*) consiste en añadir metadatos semánticos a la World Wide Web. Esas informaciones adicionales —describiendo el contenido, el significado y la relación entre los datos— deben ser dadas de manera formal, de manera de poder evaluarlas automáticamente.

[20] la mereología permite definir la propiedad de descomposición (relación Part-Of) para un concepto bajo descripción

Ontologías de Dominio, son reutilizables en un dominio específico (ej., médico, farmacéutico, ingeniería, empresa, leyes, etc.). Estas ontologías proporcionan vocabularios acerca de los conceptos y sus relaciones dentro de un dominio, acerca de las actividades que ocurren en ese dominio, y las teorías y principios elementales que dirigen dicho dominio.

Ontologías de Tareas, describen el vocabulario relacionado a una tarea o actividad genérica (por ejemplo, diagnóstico, programación, ventas, etc.) por especialización de términos. Las ontologías de tareas proporcionan un vocabulario de los términos utilizados para resolver problemas asociados con tareas que pueden o no pertenecer al mismo dominio.

Ontologías de Métodos, proporciona definiciones de los conceptos y relaciones aplicadas a procesos de razonamiento específico para realizar una tarea particular.

Ontologías de Aplicación, son dependientes de la aplicación. Contienen todas las definiciones necesarias para modelar el conocimiento requerido de una aplicación particular. Las ontologías de aplicación frecuentemente extienden y especializan el vocabulario de las ontologías de dominio y tarea de una aplicación particular.

2.5.1.2 Ontologías basadas en su estructura interna

Lassila y McGuiness [Lassila O, McGuinness D., 2001] clasificaron las ontologías de acuerdo a la información ontológica necesaria para expresar y enriquecer sus estructuras internas. Establecen las siguientes categorías: vocabularios controlados, glosarios, tesauros, jerarquías informales is-a, jerarquías formales is-a, instancias formales, frames, valor-restringido, y de restricciones lógicas generales. Las mismas se explican más abajo. En la siguiente figura se muestra esa categorización.

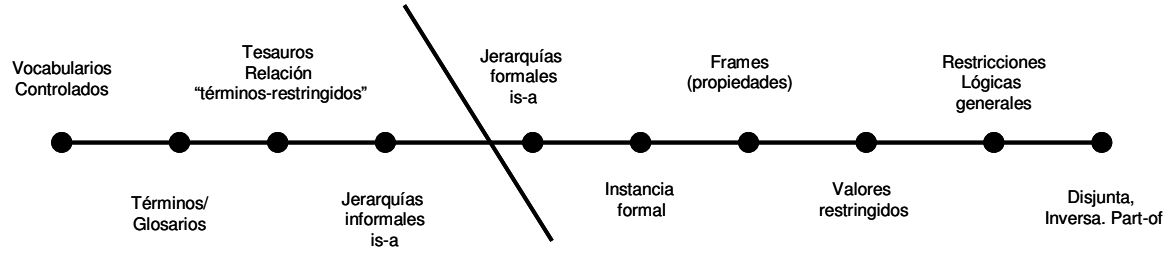

Figura 2.3 Categorización de Lassila y McGuiness

- Vocabularios controlados: una línea finita de términos. Un ejemplo típico de esta categoría es un catálogo.
- Glosarios: una lista de términos con sus significados especificados.

- Tesauros: es un vocabulario controlado y dinámico de términos que tienen entre ellos relaciones semánticas y genéricas, y que se aplica a un dominio concreto del conocimiento.
- Jerarquías informales is-a, solo definen jerarquías de clases y subclases (sin propiedades).
- Jerarquías formales is-a. En este caso se describe la jerarquía anterior, pero con propiedades. Las jerarquías de subclases estrictas son necesarias para explotar la herencia. Además, las jerarquías formales is-a incluyen instancias. Así, en estos sistemas si B es una subclase de A y un objeto es una instancia de B, entonces el objeto es una instancia de A.
- Frames. La ontología incluye clases y sus propiedades, las cuales pueden ser heredadas por clases de niveles inferiores en las taxonomías formales "is-a".
- Ontologías que expresan restricciones de valores. Estas son ontologías que pueden colocar restricciones en los valores que tiene una propiedad.
- Ontologías que expresan restricciones lógicas generales. Son las más expresivas. Los ontólogos pueden especificar restricciones lógicas de primer orden entre términos, utilizando lenguajes de descripción lógica (OWL, OIL, etc.).

Finalmente, existen una serie de ontologías que están clasificadas en función de su utilidad, propuestas por Lassila y McGuiness [Lassila O, McGuinness D., 2001]:

- Ontologías de contenidos para reutilización de conocimientos. Estas ontologías incluyen otras subcategorías: ontologías de tareas, ontologías de dominios y ontologías generales.
- Ontologías de comunicación para compartir conocimiento.
- Ontologías de indexación para la recuperación de casos.
- Meta-Ontologías, que se refieren a ontologías de representación del conocimiento.

2.5.2 Metodologías para el diseño de Ontologías

Para el diseño de una ontología existen diferentes metodologías, la mayoría comienza el proceso de diseño identificando el propósito de la ontología y adquiriendo el conocimiento del dominio. Luego, este conocimiento se representa de diferentes maneras, utilizando lenguajes formales o utilizando representaciones intermedias. Después, se construye la ontología [Gómez-Pérez et al. 2004]. Para el proceso de construcción de una ontología generalmente todas las metodologías definen tres actividades, la figura 2.6 muestra estas actividades [Gómez-Pérez et al. 2004].

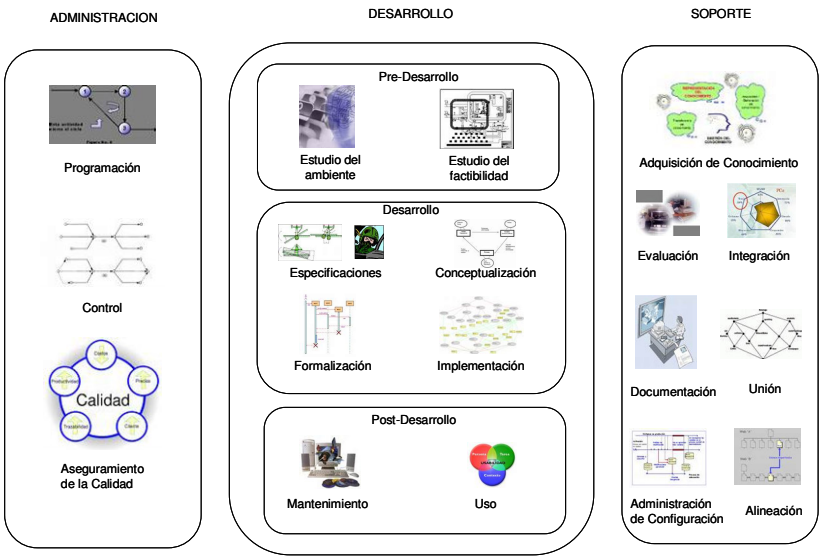

Figura 2.4 Proceso de desarrollo de una Ontología

En *administración* se realizan las siguientes actividades:

- Programación: en esta actividad se identifican las tareas a realizar, su alcance, y el tiempo y recursos para realizarla.
- Control: garantiza que las tareas programadas se finalicen como se planificaron.
- Aseguramiento de la calidad: asegura que la calidad de los productos (ontología, software y documentación) sea la adecuada.

En el *desarrollo* de una ontología se realizan las siguientes actividades:

- Pre-desarrollo: se hace un estudio del ambiente para conocer las plataformas donde la ontología será utilizada, aplicaciones con las que se integrará, etc. Además, se lleva a cabo un estudio de factibilidad para buscar las respuestas a ¿es posible construir esta ontología?, ¿es deseable construir la ontología?, etc.
- Desarrollo: tiene actividades de *especificación* que establecen por qué la ontología se desarrolla, su posible uso, y quiénes son los usuarios finales. También tiene una actividad de *conceptualización* que estructura el dominio de conocimiento en un modelo formal. La actividad de *implementación* construye modelos computarizados en un lenguaje ontológico.
- Post-desarrollo: durante el post desarrollo las actividades de mantenimiento actualizan y corrigen la ontología en caso de que sea necesario. También la ontología es reutilizada por otras ontologías o aplicaciones.

Finalmente, *las actividades de soporte* de administración y desarrollo se llevan a cabo permanentemente, y consisten en:

- Adquisición de conocimiento: adquiere conocimientos de expertos de un dominio dado, o a través de procesos (semi)automáticos llamados ontology learning.

- La actividad de evaluación realiza una evaluación técnica de las ontologías, de su ambiente de software, y de la documentación.
- La integración ocurre cuando se crea una nueva ontología por reutilización de otras ontologías. La ontología resultante debe poder unificar conceptos, terminología, restricciones, etc.
- La actividad de alineación establece diferentes clases de enlaces entre las ontologías involucradas.
- La actividad de documentación detalla claramente cada una de las etapas completadas y productos generados.
- La administración de configuración registra todas las versiones de la documentación y de la ontología para el control de cambios.

2.5.2.1 Metodología de Uschold y King

Para construir una ontología de acuerdo a Uschold y King [Uschold M., King M. 1995] se realizan los siguientes procesos: 1) Identificar el propósito de la ontología, 2) Construirla, 3) Evaluarla y 4) Documentarla. Durante el proceso de construcción los autores proponen capturar conocimiento, codificarlo e integrar otras ontologías dentro de la ontología obtenida. Estos procesos se muestran en la figura 2.7.

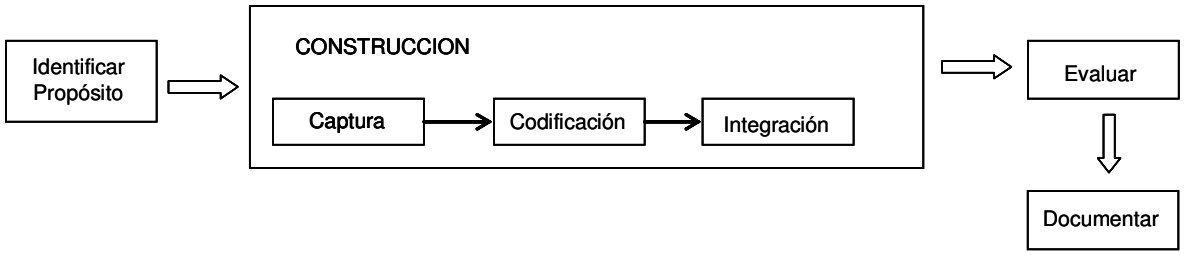

Figura 2.5 Metodología de Uschold y King

Identificar el propósito y alcance. En este proceso se clarifica el por qué se va a construir la ontología, cuáles son sus usos y cuáles serán los términos relevantes del dominio.

Construir la Ontología. Está conformada por tres actividades:

- *Captura de la Ontología*: se identifican conceptos y relaciones claves en el dominio a tratar; se producen definiciones precisas y sin ambigüedad para esos conceptos y relaciones; se identifican los términos que se refieren a tales conceptos y relaciones. Para identificar los conceptos Uschold y King utilizan tres estrategias: aquellas que construyen la ontología aplicando un proceso bottom-up, que identifica primero los conceptos más específicos y luego los generalizan en conceptos más abstractos; o proceso top-down, donde se identifican primero los conceptos más abstractos y luego los más especializados. Finalmente, esta la estrategia intermedia que recomienda identificar primero los términos básicos y los especifican o generalizan según sea necesario.

- *Codificación*: En esta actividad se reúnen los términos básicos que serán utilizados para especificar la ontología.
- *Integración de Ontologías Existentes*: Esta actividad se refiere a cómo y cuándo usar ontologías que ya existen. La integración se puede hacer en paralelo con las actividades de este proceso.

Evaluar. A partir de un marco de referencia se evalúa la ontología obtenida.

Documentar. En este proceso se establece la guía para documentar la ontología de acuerdo al tipo y propósito de la ontología. Un ejemplo puede ser localizar definiciones similares o crear convenciones de nombrado, como letras mayúsculas o minúsculas para nombrar términos, o escribir los términos de la ontología en letras mayúsculas.

2.5.2.2 Metodología de Grüninger y Fox

Esta metodología está inspirada en el desarrollo de sistemas basados en conocimientos usando lógica de primer orden. Proponen identificar intuitivamente los escenarios principales, esto es, posibles aplicaciones en las que la ontología será utilizada. Para ello utilizan un conjunto de preguntas en lenguaje natural, llamadas preguntas de competencia, que determinan el alcance de la ontología. Estas preguntas y sus respuestas son utilizadas para extraer los principales conceptos y sus propiedades, y relaciones y axiomas formales de la ontología. El conocimiento se expresa formalmente en lógica de primer orden. Esta es una metodología muy formal que toma ventaja de la robustez de la lógica clásica [Grüninger M., Fox MS 1995]. Los procesos de esta metodología se muestran en la siguiente figura.

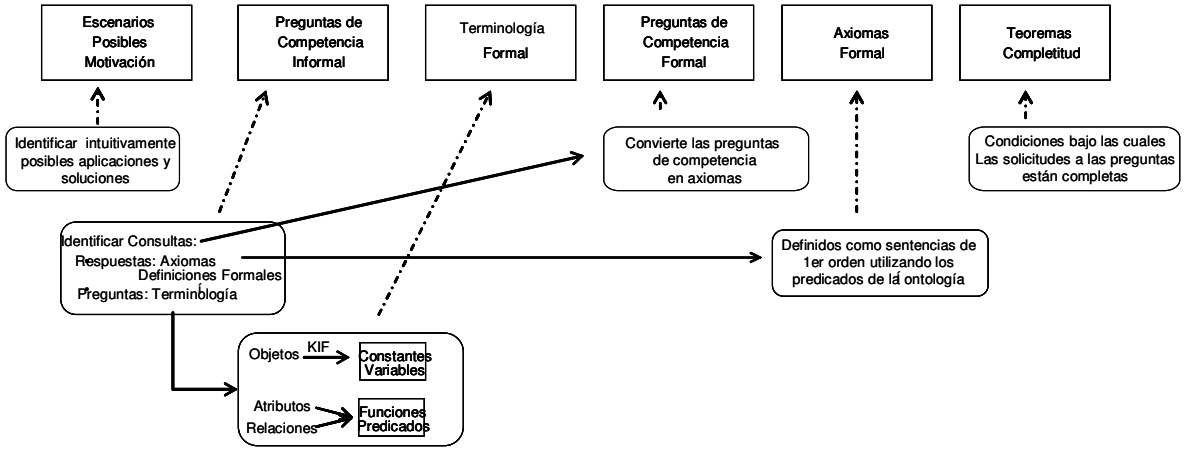

Figura 2.6 Metodología de Grüninger y Fox

Identificación de escenarios posibles. El desarrollo de la ontología está motivado por los escenarios relacionados a las aplicaciones que usaran las ontologías. Tales escenarios describen un conjunto de requerimientos ontológicos que la ontología debe satisfacer después de ser formalmente implementada.

Elaboración de Preguntas de Competencia Informales. A partir del conjunto de escenarios surgen un conjunto de preguntas de competencia. Las preguntas de

competencia son aquellas escritas en lenguaje natural a ser respondidas por la ontología una vez que la ontología esté expresada en lenguaje formal. Las preguntas de competencia juegan el rol de un tipo de especificaciones de requerimientos contra los que la ontología será evaluada. Cada pregunta de competencia es utilizada como base para obtener afirmaciones, restricciones, datos necesarios de entrada, etc.

Especificar la terminología usando lógica de primer orden. A partir de las preguntas de competencia informales los ontólogos extraen la ontología que está formalmente representada por medio de conceptos, atributos y relaciones, en lenguaje de lógica de primer orden. De las respuestas en lenguaje natural a las preguntas de competencia, los ontólogos extraen el conocimiento a ser incluido en la definición formal de conceptos, relaciones y axiomas formales.

Especificar los axiomas usando lógica de primer orden. Esta metodología propone usar axiomas para especificar la definición de términos en la ontología; los axiomas se definen como sentencias de lógica de predicado de primer orden.

2.5.2.3 Methontology

Esta metodología comprende la identificación de los procesos de desarrollo de la ontología basado en ciclos de vida que van produciendo prototipos evolutivos, además de las técnicas para llevar a cabo las actividades de administración, desarrollo y soporte a las ontologías [Fernandez et al. 1997].

Para cada prototipo, METHONTOLOGY propone comenzar con la actividad de identificar las tareas a ser realizadas, y el tiempo y los recursos necesarios para llevarlas acabo (programación). Después, comienza la primera actividad de desarrollo (especificación de la ontología), y al mismo tiempo comienzan las actividades de administración (aseguramiento de control y calidad) y de soporte (adquisición de conocimiento, integración, evaluación, documentación y administración de configuración). Todas las actividades de administración y soporte se realizan en paralelo con cada una de las actividades de desarrollo (especificación, conceptualización, formalización, implementación y mantenimiento), durante todo el ciclo de vida de la ontología [Fernandez et al., 1997]. Si se detecta alguna carencia durante cualquiera de las actividades de desarrollo, se puede regresar a cualquiera de las actividades anteriores para hacer modificaciones o refinamientos. En la siguiente figura se muestra el ciclo de vida de Methontology.

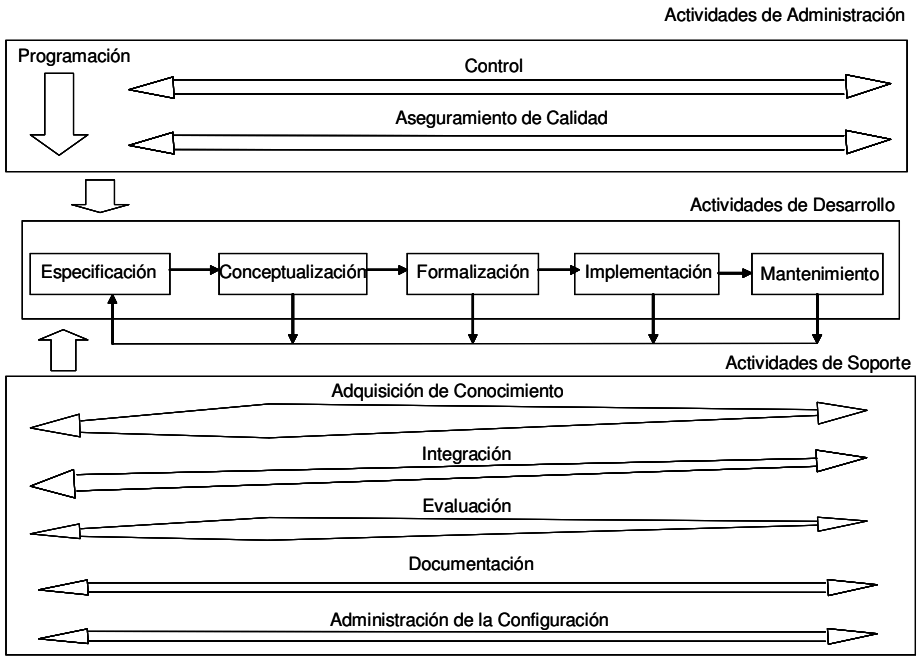

Figura 2.7 Metodología Procesos de desarrollo y ciclo de vida de Methontology [Fernandez et al. 1997]

Methontology incluye en la actividad de conceptualización un conjunto de tareas para estructurar el conocimiento, como se muestra en la siguiente figura.

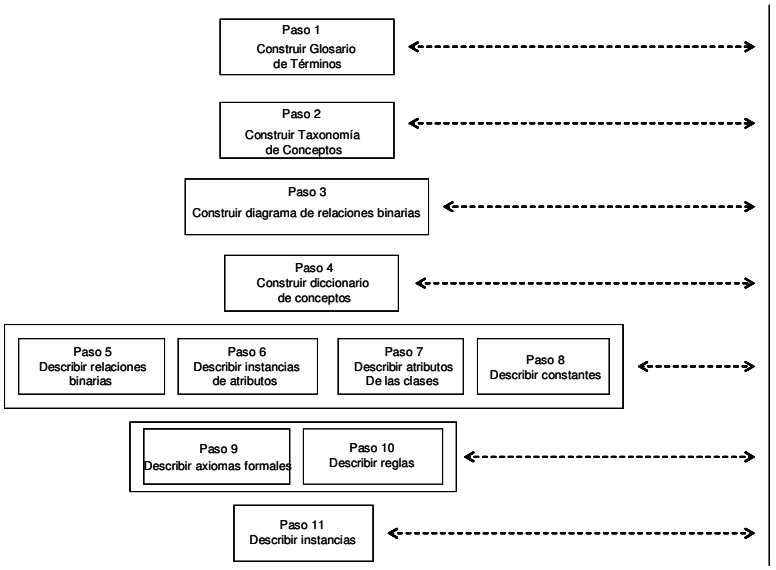

Figura 2.8 Tareas de la actividad de conceptualización de acuerdo a METHONTOLOGY [Fernandez et al. 1997]

Tarea 1: Construir el glosario de términos que identifica el conjunto de términos a incluir en la ontología, su definición en lenguaje natural, y sus sinónimos y acrónimos.

Tarea 2: Construir taxonomía de conceptos para clasificar conceptos. Los resultados de esta tarea son una o más taxonomías donde se clasifican los conceptos.
Tarea 3: Construcción del diagrama de relaciones binarias, para identificar las relaciones entre los conceptos de la ontología con los conceptos de otras ontologías, si existen.
Tarea 4: Construir diccionario de conceptos, que contiene todos los conceptos del dominio, sus relaciones, sus instancias, atributos y clases (sub-conceptos), y sus relaciones binarias.
Tarea 5: Describir detalladamente las relaciones binarias que aparecen en el diagrama de relaciones binarias y en el diccionario de conceptos. El resultado de esta tarea es una tabla de relaciones binarias donde se especifican nombre de los conceptos, fuentes y destino, su cardinalidad, relaciones inversas y propiedades matemáticas.
Tarea 6: Describir en detalle cada atributo que aparezca en el diccionario de conceptos. El resultado de esta tarea es una tabla donde se describen los atributos. Para cada atributo se pueden especificar los siguientes campos: nombre, concepto al que pertenece (los atributos pertenecen a los conceptos locales), tipo de valor, unidad de medida, precisión y rango de valores, valores por defecto si existen, cardinalidad máxima y mínima, atributos instancia, atributos clase, y constantes utilizadas para inferir valores de atributos, atributos que puedan ser inferidos usando valores de sus atributos, formulas y reglas que permitan inferir valores de los atributos y referencias usadas para definir el atributo.
Tarea 7: Describir en detalle cada clase que aparece en el diccionario de conceptos. El resultado es una tabla donde se describen las clases (objeto). La tabla contiene los siguientes campos: nombre, nombre del concepto donde la clase está definida, instancias de atributos cuyos valores puedan ser inferidos, etc.
Tarea 8: Describir en detalle cada constante y producir una tabla de constantes. Las constantes especifican información relacionada con el dominio de conocimiento; ellas siempre toman el mismo valor y normalmente son utilizadas en fórmulas. Cada fila de la tabla de constantes contiene una descripción detallada de la constante, donde se debe especificar: nombre, tipo de valor (número, masa, etc.), valor, unidad de medida para constantes numéricas, y atributos a ser inferidos usando las constantes.

Una vez definidos los conceptos, taxonomías, atributos y relaciones, los ontólogos pueden describir los axiomas formales (tarea 9) y reglas (tarea 10) que son utilizadas para el chequeo de restricciones y para inferir valores de atributos. Opcionalmente se puede introducir información acerca de las instancias (tarea 11).
Tarea 9: Describir Axiomas Formales. Se identifican los axiomas formales en la ontología y se describen de manera precisa. Para cada definición de los axiomas formales METHONTOLOGY propone especificar la siguiente información: nombre, descripción en lenguaje natural, la expresión lógica que formalmente describe el axioma utilizando lógica de primer orden, los conceptos, atributos y relaciones a los que se refiere el axioma y las variables usadas.

Tarea 10: Describir Reglas. Describir las reglas en la tabla de reglas. Para cada definición de regla METHONTOLOGY propone incluir la siguiente información: nombre, descripción en lenguaje natural, expresión que formalmente describe la regla, conceptos, atributos y relaciones a los que se refiere la regla y las variables usadas en la expresión. METHONTOLOGY propone especificar la expresión de la regla usando la plantilla *if <condición> then <consecuencia>*.

Tarea 11: Describir Instancias. Después de definido el modelo conceptual de la ontología se deben describir instancias relevantes. Para cada instancia se debe definir: nombre, nombre del concepto al que pertenece, y los valores de sus atributos si se conocen.

METHONTOLOGY sugiere reducir o extender el conjunto de representaciones intermedias de acuerdo a las necesidades de representación de conocimiento del dominio, así como también modificar los campos de representaciones intermedias agregando, eliminando o modificando los campos ya presentados. Por ejemplo se puede construir una ontología ligera con solo conceptos, atributos y relaciones entre conceptos, no se necesita usar representaciones del modelo como reglas y axiomas formales.

2.5.3 Lenguajes para construir Ontologías

Al comienzo de los 90 se crearon un conjunto de lenguajes ontológicos basados en Inteligencia Artificial. Básicamente, el paradigma de representación de conocimiento subyacente de estos lenguajes ontológicos se basaba en lógica de primer orden (por ejemplo KIF), en Frames combinados con lógica de primer orden (ej. CycL, Ontolingua, OCML, y FLogic), y en descripción lógica[21] (ej. LOOM). OKBC (Open Knowledge Base Connectivity) [Chaudri et al., 1998] es un protocolo que permite accesar bases de conocimiento basadas en diferentes paradigmas de representación de conocimiento. En la siguiente figura se muestra estos lenguajes tradicionales [Gómez-Pérez et al., 2004].

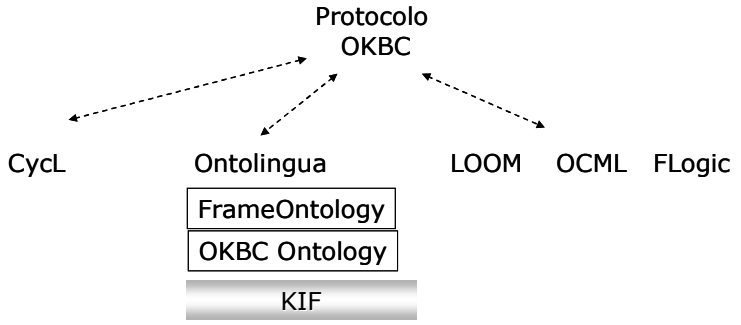

Figura 2.9. Lenguajes ontológicos tradicionales

[21] familia de lenguajes de representación del conocimiento que pueden ser usados para representar conocimiento terminológico de un dominio de aplicación de una forma estructurada y formalmente bien comprendida

De este conjunto de lenguajes, **CycL** [Lenat and Guha, 1990] fue el primero. CycL está basado en frames y lógica de predicado de primer orden, y se utilizó para construir la ontología Cyc.

KIF [Genesereth and Fikes, 1992; NCITS, 1998] fue creado después, en 1992, y se diseñó como formato de intercambio de conocimiento; KIF está basado en lógica de primer orden. A pesar de que era difícil crear ontologías directamente en KIF, **Ontolingua** [Farquhar et al., 1997] fue creada encima de él. Ontolingua es un lenguaje soportado por el servidor Ontolingua, que tiene una sintaxis similar a Lisp. Su paradigma de representación de conocimiento subyacente son frames (marcos) y lógica de primer orden. Ontolingua fue considerado un estándar de facto por la comunidad ontológica en los 90. Al mismo tiempo se construyó **LOOM** [MacGregor, 1991], aunque no fue construido para implementar ontologías, sino para bases de conocimiento. LOOM está basado en descripción lógica (DL, por sus siglas en inglés Description Logic) y reglas de producción, y proporciona característica como clasificación automática de conceptos. **OCML** [Motta, 1999] fue desarrollado después como una clase de "Ontolingua operacional". Finalmente, FLogic [Kifer et al., 1995] fue desarrollado como un lenguaje que combina Frames y lógica de primer orden aunque no posee sintaxis como Lisp.

Internet condujo a la creación de lenguajes ontológicos para explotar las caracteríticas de la Web. Tales lenguajes son llamados *lenguajes ontológicos basados en web, o lenguajes de marca ontológicos*. Su sintaxis está basada en lenguajes de marca existentes como HTML, y XML, cuyo propósito es la presentación e intercambio de datos. La relación entre estos lenguajes se muestra en la siguiente figura.

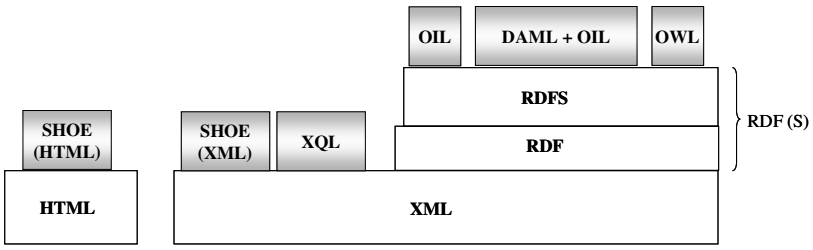

Figura 2.10. Lenguajes Ontológicos de Marca

El primer lenguaje ontológico de marca fue **SHOE** [Luke and Helfin, 2000]. SHOE es un lenguaje que combina frames y reglas. Fue construido como una extensión de HTML, usa diferentes etiquetas a las del HTML, que permiten la inserción de ontologías en documentos HTML. Su sintaxis fue adaptada a XML después.

El resto de los lenguajes ontológicos de marca están basados en XML. **XOL** [Karp et al., 1999] fue desarrollado como una conversión al XML de un subconjunto de primitivas del protocolo OKBC, llamado OKBC-Lite. **RDF** [Lassila and Swick, 1999] fue desarrollado

por el W3C (World Wide Consortium), es un lenguaje basado en redes semánticas[22] para describir recursos Web. El **RDF Schema** [Brickley and Guha, 2003] es una extensión de RDF. La combinación de RDF y RDF Schema es conocido como RDF(S). Estos lenguajes establecieron las bases de la Web Semántica. En este contexto se han desarrollado tres lenguajes más: OIL, DAM+OIL, OWL y SWOOP. **OIL** [Horrocks et al., 2000] fue desarrollado en el marco del proyecto europeo OntoKnowledge[23], agrega primitivas de representación de conocimiento a RDF(S), y su semántica formal está basada en descripción lógica. **DAML+OIL** [Horrocks and Van Harmelen, 2001] está basado en especificaciones de una versión anterior de DAML-ONT y OIL, agregándole primitivas de representación de conocimiento basadas en descripción lógica a RDF(S). **OWL** [Dean and Schreiber, 2003] se deriva de DAM+OIL y lo reemplaza. Cubre la mayoría de las características de DAM+OIL, y renombra la mayoría de sus primitivas. OWL se usa para publicar y compartir ontologías en la Web.

2.5.4 Herramientas Ontológicas

Gómez-Pérez [Gómez-Pérez et al. 2002], distinguen los siguientes grupos de herramientas ontológicas:

Herramientas para el desarrollo de Ontologías: Este grupo incluye herramientas y suites integradas que pueden utilizarse para construir nuevas ontologías. Además de las funciones de búsqueda y edición, estas herramientas usualmente dan soporte para documentar las ontología, exportar e importar a y desde diferentes formatos y lenguajes ontológicos, edición gráfica de ontologías, administración de librerías ontológicas, etc. Algunas de ellas son: KAON [Maedche et al., 2003], OilEd [Bechofer et al., 2001], Ontolingua Server [Farqhuar et al., 1997], OntoSaurus [Swartout et al., 1997], Protégé-OWL [Holger Knublauch et al., 2004], SWOOP [Aditya Kalyanpur et al., 2005], WebODE [Arpirez et al., 2003] y WebOnto [Domingue, 1998].

Herramientas de Evaluación de Ontologías: Son utilizadas para evaluar el contenido de las ontologías y sus tecnologías relacionadas. La evaluación del contenido de las ontologías reduce problemas cuando se necesita integrar y usar ontologías y tecnologías basadas en ontologías en diferentes sistemas de información.

Herramientas de alineación y unión de ontologías: Estas herramientas son usadas para resolver problemas de unión y alineación[24] de diferentes ontologías en un mismo dominio. Ejemplo de ellas es el plug-in PROMPT integrado a Protégé.

Herramientas de anotación basadas en ontologías: Con estas herramientas los usuarios pueden guardar como anotaciones clasificadas, objetos o propiedades correspondientes a un concepto de una ontología. La mayoría de estas herramientas han aparecido recientemente en el contexto de la Web Semántica. Algunas de ellas son: AeroDAML (Reeve L. and Hyoil Hann, 2005), COHSE (Bechhofer et al., 2002),

[22] Grafos orientados que proporcionan una representación declarativa de objetos, propiedades y relaciones
[23] http://www.ontoknowledge.org/
[24] Alineamientos de ontologías: consiste en encontrar correspondencias o mapeos existentes entre entidades pertenecientes a ontologías diferentes, pero que están relacionadas semánticamente

OntoAnnotate (Staab et al., 2001) y el SHOE Knowledge Annotator (Heflin and Hendler, 2001).

Herramientas de Aprendizaje Ontológico: pueden generar ontologías (semi) automáticamente desde textos en lenguaje natural, así como también desde fuentes semi-estructuradas y bases de datos, por medio de técnicas de máquinas de aprendizaje y lenguaje natural.

Capítulo 3

3. MODELO ONTOLOGICO PARA BASES DE DATOS FEDERADAS

3.1 Introducción

En este capítulo se muestra el diseño de un "Modelo de Integración para Bases de Datos Federadas". Particularmente, se propone diseñar el modelo común para el proceso de integración a través del manejo ontológico de la información. Para esto, se diseñan ontologías que permitirán describir los diferentes tipos de bases de datos como un conjunto de términos representacionales de sus diferentes componentes. En estas ontologías, las definiciones asocian conceptos, relaciones, funciones, entre otras cosas, de entidades en el universo del discurso de las bases de datos, para describir sus elementos, sus operaciones y sus restricciones.

La razón de usar ontologías es que ellas definen conceptos y relaciones dentro de un marco taxonómico, cuya conceptualización está representada de una manera formal, legible y reutilizable. Además, una ontología representa un entendimiento común y compartido de un dominio, que puede comunicarse entre sistemas heterogéneos [Weigand, H**.** 1997].

En trabajos anteriores [Muñoz A, Aguilar J., Martínez R, 2006, Muñoz A, Aguilar J., Martínez R. 2005, Muñoz A., Aguilar J. 2005 (a)] se han propuesto modelos de referencia y una arquitectura para la integración de bases de datos federadas en donde se plantea la necesidad de definir un "Modelo Común para Sistemas de Bases de Datos Federadas". En este capítulo se describen los modelos ontológicos que componen el proceso de integración de bases datos federados, y los modelos ontológicos para las bases de datos componentes de una federación. Para modelar las ontologías se utilizó como metodología Methontology, que identifica conceptos, relaciones, atributos de los conceptos, axiomas e instancias.

3.2 Ontología de Integración de Bases de Datos

El diseño del modelo común para la Integración de Base de Datos se basa en Ontologías que caracterizan los tipos de base de datos a integrar a través de sus conceptos, operaciones y restricciones. Así, el modelo común es el resultado de la unión de la ontología que describe el proceso de integración y de las ontologías de cada una de las bases de datos componentes de una federación. En la siguiente figura se muestra el modelo de integración propuesto basado en ontologías (modelada a través de una Base de

Conocimientos), el cual además posee mecanismos de razonamiento y de aprendizaje para llevar a cabo el proceso de integración [Muñoz A., Aguilar J. 2005, Muñoz A, Aguilar J., Martínez R, 2006].

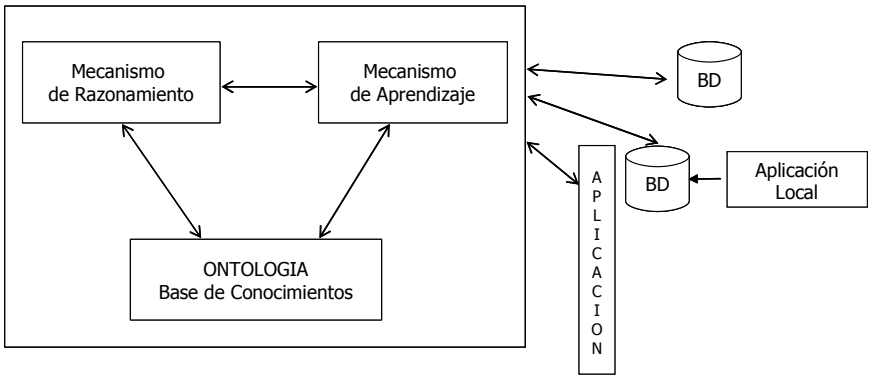

Figura 3.1. Modelo Inteligente de Integración para Bases de Datos Federadas

A través de la ontología se define el modelo de integración que contiene el conocimiento necesario para realizar el proceso de integración de bases de datos federadas. El proceso de integración de bases de datos que componen la federación propuesto se realiza utilizando un modelo común para el acceso a través de la federación, y el modelo propio de cada base de datos componente de la federación para el acceso local. Este modelo común es definido como una ontología que contiene los conceptos, operaciones y restricciones necesarios para poder realizar la integración inteligente de las bases de datos componentes. El mecanismo de razonamiento (inductivo, deductivo o abductivo) realiza procesos de inferencia usando las reglas y los hechos de la base de conocimientos, que permiten resolver el problema de integración. El mecanismo de aprendizaje contiene los algoritmos y técnicas que permiten extraer patrones de la base de conocimientos, o generar nuevas reglas de comportamiento. La base de conocimientos contiene las reglas y hechos que describen el dominio de las bases de datos federadas. A continuación se describe la ontología del proceso de integración.

3.2.1 Conceptos para Bases de Datos Federadas

Las Bases de Datos Federadas están conformadas por bases de datos componentes y tienen operaciones y restricciones de integración. Las Bases de Datos Componentes son aquellas bases de datos que van a conformar una federación. En este estudio, estas bases de datos componentes pueden ser: Bases de Datos Relacionales, Bases de Datos Orientadas a Objeto, Bases de Datos Multimedia, Bases de Datos Inteligentes y también otra Base de Datos Federada. Cada una de estas bases de datos componentes posee sus conceptos, operaciones y restricciones. A continuación se describe el modelo ontológico de las bases de datos federadas. En la figura 3.2 se puede ver la taxonomía de los conceptos de una base de datos federada. Los conceptos son cada uno de los nodos y las relaciones son las etiquetas en las flechas. Por otro lado, las funciones y axiomas se representan a

través de sentencias en lógica de predicado de primer orden, las cuales se muestran en tablas.

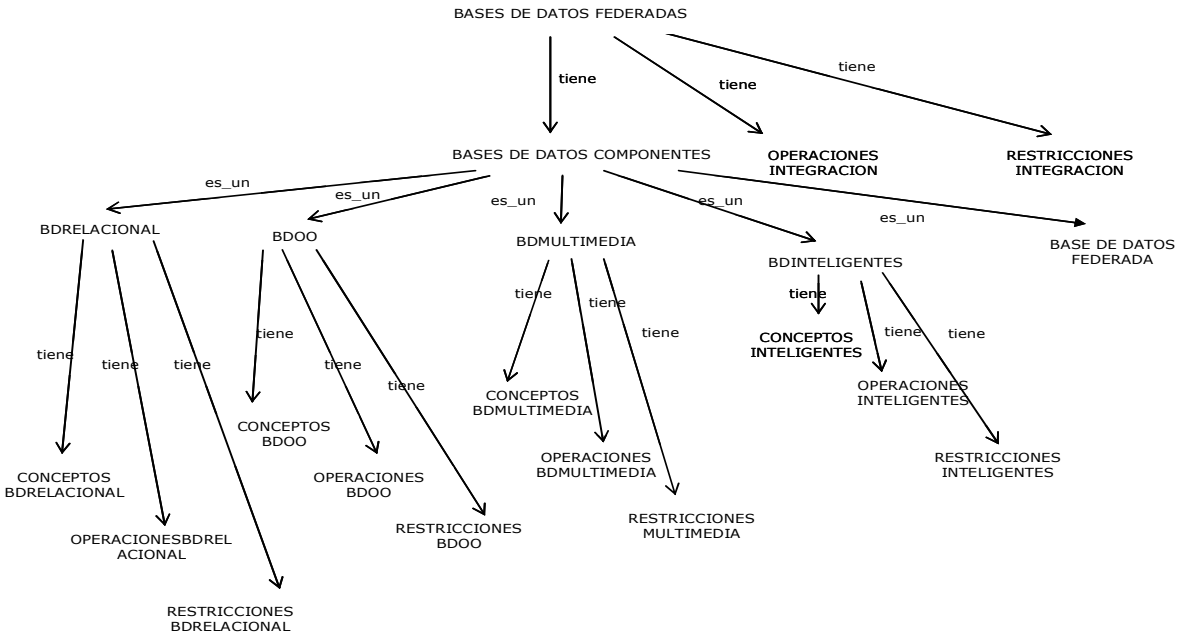

Figura 3.2. Taxonomía de los Conceptos para Bases de Datos Federadas

Cada uno de los conceptos para las bases de datos federadas tiene atributos y relaciones que los describen, estos son:

Conceptos y sus atributos:

BD_FEDERADA (NombreBDF, ID_BDF, BD_Componente, Fecha_Inicio_Federación, Fecha_Fin_Federacion)

BD_COMPONENTE (NombreBDC, Tipo_SABDC, Tipo_SOP, Lenguaje_Consulta)

BD_RELACIONAL (Nombre_BDR, ID_BDR, Esquemas_R)

BD_OO (Nombre_BDOO, ID_BDOO, Esquemas_OO)

BD_MM (Nombre_BDMM, ID_BDMM, Esquemas_MM)

BD_INTELIGENTE (Nombre_BDI, ID_BDI, Base_H, Base_R)

Relaciones: tiene, es_un

Donde:

BDF: Base de Datos Federada

BDC: Base de Datos Componente

SABDC: Sistema Administrador de Base de Datos Componente

SOP: Sistema Operativo

BDR: Base de Datos Relacional

BDOO: Base de Datos Orientada a Objeto

BDMM: Base de Datos Multimedia

BDI: Base de Datos Inteligente

Los modelos ontológicos para las Bases de Datos Inteligentes, Bases de Datos Multimedia y Bases de Datos Orientadas a Objeto fueron desarrollados en [Muñoz A, Aguilar J. 2007, Muñoz A, Aguilar J. Julio 2007].

A continuación, en la tabla 1, se describe el comportamiento de la taxonomía de conceptos para bases de datos federadas a través de axiomas. Cada axioma definido incluye: su descripción en lenguaje natural y la expresión lógica que describe formalmente el axioma.

Tabla 1. Axiomas que describen en Lógica de Predicado de Primer Orden (LPO) los conceptos de bases de datos federadas.

Sentencia	LPO
Una base de datos federada tiene base de datos componentes y tiene operaciones de integración y restricciones de integración	∀ x BDFederada(x) => tiene (x,BDComponente) Λ tiene (x,OperacionIntegracion) Λ tiene (x,RestriccionesIntegracion)
Las bases de datos componentes pueden ser bases de datos relacionales, bases de datos orientadas a objeto, bases de datos multimedia, bases de datos inteligentes o bases de datos federadas	∀ x BDComponente(x) => es_un(x,BDRelacional) V es_un (x,BDOO) V es_un (x, BDMultimedia) V es_un (x, BDInteligente) V es_un (x, BDFederada)
Las BDRelacionales tienen Conceptos, Operaciones, Restricciones	∀ x BDRelacional(x) => tiene(x,ConceptosR) Λ tiene(x, OperacionesR) Λ tiene(x, RestriccionesR)
Las BDOO tienen Conceptos, Operaciones y Restricciones	∀ x BDOO(x) => tiene(x,ConceptosOO) Λ tiene(x, OperacionesOO) Λ tiene(x, RestriccionesOO)
Las BDMultimedia tienen Conceptos, Operaciones y Restricciones	∀ x BDMultimedia(x) => tiene(x,ConceptosMM) Λ tiene(x, OperacionesMM) Λ tiene(x, RestriccionesMM)
Las BDInteligentes tienen Conceptos, Operaciones y Restricciones	∀ x BDInteligentes(x) => Tiene(x,ConceptosInt) Λ Tiene(x, OperacionesInt) Λ Tiene(x, RestriccionesInt)

3.2.2 Operaciones de Integración de las Bases de Datos Federadas

Se utilizaran las operaciones de integración propuestas en [Batini, C.; Lenzerini, M.; Navathe, S. 1986], como base para esta metodología de integración, las cuales se realizan en fases. A continuación se describen las características de estas fases.

- *Preintegración*. En esta fase se deciden las bases de datos a integrar, el orden de integración (puede ser binario cuando se integran dos esquemas a la vez, y n-arias cuando se integran n esquemas a la vez), y las partes de las bases de datos a integrar. También se establecen las políticas de integración en cuanto a las restricciones de acceso (datos a compartir en la federación) y prioridad en el acceso a las Bases de Datos Componentes (orden de acceso de acuerdo a las reglas de negocio). Además, aquí se establece (negocia) si se requiere formar una nueva federación o incorporar una base de datos componente a un sistema de base de datos federados existente.
- *Comparación de los esquemas*. Las bases de datos se comparan y analizan para determinar la correspondencia entre conceptos y detectar los posibles conflictos. En este caso, una vez que se detectan los conflictos se incorporan a la base de

conocimientos de conflictos para manejar sus soluciones a través de un sistema de reglas. Para la comparación de esquemas, Rahm et al [Rahm et al. 2001] definen el operador Match, el cual es una función que toma dos esquemas E1 y E2 como entrada, y regresa un valor indicando el grado de semejanza. Las técnicas de comparación se clasifican a nivel de Elemento y a nivel de Estructura. Las técnicas a nivel de elementos calculan el grado de semejanza entre cada elemento que compone cada uno de los esquemas. Las técnicas a nivel de estructura calculan el grado de semejanza de la estructura de los esquemas [Rahm et al. 2001]. A nivel de Elemento y a nivel de Estructura las técnicas también se pueden clasificar en sintácticas, externas (solo para el caso de esquema) y semánticas. Las técnicas sintácticas interpretan la entrada en función de una estructura siguiendo un algoritmo establecido. Las técnicas externas explotan recursos auxiliares (externos) de un dominio y un conocimiento común de manera de interpretar la entrada. Las técnicas semánticas usan una semántica formal para interpretar la entrada y justificar su resultado. En la siguiente figura se presenta una tipología de las técnicas de comparación.

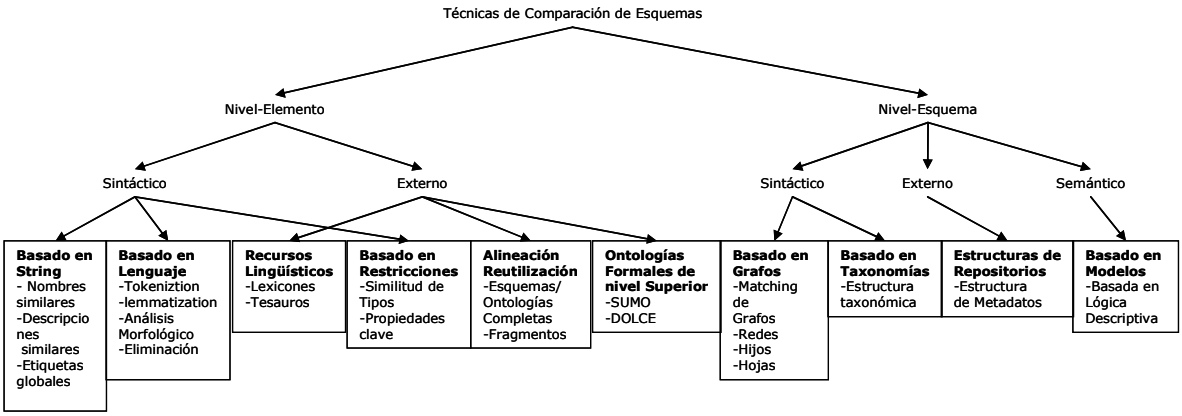

Figura 3.3. Técnicas de comparación de esquemas

- *Unión y Reestructuración*. Una vez detectados los conflictos, se resuelven de manera tal que la unión de los diferentes esquemas de las bases de datos componentes sea posible. La meta de esta actividad es generar esquemas compatibles para su integración. Tiene operaciones como: renombrado, transformación de conceptos, eliminación de relaciones redundantes, generalización, etc.

En la figura 3.4 se muestra la taxonomía que describe las operaciones de integración de bases de datos federadas.

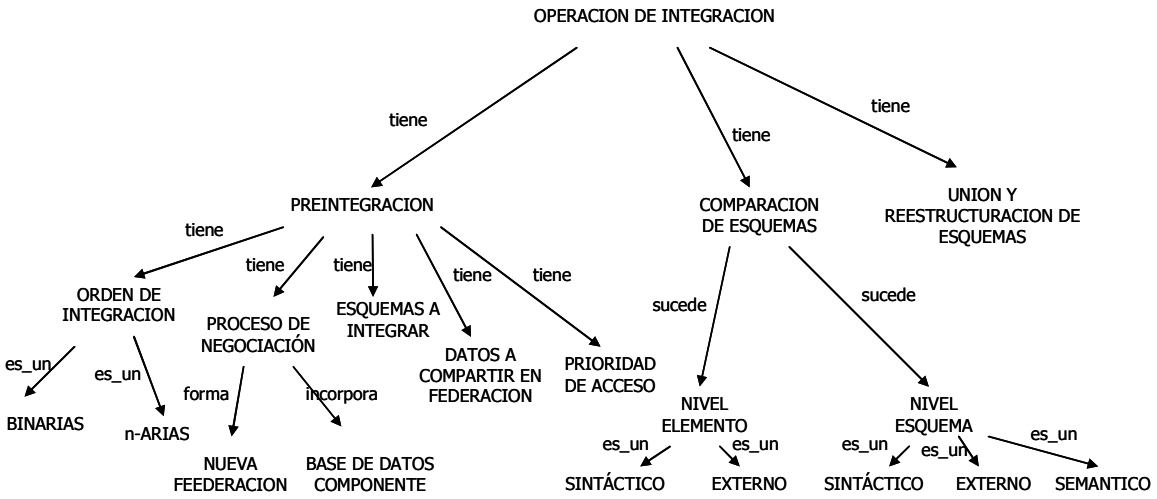

Figura 3.4. Taxonomía de Operaciones de Integración para Bases de Datos Federadas

Los axiomas para las operaciones de integración de las Bases de Datos Federadas se muestran en la tabla 2.

Tabla 2. Axiomas que describen en LPO las operaciones de integración para las bases de datos federadas.

Tabla 2. Axiomas para las operaciones de integración.

Sentencia	LPO
La operación de integración tiene la fase de preintegración, de comparación de esquemas, de unión y reestructuración de esquemas	∀ x OperaciónIntegración(x) => tiene (x,Preintegración) ∧ tiene(x,ComparacióndeEsquemas) ∧ tiene (x,Union_ReestructuraciónEsquemas) ∧ tiene (x, UnionyReestructuraciondeEsquemas)
La Preintegración tiene orden de integración, proceso de negociación y tiene datos a compartir en la federación y prioridad de acceso	∀ Preintegración(x) =>tiene(x, OrdenIntegracion) ∧ tiene(x,ProcesoNegociación) ∧ tiene(x,EsquemasaIntegrar) ∧ tiene(x, DatosaCompartirenFederacion) ∧ tiene(x, PrioridaddeAcceso)
El orden de integración de la Base de Datos puede ser binario o n-ario	∀ x OrdenIntegracion(x) => es_un(x,IntegracionBinario) V es_un(x,IntegracionN-ario)
Un orden de integración binario es aquel integra dos esquemas a la vez	∀ x IntegracionBinaria(x) => Integra(x,dos_Esquemas)
El orden de integración n-ario es el que integra n esquemas a la vez	∀ x IntegracionNaria(x) => Integra(x, n_Esquemas)
En el proceso de negociación se forma una nueva federación o se incorpora una base de datos componente	∀ x ProcesoNegociación(x) => Forma(x,NuevaBDFederada) V Incorpora(x,BDComponente)
La prioridad de acceso establece el orden de acceso a las bases de datos componentes	∀ x PrioridaddeAcceso(x) => Establece(x,OrdendeAcceso)
La comparación de esquemas sucede a nivel de elemento o a nivel de esquemas	∀ x,y ComparacionEsquemas(x,y) => Sucede ((x,y),NivelElemento) V Sucede((x,y),NivelEsquema)
Las técnicas de comparación de esquemas a nivel de elemento pueden ser sintácticas o externas	∀ x NiveldeElemento(x) => es_un (x,Sintacticas) V es_un (x, Externas)

Las técnicas de comparación de esquemas a nivel de esquema pueden ser sintácticas, externas o semánticas	∀ x NiveldeEsquema (x) => es_un (x,Sintácticas) V es_un (x, Externas) V es_un (x, Semánticas)

3.2.3 Restricciones de Integración en Bases de Datos Federadas

En la integración de las bases de datos pueden presentarse los siguientes tipos de conflictos [Kim y Seo, 1991]:

***Conflictos de Esquemas*.**

- Conflictos en Tablas.
    - Conflictos en el Nombrado de tablas: pueden surgir cuando existen nombres diferentes para tablas equivalentes o nombres iguales para tablas diferentes.
- Conflictos en las diferencias que surgen con respecto a la selección de claves primarias y ajenas.
- Conflictos de Atributos.
    - Conflictos en nombrado de Atributos, pueden surgir cuando existen nombres diferentes para atributos equivalentes o nombres iguales para atributos diferentes.
    - Conflictos en Valores por Defecto, es derivado del hecho de que los valores deducidos por defecto para un mismo atributo son diferentes.
    - Conflictos por Restricciones de Asignación de Valores a los Atributos, pueden ser conflictos en los tipos de datos y conflictos en las restricciones de dominio.
    - Conflictos por la Cardinalidad y Grado de Atomicidad, el grado de detalle de cada atributo puede ser distinto en cada aplicación, por ejemplo atributos que se omiten en una tabla.
- Conflictos en la Representación de la Información, ocurre cuando un mismo concepto representa tanto a un atributo como a una entidad.

Conflictos de Datos.

- Conflictos entre los valores, cuando instancias equivalentes no tienen los mismos valores debido a que los datos obtenidos son incorrectos o son obsoletos.
- Diferencias en la representación, vienen dadas por situaciones de contexto y cultura organizacional. Estas diferencias vienen dadas por: notaciones diferentes (cuando existen diversas maneras para nombrar un dato, por ejemplo para nombrar una ciudad: Caracas o CCS), unidades distintas (por Ej., el sistema de medición inglés y el internacional), y diferencias en la representación (cuando un dato se puede representar de diferentes maneras, por ejemplo calificaciones numéricas y letras)

***Conflictos en Reglas*:**

- Disparo Simultáneo de Reglas, cuando un evento o consulta tiene diferentes reglas asociadas y el sistema permite que solo una regla se active.

- Contradicción entre reglas, que es cuando un evento o consulta dispara dos reglas, donde la acción de una es la negación de la acción generada por la otra regla. En este caso, se puede resolver al inhibir la activación de una de ellas.

En la figura 3.5 se muestra el esquema ontológico que describe los conflictos anteriormente nombrados.

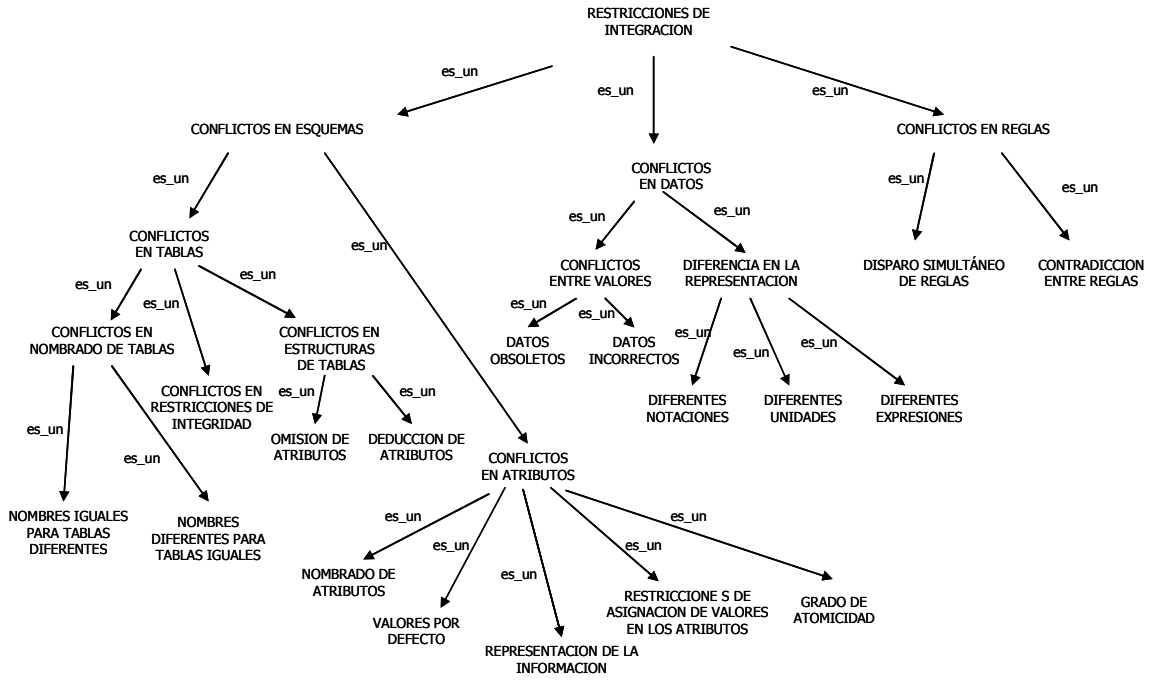

Figura 3.5. Esquema Ontológico de las Restricciones de Integración para Bases de Datos Federadas

Los axiomas de las restricciones de integración para las Bases de Datos Federadas se muestran en la tabla 3.

Tabla 3. Axiomas de las restricciones de integración.

Sentencia	LPO
Las restricciones de integración pueden ser conflictos en esquemas, conflictos en datos o conflictos en reglas	∀ x RestriccionIntegracion (x) => es_un(x,ConflictoEsquema) V es_un (x, ConflictoenDatos) V es_un(x,ConflictoenReglas)
Los conflictos en esquema pueden ser conflictos en tablas o conflictos en atributos	∀ x ConflictodeEsquema(x) => es_un(x,ConflictoTabla) V es_un(x,ConflictoAtributo)
Los conflictos en tablas pueden ser en nombrado de tablas, en estructura de tabla	∀ x ConflictoTabla(x) => es_un (x,ConflictoNombreTabla) V (x,ConflictoRestriccionesIntegridad) V es_un (x, ConflictoEstructuraTabla)
El conflicto en nombrado de tablas surge cuando existen nombres diferentes para tablas iguales o nombres iguales para tablas diferentes	∀ x,y ConflictoNombreTabla(x,y) => es_un((x,y),DiferentesNombre_TablasIguales) V es_unl((x,y),IgualNombre_TablasDiferentes)
El conflicto en estructura de tabla ocurre cuando hay atributos que se omiten o cuando hay atributos que se deducen	∀ x ConflictoEstructuraTabla (x) => es_un(x,AtributosTOmitidos) V es_un(x,AtributosTDeducidos)
Los conflictos en atributos pueden ser conflictos en nombre de atributo o conflictos	∀ x ConflictoAtributo(x,y) => es_un(x,ConflictoNombreAtributo) V

en valores por defecto o conflictos de restricciones de asignación de valores de los atributos o conflictos de grado de atomicidad o conflictos en la representación de la información	es_un(x,ConflictoValorporDefecto) V es_un (x,ConflictoRestricciondeAsignacióndeValores) V es_un (x,ConflictoGradodeAtomicidad) V es_un (x,ConflictoRepresentacióndeInformacion)
Los conflictos en nombrado de Atributos surgen cuando existen nombres diferentes para atributos equivalentes o nombres iguales para atributos diferentes	∀ x,y ConflictoNombreAtributo(x,y) => es_un ((x,y), NombreDiferentes_AtributosEquivalentes) V es_un ((x,y), NombresIguales_AtributosDiferentes)
Los conflictos en valores por defecto es la definición de los valores deducidos por defecto	∀ x ConflictoValorporDefecto(x) => es_un(x,DefinicióndeValoresDeducidospordefecto)
Los conflictos por Restricciones de Asignación de Valores a los Atributos, pueden ser conflictos en los tipos de datos y conflictos en las restricciones de dominio.	∀ x ConflictoRestricciondeAsignacióndeValores(x) => es_un (x,ConflictosenTipodeDatos) V es_un(x,ConflictosenRestriccionesdeDominio)
Un conflicto de cardinalidad es la diferencia de detalles de los atributos	∀ x ConflictoCardinalidad(x) => Tiene(x,DiferenteNiveldeRepresentaciondeAtributos)
Los Conflicto en la representación de información son las diferentes dominios que representa un atributo	∀ x ConflictoRepresentacióndeInformacion(x) => Tiene(x,DominiosDiferentes)
Los conflictos en datos pueden ser conflictos entre valores o un conflicto de diferencias en la representación de datos	∀ x ConflictoDatos(x) => es_un (x,ConflictoentreValores) V es_un (x,ConflictodeDiferenciadeRepresentacióndeDatos)
Un conflicto entre valores ocurre por datos obsoletos o datos de entrada incorrectos	∀ x ConflictoentreValores(x) => tiene (x,DatosObsoletos) Λ tiene(x,ValoresIncorrectos)
Las diferencias de representación ocurre cuando tienen diferentes notaciones o diferentes unidades o diferentes expresiones	∀ x ConflictodeDiferenciadeRepresentación(x) => tiene(x, DiferenteNotacion) V tiene (x, DiferentesUnidades) V tienen (x, DiferentesExpresiones)
Un conflicto en regla puede ser un disparo simultaneo de reglas, o puede ser una contradicción entre reglas	∀ x ConflictoenReglas(x) => es_un (x,DisparoSimultaneoReglas) V es_un (x,ContradiccionEntreReglas)
Un disparo simultaneo de reglas es cuando se dispara mas de una regla y solo se activa una	∀ x,y DisparoSimultaneoReglas(x,y) => es_un((x,y),DisparodemasdeunaRegla) Λ es_un((x,y), ActivaunaRegla)
La contradicción entre reglas existe cuando se disparan dos reglas y una regla es la negación de la otra	∀ x,y ContradiccionEntreReglas(x,y) => Tiene((x,y),Reglax=¬ReglaY)

3.3 Ontologías para Bases de Datos Componentes

Las bases de datos componentes de una federación pueden ser: bases de datos tradicionales como las bases de datos relacionales que contienen tablas y relaciones; las bases de datos orientadas a objeto que se describen a través de objetos, atributos y métodos; las bases de datos multimedia que contienen datos multimedia como audio, video e imágenes, además de los datos tradicionales; y finalmente, las bases de datos inteligentes en las que se agrupan a las bases de datos activas, deductivas y los sistemas basados en conocimiento. A continuación se presentan los modelos ontológicos de cada una de estas bases de datos.

3.3.1 Bases de Datos Relacionales

Una base de datos relacional consiste en una colección de tablas (esquemas o relaciones) en las que todas las entradas son datos atómicos como número, texto, fecha, etc. Las bases de datos relacionales están basadas en la teoría de conjuntos. Las relaciones están conformadas por un conjunto de atributos que las definen, son las tablas de la base de datos, así como también son tablas los distintos tipos de relaciones que se pueden generar mediante consultas a las relaciones base. Los elementos de una relación se denominan tuplas, el número de tuplas en una relación es la cardinalidad de la relación, y el número de atributos indica el grado de una relación

3.3.1.1 Conceptos de Bases de Datos Relacionales

En una base de datos relacional se encuentran los siguientes conceptos:

Esquemas definen los atributos que conforman una relación, sus claves propias y ajenas. Es lo que corresponde al concepto de tabla.

Interrelaciones**:** tipo de relación que poseen las relaciones entre sí y que definen el comportamiento de la base de datos. El tipo de relación está dado de acuerdo a la cardinalidad en las relaciones: uno-a-uno (donde la clave primaria o propia de una relación se convierte en la clave ajena de la otra), uno-a-muchos (donde la clave de la relación del lado de uno se convierte en clave ajena de la relación del lado de muchos), y muchos-a-muchos (donde las claves propias de ambas relaciones generan una nueva tabla).

El ***dominio D*** de una relación *R* es la colección de valores posibles para un determinado atributo.

La ***clave primaria*** de la relación, que se define como un subconjunto *C* de los atributos de *R*, y cuyos valores no pueden ser repetidos, es *mínima* en el sentido de que en su composición intervienen solamente los atributos estrictamente requeridos para identificar las tuplas de forma única. Una clave primaria puede ser simple (formada por un solo atributo) o compuesta (formada por más de un atributo) [Date C. J, 2005].

Las ***claves ajenas o externas*** son aquellas claves primarias de una relación que se encuentran en otras relaciones, consiguiendo así la interrelación entre ellas.

En la siguiente figura se muestra la taxonomía de conceptos para las bases de datos relacionales.

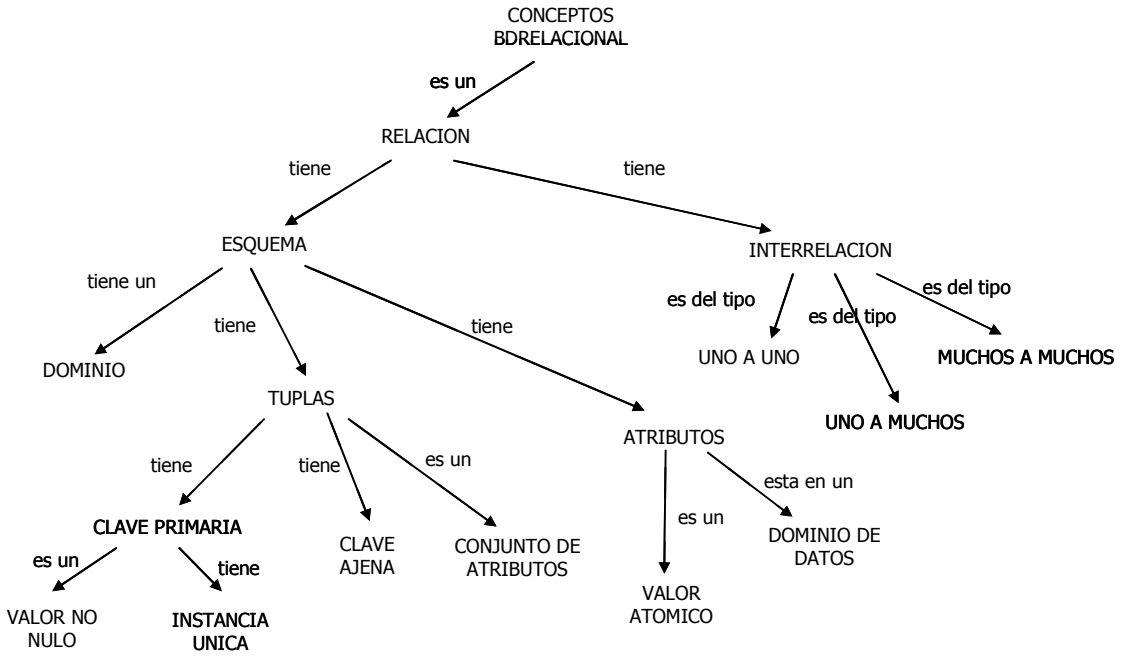

Figura 3.6. Taxonomía de Conceptos para Bases de Datos Relacionales

Las bases de datos relacionales poseen los siguientes atributos que describen sus características:

BASE_DE_DATO_RELACIONAL (Id_BDR, Nombre_BDR, Dirección, Usuario, Tipo_BDR, Esquema_BDR, Lenguaje_de_Consulta, SABD)

Donde:

Id_BDR: identificador de la BDR, este identificador es único

Nombre_BDR, donde se coloca el nombre de la base de datos

Dirección: Ubicación de la base de datos

Usuario: Propietarios de la BDR

Esquema de la BDR: nombre de los esquemas que conforman la base de datos relacional.

Lenguaje de Consulta: Lenguaje de Consulta que utiliza el Sistema Manejador de Base de Datos. Por ejemplo: SQL

SABD: Sistema Administrador de Base de Datos. Por ejemplo: MySQL, Informix, etc.

Relaciones: tiene, es del tipo, esta_en_un

Los axiomas de los conceptos para Bases de Datos Relacionales se muestran en la tabla 4.

Tabla 4. Axiomas para los conceptos de bases de datos relacionales

Sentencia	LPO
Un concepto de base de datos relacional es una relación	∀ x ConceptoBDRelacional(x) => tiene (x, Relación)
Una relación tiene esquemas y tiene interrelaciones	∀ x Relacion(x) => tiene (x, Esquema) Λ Tiene(x,Interrelaciones)

Un esquema tiene un dominio, tuplas y atributos	∀ x Esquema(x) => tiene(x,Dominio) Λ tiene(x,Tuplas) Λ tiene(x,Atributos)
Una tupla tiene clave primaria, clave ajena y un conjunto de atributos	∀ x Tupla(x) => tiene(x,Clave Primaria) Λ tiene(x,ClaveAjena) Λ tiene(x,ConjuntodeAtributos)
Un atributo es un valor atómico y está en un dominio	∀ x Atributo(x) => es_un(x,ValorAtomico) Λ esta_en_un(x,Dominio)
Una interrelación es del tipo uno a uno, uno a muchos, o muchos a muchos	∀ x Interrelacion(x) => es_del_tipo(x,UnoaUno) V es_del_tipo (x,UnoaMuchos) V es_del_tipo (x,MuchosaMuchos)

3.3.1.2 Operaciones de Bases de Datos Relacionales

Las operaciones del modelo relacional se clasifican en consultas y actualizaciones. Las operaciones de actualización pueden ser insertar, modificar y eliminar una o más tuplas de una relación dada [Date C. J, 2005]. La operación **Insertar** proporciona una lista de valores de los atributos para una nueva tupla t que se insertará en una relación R. La operación **Eliminar** elimina una tupla t de una relación R. La operación **Modificar** sirve para cambiar los valores de uno o más atributos en una tupla de una relación R. Surgen violaciones solo si se modifica un atributo clave, clave externa o clave primaria.

Las **operaciones de consulta** son las operaciones estándar del álgebra relacional, que permiten seleccionar tuplas y combinar tuplas a partir de varias relaciones, dando como resultado una nueva relación. Se clasifican en dos grupos: las operaciones de la teoría de conjuntos que son la **Unión**, la **Intersección**, la **Diferencia** y el **Producto Cartesiano**, y las operaciones propias del álgebra relacional como **seleccionar**, **proyectar** y **reunir** [Date C. J. 2005].

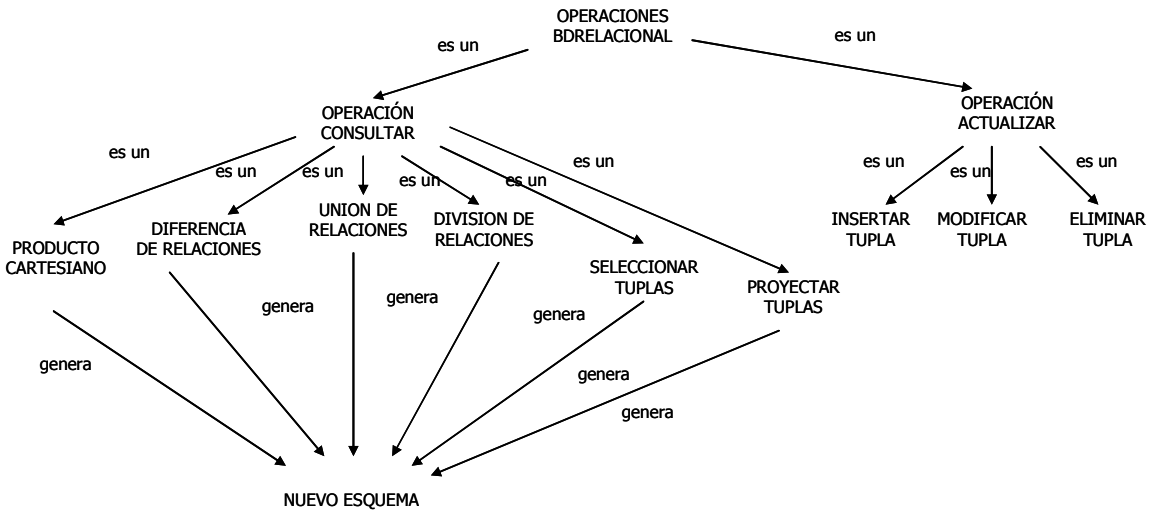

Figura 3.7. Taxonomía de Operaciones para las Bases de Datos Relacionales

Los axiomas de las operaciones para Bases de Datos Relacionales se muestran en la tabla 5.

Tabla 5. Axiomas para las operaciones de bases de datos relacionales

Sentencias	LPO
Una operación de Base de Datos Relacional es una operación consultar relacional o una operación actualizar relacional	∀ x OperacionBDRelacional(x) => es_un(x,OperacionConsultarR) V es_un(x,OperaciónActualizarR)
Una operación consultar relacional es un producto cartesiano o una diferencia relacional, o una unión de relaciones, o una división de relaciones o una selección de tuplas o una proyección de tuplas	∀ x OperaciónConsultarR(x) => es_un(x,OperacionProductoCartesianoR) V es_un (x,OperacionDiferenciadeRelaciones) V es_un (x,OperacionUnionRelaciones) V es_un (x,OperacionDivisiondeRelaciones) V es_un(x,OperacionSeleccionde Tuplas) V es_un (x,OperacionProyeccionTuplas)
Una operación producto cartesiano genera un nuevo esquema	∀ x OperacionProductoCartesianoR(x) => Genera(x,NuevoEsquema)
Una operación diferencia de relaciones genera un nuevo esquema	∀ x OperacionDiferenciaRelaciones(x) => Genera(x,Nuevoesquema)
Una operación unión de relaciones genera un nuevo esquema	∀ x OperacionUniondeRelaciones(x) => Genera(x,Nuevoesquema)
Una operación unión de relaciones genera un nuevo esquema	∀ x OperacionUniondeRelaciones(x) => Genera(x,Nuevoesquema)
Una operación división de relaciones genera un nuevo esquema	∀ x OperacionDivisiondeRelaciones(x) => Genera(x,Nuevoesquema)
Una operación selección de tuplas genera una nuevo esquema	∀ x OperacionSeleccionde Tuplas(x) => Genera(x,Nuevoesquema)
Una operación de proyección genera nuevo esquema	∀ x OperacionProyeccion(x) => Genera(x,NuevoEsquema)
Un nuevo esquema es un esquema de base de datos	∀ x NuevoEsquema(x) => es_un(x,Esquema)
Una operación actualizar es una operación insertar o una operación modificar o una operación eliminar	∀ x OperacionActualizar(x) => es_un(x,OperacionInsertar) V es_un (x,OperaciónModificar) V es_un (x,OperacionEliminar)

3.3.1.3 Restricciones de Bases de Datos Relacionales

A continuación se describen las restricciones a ser consideradas al realizar operaciones en las bases de datos relacionales, y en el caso de que se violen qué acciones tomar. Particularmente, consideraremos las restricciones de integridad, llamadas también las reglas del negocio que gobiernan el universo del discurso en las bases de datos relacionales. Los tipos de restricciones de integridad son:

- *Restricciones de Dominio*. Son las restricciones sobre los tipos de valores que puede tener un atributo. Especifican que el valor de cada atributo A debe ser un valor atómico del dominio dom(A) para ese atributo.
- *Restricciones de clave*. El valor del atributo que identifica de manera única una tupla en una relación. Se especifica sobre relaciones individuales.
- *Restricciones de Integridad de Entidades*. Establece que ningún valor de clave primaria puede ser nulo. Se especifica sobre relaciones individuales.

- *Restricción de Integridad Referencial*. Se especifica entre dos relaciones, y permite mantener la consistencia entre las tuplas de las dos relaciones. Se lleva a cabo a través de las claves externas o ajenas.
- *Restricciones generales o restricciones de integridad semántica*, son aquellas que se establecen en las solicitudes a realizar en las consultas o vistas, por ejemplo: Buscar profesores cuyo número máximo de horas trabajadas en un proyecto por semana es 40.

La operación de inserción puede violar cualquiera de los cuatro tipos de restricciones: *de dominio* (si proporciona un valor de atributo que no se encuentra en el dominio correspondiente), *de clave* (si el valor clave de la nueva tupla t ya existe en otra tupla de la relación r(R)), de *entidades* (si la clave primaria de la nueva tupla t es nula), y la *integridad referencial* (si el valor de cualquier clave externa de t hace referencia a una tupla que no existe en la relación referida). Si una operación de inserción viola una o más restricciones se pueden realizar las siguientes acciones: rechazar la inserción o corregir la razón por la que se rechazó la inserción.

La operación Eliminar sólo puede violar la integridad referencial (que sucede cuando las claves externas de otras tuplas de la base de datos hacen referencia a la tupla que se va a eliminar). Existen tres acciones a tomar: 1) rechazar la eliminación; 2) tratar de propagar la eliminación, eliminando las tuplas que hacen referencia a la tupla que se desea eliminar; 3) modificar los valores del atributo de referencia que provoca la violación: todos esos valores se pondrían nulos, siempre y cuando no sean claves, o se cambiarían a otra tupla válida.

En la operación Modificar surgen violaciones solo si se realizan modificaciones a un atributo clave, externa o primaria, y se resuelven de la misma manera que para insertar o eliminar.

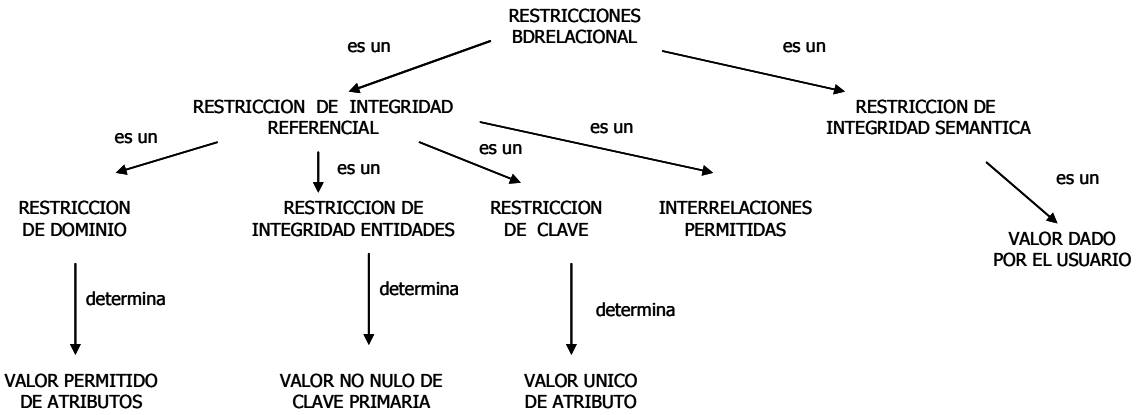

Figura 3.8. Taxonomía de Restricciones para Bases de Datos Relacionales

Los axiomas de las restricciones para Bases de Datos Relacionales se muestran en la tabla 6.

Tabla 6. Axiomas para las restricciones de bases de datos relacionales

Sentencia	LPO
La restricción de una BDRelacional es una restricción de integridad referencial o una restricción de integridad semántica	∀ x RestriccionBDRelacional(x) => es_un(x,RestricciónIntegridadReferencial) V es_un(x,RestriccionIntegridadSemantica)
Una restricción de integridad semántica es un valor dado por el usuario	∀ x RestriccionIntegridadSemantica(x) => es_un(x,ValordeUsuario)
Una restricción de integridad referencial es una restricción de dominio	∀ x RestricciondeIntegridadReferencial(x) => es_un(x,RestriccióndeDominio)
Una restricción de dominio determina los valores permitidos de los atributos	∀ x RestriccióndeDominio(x) => Determina(x,ValorPermitidodeAtributo)
La restricción de integridad de entidades determina un valor no nulo de clave	∀ x RestricciondeIntegridaddeEntidad(x) => Determina(x,ValornoNulodeClave)
La restricción de clave determina un valor único de la clave	∀ x RestricciondeClave(x) => Determina(x,ValorUnicodeClave)
La restricción de integridad referencial es una interrelación permitida	∀ x RestricciondeIntegridadReferencial(x) => es_un(x,InterrelacionPermitida)

3.3.2 Bases de Datos Orientadas a Objeto

Las bases de datos orientadas a objetos están basadas en varios conceptos fundamentales: cada entidad del mundo real es modelada como un *objeto*. Cada objeto está asociado con un identificador único y tiene un conjunto de atributos y métodos; a su vez, el valor de un atributo puede ser un objeto o un conjunto de objetos. El conjunto de atributos de un objeto representa la estructura, y el conjunto de métodos el comportamiento. El estado de un objeto es accesado ó modificado a través del envío de mensajes al objeto para invocar los métodos correspondientes. Los objetos que comparten la misma estructura y comportamiento se agrupan en clases. Una clase representa una plantilla para un conjunto de objetos similares. Cada objeto es una instancia de una clase. Una clase puede ser definida como una especialización de una o más clases. Una clase definida como una especialización se le llama subclase y hereda atributos y métodos de sus superclases.

3.3.2.1 Conceptos de Bases de Datos Orientadas a Objeto

Objeto: es cualquier cosa real ó abstracta acerca de la cual se almacenan datos y los métodos que controlan dichos datos. Un objeto está compuesto por el identificador de objetos, el constructor del tipo de objeto y el valor del objeto.

Identificador de Objetos: Es un concepto o valor que identifica al objeto de manera única y es inmutable. Se le genera al objeto en el momento de ser creado.

Tipo de Objeto: Es un conjunto de objetos que tienen un mismo comportamiento. Los tipos de objeto se definen a través de los constructores de tipo de objeto: átomos, tuplas, conjuntos, listas y arreglos.

Los objetos tienen un ***Dominio*** que contienen los valores atómicos básicos disponibles como son: enteros, reales, cadena de caracteres, booleanos, fechas, y cualquier otro tipo

de datos que maneje la BDOO directamente. Además, puede tener valores no atómicos, que se refieren a un conjunto de objetos del mismo tipo como son tuplas, listas y arreglos.

Clase: Es la agrupación de objetos parecidos. Especifica una estructura de datos y los métodos operativos permitidos que se aplican a cada uno de sus objetos [Cattell R., Barry D. 2000].

Jerarquía de clases: definición de clases a partir de clases predefinidos. Cuando las clases son parecidas se agrupan en subclases.

Herencia: tanto los atributos, como las operaciones de las clases, son heredados por las subclases.

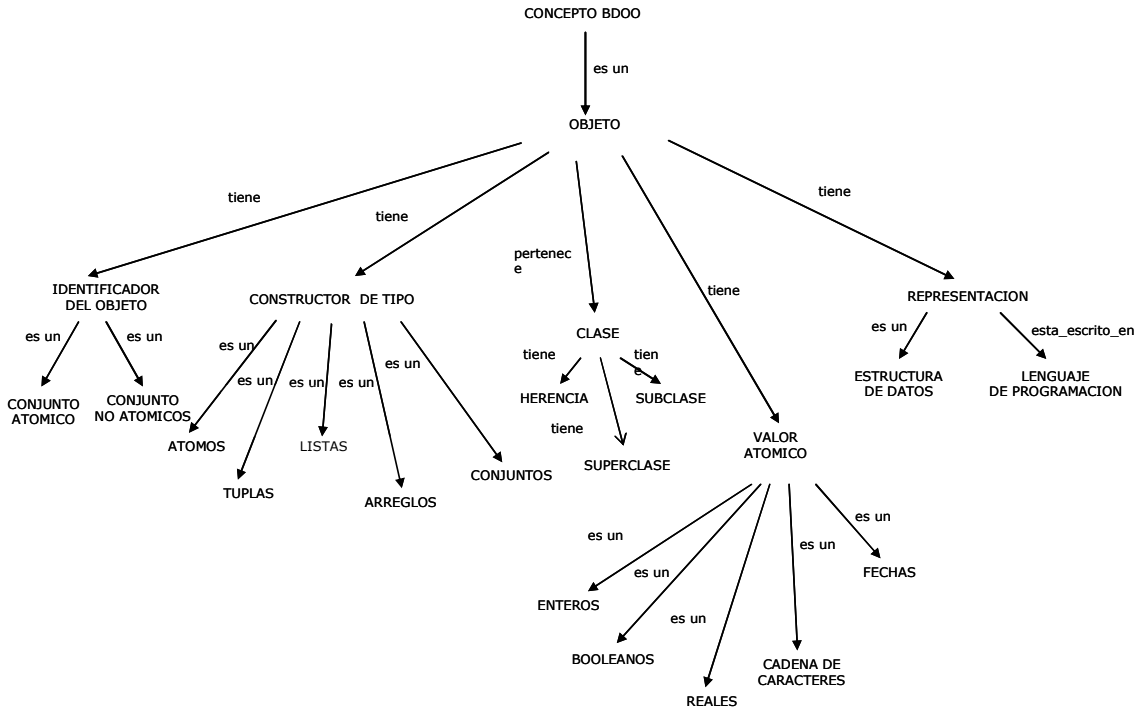

Figura 3.9. Taxonomía de Conceptos para Bases de Datos OO

Las bases de datos relacionales poseen los siguientes atributos que describen sus características:

BASE_DE_DATO_ORIENTADA_A_OBJETOS (Id_BDOO, Nombre_BDOO, Dirección, Usuario, Esquema_BDOO, Lenguaje_de_Consulta, SABD)

Donde:

Id_BDOO: identificador de la BDOO, este identificador es único

Nombre_BDOO, donde se coloca el nombre de la base de datos

Dirección: Ubicación de la base de datos

Usuario: Propietarios de la BDOO

Esquema de la BDOO: nombre de los esquemas que conforman la base de datos orientada a objetos.

Lenguaje de Consulta: Lenguaje de Consulta que utiliza el Sistema Manejador de Base de Datos. Por ejemplo: SQL

SABD: Sistema Administrador de Base de Datos. Por ejemplo: MySQL, Informix, etc.
Relaciones: es_un, tiene, pertenece

Los axiomas de los conceptos para las Bases de Datos Orientadas a Objetos se muestran en la tabla 7.

Tabla 7. Axiomas que conforman los conceptos para las bases de datos orientadas a objetos

Sentencia	LPO
Un concepto de base de datos orientada a objetos es un objeto	∀ x ConceptoBDOO(x) => es_un(x,Objeto)
Un objeto tiene identificador de objeto, constructor de tipo, y valor atómico	∀ x Objeto(x) => Tiene(x,IdentificadorObjeto) Λ Tiene(x,ConstructordeTipo) Λ Tiene(x,Valor_atómico)
Un objeto pertenece a una clase	∀ x Objeto(x) => Pertenece_a (x, Clase)
Una clase tiene herencia, superclases y subclases	V x Clase(x) => Tiene (x, Herencia) Λ Tiene(x, Subclases) Λ Tiene(x, Superclases)
Un identificador de objeto puede ser un conjunto atómico o un conjunto no atómico	∀ x IdentificadordeObjeto(x) => es_Un(x,ConjuntoAtómico) V es_Un (x,ConjuntoNoAtómico)
Un constructor de tipo puede ser un átomo o una tupla o una lista o un arreglo o un conjunto	∀ x ConstructordeTipo(x) => es_Un(x,Atomo) V es_Un(x,Tupla) V es_Un(x,Lista) V es_Un(x.Conjunto) V es_Un(x,Arreglo)
Un valor atómico es un entero, booleano, real, cadena de caracteres, fecha	∀ x ValorAtomico(x) => es_Un(x,Entero) V es_Un(x,Booleano) V es_Un(x,Real) V es_Un(x,Cadena) V es_Un(x,Fecha)
Una representación es una estructura de datos	∀ x Representación (x) => es_un (x,EstructuraDatos)
Una representación está escrita en un lenguaje de programación	∀ x Representación (x) => esta_escrita_en_un (x, Lenguaje de_Programacion)

3.3.2.2 Operaciones de Bases de Datos Orientadas a Objeto

Los objetos son manipulados a través de los métodos que consisten de dos partes: un primer componente llamado signatura o interfaz de la operación, que especifica el nombre del método, los nombres y clases de los argumentos, y los resultados si existen; y un segundo componente, que especifica la implementación de la operación, que es un código escrito en un lenguaje de programación.

Las operaciones se invocan pasándole un mensaje a un objeto, que incluye el nombre de la operación y los parámetros. Luego el objeto ejecutará el método para esa operación. Se denomina ***Solicitud*** a la invocación de una operación específica, con uno ó más objetos como parámetros. Por otro lado, los objetos tienen una representación que viene dada por una estructura de datos y un lenguaje de implantación.

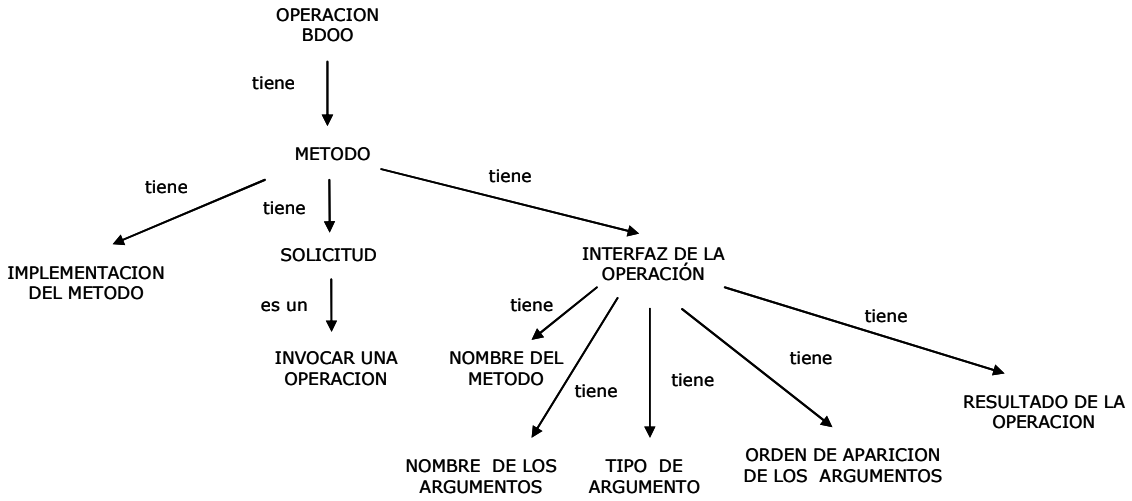

Figura 3.10. Taxonomía de Operaciones para Bases de Datos Orientadas a Objetos

Los Axiomas de las operaciones para las Bases de Datos Orientadas a Objetos se muestran en la tabla 8.

Tabla 8. Axiomas que conforman las operaciones para las bases de datos orientadas a objeto

Sentencia	LPO
Una operación de bases de datos orientadas tiene métodos y tiene representación	∀ x OperacionBDOO(x) => tiene (x,Metodos) Λ tiene (x,Representación)
Un método tiene implementación del método, tiene solicitudes y tiene interfaz de la operación	∀ x Metodo(x) => Tiene(x,ImplementaciondelMetodo) Λ Tiene (x,Solicitud) Λ Tiene(x,InterfazdeOperacion)
Una solicitud es una invocación de una operación	∀ x Solicitud(x) => es_un(x,InvocaciónOperación)
Una interfaz de operación tiene nombre del método, tiene nombre de los argumentos y tiene resultados de la operación	∀ x InterfazdeOperación(x) => tiene(x,NombredeMetodo) Λ tiene(x,NombredeArgumento) Λ tiene(x,TipodeArgumento) Λ tiene(x,OrdendeAparicionesdelosArgumentos) Λ tiene(x,ResultadosdeOperacion)

3.3.2.3 Restricciones de Bases de Datos Orientadas a Objeto

Las bases de datos orientadas a objeto poseen las mismas restricciones que las bases de datos relacionales, con la diferencia de que en lugar de referirse a tuplas y relaciones se refieren al objeto. Cada tipo de objeto tiene sus restricciones de integridad programadas en los métodos que crean, eliminan y actualizan los objetos.

- Las restricciones de Integridad Referencial de Objetos vienen dadas por:
  - **Restricción de dominio**, que son las restricciones sobre los valores que pueden tomar las variables que conforman un objeto.
  - **Restricción de integridad de objeto**, establece que ningún valor del identificador de objeto puede ser nulo.

- **Asociaciones permitidas**, especifican el tipo de asociaciones entre objetos.

- Las restricciones de integridad semánticas son aquellas que se establecen en las consultas.

Las operaciones de actualización (insertar, modificar y eliminar) no deben violar las restricciones de integridad especificadas.

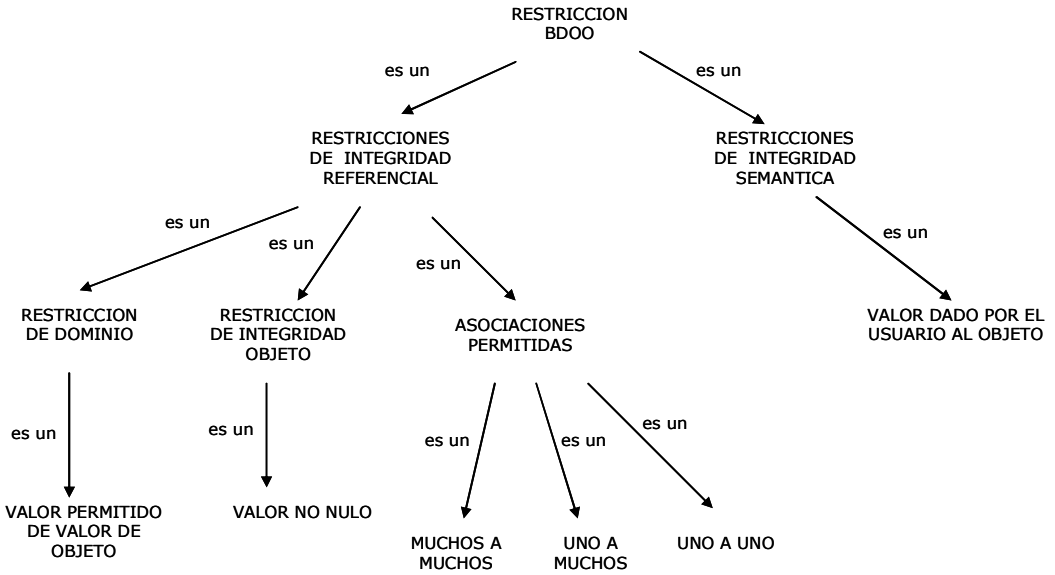

Figura 3.11. Taxonomía de las Restricciones de Bases de Datos OO

Los axiomas de las restricciones para Bases de Datos Orientadas a Objeto se muestran en la tabla 9.

Tabla 9. Axiomas que conforman las restricciones para bases de datos orientadas a objetos.

Sentencia	LPO
Una restricción de BDOO puede ser una restricción de integridad semántica o una restricción de integridad referencial	∀ x RestriccionBDOO(x) => es_un(x,RestricciónIntegridadSemantica) V es_Un(x, RestricciónIntegridadReferencial
Una restricción de integridad semántica es un valor de objeto dado por el usuario	∀ x RestricciónIntegridadSemantica(x) => EsUn(x,ValordefinidodeObjeto)
Una restricción de integridad referencial puede ser una restricción de dominio, o una restricción de integridad de objeto, o el tipo de asociación permitida	∀ x RestriccionIntegridadReferencial(x) => es_un (x,RestricciondeDominio) V es_un(x,RestriccionIntegridadObjeto) V es_un (x,AsociaciónPermitida)
Una restricción de dominio es un valor permitido de objeto	∀ x RestricciondeDominio(x) => es_un(x,ValorPermitido)
Una restricción de integridad de objeto es un valor no nulo	∀ x RestriccionIntegridadObjeto(x) => es_un(x,ValornoNulo)
Una asociación permitida puede ser del tipo: uno-a-uno o uno-a-muchos o muchos-a-muchos	∀ x AsociaciónPermitida(x) => es_un(x,UnoaUno) V es_un (x,UnoaMuchos) V es_un (x,MuchosaMuchos)

3.3.3 Bases de Datos Multimedia

Siguiendo la metodología Methontology (Fernandez-López et al, 1997) para desarrollar ontologías, y la arquitectura ontológica para bases de datos propuesta en (Muñoz et al, 2006); (Muñoz, Aguilar, Martínez 2005); (Muñoz, Aguilar, 2005), a continuación se presenta el modelo ontológico para las bases de datos multimedia. En general, las bases de datos multimedia poseen conceptos, operaciones y restricciones.

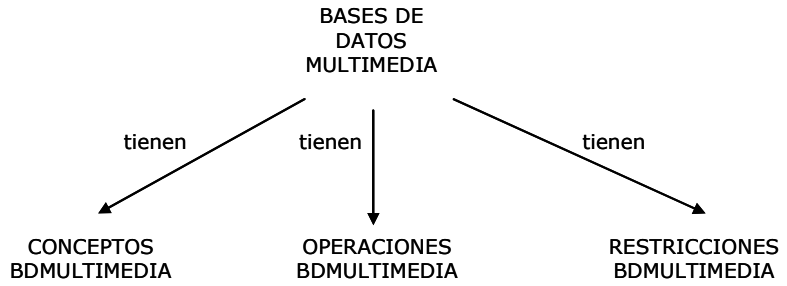

Figura 3.12. Taxonomía de las Bases de Datos Multimedia

Las bases de datos multimedia poseen los siguientes atributos que describen sus características.

BASE_DE_DATO_MULTIMEDIA (Id_BDMM, Nombre_BDMM, Dirección, Usuario, Tipo_BDMM, Esquema_BDMM, Modelo_BDMM, Lenguaje_de_Consulta, SABD)

Donde:

Id_BDMM: identificador de la BDMM, este identificador es único

Nombre_BDMM, donde se coloca el nombre de la base de datos

Dirección: Ubicación de la base de datos

Usuario: Propietarios de la BDMM

Tipo_BDMM: tipo de base de datos multimedia, por ejemplo de video, de música, hibrida, etc.

Esquema de la BDMM: nombre de los esquemas que conforman la base de datos multimedia. Por ejemplo: General, Metadata, etc.

Modelo_BDMM: modelo utilizado por la Base de Datos Multimedia para representar sus esquemas. Por ejemplo: Orientados a Objeto, Entidad-Relación.

Lenguaje de Consulta: Lenguaje de Consulta que utiliza el Sistema Manejador de Base de Datos. Por ejemplo: SQL

SABD: Sistema Administrador de Base de Datos. Por ejemplo: MySQL, Informix, etc.

Relaciones: tiene

En la siguiente tabla se muestran los axiomas para esta taxonomía.

Tabla 10. Axiomas de las bases de datos multimedia

Sentencia	LPO
Una base de datos Multimedia tiene conceptos, operaciones y restricciones	∀ x BDMultimedia(x) => tiene(x,ConceptoBDMM) ∧ tiene(x,OperacionesBDMM) ∧ tiene(x,RestriccionesBDMM)

3.3.3.1 Conceptos de Bases de Datos Multimedia

La figura 3.10 muestra la representación taxonómica de la ontología de los conceptos de las bases de datos multimedia. A continuación se listan los atributos que poseen los conceptos de la Base de Datos Multimedia, se describen a manera de esquemas de Bases de datos.

BDMM (Nombre, ModeloBDMM)
METADATO_TEXTO (Formato_Texto, Tipo_Font, Tamaño_Pagina, Descripción_Contenido_Texto, Representación_Texto, Documento_Histórico, Localizacion_Documento)
METADATO_HABLA (Formato_Compresion, TiempoInicial_Discurso, Tiempo_Final_Discurso, Nombre_del_Hablante, Información_Prosodica=Significado)
IMAGEN_SATELITAL(Fila, Columna, Capa_Profundidad)
IMAGEN_DISEÑO_ARQUITECTÓNICO(Nombre_Edificación, Ubicación, Nombre_Arquitecto, Costo_Estimado, Nombre_Compañía)
METADATO_IMAGEN_FACIAL(Color_Cabello, Descripción_de_Ojos, descripción_Nariz, Descripción_Boca, Nombre_Persona, Sexo_Persona, Identificación_Persona)
METADATO_VIDEO (Frames, Formato, Movimiento_de_Camara, Altura_Camara, Nivel_de_Luz, Objetos_en_Secuencia_de_Video, Enfoque_de_Camara, Angulo_de_Toma, Movimiento_de_la_Toma, Descripción_de_la_Acción, Tipo_de_Objetos_en_la_Toma, Fecha_de_Produccion, Nombre_Productor, Nombre_Director)
HERRAMIENTA_SOPORTE_USUARIO (NombreHerramientaSU, SistemaOperativoSU, LenguajeSU)
HERRAMIENTA_SOPORTE_APLICACION (Nombre_HerramientaSA, SistemaOperativoSA, LenguajeSA)
DATO_TEXTO (Titulo, Autor, Afiliacion_Autor, Resumen, Secciones, Subsecciones, Parrafos, Lenguaje_de_Representacion)
DATO_AUDIO (Tipo_Audio, Calidad_Digitalización)
DATO_IMAGEN (Tipo_Imagen, Color, Formato)
DATO_GRAFICO (Descripción, Nombre_Aplicacion)
DATO_VIDEO (Estandar, Esquema_Compresion)
DATO_MEDIA_GENERADO (Tipo_Presentacion, Formato)
Relaciones: tiene, es_un

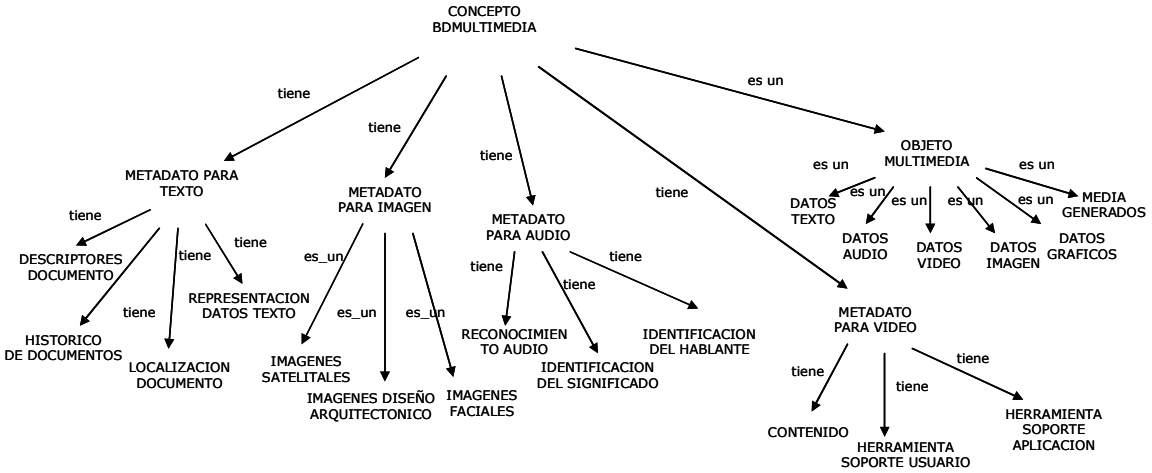

Figura 3.13. Taxonomía de Conceptos para Bases de Datos Multimedia

Los axiomas de los conceptos para las Bases de Datos Multimedia se muestran en la tabla 11.

Tabla 11. Axiomas de los conceptos de bases de datos multimedia

Sentencia	LPO
Un concepto de base de datos multimedia tiene metadato para texto, metadato para imagen, metadato para audio, o metadato para video	∀ x ConceptoBDMultimedia(x) => tiene(x,MetadatoTexto) V tiene(x,MetadatoImagen) V tiene(x,MetadatoAudio) V tiene(x,MetadatoVideo)
Un concepto de base de datos multimedia es un objeto multimedia	∀ x ConceptoBDMultimedia(x) => es_un(x,ObjetoMM)
Un metadato para texto tiene descriptores de documento, histórico del documento, Localización del documento y representación de datos texto	∀ x MetadatoTexto(x) => tiene(x,DescriptorDocumento) Λ tiene(x,HistoricoDocumento) Λ tiene(x,LocalizacionDocumento) Λ tiene(x,Representación_Datos_Texto)
Un metadato para imagen puede ser una imagen satelital, imagen de Diseño arquitectónico, o una imagen facial	∀ x MetadatoImagen(x) => es_un(x,ImagenSatelital) V es_un(x,ImagendeDiseñoArquitectonico) V es_un(x,ImagenFacial)
Un metadato para video tiene contenido de video, herramientas de soporte al usuario y herramientas de soporte de aplicación	∀ x MetadatoVideo(x) => tiene (x,Contenido_de_Video) Λ tiene (x,Herramientas_Sop_Usurio) Λ tiene (x,Herramientas_Sop_Aplicacion)
Un metadato para audio tiene reconocimiento del habla, Identificación del significado e identificación del hablante	∀ x MetadatoAudio(x) => tiene (x,Reconocimiento_del_Habla) Λ tiene (x,Identificacion_del_Significado) Λ tiene (x,Identificacion_del_Hablante)
Un objeto multimedia es un dato texto, dato audio,	∀ x ObjetoMM(x) => es_Un(Dato_Texto, x) V es_un(x, Dato_Audio) V es_un (x, Dato_Imagen) V es_un(x, Dato_Grafico) V es_un (x,

dato imagen, dato gráfico, dato video o datos media generados.	Dato_Video) V es_un (x, Dato_MediaGenerado)

3.3.3.2 Operaciones de Bases de Datos Multimedia

En estas bases de datos, además de las operaciones propias de las bases de datos relacionales, ellas poseen operaciones dadas por los tipos de datos que ellas manejan: video, imagen, texto y audio. Las operaciones de las bases de datos multimedia son: representación de datos, consultas y actualizaciones, búsqueda y edición, determinación de calidad de servicios procesados, y administración de seguridad/integridad. Por otro lado, los datos multimedia manejan el tiempo de la siguiente manera:

- *Instantes*. Representan chronons[25] simples, son utilizados para representar el instante en el que ocurre el hecho.
- *Periodos temporales*. Representa el conjunto de instantes contenidos entre dos instantes específicos comienzo y fin.
- *Intervalos Temporales*. Denota un tamaño de tiempo específico, sin precisar el instante de inicio y de fin. Por ejemplo, un día.
- *Conjunto de instantes*. Es utilizado para representar un conjunto de instantes finitos, no necesariamente contiguos, que eventualmente se repiten.
- *Elementos temporales*. Es un conjunto finito de períodos temporales no necesariamente contiguos.

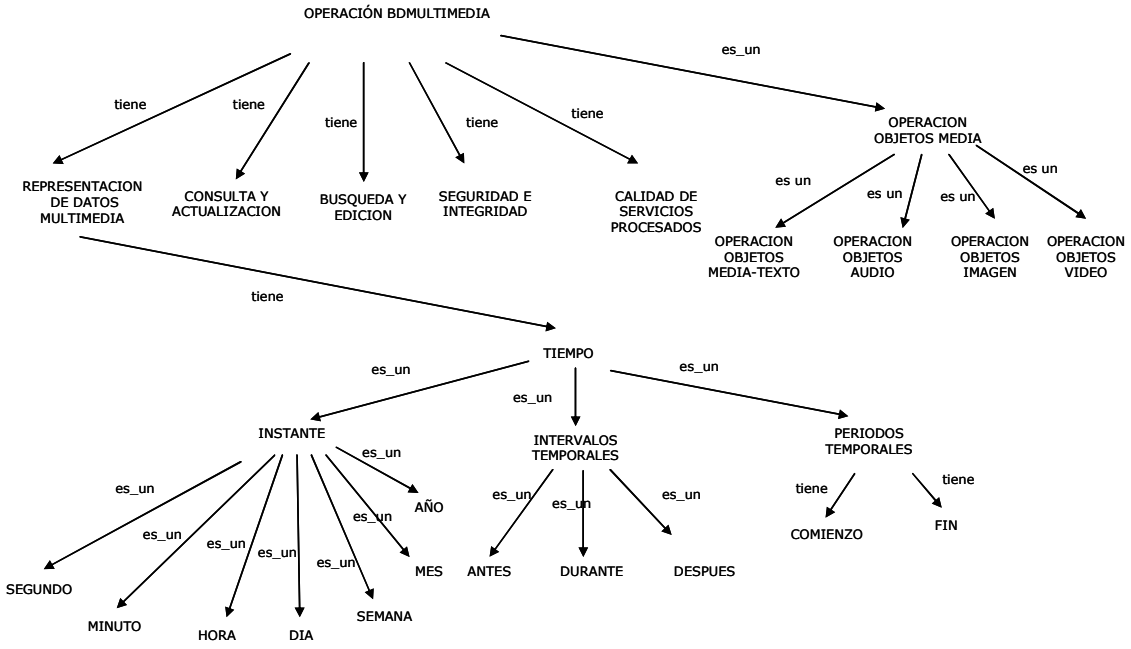

Figura 3.14. Taxonomía de Operaciones para Bases de Datos Multimedia

A continuación se presentan los atributos para las operaciones de las bases de datos Multimedia

[25] Unidad indivisible y discreta de tiempo ($2x10^{-23}$ seg.)

OPERACIÓN_OBJETOSMEDIA (Iniciar_Presentación, Parar_Presentación)
OPERACIÓN_OBJETOSMEDIA_TEXTO (Presentar_Texto)
OPERACIÓN_OBJETOSMEDIA_AUDIO (Presentar_Audio)
OPERACIÓN_OBJETOSMEDIA_IMAGEN (Presentar_Imagen)
OPERACIÓN_OBJETOSMEDIA_VIDEO (Presentar_Video)

Los axiomas de las restricciones se muestran en la tabla 12.

Tabla 12. Axiomas que conforman las operaciones para bases de datos multimedia

Sentencia	LPO
Una operación de Base de Datos Multimedia es una consulta, una actualización, una búsqueda y edición o una operación de objetos media	∀ x OperacionBDMM(x) => es_un(x,ConsultaBDMM) V es_un(x,ActualizacionBDMM) V es_un(x,BusquedaEdicion) V es_un (x,Operación_Objetos_Media)
Una operación de bases de datos multimedia tiene seguridad, integridad y calidad de servicios procesados	∀ x OperacionBDMM(x) => tiene (x, Seguridad) Λ tiene (x, Integridad) Λ tiene (x, CalidadServiciosProcesados)
Una representación de datos multimedia tiene tiempo	∀ x RepresentacionDatosMM(x) => tiene(x,Tiempo)
El tiempo es un instante, un periodo temporal, o un intervalo temporal	∀ x Tiempo(x) => es_un(x,Instante) V es_un (x,PeriodoTemporal) V es_un(x,IntervaloTemporal)
Un instante puede ser un segundo o un minuto o una hora o un día o una semana o un año	∀ x Instante(x) => es_un(x,Segundo) V es_un (x,Minuto) V es_un (x,Dia) V es_un (x,Hora) V es_un(x,Semana) V es_un (x,Mes) V es_un (x,Año)
El intervalo temporal es un antes, un durante o un después	∀ x IntervaloTemporal(x) => es_un(x,Antes) V es_un(x,Durante) V es_un(x,Después)
Un periodo temporal tiene un comienzo y tiene un fin	∀ x PeriodoTemporal(x) => tiene(x,Comienzo) Λ tiene (x,Fin)
Una Operación de Objetos media es una operación de objetos media texto, objetos media audio, objetos media video u objetos media imagen	∀ x Operación_Objetos_media (x)=> es_un(x,ObjetoMedia_Texto) V es_un(x,ObjetoMedia_Audio) V es_un(x,ObjetoMedia_Video) V es_un (x,ObjetoMedia_Imagen)

Los objetos multimedia poseen además las operaciones propias de un objeto, como se muestran en la figura 3.9 y los axiomas de las tablas 7 y 8.

3.3.3.3 Restricciones de Bases de Datos Multimedia

Las restricciones vienen dadas por las restricciones de integridad de las bases de datos orientadas a objeto, así como de la ocurrencia de eventos para la presentación de datos, tales como la representación de espacio y tiempo. Se debe tener en cuenta la ubicación del objeto multimedia en la pantalla, por lo que se debe verificar que se encuentre dentro de las coordenadas especificadas y el punto de referencia. En la figura 3.15 se muestra la taxonomía para las restricciones de bases de datos multimedia.

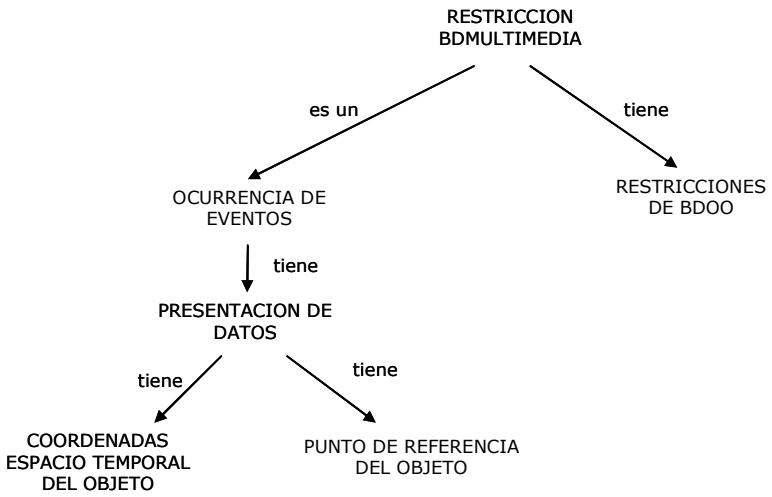

Figura 3.15. Taxonomía de Restricciones para Bases de Datos Multimedia

Los axiomas de las operaciones se muestran en la tabla 13.

Tabla 13. Axiomas que conforman las restricciones de bases de datos multimedia

Sentencia	LPO
Una restricción de bases de datos multimedia es una ocurrencia de eventos y tiene restricciones de BDOO	∀ x RestriccionBDMultimedia(x) => es_un (x,OcurrenciaEventos) Λ tiene (x,RestriccionBDOO)
Una ocurrencia de eventos tiene presentación de datos	∀ x OcurrenciadeEventos(x) => tiene(x,PresentacionDatos)
La presentación de datos tiene coordenadas espacio-temporal del objeto y tiene punto de referencia del objeto	∀ x PresentaciondeDatos(x) => tiene(x,CoordenadasEspacioTemporal) Λ tiene(x,PuntodeReferencia)

Los objetos multimedia poseen, además, las restricciones propias de un objeto como se mostraron en la figura 3.10 y los axiomas de la tabla 9.

3.3.4 Bases de Datos Inteligentes

Se consideran dentro de las Bases de Datos Inteligentes: las bases de datos activas, las bases de datos deductivas y los sistemas basados en conocimiento [Muñoz A, Aguilar J. Julio 2007, Muñoz A, Aguilar J., Martinez R, 2006]. En la figura 3.16 se muestra la taxonomía para las bases de datos inteligentes.

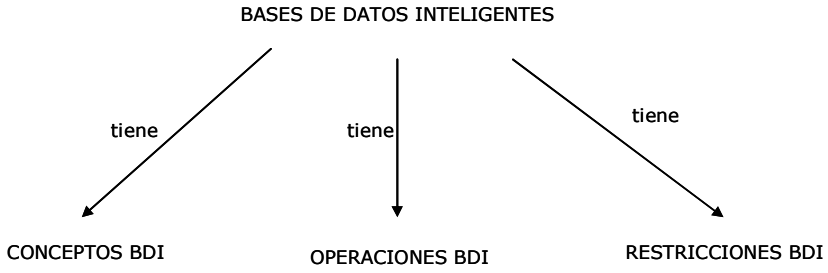

Figura 3.16. Taxonomía de Bases de Datos Inteligentes

Conceptos: BASEDATOSINTELIGENTES, CONCEPTOSBDI, OPERACIONESBDI, RESTRICCIONESBDI.
Relaciones: tiene

Las Bases de Datos Inteligentes poseen los siguientes atributos:
BasesdeDatosInteligentes (ID_BDI, Nombre_BDI, Dirección, Dominio, Esquemas, Modelo), donde:
ID_BDI: identificador de la Base de Datos Inteligente, este identificador es único y permite identificar de manera única cada base de datos
Nombre_BDI: nombre de la base de datos inteligente
Dirección: Dirección electrónica de la base de datos inteligente, que permite ubicar el sitio en que se encuentra la base de datos inteligente para posibles actualizaciones y/o consultas.
Dominio: Dominio al que pertenece la base de datos inteligente, que permite identificar los datos y en con qué áreas de conocimiento trabajan
Esquema de la Base de Datos Inteligente: Descripción de los esquemas que conforman la base de datos, donde se muestran las tablas, tipos de datos y relaciones entre ellas, así como su diccionario.
Modelo de la Base de Datos Inteligente: modelo de datos que utiliza la base de datos inteligente para describir sus esquemas, tales como modelo relacional, orientado a objeto, modelo semántico entre otros.

En la Tabla 14 se muestran los axiomas que conforman el esquema general ontológico de las bases de datos inteligentes.

Tabla 14. Esquema general ontológico de las Bases de Datos Inteligentes que representan los axiomas

Sentencia	LPO
Una Base de Datos Inteligente (BDI) tiene conceptos, operaciones y restricciones	∀ x BDI(x) =>tiene(x,ConceptosBDI) Λ tiene(x,OperacionesBDI) Λ tiene(x,RestriccionesBDI)

Las bases de datos inteligentes poseen conceptos que definen los elementos que las conforman, operaciones que pueden realizar, y restricciones que definen el comportamiento de las reglas y las operaciones de las bases de datos inteligentes.

3.3.4.1 Conceptos de Bases de Datos Inteligentes

En general, las Bases de Datos Inteligentes están conformadas por una base de conocimientos y un mecanismo de razonamiento. La base de conocimientos está compuesta por reglas y hechos. Así, son los sistemas basados en reglas que mediante un esquema de razonamiento van determinando las reglas que se van activando hasta obtener la respuesta a una entrada dada (consulta, evento, etc.). El esquema de razonamiento puede ser deductivo, inductivo o abductivo. Por ejemplo, se podría empezar con una evidencia inicial de una determinada situación para dar con una solución, o bien con hipótesis sobre posibles soluciones y volver hacia atrás para encontrar una evidencia que apoye la hipótesis, entre otras formas de razonamiento. Así, dos conceptos son claves para las bases de datos inteligentes:

- **Base de Conocimientos**: Es una colección de hechos y reglas. Los *hechos* se especifican de manera similar a como se especifican las relaciones en una base de datos relacional. Las reglas pueden ser referidas como "*situación-acción*" o "*if-then*". De esta manera, son invocadas por eventos o consultas que van apareciendo en la base de conocimiento (específicamente en la base de hechos). Las reglas se conectan una a otra por enlaces de asociación para formar redes de reglas.
- **Mecanismo de Razonamiento**: Proceso de razonamiento a partir de los datos de entrada y de la base de conocimientos. Este mecanismo es genérico en el sentido de que puede aplicarse a diferentes dominios con solo cambiar la base de conocimientos.
  La aplicación de las reglas cambian el estado del sistema, y por consiguiente, la base de conocimientos, habilitando unas reglas y deshabilitando otras. El intérprete de reglas usa una estrategia de control para encontrar las reglas disponibles y decidir qué regla debe aplicarse basado en la forma de razonamiento utilizada (deductivo, inductivo, y abductivo) [Bertino E., Catania B., Zarri Gian P. 2001].
  El razonamiento deductivo es el razonamiento a partir de un principio conocido hacia uno desconocido, de lo general a lo particular o de la premisa a la conclusión lógica. El razonamiento abductivo es un método de razonamiento utilizado para generar explicaciones. En la abducción se empieza por una conclusión y se procede a derivar las condiciones que podrían hacer ésta conclusión válida. La abducción trata de explicar la conclusión. El razonamiento inductivo es el razonamiento a partir de hechos particulares para llegar a una conclusión general [Eberhart A 2004].
  Un ejemplo de deducción:
  Regla: "Todas las metras de la bolsa x son blancas".
  Caso:"Estas metras provienen de la bolsa x".
  Conclusión:"Estas metras son blancas".
  Un ejemplo de inducción:

Caso:"Estas metras proceden de la bolsa x"
Conclusión: "Estas metras son blancas".
Regla General:"Todas las metras de la bolsa x son blancas"
Un ejemplo de abducción:
Conclusión:"Estas metras son blancas"
Caso:"Estas metras proceden de la bolsa x".
Regla:"Todos las metras de la bolsa x son blancas".

En la figura 3.17 se muestra la taxonomía de los conceptos para Bases de Datos Inteligentes.

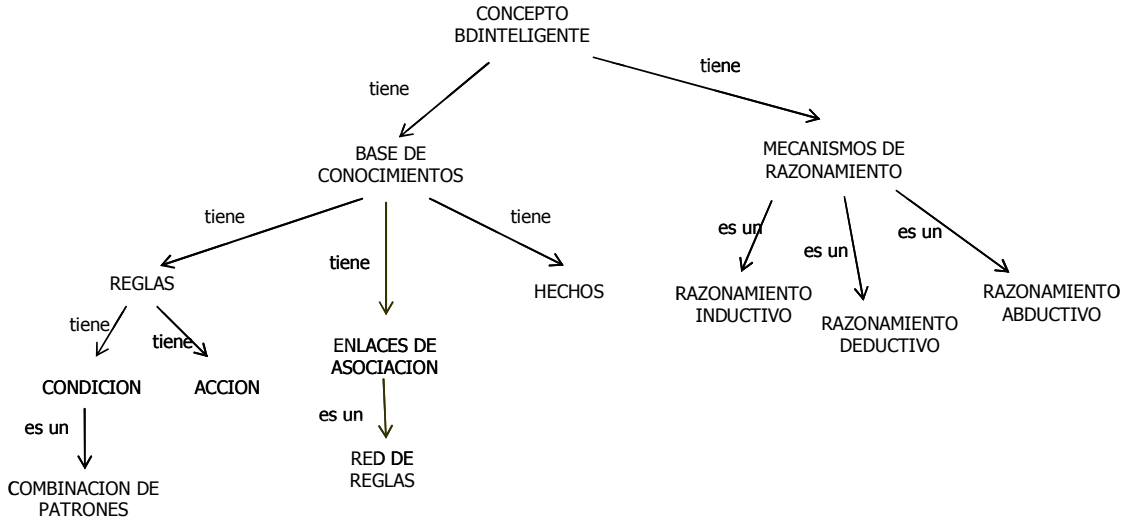

Figura 3.17. Esquema Ontológico de Conceptos para Bases de Datos Inteligentes

En la tabla 15 se muestran los axiomas de los conceptos de las bases de datos inteligentes

Tabla 15. Axiomas de los conceptos de las Bases de Datos Inteligentes

Sentencia	LPO
Un concepto de BDI tiene base de conocimiento y mecanismo de razonamiento	∀ x ConceptoBDInteligente(x) => tiene(x,BasedeConocimiento) Λ tiene(x,MecanismodeRazonamiento)
La base de conocimientos tiene reglas y hechos	∀ x BasedeConocimientos(x) => tiene(x,Reglas) Λ tiene(x,Hechos)
Una regla tiene una condición, una acción y enlaces de asociación	∀ x Regla(x) => tiene(x,Condición) Λ tiene(x,Acción) Λ tiene(x,EnlaceAsociación)
Una condición es una combinación de hechos que se deben dar en la entrada para que se active la regla	∀ x Condicion(x) => es_un(x,CombinaciondeHechos) Λ es_un (x, ActivaciondeRegla)
Un enlace de asociación es un enlace de reglas	V x EnlacedeAsociacion(x) => es_un(x,EnlacedeReglas)
Un mecanismo de razonamiento puede ser deductivo, inductivo o abductivo	V x MecanismoRazonamiento(x) => es_unr(x,RazonamientoDeductivo) V es_un(x,RazonamientoInductivo) V

	es_un(x,RazonamientoAbductivo)
En el razonamiento deductivo la conclusión se obtiene de la hipótesis	∀ x RazonamientoDeductivo(x) => es_un(x,ConclusiondeHipotesis)
En el razonamiento inductivo las conclusiones se obtienen de los hechos	∀ x RazonamientoInductivo(x) => es_un(x, ConclusionesdeHechos)
El razonamiento abductivo trata de explicar la conclusión	∀ x RazonamientoAbductivo(x) => es_un(x, ConclusionesdeHipotesis)

3.3.4.2 Operaciones de Bases de Datos Inteligentes

Las operaciones en las bases de datos inteligentes se realizan a través de la máquina de razonamiento, que es quien controla el disparo de reglas. El ciclo comienza con un evento, y finaliza cuando no existen reglas aplicables. La máquina de razonamiento busca los elementos de la base de conocimientos que cumplen la condición. Luego aplica las reglas, ejecutando acciones que pueden involucrar cambios en la base de conocimiento y/o ambiente.

Existen diferentes estrategias de razonamiento, según el tipo de razonamiento que se esté utilizando: clásicamente pueden ser del tipo *encadenamiento hacia adelante* o *encadenamiento hacia atrás*. Encadenamiento hacia delante, parte de hechos para cumplir condiciones y ejecutar acciones (creando nuevos hechos). Encadenamiento hacia atrás, parte de los estados deseados y trata de cumplir las condiciones necesarias para llegar a ellos [Eun-Hye Choi et al, 2006].

La semántica de la ejecución de las reglas dependen de la forma en cómo se ejecutan las reglas. Existen tres modos de ejecución: inmediato, diferido y desacoplado. Bajo el modo inmediato la regla se procesa tan rápido como sea posible; bajo el modo diferido la regla se procesa al final de la transacción[26]; bajo el modo desacoplado la regla se procesa fuera de la transacción como parte de una transacción separada.

En la siguiente figura se muestra la taxonomía de las operaciones para bases de datos inteligentes.

[26] Interacción con una estructura de datos compuesta por varios procesos u ordenes que se ejecutan formando una unidad de trabajo

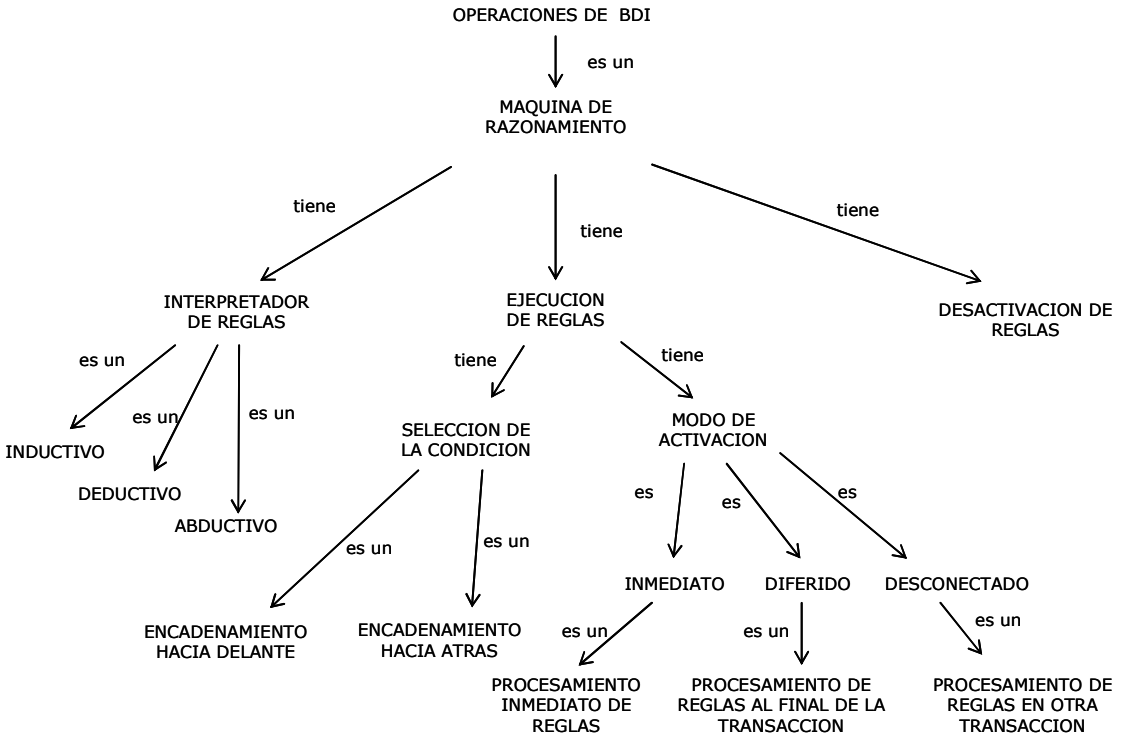

Figura 3.18. Taxonomía de las Operaciones para Bases de Datos Inteligentes

En la tabla 16 se muestran los axiomas de las Operaciones de Bases de Datos Inteligentes

Tabla 16. Axiomas de las Operaciones de las Bases de Datos Inteligentes

Sentencia	LPO
Una operación de BDI es una máquina de razonamiento	∀ x OperaciónBDI(x) => es_un (x, MaquinadeRazonamiento)
Una máquina de razonamiento tiene un interpretador de reglas, una ejecución de reglas, y un desactivador de reglas	∀ x MaquinadeRazonamiento(x) => tiene(x,InterpretadorReglas) Λ tiene(x, EjecuciónReglas) Λ tiene(x, DesactivadorReglas)
Un interpretador de reglas puede realizar razonamiento deductivo, inductivo o abductivo	∀ x Interpretadorde Reglas(x) => es_un(x,RazonamientoDeductivo) V es_un(x,RazonamientoInductivo) V es_un(x,RazonamientoAbductivo)
La ejecución de reglas tiene selección de la condición y modo de acoplamiento	∀ x EjecuciondeReglas(x) => tiene(x,SeleccióndeCondición) Λ tiene(x,MododeActivación)
La selección de la condición se lleva a cabo por encadenamiento hacia delante o encadenamiento hacia atrás	∀ x SeleccióndeCondicion(x) => es_un(x,EncadenamientoHaciaDelante) V es_un(x,EncadenamientoHaciaAtras)
El modo de activación de la regla puede ser inmediata, diferida o desconectada	∀ x MododeActivacion(x) => es_un (x, TransaccionInmediata) V es_un (x,Diferido) V es_un (x,Desacoplada)

El modo de activación inmediato es el procesamiento inmediato de la regla en la transacción	∀ x TransaccionInmediata(x) => es_un(x,ProcesamientoInmediatodelaRegla)
El modo de activación diferido es un procesamiento de la regla al final de la transacción	∀ x Diferido(x) => es_un(x,ProcesamientoFindeTransaccion)
El modo de activación de la regla desacoplado es cuando la regla se procesa como otra transacción	∀ x Desacoplado(x) => es_un(x,ProcesoReglaComo OtraTransacción)

3.3.4.3 Restricciones de Bases de Datos Inteligentes

Las restricciones en las bases de datos inteligentes vienen dadas de acuerdo a las siguientes condiciones:

- Si surge un Disparo Simultáneo de Reglas, ocurre cuando un evento o consulta tiene diferentes reglas asociadas, y el sistema permite que solo una regla se active. Se puede resolver por: selección aleatoria, uso de prioridades, establecer tiempo de activación de la regla, etc.
- Si surge Contradicción entre Reglas, que ocurre cuando un evento o consulta dispara dos reglas, donde la acción de una es la negación de la acción generada por la otra regla. En este caso, se puede resolver al inhibir la activación de una de ellas.

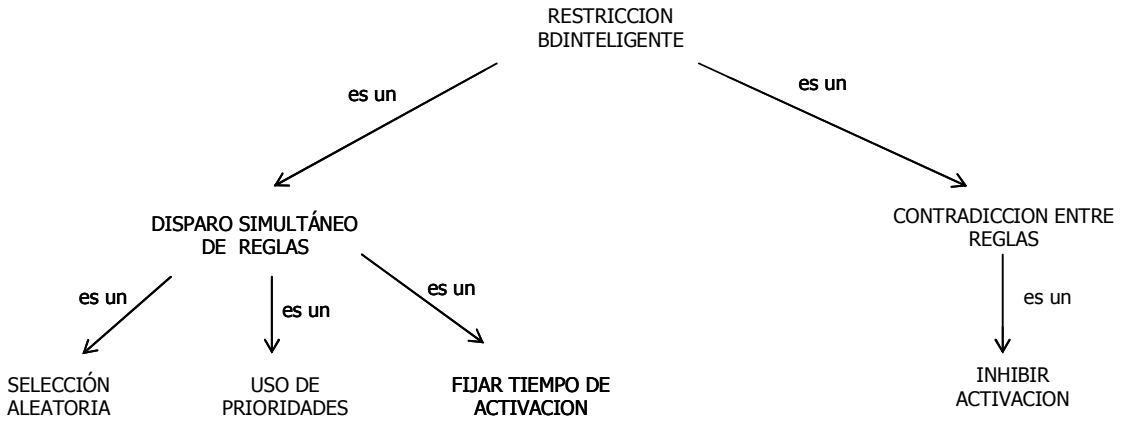

Figura 3.19. Esquema Ontológico de Restricciones en Bases de Datos Inteligentes

A continuación en la tabla 17 se muestran los axiomas de las restricciones de las Bases de Datos Inteligentes.

Tabla 17. Axiomas de las restricciones de las Bases de Datos Inteligentes

Sentencia	LPO
La restricción en las BDI ocurren por un disparo simultáneo de reglas o contradicción entre reglas	∀ x RestricciónBDInteligente(x) => es_un(x,DisparoSimultáneodeReglas) V es_un(x,Contradiccionentrereglas)
En un disparo simultaneo de reglas se realiza la selección aleatoria de la regla, el uso de prioridades, o se fija el tiempo de activación de la regla	∀ x DisparoSimultaneodeReglas(x) => es_un(x,SelecciónAleatoriadeRegla) V es_un(x, UsodePrioridades) V es_un(x, FijarTiempodeActivaciondelaRegla)

La contradicción entre reglas se soluciona con inhibir activación de la regla	∀ x ContradiccióntreReglas(x) => es_un(x,InhibirActivaciondeRegla)

Capítulo 4

4. CASO DE ESTUDIO PARA EL MODELO ONTOLÓGICO DE INTEGRACIÓN DE BASES DE DATOS FEDERADAS

4.1 Introducción

En este capítulo se describen tres ejemplos de uso de los modelos ontológicos propuestos en este trabajo. El primero para describir una base de datos inteligente para un sistema de inscripción inteligente, el siguiente es en el diseño de una base de datos multimedia para el uso de objetos de aprendizaje, y finalmente, la instanciación del proceso de integración de estas dos bases de datos en la ontología de integración de bases de datos federadas.

4.2 Caso de Estudio utilizando el Modelo Ontológico de Bases de Datos Inteligentes

El Sistema Inteligente de Inscripciones (SII) contiene una base de conocimientos conformada por una base de hechos de estudiantes y una base de reglas que contiene las condiciones para realizar la inscripción de los estudiantes. Algunos ejemplos de la información contenida en cada una de ellas son:

En la **Base de Hechos** almacena datos de alumnos, materias cursadas, materias a cursar, materias aprobadas, régimen de prelaciones, historial académico del alumno, materias a inscribir, entre otros.

En la **Base de Reglas** almacena reglas que determinen las condiciones en las que se pueden autorizar las inscripciones de los estudiantes en los diferentes cursos que se ofrecen, así como que gestionan los diferentes cursos. Algunos ejemplos son:

- Reglas para establecer el orden de inscripción de los estudiantes: Por ejemplo:
  - SI el estudiante tiene un promedio mayor o igual a 18 ENTONCES inscribirlo en la primera fecha establecida para las inscripciones del nuevo año escolar.
  - SI el estudiante tiene un promedio entre 18 y 15 ENTONCES inscribirlo en la segunda fecha de inscripciones establecida.
- Reglas que permiten inscribir a los alumnos de acuerdo a un estatus establecido (nuevo ingreso, regulares, repitientes y reincorporaciones). Por ejemplo:
  - SI el alumno es repitiente ENTONCES el número de créditos a inscribir es menor a 12

- SI el alumno es nuevo ingreso ENTONCES asignarle las materias correspondientes al primer curso.

- Reglas que permiten inscribir las materias de acuerdo a la carrera que esté cursando y las prelaciones de cada una de las materias a inscribir. Por ejemplo:
  - SI tiene Sistemas y Organizaciones aprobada e Ingeniería y Sociedad aprobada ENTONCES se puede inscribir en Sintaxis y Semántica del Lenguaje.
- Reglas de excepción al inscribir un estudiante:
  - SI es estudiante de último semestre y solicita paralelo ENTONCES otorgar paralelo.
- Reglas que establecen las capacidades de cupo en las materias. Por ejemplo
  - SI computación tiene laboratorio ENTONCES numero de estudiantes=24;
  - SI Análisis no requiere laboratorio ENTONCES número de estudiantes=45.
- Reglas que establecen la apertura de nuevas materias o secciones. Por ejemplo
  - SI existe solicitud de apertura de nueva materia o sección ENTONCES Verificar si existe profesor
  - SI existe Profesor ENTONCES verificar si existe aula disponible.
  - SI existe aula disponible ENTONCES abrir materia o sección nueva

A continuación se describe la Base de Datos Inteligente de acuerdo al marco ontológico propuesto anteriormente (ver figuras 3.17, 3.18, y 3.19 y axiomas contenidos en la tablas 15, 16 y 17).

4.2.1 Descripción Conceptual de la Bases de Datos Inteligente utilizando el marco ontológico planteado

A través del marco ontológico para Bases de Datos Inteligentes se identifican los conceptos y componentes del Sistema Inteligente de Inscripción. La tabla 18 muestra el uso del marco ontológico para realizar la descripción del sistema. En ella se describen algunos de los componentes conceptuales de la base de datos inteligentes del SII, según lo descrito en la sección 3.3.4.1 y la figura 3.17 y tabla 15. Los atributos de la base de datos inteligentes son:

ID_BDI: BDI01
Nombre_BDI: Sistema Inteligente de Inscripción
Dirección: www.universidad.inscripciones
Dominio: Académico
Esquemas:
a) Base de Hechos comprendida por: ESTUDIANTES, CURICULUM, CARRERAS, NOTAS, PROFESORES; AULAS, INSCRIPCION.
b) Base de Reglas, que contiene las condiciones en las que se pueden autorizar las inscripciones de los estudiantes en los diferentes cursos que se ofrecen, así como las que gestionan los diferentes cursos.

Modelo: Modelo Orientado a Objeto

Tabla 18**.** Ejemplo de los componentes conceptuales de la BDI bajo estudio usando el esquema ontológico de conceptos.

CONCEPTO	LPO	Comentario
El Sistema Inteligente de Inscripción (SII) tiene una base de conocimiento y un mecanismo de razonamiento	ConceptoBDInteligente(SII) => tiene(SII,BasedeConocimiento) Λ tiene(SII,MecanismodeRazonamiento)	∀ x ConceptoBDInteligente(x) => tiene(x,BasedeConocimiento) Λ tiene(x,MecanismodeRazonamiento) (Axioma Tabla 15)
La base de conocimientos del SII *tiene reglas y tiene hechos*. La **base de Reglas** está conformada por Reglas que establecen el orden de inscripción de los estudiantes, Reglas que permiten inscribir a los alumnos de acuerdo a un estatus establecido, Reglas que permiten inscribir las materias de acuerdo a la carrera que esté cursando y las prelaciones de cada una de las materias a inscribir, Reglas de excepción al inscribirse un estudiante, Reglas que establecen las capacidades de cupo en las materias, entre otras. La **base de hechos** está conformada por los esquemas: alumnos, materias cursadas, materias a cursar, materias aprobadas, régimen de prelaciones, historial académico del alumno, materias a inscribir, etc.	BasedeConocimientos(SII) => [tiene(SII,ReglasdeOrdendeInscripcion) Λ tiene(SII,ReglasdeestatusAlumno) Λ tiene(SII,ReglasMateriaCarreraPrelacion) Λ tiene(SII,ReglasExcepcionInscripcion) Λ tiene(SII,ReglasCapacidadMaterias] Λ [tiene(SII, Datosdealumnos) Λ tiene(SII,MateriasCursadas) Λ tiene(SII, MateriasaCursar) Λ (tiene(SII, MateriasAprobadas) Λ tiene (SII , RegimendePrelaciones) Λ tiene(SII, HistorialAcadémicodelAlumno) Λ tiene (SII, MateriasaInscribir)]	∀ x BasedeConocimientos(x) => tiene(x,Reglas) Λ tiene(x,Hechos) (Axioma Tabla 15)
Las reglas tienen una condición, y una acción. La regla "OrdendeInscripcion": se activa con la Solicitud de Inscripción, y tiene la CONDICION verificar la selección de Estudiante por Promedio para realizar la ACCION Inscribir	Regla(OrdenInscripcion) => tiene(OrdenInscripcion, Solicitudde InscripcionANDSeleccionaEstudianteporPromedio) Λ tiene(OrdenInscripcion, EstablecefechasdeInscripción)	∀ x Regla(x) => tiene(x,Condición) Λ tiene(x,Acción) Λ tiene(x,EnlaceAsociación) (Axioma Tabla 15)
El mecanismo de razonamiento para el SII es deductivo. De la condición "SolicitudInscripcion" se activan, entre otras, las reglas para establecer el "orden de Inscripción por promedio del estudiante" y de las "Materias de la Carrera", para que con el hecho	MecanismoRazonamiento(SII) => es_un(SII,Deductivo) Por ejemplo: ∀ x SolicitudInscripcion(x) => tiene (x, ReglasdeOrdenInscripcion) Λ tiene (x, ReglasMateriaCarreraPrelacion)	∀ x MecanismoRazonamiento(x) => es_un(x,Deductivo) (Axioma Tabla 15)

"materias a inscribir" determinar qué materias puede inscribir el estudiante	Λ…]	

4.2.2 Operaciones del Modelo Ontológico para el caso de uso

A continuación se describen algunas de las operaciones que se pueden realizar con el Sistema Inteligente de Inscripción (SII), para lo cual se utiliza el marco ontológico descrito en la sección 3.3.4.

4.2.2.1 Inscripción de un Estudiante

Si el evento que activa la base de conocimientos es la inscripción de un estudiante se verifica si el estudiante existe, se revisan las materias que desea inscribir para ver si existen, y sus materias aprobadas, entre otras cosas. Luego, el mecanismo de razonamiento comienza a activar las reglas que permiten realizar la inscripción, tales como las de prelaciones de las materias a inscribir, fecha en la cual puede realizar la inscripción, etc. El modo de activación de las reglas es inmediato. Las sentencias que se formularían en la ontología para realizar esa operación se muestran en la siguiente tabla.

Tabla 19. Operaciones en el SII para una inscripción de un alumno

OPERACIONES	LPO	Comentario
Existe la condición de Solicitud de Inscripción que activa los Hechos: Estudiante, PromedioEstudiante, HistoricoEstudiante y Materias; y activa las reglas: SeleccíondeMaterias, Cupo, y OrdenInscripcion entre otras	Condicion (SolicitudInscripcion) => [es_un(SolicitudInscripcion, Estudiante) Λ es_un (SolicitudInscripcion, PromedioEstudiante) Λ es_un (SolicitudInscripcion, HistoricoEstudiante] Λ es_un (SolicitudInscripcion,Materias)] Λ [es_un (SolicitudInscripcion,ReglasSeleccíondeMaterias) Λ es_un (SolicitudInscripcion, ReglasCupo) Λ es_un (SolicitudInscripcion, ReglasOrdenInscripcion) Λ …]	∀ x Condicion(x) => es_un(x,CombinaciondeHechos) Λ es_un (x, ActivaciondeRegla) (Axioma Tabla 15)
La máquina de Razonamiento del SII interpreta, ejecuta y desactiva reglas	MaquinaRazonamiento(SII) => tiene(SII,InterpretadorReglas) Λ tiene(SII, EjecuciónReglas) Λ tiene(SII, DesactivadorReglas)	∀ x MaquinadeRazonamiento(x) => tiene(x,InterpretadorReglas) Λ tiene(x, EjecuciónReglas) Λ tiene(x, DesactivadorReglas) (Axioma Tabla 16)
El interpretador de reglas del SII realiza razonamiento deductivo. Por ejemplo: SI existe Solicitud de Inscripción ENTONCES activar Reglas de	Interpretadorde ReglasSII(SII) => es_un(SII,RazonamientoDeductivo)	∀ x InterpretadordeReglas(x) => es_un(x,RazonamientoDeductivo) V es_un(x,RazonamientoInductivo) V

Orden de Inscripción, Estatus del Alumno, Prelación de materias, capacidad de aula y reglas de excepción		es_un(x,RazonamientoAbductivo) (Axioma Tabla 16)
Ejecución de Regla que inicia proceso de razonamiento en el SII es un encadenamiento hacia delante. Por ejemplo: SI SolicitudInscripcion ENTONCES SeleccionaEstudianteporPromedio Y … Y Selección deMaterias	SeleccíondeCondicion (SolicitudInscripcion) => es_un(SolicitudInscripcion,EncadenamientoHaciaDelante)	∀ x SeleccíondeCondicion(x) => es_un(x,EncadenamientoHaciaDelante) V es_un(x,EncadenamientoHaciaAtras) (Axioma Tabla 16)
Ejecuta las reglas de Orden de inscripción como una transacción inmediata Ejemplo de ReglasdeOrdenInscripcion SI SeleccionaEstudianteporPromedio ENTONCES EstablecefechasdeInscripción	EjecuciondeReglas(ReglasOrdenInscripcion) => tiene(ReglasdeOrdenInscripcion, SolicitudInscripcion) Λ tiene(ReglasdeOrdenInscripcion,TransaccionInmediata) ∀ x OrdendeInscripcion(x) => tiene(x, CONDICION(x,SeleccionaEstudianteporPromedio)) Λ tiene(x,ACCION(EstablecefechasdeInscripción))	∀ x EjecuciondeReglas(x) => tiene(x,SeleccíondeCondición) Λ tiene(x,MododeActivación) ∀ x MododeActivacion(x) => es_un (x, TransaccionInmediata) V es_un (x,Diferido) V es_un (x,Desacoplada) ∀ x Regla(x) => tiene(x,Condición) Λ tiene(x,Acción) Λ tiene(x,EnlaceAsociación) (Axiomas Tablas 15 y 16)
Ejemplo de ReglasSelecciónde Materias SI Selecciónde Materias ENTONCES InscribirEstudiante	EjecuciondeReglas(SeleccíondeMaterias) => tiene(SeleccíondeMaterias, SolicitudInscripcion) Λ tiene(SeleccíondeMaterias,TransaccionInmediata) Regla (SeleccíondeMaterias) => tiene(SeleccíondeMaterias, Materias a Inscribir) Λ tiene(SeleccíondeMaterias, InscribirEstudiante)	∀ x EjecuciondeReglas(x) => tiene(x,SeleccíondeCondición) Λ tiene(x,MododeActivación) ∀ x Regla(x) => tiene(x,Condición) Λ tiene(x,Acción) Λ tiene(x,EnlaceAsociación) (Axiomas Tabla 15, 16)
El modo de activación de las Reglas del sistema SII es inmediato	∀ x ModoActivacionReglasSII (x) => es_un(x,TransaccionInmediatadelaReglaSII)	∀ x MododeActivacion(x) => es_un (x, TransaccionInmediata) V es_un (x,Diferido) V es_un (x,Desacoplada) (Axioma Tabla 16)

Otras Reglas que se deben activar para inscribir al estudiante son aquellas que verifican los cursos que se pueden tomar, las capacidades de los cursos, etc.

4.2.2.2 Apertura de una materia

En esta operación se apertura una materia. Para abrir una nueva materia se debe verificar que exista dicha materia en el currículo de la carrera, utilizando los datos de la

materia a aperturar y currículo, junto con las reglas de apertura de materias. Las reglas a utilizar en este proceso de apertura de una materia son las establecidas en las tablas 3.16 y 3.17 para las operaciones y restricciones del modelo ontológico de bases de datos inteligentes. Las sentencias que se formularían para realizar esta operación se muestran en la siguiente tabla.

Tabla 20. Apertura de una materia.[27]

OPERACION	LPO	Comentario
Existe la condición: Apertura de Materia.	Condicion (AperturaMateria) => [es_un (AperturaMateria, Materia) Λ es_un (AperturaMateria, CurrículoCarrera)] Λ es_un (AperturaMateria,ReglasProfesoresDisponible) Λ es_un (AperturaMateria,ReglasEspacioFisico) Λ …)	∀ x Condicion(x) => es_un(x,CombinaciondeHechos) Λ es_un (x, ActivaciondeRegla) (Axioma Tabla 15)
Ejecución de Regla que inicia proceso de razonamiento en el SII SI hay solicitud de Apertura de Materia ENTONCES Revisar disponibilidad de Espacio Físico Y RevisarPertinenciadeMateria… Y Disponibilidad de Profesores	SelecciondeCondicion (AperturaMateria) => es_un(AperturaMateria,EncadenamientoHaciaDelante) ∀ x AperturaMateria(x) => es_un (x,ReglasDisponibilidaddeProfesores) Λ (x,RegladeVerificaciondePertinenciadeMateria)… Λ es_un (x,ReglasDisponibilidadEspacioFisico)	∀ x SelecciondeCondicion(x) => es_un(x,EncadenamientoHaciaDelante) V es_un(x,EncadenamientoHaciaAtras) (Axioma Tabla 16)
Disparo simultaneo de reglas Se debe establecer la prioridad entre ambas reglas. En este caso se establece que primero se verifica la disponibilidad de profesores, y luego la de capacidad física	DisparoSimultaneodeReglas(ReglasDisponibilidaddeProfesores, ReglasDisponibilidadEspacioFisico) => es_un((ReglasDisponibilidaddeProfesores,ReglasDisponibilidadEspacioFisico), UsodePrioridades)	∀ x,y DisparoSimultaneodeReglas(x,y) => es_un((x,y),SelecciónAleatoriadeRegla) V es_un((x,y), UsodePrioridades) V es_un((x,y), FijarTiempodeActivaciondelaRegla) (Axioma tabla 17)
SI ProfesorDisponible ENTONCES AsignarCurso	EjecuciondeReglas(ProfesorDisponible) => tiene(ProfesorDisponible, AperturaMateria) Λ tiene(ProfesorDisponible,TransaccionInmediata)	∀ x EjecuciondeReglas(x) => tiene(x,SelecciondeCondición) Λ tiene(x,MododeActivación) ∀ x MododeActivacion(x) => es_un (x, TransaccionInmediata) V es_un (x,Diferido) V es_un (x,Desacoplada)

[27] Las sentencias que activan la base de conocimiento y la máquina de razonamiento son idénticas a la tabla anterior

		(Axiomas Tablas 15 y 16)
SI EspacioFisicoDisponible ENTONCES AsignarCurso	EjecuciondeReglas(EspacioFisicoDisponible) => tiene(EspacioFisicoDisponible, AperturaMateria) Λ tiene(EspacioFisicoDisponible, TransaccionInmediata) ∀ x Regla (EspacioFisico) => tiene(EspacioFisico,DisponibilidadEspacioFisico)) Λ tiene(EspacioFisico,AsignarCurso))	∀ x EjecuciondeReglas(x) => tiene(x,SeleccciondeCondición) Λ tiene(x,MododeActivación) ∀ x MododeActivacion(x) => es_un (x, TransaccionInmediata) V es_un (x,Diferido) V es_un (x,Desacoplada) ∀ x Regla(x) => tiene(x,Condición) Λ tiene(x,Acción) Λ tiene(x,EnlaceAsociación) (Axiomas Tablas 15 y 16)

4.3 Caso de Estudio utilizando el Modelo Ontológico de Bases de Datos Multimedia

A continuación se muestra un ejemplo de uso del modelo ontológico, utilizando una Base de Datos Multimedia llamada "Sistema Multimedia de Control de Curso" en un repositorio de objetos de Aprendizaje similar al de la Universidad de Calgary, en Canadá, llamado CAREO (careo.ucalgary.ca) (Campus Alberta Repository Objects.

4.3.1 Generalidades de los Objetos de Aprendizaje

Un objeto de aprendizaje es un elemento con contenido educacional, para desarrollar cursos en línea. Puede ser un texto, un gráfico, una animación, un audio clip, un video, un quiz, etc. Una colección de objetos de aprendizaje conforman un módulo, y una colección de módulos conforman un curso.

Los objetos de aprendizaje están conformados en dos partes: 1) la parte que se ve o se oye, que puede ser un texto, un gráfico, una animación, un audioclip, etc., 2) los metadatos, que son los descriptores utilizados para indexar objetos (por ejemplo: tipo de archivo, copyright, etc.). Así, estos son etiquetas que permiten buscar, recuperar y reutilizar los objetos.

Los metadatos para los objetos de aprendizaje están basados en estándares como: IEEE LOM (Learning Object Metadata); IMS Learning Resource Meta-Data Information Model; SCORM (Shareable Courseware Object Referente Model); Dublín Core Metadata Initiative, etc.

Una vez que los objetos se han creado y etiquetado, se almacenan en repositorios de objetos de aprendizaje que se encuentran en bases de datos multimedia. Este sistema posee categorías y usuarios de objetos de aprendizaje, que son manejados por el administrador de la base de datos.

4.3.1.1 Descripción de las características generales de los Objetos de Aprendizaje.

Comencemos por definir los conceptos de los metadatos de los objetos de aprendizaje, junto con sus atributos.

RESUMEN es un metametadato[28] Texto que contiene los siguientes atributos:

General: Información general que describe el objeto de aprendizaje como un todo (Titulo, Descripción, Idioma, Alcance, Palabras Clave)

Educacional: Condiciones de uso educativo del recurso (Idioma, Tipo de Recursos de Aprendizaje (Simulación, Animación, Tutorial, Practicas, Quiz/Test; Lecturas/Presentación, Referencias), Tiempo establecido de Aprendizaje, Nivel de Dificultad, Contexto, Descripción)

Clasificación: Descripción Temática del objeto en algún sistema de clasificación (Propósito, Descripción, Palabras Clave)

Ciclo de Vida: Características relacionadas con la historia y el estado presente del objeto de aprendizaje, y de aquellos que han afectado a éste objeto durante su evolución (Versión, Status, Contribuyente)

Características Técnicas: Agrupa los requerimientos y características técnicas del objeto de aprendizaje (Ubicación, Formato, Tamaño, Requerimientos (tipo, nombre, fuente, versión máxima, versión mínima), requerimientos de otras plataformas, detalles de instalación)

Relación: Define la relación del objeto descrito con otros objetos de aprendizaje (Recursos (Nro. de Catálogo, Descripción, Identificador), Clase (Fuente, Valor))

Derechos: Condiciones de uso para la explotación del objeto (Descripción, Costo (si o no), Copyright (si o no))

Metametadata: Agrupa información sobre los metadatos, no sobre el objeto de aprendizaje que está describiendo.

FUENTE es un archivo con las fuentes y enlaces a sitios Web que contiene información, recursos, actividades, etc., relacionados con el objeto de aprendizaje.

HISTORIA es historia y estado del objeto de aprendizaje. Contiene las versiones del objeto de aprendizaje y su estado.

DISCUSION es un WEBLOG para escribir allí sobre el tema que trata el objeto de aprendizaje.

[28] metadatos marcados. Información descriptiva que permite encontrar el metadato por medio de búsquedas.

4.3.2 Descripción del caso de estudio de acuerdo al modelo ontológico para bases de datos multimedia

El ejemplo a considerar se trata del ingreso y consulta de un CURSO, llamado "**Gestión de Conocimiento**", en la Base de Datos Multimedia "Sistema Multimedia de Control de Curso" (SMCC). El CURSO contiene una colección de módulos, descritos como objetos de aprendizaje, como se muestra en la siguiente figura.

Figura 4.1. Módulos que conforman un CURSO

Donde

Objetivos Generales: describe el curso, está elaborado en XML

Clases: objetos de aprendizaje que contienen los objetivos del curso desarrollados utilizando diferentes tipos de datos multimedia (video, texto, audio y presentaciones)

Resumen: contiene un resumen de cada una de las clases elaborado en XML

Actividades: objetos de aprendizaje que contienen actividades a realizar por el estudiante del curso desarrollados utilizando diferentes tipos de datos multimedia (video, texto, audio y presentaciones)

Evaluación: en el que se tienen varios tipos de archivos (presentaciones y texto)

Una de las operaciones para el objeto de aprendizaje CURSO es una consulta, donde se puede mostrar un elemento de CLASES (se consulta la clase1), un examen, o el listado de los objetivos de la materia. Otra posible operación es una actualización de la materia, por ejemplo cuando se quiere agregar un video (Video01) al objeto de aprendizaje CLASES.

4.3.2.1 *Caracterización de los conceptos del caso de estudio en el modelo ontológico*.

En el caso particular del ejemplo: CURSO de "Gestión del Conocimiento", los atributos de RESUMEN son los siguientes:

General:

Titulo: Gestión de Conocimiento

Descripción: Permite al alumno adquirir los conocimientos sobre técnicas y herramientas para implementar Sistemas Basados en Conocimiento,

Idioma: Español

Alcance: Este curso está dirigido a estudiantes del 7mo semestre que hayan cursado Inteligencia Artificial y Sistemas de Información

Palabras Clave: Sistemas Basados en Conocimiento, Gestión de Conocimientos, Ontologías, Web Semántica

Educacional

Idioma: Español

Tipo: Tutorial

Tiempo establecido de Aprendizaje: 1 Semestre (16 semanas)

Nivel de Dificultad: Media

Tipo de Recursos de Aprendizaje: Simulación, Animación, Tutorial, Prácticas, Quiz/Test; Lecturas/Presentación, Referencias.

Contexto: Se ubica en el pensum como electiva a partir del 7mo semestre de Ingeniería de Sistemas

Descripción: Recurso que contiene actividades, herramientas, etc., relacionados con sistemas basados en conocimientos

Clasificación

Propósito: Estos objetos se utilizan como herramienta de soporte para la materia Gestión de Conocimiento que se imparte en CEMISID

Descripción: Permite al alumno adquirir los conocimientos sobre técnicas y herramientas para implementar Sistemas Basados en Conocimiento,

Palabras Clave: IMS

Ciclo de vida

Versión: 1.0

Status:

Contribuyente: Profesor

Fecha: 21 de Abril 2007

Carlos García

Contribuyente: Profesor

Fecha: 21 de Abril 2007

Alba Rojas

Características técnicas

Ubicación: http://www.cemisid.ula.ve/GC.htm

Formato: XML, HTML,

Tamaño: bytes

Requerimientos: XML, GC.htm (tiene un enlace para cargar el archivo en la base de datos)

Otros Requerimientos:

Detalles de Instalación:

Relación

Descripción: Video que muestra procesos en una organización

Recursos: GC01, Video, VGC01

Clase: http://www.cemisid.ula.ve/VGC01.mpeg; VGC01.mpeg

Derechos

Costo: no
Copyright: si
Descripción: Este video requiere establecer los derechos para visualizarlo

FUENTE: http://www.cemisid.ula.ve
HISTORICO:

A continuación se describen los conceptos de la base de datos multimedia bajo estudio de acuerdo al marco ontológico propuesto en este trabajo.

Tabla 21. Axiomas para los componentes conceptuales de la BDMM bajo estudio usando el esquema ontológico de conceptos

CONCEPTO	LPO	COMENTARIO
El objeto de aprendizaje curso Gestión de Conocimiento tiene metadatos texto, metadatos audio, metadatos imagen	ConceptoBDMultimedia (ObjetoAprendizajeGC) => tiene(ObjetoAprendizajeGC,MetadatoTexto) V tiene (ObjetoAprendizajeGC,MetadatoAudio) V tiene (ObjetoAprendizajeGC,MetadatoImagen) V tiene (ObjetoAprendizajeGC,MetadatoVideo)	∀ x ConceptoBDMultimedia(x) => tiene(x,MetadatoTexto) V tiene(x,MetadatoImagen) V tiene(x,MetadatoAudio) V tiene(x,MetadatoVideo) (Axioma tabla 11)
El objeto de aprendizaje curso Gestión de Conocimiento es un objeto multimedia	ConceptoBDMultimedia (ObjetoAprendizajeGC) => es_un (ObjetoAprendizajeGC,ObjetoMM)	∀ x ConceptoBDMultimedia(x) => es_un(x,ObjetoMM) (Axioma tabla 11)
El metadato texto del curso GC tiene Descriptores de documento, histórico de documento, localización de documento y fuentes	MetadatoTexto (CursoGC) => tiene(CursoGC,Resumen) Λ tiene(CursoGC,Historico) Λ tiene(CursoGC,Localizacion) Λ tiene(CursoGC,Fuente)	∀ x MetadatoTexto(x) => tiene(x,DescriptorDocumento) Λ tiene(x,HistoricoDocumento) Λ tiene(x,LocalizacionDocumento) Λ tiene(x,Representación_Datos_Texto) (Axioma tabla 11)
Un objeto de aprendizaje del curso Gestión de Conocimiento puede contener metadatos imagen	MetadatoImagen(CursoGC) => es_un (CursoGC,ImagenSatelital) V es_un (CursoGC, ImagenDiseñoArquitectonico) V es_un (CursoGC, ImagenFacial)	∀ x MetadatoImagen(x) => es_un(x,ImagenSatelital) V es_un(x,ImagendeDiseñoArquitectonico) V es_un(x,ImagenFacial) (Axioma Tabla 11)
La materia GC tiene Metadatos audio	∀ x MetadatoAudio(CursoGC) => tiene (CursoGC,ReconocimientoAudio) Λ tiene (CursoGC, IdentificacionSignificado) Λ tiene (CursoGC, Identificaciondel Hablante)	∀ x MetadatoAudio(x) => tiene (x,Reconocimiento_del_Habla) Λ tiene (x,Identificacion_del_Significado) Λ tiene (x,Identificacion_del_Hablante) (Axioma Tabla 11)
El curso GC tiene Metadato	∀ x MetadatoVideo(CursoGC) =>	∀ x MetadatoVideo(x) =>

Video	tiene (CursoGC,Contenido) Λ tiene (CursoGC, HerramientaSoporteUsuario) Λ tiene (CursoGC, HerramientaSoporteAplicacion)	tiene (x,Contenido_de_Video) Λ tiene (x,Herramientas_Sop_Usuario) Λ tiene (x,Herramientas_Sop_Aplicacion) (Axioma Tabla 11)
El objeto Video01 es un objeto multimedia	Objeto (Video01) => es_un (Video01, ObjetoMM)	∀ x Objeto (x) => es_un (x, ObjetoMM) (Axioma Tabla 7)
Video01 es un dato video como mpeg, mov, avi	ObjetoMM (Video01) => es_un (Video01, Dato_Video)	∀ x ObjetoMM(x) => es_un(x, Dato_Texto) V es_un(x, Dato_Audio) V es_un (x, Dato_Imagen) V es_un(x, Dato_Grafico) V es_un (x, Dato_Video) V es_un (x, Dato_MediaGenerado) (Axioma Tabla 7)

4.3.2.2 Operación de consulta

La operación de consulta del curso de "Gestión de Conocimiento" se lleva a cabo cuando cualquier estudiante quiera realizar el curso. En esta operación se ejecutan operaciones de consulta a una base de datos multimedia, para lo cual se va caracterizando el objeto de aprendizaje a acceder (duración, tipo, etc.). A continuación, en la tabla 22, se muestran los axiomas requeridos según el modelo ontológico para realizar esa operación.

Tabla 22. Ejemplo de axiomas para operaciones de consulta

OPERACIONES	**LPO**	**Comentarios**
Consultar la Clase1 del CURSO Gestión de Conocimientos	OperacionBDMM(ConsultaClase1) => es_un (ConsultaClase1, ConsultaBDMM) V es_un (ConsultaClase1,Operación_Objetos_Media)	∀ x OperacionBDMM(x) => es_un(x,ConsultaBDMM) V es_un(x,ActualizacionBDMM) V es_un(x,BusquedaEdicion) V es_un (x,Operación_Objetos_Media) (Axioma Tabla 12)
La CLASE1 tiene un tiempo de duración establecido	RepresentacionDatosMM (CLASE1) => tiene(CLASE1,Tiempo)	∀ x RepresentacionDatosMM(x) => tiene(x,Tiempo) (Axioma Tabla 12)
El tiempo de duración de la CLASE1 es de 45 minutos	Tiempo(CLASE1) => es_un (CLASE1, 45Minutos)	∀ x Tiempo(x) => es_un(x,Instante) V es_un (x,PeriodoTemporal) V es_un(x,IntervaloTemporal) (Axioma Tabla 12)

La CLASE1 muestra un video y una presentación que contiene texto e imágenes	Operación_Objetos_media (ConsultarCLASE1GC) => es_un(ConsultarCLASE1GC,ObjetoMedia_Texto) V es_un(ConsultarCLASE1GC,ObjetoMedia_Audio) V es_un(ConsultarCLASE1GC ,ObjetoMedia_Video) V es_un (ConsultarCLASE1GC,ObjetoMedia_Imagen)	∀ x Operación_Objetos_media (x)=> es_un(x,ObjetoMedia_Texto) V es_un(x,ObjetoMedia_Audio) V es_un(x,ObjetoMedia_Video) V es_un (x,ObjetoMedia_Imagen) (Axioma Tabla 12)
La CLASE1 no puede tener un valor nulo	RestriccionIntegridadObjeto (CLASE1) => es_un(CLASE1, ValornoNulo)	∀ x RestriccionIntegridadObjeto(x) => es_un(x,ValornoNulo) (Axioma de Tabla 13 y 9)
La CLASE1 puede ser un archivo de Video	∀ x RestricciondeDominio (CLASE1) => es_un(CLASE1, ValorPermitido)	∀ x RestricciondeDominio(x) => es_un(x,ValorPermitido) (Axioma de Tabla 9)

4.3.2.3 Operación de actualización del sistema con el Video01

A continuación se muestran los axiomas para la operación insertar el Video01.mpeg en la Clase1 de la materia Gestión de Conocimientos. En este caso, se deben definir las operaciones como el tipo de datos a insertar (entre otras cosas), el método que se invocará para realizar la operación de inserción (considerando que es un objeto), etc. Se supone que el software que soporta el video es QuickTime.

Tabla 23. Ejemplo de conceptos, operaciones y restricciones de la BDMM bajo estudio usando el modelo ontológico para BDMM

Operaciones	LPO	Acción
El Video01.mpeg es un metadato video y se inserta a través de la Aplicación de Soporte Usuario QuickTime	∀ x MetadatoVideo(Video01) => tiene (Video01,Contenido_de_Video) Λ tiene (Video01,Herramientas_Sop_Usurio) Λ tiene (Video01,Herramientas_Sop_Aplicacion)	∀ x MetadatoVideo(x) => tiene (x,Contenido_de_Video) Λ tiene (x,Herramientas_Sop_Usurio) Λ tiene (x,Herramientas_Sop_Aplicacion) (Axioma Tabla 11)
Actualizar la CLASE1 con el Video01.jpg	∀ x OperacionBDMM (ActualizarCLASE1) => es_un(ActualizarCLASE1, ActualizacionBDMM)	∀ x OperacionBDMM(x) => es_un(x,ActualizacionBDMM) V es_un (x,Operación_Objetos_Media) (Axioma Tabla 12)
Insertar el Video01.jpeg es una operación de objetos media	∀ x Operación_Objetos_media (InsertarVideo01.mpeg) => es_un(InsertarVideo01,ObjetoMedia_Video)	∀ x Operación_Objetos_media (x)=> es_un(ObjetoMedia_Texto, x) V es_un(ObjetoMedia_Audio, x) V es_un(ObjetoMedia_Video, x) V es_un (ObjetoMedia_Imagen) (Axioma Tabla 12)

Ejecutar método Ingresar Video01	Solicitud (IngresarVideo01) => es_un(IngresarVideo01,InvocaciónOperación)	∀ x Solicitud(x) => es_un(x,InvocaciónOperación) (Axioma Tabla 8 Operación BDOO)
Actualizar Video01.mpeg en el Objeto de Aprendizaje Clase1 de la materia Gestión de conocimientos	InterfazdeOperación(ActualizarVideo01) => tiene(ActualizarVideo01, InsertarVideo) ∧ tiene(ActualizarVideo01, Video01.mpeg) ∧ tiene(ActualizarVideo01, "CargadoVideo01.mpeg",)	∀ x InterfazdeOperación(x) => tiene(x,NombredeMétodo) ∧ tiene(x,NombredeArgumento) ∧ tiene(x,TipodeArgumento) ∧ tiene(x,OrdendeApariciondelosArgumentos) ∧ tiene(x,ResultadosdeOperacion) (Axioma Tabla 8 Operación BDOO)
El archivo debe ser de tipo de video	RestriccionIntegridadReferencial(Video01) => es_un (Video01,TipoVideo)	∀ x RestriccionIntegridadReferencial(x) => es_un (x,RestricciondeDominio) (Axioma Tabla 9)
El archivo Video01 es de tipo mpeg	∀ x RestricciondeDominio(Video01) => es_un(Tipo_mpeg,ValorPermitido)	∀ x RestricciondeDominio(x) => es_un(x,ValorPermitido) (Axioma Tabla 9)

4.4 Caso de Estudio utilizando el Modelo Ontológico de Integración de Bases de Datos Federadas

A continuación se muestra un caso de estudio, para la integración de bases de datos. Utilizaremos dos bases de datos que se van a integrar para manejar toda la información requerida sobre los alumnos que cursan la materia Gestión de Conocimiento. Así, el contenido de los exámenes y el resultado de los mismos se encuentra en una Base de Datos Multimedia llamada: "Sistema Multimedia de Control de Curso" (SMCC). Los datos de los alumnos y su registro de notas por materia se encuentran una base de datos inteligente llamada "Sistema Inteligente de Inscripción" (SII). Por lo tanto debemos integrar una base de datos multimedia y una base de datos inteligente.

El curso Gestión de Conocimiento ya se describió en la sección 4.3.2 el cual contiene una serie de módulos.

El SII contiene una base de conocimientos conformada por una base de hechos de estudiantes, que contiene datos de alumnos, materias cursadas, materias inscritas, materias aprobadas, régimen de prelaciones, historial académico del alumno, entre otros, y una base de reglas que determinan las condiciones en las que se pueden autorizar las inscripciones de los estudiantes en los diferentes cursos que se ofrecen (se describió en la sección 4.2). Los atributos para la federación son:

- BD_FEDERADA (Inscripcion_Clases, BDF01, 01/01/2007)
- BD_COMPONENTE ("Sistema Multimedia de Control de Curso", "Sistema Inteligente de Inscripción")
- BD_MM (Sistema Multimedia de Control de Curso, ESQ_ObjetivosCurso, ESQ_CLASES, ESQ_EXAMEN, ESQ_ESTUDIANTE)
- BD_INTELIGENTE (Sistema Inteligente de Inscripción, BDI01, ESQ_Alumnos, ESQ_MATERIAS, ESQ_NotasALUMNOSMATERIA, Base_Reglas)

4.4.1 Proceso de Integración de la Base de Datos Federada Inscripcion_Clases.

El primer paso en el proceso de integración es la definición de la base de datos federada y sus bases de datos componentes, según lo establecido en la tabla de conceptos. Así, el administrador de la Base de Datos Federada define el nombre de la Base de Datos Federada, la cual se llamará: Inscripción_Clases y las bases de datos que componen la federación: "Sistema Multimedia de Control de Curso", "Sistema Inteligente de Inscripción". A continuación se muestra una tabla con la instanciación de los axiomas descritos en la tabla 1 que se encuentra en el capitulo 3, de la ontología de integración.

Tabla 24. Instanciación de los axiomas de conceptos en LPO para la composición de la federación.

Sentencia	LPO	Comentario
La base de datos federada Inscripción_Clase tiene bases de datos componentes, tiene operaciones de integración y tiene restricciones de integración	BDFederada(Inscripción_Clase) => tiene (Inscripción_Clase,BDComponentes) Λ tiene (Inscripción_Clase,OperacionesIntegracion) Λ tiene (Inscripción_Clase,RestriccionesIntegracion)	∀ x BDFederada(x) => tiene (x,BDComponente) Λ tiene (x,OperacionIntegracion) Λ tiene (x,RestriccionesIntegracion) (Axioma de Tabla 1)
Las bases de datos componentes de Inscripción_Clase son: "Sistema Multimedia de Control de Curso" que una BDMM, y el "Sistema Inteligente de Inscripción" que es una BDI	BDComponente(Inscripción_Clase) => es_un (Inscripción_Clase , SMMControldeCurso) V es_un (Inscripción_Clase , SInscripciónInteligente)	∀ x BDComponente(x) => es_un(x,BDRelacional) V es_un (x,BDOO) V es_un (x, BDMultimedia) V es_un (x, BDInteligente) V es_un (x, BDFederada) (Axioma Tabla 1)
El Sistema Multimedia de Control de Curso tiene conceptos,	BDMultimedia (SMMControldeCurso) => tiene(SMMControldeCurso,ConceptosMM) Λ tiene(SMMControldeCurso, OperacionesMM) Λ tiene(SMMControldeCurso, Restricciones	∀ x BDMultimedia(x) => tiene(x,ConceptosMM) Λ tiene(x, OperacionesMM) Λ tiene(x, RestriccionesMM)

operaciones y restricciones de BDMM	MM)	(Axioma Tabla 1)
El Sistema de Inscripción Inteligente tiene conceptos, operaciones y restricciones de BDI	BDInteligentes (SInscripciónInteligente) => tiene(SInscripciónInteligente,ConceptosInt) Λ tiene(SInscripciónInteligente, OperacionesInt) Λ tiene(SInscripciónInteligente, RestriccionesInt)	∀ x BDInteligentes(x) => tiene(x,ConceptosInt) Λ tiene(x, OperacionesInt) Λ tiene(x, RestriccionesInt) (Axioma Tabla 1)

4.4.2 Ejemplo de operaciones una vez creada la federación.

En este caso, el orden de integración es binario, es decir el proceso de integración consiste en ir integrando dos esquemas a la vez. Para la inscripción de un alumno en una materia se debe registrar en ambas bases de datos. Se establece que primero debe estar registrado en la base de datos inteligente para luego registrarse en la base de datos multimedia. El proceso se lleva a cabo de la siguiente manera:

1. El alumno se inscribe en la base de datos inteligente en la materia "Gestión del Conocimiento". Este proceso de inscripción se describe en 4.2.2.1, tabla 19 [Muñoz A, Aguilar J, 2007].
2. Luego, el alumno es registrado en la materia Gestión del Conocimiento en la base de datos multimedia, esto se realiza una vez que las bases de datos estén integradas. Así, aquí es donde se activa el proceso de integración de las dos bases de datos heterogéneas.

Los esquemas a integrar de la base de datos inteligentes son:
ESQ_Estudiante (**ID_Estudiante**, Nombre, Carrera)
ESQ_MATERIAS (**ID_Materia**, Descripcion, UC, Prelacion)
ESQ_NotasESTUDIANTEMATERIA (***ID_Estudiante, ID_Materia***, Nota1, Nota2, Nota3, Nota4, Definitiva)

Las reglas de la Base_Reglas a usar son las siguientes: Orden de Inscripción, Estatus del Estudiante, Materias por Carrera, Capacidad de la Materia.

Los esquemas a integrar de la base de datos multimedia son:
ESQ_Alumno (**ID_Alumno**, Nombre, Materia)
ESQ_ObjetivosCurso (**ID_Curso**, NroObjetivo, Descripcion, ContenidoObjetivos, Resumen, Fuente, Historia, Discusion)
ESQ_CLASES (**ID_Clase**, ***ID_Curso***, ContenidoClases, Ejercicios)
ESQ_EXAMEN (***ID_Alumno, ID_Clase, ID_Curso***, NumeroExamen, TipoExamen, Nota)

Las operaciones iniciales para integrar las dos bases de datos vienen dadas por los siguientes axiomas.

Tabla 25. Operaciones y Restricciones de integración de la base de datos federada de acuerdo al modelo ontológico presentado en el capitulo 3 a través de las tablas 2 y 3.

OPERACIONES	LPO	ACCIÓN
La operación de integración de la BDF Inscripción_Clase tiene preintegración, comparación de esquemas y la reunión y reestructuración de los esquemas	OperaciónIntegración(Inscripción_Clase) => tiene (Inscripción_Clase,Preintegración) Λ tiene(Inscripcion_Clase,ComparacióndeEsquemas) Λ tiene (Inscripción_Clase,Union_ReestructuraciónEsquemas) [29]	∀ x OperaciónIntegración(x) => tiene (x,Preintegración) Λ tiene(x,ComparacióndeEsquemas) Λ tiene (x,Union_ReestructuraciónEsquemas) Λ tiene (x, UniondeEsquemas) Λ tiene (x, ActualizaciondeInformación) (Axioma Tabla 2)
La base de datos federada Inscripción_Clases integra dos esquemas: ESQ_Estudiante de la Base de datos inteligente y ESQ_Alumno de la base de datos multimedia	IntegracionBinaria(Inscripción_Clases) => Integra(Inscripción_Clases,DosEsquemas)	V x IntegracionBinaria(x) => Integra(x,DosEsquemas) (Axioma Tabla 2)
La comparación de esquemas ocurre a nivel de elemento para los atributos: ID_Estudiante de ESQ_Estudiante, e ID_Alumno de ESQ_Alumno	ComparacionEsquemas (ESQ_Estudiante, ESQ_Alumno) => Sucede ((ESQ_Estudiante, ESQ_Alumno),NivelElemento)	∀ x,y ComparacionEsquemas(x,y) => Sucede ((x,y),NivelElemento) Λ Sucede((x,y),NivelEsquema) (Axioma Tabla 2)
La comparación de esquemas ocurre a nivel de elemento para los atributos: Definitiva de ESQ_ NotasESTUDIANTEMATERIA, y NotaDef de ESQ_EXAMEN	ComparacionEsquemas (ESQ_NotasESTUDIANTEMATERIA, ESQ_EXAMEN) => Sucede((ESQ_NotasESTUDIANTEMATERIA, ESQ_EXAMEN),NivelElemento)	∀ x,y ComparacionEsquemas(x,y) => Sucede ((x,y),NivelElemento) Λ Sucede((x,y),NivelEsquema) (Axioma Tabla 2)
Se presentan conflictos en nombrado de atributos para ID_Estudiante en el esquema ESQ_Estudiante e ID_Alumno en el esquema ESQ_Alumno:, pues los nombres de los atributos son diferentes y significan lo mismo Lo mismo sucede con Definitiva para el esquema ESQ_ NotasESTUDIANTEMATERIA y NotaDef en el esquema ESQ_EXAMEN	ConflictoNombreAtributo (ESQ_Estudiante, ESQ_Alumno) => es _ un ((ESQ_Estudiante, ESQ_Alumno), NombreDiferentes_AtributosEquivalentes) ConflictoNombreAtributo (ESQ_ NotasESTUDIANTEMATERIA, ESQ_EXAMEN) => es_un ((ESQ_ NotasESTUDIANTEMATERIA, ESQ_EXAMEN), NombreDiferentes_AtributosEquivalentes)	∀ x,y ConflictoNombreAtributo (x,y) => es_un ((x,y) NombreDiferentes_AtributosEquivalentes) V es_un ((x,y) NombresIguales_AtributosDiferentes) (Axioma Tabla 3)

[29] Ver tabla 2 para la descripción del axioma de Preintegración(x)

Se unen los esquemas ESQ_Estudiante, ESQ_Alumno, así como también los esquemas ESQ_ NotasESTUDIANTEMATERIA, ESQ_EXAMEN	UnionyReestructuracionEsquemas (ESQ_Estudiante, ESQ_Alumno) => tiene((ESQ_Estudiante, ESQ_Alumno),UnionEsquemas) UnionyReestructuracionEsquemas (ESQ_ NotasESTUDIANTEMATERIA, ESQ_EXAMEN) => tiene((ESQ_Estudiante, ESQ_Alumno),UnionEsquemas)	∀ x UnionyReestructuracion Esquemas(x,y) => tiene((x,y),UnionEsque mas) (Axioma Tabla 2)
Prioridad de Acceso: Acceder a la base de datos Inteligente primero pues es quien realiza la inscripción	PrioridaddeAcceso(Inscripción_Cla se) => Establece(Inscripción_Clase, PrimeroBDI)	∀ x PrioridaddeAcceso(x) => Establece(x,OrdendeAcc eso) (Axioma Tabla 2)

A continuación se presentan otros posibles casos de uso en los que se reflejan el uso de los axiomas descritos en las tablas 2 y 3 del modelo ontológico para integración de bases de datos federadas. En el primero se utilizan los axiomas descritos para las operaciones de integración, y en el segundo las restricciones de la integración.

4.4.2.1 Un estudiante consulta una clase de una materia una vez que está inscrito

Cuando el alumno desea acceder a la clase se verifica que su código de estudiante se encuentre registrado en la base de datos inteligente como estudiante de la materia gestión del conocimiento, y en la BDMM en la tabla Estudiante. Para ello se realizan las operaciones de integración de los esquemas: ESQ_Estudiante y ESQ_Alumno, como se muestra en la tabla 25. Una vez integrados los esquemas y resuelto el conflicto de nombrado, se verifica el tipo de dato de cada uno de los atributos claves de los dos esquemas a integrar (ID_Estudiante, ID_Alumno). Ahora surge el conflicto entre valores diferentes de datos, por las representaciones diferentes para un mismo dato en las diferentes bases de datos a integrar (en este caso, suponemos que los identificadores de los estudiantes en cada una de las BD son diferentes, pues ID_Estudiante es tipo numérico e ID_Alumno es tipo texto). Suponiendo lo anterior, el orden de ejecución de los axiomas para poder realizar la consulta es el siguiente:

- Verificar si el estudiante esta inscrito en la materia (en ambas bases de datos).
    - Previamente se deben resolver los conflictos existentes entre los identificadores de estudiantes que existen entre ambas bases de datos (tanto el nombre como el tipo de datos son diferentes).
    - Consultar curso especifico deseado (esta operación sobre la BDMM se describió en la tabla 22).

A continuación se muestran los axiomas para esta operación.

Tabla 26. Instancias de los axiomas de las operaciones de integración (los axiomas para consultar la BDMM se muestran en la tabla 22).

OPERACIONES	LPO	ACCIÓN
El estudiante con ID_Estudiante=12345 se encuentra registrado en el Sistema de Registro de Inscripción en el ESQ_Estudiante	BasedeConocimientos(SistemaInteligentedeInscripcion) => tiene(SistemaInteligentedeInscripcion, (12345,Juan Pérez, Informática))	∀ x BasedeConocimientos(x) => tiene(x,Reglas) Λ tiene(x,Hechos) (Axioma Tabla 15)
El alumno con ID_Alumno=12345 se encuentra registrado en el Sistema Multimedia de Control de Curso dentro del ESQ_Alumno	ObjetoMM(12345) => es_un(12345 ,Dato_Texto)	∀ x ObjetoMM(x) => es_un(x,Dato_Texto) V es_un(x, Dato_Audio) V es_un (x, Dato_Imagen) V es_un(x, Dato_Grafico) V es_un (x, Dato_Video) V es_un (x, Dato_MediaGenerado) (Axioma Tabla 11)
La comparación de esquemas ocurre a nivel de elemento para los atributos: ID_Estudiante de ESQ_Estudiante, e ID_Alumno de ESQ_Alumno	ComparacionEsquemas (ESQ_Estudiante, ESQ_Alumno) => Sucede ((ESQ_Estudiante, ESQ_Alumno),NivelElemento)	∀ x,y ComparacionEsquemas(x,y) => Sucede ((x,y),NivelElemento) Λ Sucede((x,y),NivelEsquema) (Axioma Tabla 2)
Se presentan conflictos en nombrado de atributos para ID_Estudiante en el esquema ESQ_Estudiante e ID_Alumno en el esquema ESQ_Alumno:, pues los nombres de los atributos son diferentes y significan lo mismo	ConflictoNombreAtributo (ESQ_ESTUDIANTE, ESQ_Alumno) => es_un ((ESQ_ESTUDIANTE, ESQ_Alumno), NombreDiferentes_AtributosEquivalentes)	∀ x,y ConflictoNombreAtributo(x,y) => es_un ((x,y) NombreDiferentes_AtributosEquivalentes) V es_un ((x,y) NombresIguales_AtributosDiferentes) (Axioma Tabla 3)
El tipo de datos de ID_Estudiante de la BDInteligente e ID_Alumno de la BDMM tienen conflicto en el tipo de datos pues ID_Estudiante es numérico e ID_Alumno es tipo texto	ConflictodeDiferenciadeRepresentación(ESQ_Estudiante, ESQ_Alumno) => tiene((ESQ_Estudiante,ESQ_Alumno), DiferenteNotacion)	∀ x,y ConflictodeDiferenciadeRepresentación(x,y) => tiene((x,y), DiferenteNotacion) V tiene ((x,y), DiferentesUnidades) V tienen ((x,y), DiferentesExpresiones) (Axioma Tabla 3)
Se unen los esquemas ESQ_Estudiante, ESQ_Alumno	UnionyReestructuracionEsquemas(ESQ_Estudiante, ESQ_Alumno) => tiene((ESQ_Estudiante, ESQ_Alumno),UnionEsquemas)	∀ x,y UnionyReestructuracionEsquemas(x,y) => tiene((x,y),UnionEsquemas) (Axioma Tabla 2)
Consultar la Clase1 del CURSO Gestión de Conocimientos	OperacionBDMM(Clase1) => es_un (Clase1, ConsultaBDMM) V es_un (Clase1,Operación_Objetos_Media)	∀ x OperacionBDMM(x) => es_un(x,ConsultaBDMM) V es_un(x,ActualizacionBDMM) V es_un(x,BusquedaEdicion) V es_un (x,Operación_Objetos_Media)

		(Axioma Tabla 12)

4.4.2.2 Un estudiante realiza un examen

Un estudiante realiza un examen en la BDMM dentro del objeto multimedia Examen, y luego de actualizar la definitiva debe ser actualizada en la base de datos inteligente en el ESQ_NotasESTUDIANTEMATERIA. Asi, al existir una nota en ESQ_Examen de la BDMM se debe actualizar dicha nota en la base de datos inteligente en el esquema ESQ_NotasALUMNOSMATERIA. El cálculo de la nota en el examen ocurre cuando el estudiante finaliza el examen, disparando la regla cálculo de la nota en la BDMM. El proceso es el siguiente:

1. El estudiante ingresa al SMCC y solicita realizar un examen. El sistema le presenta el examen y el estudiante lo ejecuta. Al finalizar el examen el sistema calcula y registra la nota (para ello se dispara la regla respectiva).
2. Una vez que el sistema registra la nota, debe activar las operaciones de integración para registrar la nota en la BDI, en este momento se deben resolver los siguientes problemas de integración:
   - Conflictos en nombrado de Atributos, existen nombres diferentes para atributos equivalentes: **ID_Estudiante** es un atributo perteneciente a la BDI, y contiene el mismo dato que **ID_Alumno,** que es un atributo perteneciente a un esquema de la BDMM (presentados en las tablas 25 y 26)
   - El conflicto en estructura de tabla ocurre cuando hay atributos que se omiten o cuando hay atributos que se deducen: en ESQ_NotasALUMNOSMATERIA la definitiva es un atributo que se deduce (Disparo de Regla: cálculo de definitiva).
   - Conflicto en disparo simultáneo de Reglas: En cada una de las bases de datos existe una regla, en el caso de la BDMM se dispara para realizar el cálculo de la nota de un examen, y en el caso de la BDI para calcular la definitiva de la materia. Se debe decidir cuál es la prioridad de disparo, que en este caso es la de la BDMM (ver paso 1).
3. Luego se actualizan los datos en la base de datos inteligente.

Tabla 27. Ejemplo de las restricciones de integración de las bases de datos federadas (aquí no se muestran los axiomas para actualizar ambas bases de datos componentes (ver secciones 4.2 y 4.3)

Sentencia	LPO	Comentarios
En ESQ_NotasALUMNOSMATERIA la **Definitiva** es un atributo que se deduce	ConflictoEstructuraTabla (ESQ_NotasALUMNOSMATERIA) => es_un(ESQ_NotasALUMNOSMATERIA,AtributosTDeducidos)	∀ x ConflictoEstructuraTabla (x) => es_un(x,AtributosTOmitidos) V es_un(x,AtributosTDeducidos) (Axioma Tabla 3)
Existe un conflicto de disparo simultáneo de	DisparoSimultaneoReglas(ReglaCalcu	∀ x, y

reglas, cuando se indica realizar el Calculo de nota según la Regla ”CalculoDefinitiva” en el esquema ESQ_Estudiantes de la BDInteligente y el cálculo de *definitiva* según la Regla ”CalculoDef” en el ESQ_Examen de la BDMM	loDefinitiva, ReglaCalculoDef) => es_un((ReglaCalculoDefinitiva, ReglaCalculoDef),Disparodemasdeuna Regla)	DisparoSimultaneoReglas(x,y) => es_un((x,y),Disparodemasdeuna Regla) (Axioma Tabla 3)

Capítulo 5

5. VALIDACIÓN DE LA ONTOLOGÍA PARA LA INTEGRACIÓN DE BASES DE DATOS FEDERADAS

5.1 Introducción

Este capítulo presenta la validación de la ontología para el proceso de integración de las bases de datos federadas. Inicialmente se describe la técnica de validación usada, y luego el proceso de validación de la ontología propuesta en este trabajo. Para ello se utilizó la herramienta Protégé-OWL, que es una extensión de Protégé (editor de ontologías basado en Java) que soporta el lenguaje Ontológico OWL (Ontology Web Language). OWL[30] es un lenguaje estándar de ontologías, aprobado por el World Wide Web Consortium (W3C), el cual permite definir una semántica a través de clases, propiedades y restricciones y posee mecanismos de razonamiento.

5.2 Base Teórica para validar una Ontología

Como cualquier otro recurso de software, el contenido de las ontologías (es decir, la definición de sus conceptos, taxonomía y axiomas) debe ser evaluado antes de ser usado o reutilizado en otras ontologías o aplicaciones. Gómez-Pérez (Gómez-Pérez, 1996) proporcionan una estructura para la evaluación ontológica que incluye la siguiente terminología:

Evaluación de la Ontología: es un juicio técnico del contenido de la ontología con respecto a un marco de referencia (un marco de referencia puede ser: especificación de requerimientos, preguntas de competencia, mundo real, etc.) durante su ciclo de vida. La evaluación de una ontología puede realizarse para los siguientes componentes de ella:

- Cada definición y axioma establecido explícitamente en la ontología.
- Definiciones importadas de otras ontologías usadas por ella.
- Definiciones que pueden ser inferidas desde otras definiciones y axiomas.

Verificación de la Ontología: se refiere a construir la ontología correctamente, esto es, asegurar que sus definiciones (una definición escrita en lenguaje natural es una definición informal, y en lenguaje formal es una definición formal) implementan correctamente los requerimientos de la ontología.

[30] http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial.pdf

Validación de la Ontología: se refiere a garantizar que las definiciones ontológicas modelan realmente el mundo real para el que la ontología fue creada. La meta es probar que el modelo del mundo (si existe y es conocido) es similar al mundo modelado formalmente.

En general, la meta del proceso de evaluación de una ontología es determinar qué define correctamente una ontología y qué la hace incorrecta. Por otro lado, para evaluar el contenido de una ontología dada, Gómez-Pérez [Gómez-Pérez, 1996] identificaron los siguientes criterios:

- **Consistencia**, se refiere a cuando no es posible obtener conclusiones contradictorias desde definiciones de entrada válidas. Una definición dada es consistente sí y solo sí la definición individual es consistente y no contradictoria con el conocimiento que puede ser inferido desde otras definiciones y axiomas.
- **Completitud**. Una ontología es completa sí y solo sí:
  - Todo lo que es la ontología está explícitamente establecido o puede ser inferido.
  - Cada definición es completa sí: (a) el conocimiento que especifica la definición lo define explícitamente el mundo o; (b) el conocimiento requerido no explícito puede ser inferido desde otras definiciones y axiomas.
- **Concisión**. Una ontología es concisa sí: (a) no almacena definiciones innecesarias o no utilizadas; y (b) no existen redundancias explícitas entre definiciones de términos.

5.2.1 Marco teórico de OWL para la validación

Como se dijo antes, para validar la ontología del proceso de integración de bases de datos federadas de acuerdo a los conceptos formulados en la sección anterior se usa Protégé-OWL (un editor de ontologías), que permite representar las clases, propiedades y restricciones de la ontología. El Protégé-OWL utiliza las propiedades de OWL y genera un sistema basado en conocimiento; una ontología OWL incluye descripciones de las clases y de las propiedades de ellas. La semántica formal OWL especifica la forma de obtener consecuencias lógicas, es decir, hechos no literalmente presente en la ontología, sino que supondría la semántica. Protégé-OWL permite verificar que:

- La definición de cada clase sea consistente y no contradictoria con el conocimiento almacenado, puesto que se obtienen a través de las propiedades y restricciones. En Protégé-OWL llas *propiedades* se utilizan para establecer relaciones entre individuos[31], o de individuos con datos, ya que están formuladas en base a las clases existentes. Las propiedades poseen un dominio y un rango, a través de ellos se limita los individuos a los cuales se les aplica la propiedad. Las *restricciones* indican cómo pueden utilizar las propiedades las instancias de una clase. Todo lo anterior limita qué y cuántos

[31] Instancia de una clase

valores pueden ser usados, en que contextos estos pueden ser usado, etc., basado en la información almacenada. Pero lo más importante, permite generar conclusiones no contradictorias con el conocimiento que puede ser inferido desde el marco ontológico bajo descripción, ni con otras definiciones de clases del mismo marco ontológico (es validado al describir las restricciones y propiedades de cada clase).[32].

- La ontología sea completa, pues Protégé-OWL define explícitamente, a través de las clases y subclases, todo el marco ontológico, y válida las relaciones que se dan entre ellas, garantizando que el conocimiento pueda ser inferido desde las definiciones de las clases, sus propiedades, y las restricciones que se establecen para ellas[33].
- La ontología sea concisa, puesto que no permite redundancias dado que en Protégé-OWL no se le puede dar el mismo nombre a clases, subclases, o propiedades. Por lo tanto, no ocurren inferencias de clases redundantes.

A continuación se muestran en detalle algunas características de Protégé-OWL que nos interesan en este trabajo.

5.2.1.1 Características del Protégé OWL DL de interés para la validación

OWL es el acrónimo del inglés **Ontology Web Language**, un lenguaje de marcado para publicar y compartir datos usando ontologías en la WWW. **OWL** tiene como objetivo facilitar un modelo de marcado construido sobre RDF y codificado en XML. OWL proporciona tres sublenguajes: OWL Lite, OWL DL y OWL Full. Cada sublenguaje de OWL se define de acuerdo a su expresividad. OWL Lite es el sublenguaje sintácticamente más simple; se utiliza cuando se requiere de jerarquías de clases y restricciones simples. OWL DL es más expresivo que el OWL-Lite, y está basado en Lógica Descriptiva, por lo tanto permite realizar clasificación de jerarquías y chequeo de inconsistencias de la ontología expresada en este sublenguaje. OWL Full posee la máxima expresividad y libertad sintáctica del RDF, pero sin garantía computacional.

El Protégé OWL se utiliza para cargar y guardar archivos de OWL, para editar ontologías OWL con adaptadores gráficos, y para realizar razonamiento inteligente basado en Lógica Descriptiva. Como se muestra en la Figura 5.1, la interfaz de usuario de Protégé-OWL proporciona diversas pestañas:

- La pestaña OWLClasses muestra la ontología de la jerarquía de las clases, permite crear y editar clases, y muestra el resultado de la clasificación.
- La pestaña de Propiedades se utiliza para crear y editar las propiedades de la ontología. Las propiedades describen las relaciones que existen entre clases.

[32] Ver Anexo manual de Protégé-OWL
[33] Ver Anexo manual de Protégé-OWL

- La pestaña de Individuos se utiliza para crear y editar los individuos (instancias de clases). Estos individuos se crean en función de las clases creadas y las propiedades entre ellas.
- La pestaña Forms permite personalizar los formularios utilizados para la edición de las clases, las propiedades y los individuos.
- La pestaña Metadatos muestra los metadatos de la ontología, tales como prefijos de nombres.

Como se dijo antes, las clases en Protégé OWL se definen a través de la pestaña de OWLClasses (ver figura 5.1). Las clases OWL se interpretan como un conjunto de individuos (o conjunto de objetos) que permanecen juntos porque comparten propiedades. Las clases están organizadas en una jerarquía de especialización usando ***subClassOf***. La clase owl:Thing es la clase que representa el conjunto que contiene todos los individuos, es por esto que todas las clases son subclases de owl:Thing. Los detalles de la clase seleccionada se muestran en el lado derecho de la pantalla. La parte superior de esta área permite a los usuarios agregar comentarios, etiquetas y otras anotaciones. La parte inferior muestra las características lógicas y las propiedades de la clase seleccionada. Así, a través de las clases Protégé-OWL representa los conceptos del dominio, en nuestro caso del proceso de integración de las bases de datos federadas.

En OWL, las restricciones se definen a través de condiciones, las cuales pueden ser *condiciones necesarias & suficientes*[34] o *condiciones necesarias*[35]. Estas condiciones son las que permiten realizar procesos de inferencia usando las clases, y proporcionan la definición del comportamiento lógico para cada clase. Así, dicho comportamiento lógico de cada una de las clases se describe a través de axiomas (cada condición es un axioma). Ejemplos de posibles axiomas fueron presentados en las tablas del capítulo 3, los cuales fueron definidos en lenguaje natural y expresados en lógica de primer orden. La implementación de estos axiomas se visualiza en el centro de la ventana de la figura 5.1.

En las figuras 5.1 y 5.2 se muestran encerradas en una elipse los botones que activan la vista lógica (Figura 5.1) y la vista de propiedades (Figura 5.2) de la ontología. La vista lógica muestra las propiedades lógicas, como son las restricciones, y la vista de propiedades muestra las propiedades de objeto (ObjectProperty) y las propiedades del tipo de dato (datatype) para las clases y subclases (identificados por símbolos en la ventana donde se describen las propiedades). Todo esto se explica luego.

[34] La condición *necesaria&suficiente* se lee como: "Si un individuo es miembro de una clase debe satisfacer las condiciones, y si ese individuo satisface las condiciones entonces el individuo debe ser un miembro de la clase"
[35] La condición *necesaria* se lee como "Si alguien es miembro de esta clase entonces es necesario que cumpla estas condiciones" (Manual de Protégé OWL Edición 1.0 Pág. 59).

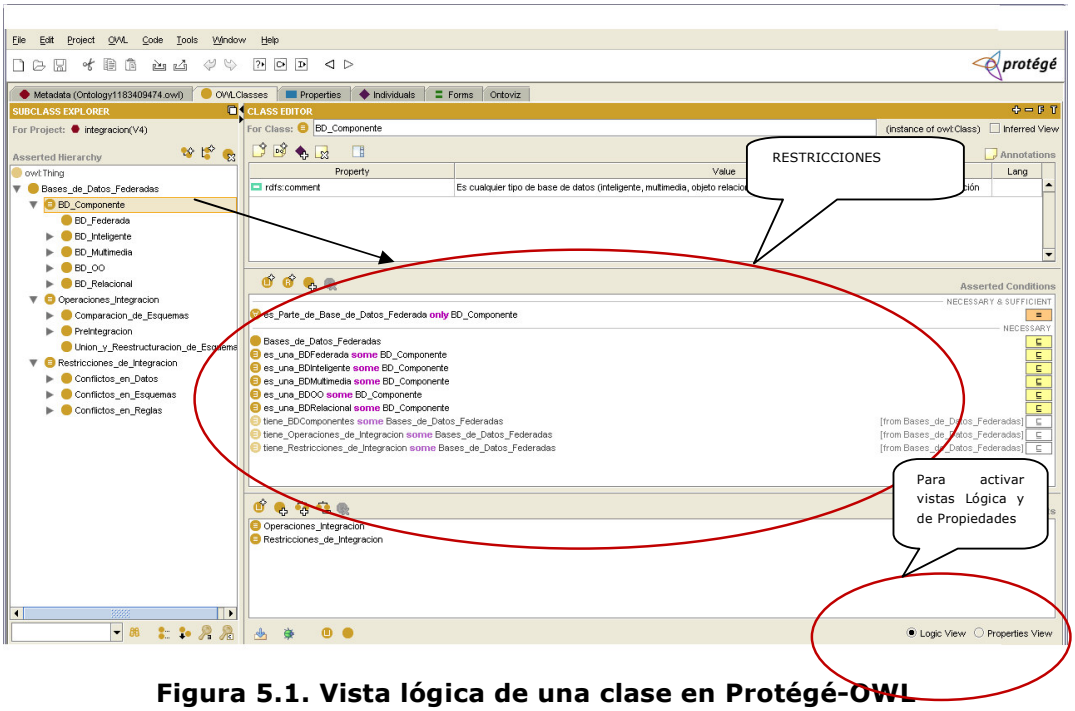

Figura 5.1. Vista lógica de una clase en Protégé-OWL

Las propiedades de cada una de las clases se muestran en el área media del lado derecho de la pantalla (ver figura 5.2). Las propiedades se utilizan para establecer relaciones entre individuos o de individuos con datos. En las dos últimas ventanas del lado derecho se muestra la superclase a la que pertenece la clase seleccionada (ventana izquierda) y las clases disjuntas de la clase seleccionada (ventana derecha). Por ejemplo, Hombre y Mujer son clases disjuntas, por lo que con la declaración disjointWith un razonador puede deducir que existe una inconsistencia cuando un individuo establece que es una instancia de ambos; similarmente, un razonador puede deducir que si A es una instancia de Hombre entonces A no es una instancia de Mujer.

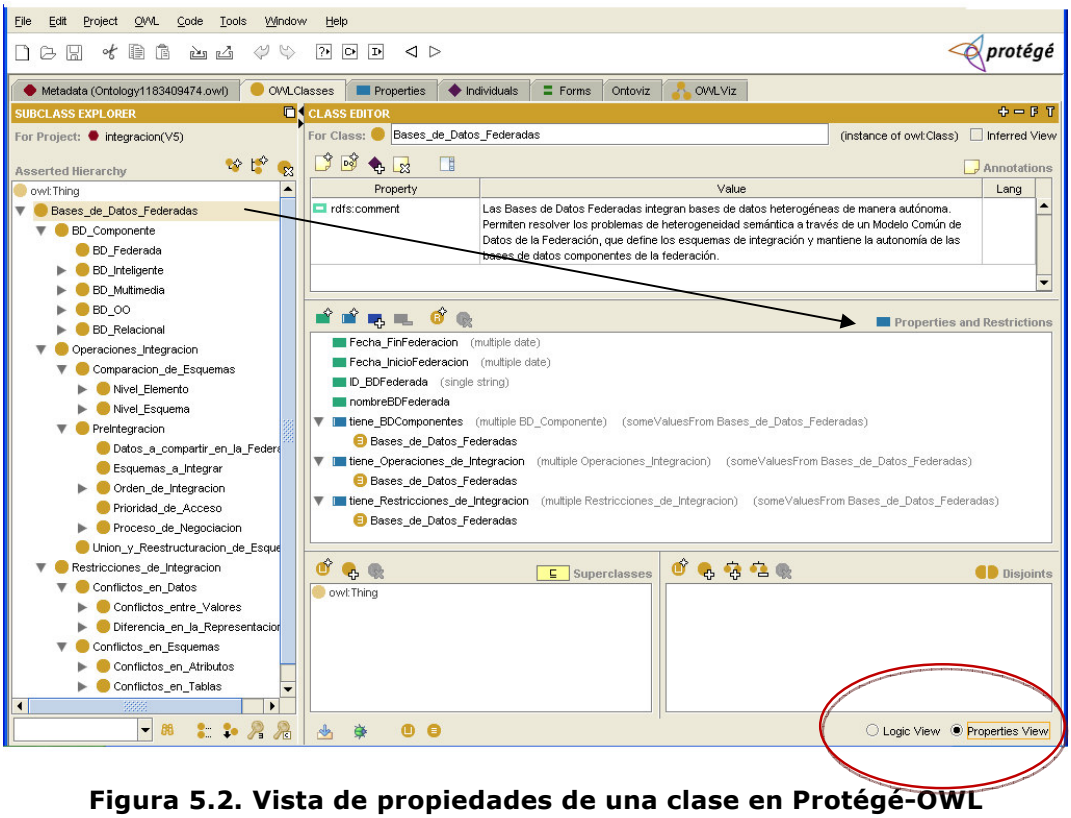

Figura 5.2. Vista de propiedades de una clase en Protégé-OWL

En la siguiente figura se muestra como se describen las propiedades para cada una de las clases que conforman la taxonomía del modelo (pestaña de propiedades). Como se dijo antes, las propiedades se utilizan para establecer relaciones entre dos individuos, o entre individuos y datos. Así, estas propiedades pueden ser propiedades de objeto (ObjectProperty) que definen las relaciones entre dos clases (relación binaria), y propiedades de tipo datatype que relacionan clases con datos. En la ventana del lado izquierdo se muestran varias pestañas para definir el tipo de propiedad que utiliza Protégé-OWL. La primera Propiedad Objeto (llamada también ObjectProperty, señalada con el rectángulo azul) relaciona instancias de clases (individuo con individuo). Estas propiedades de objeto tienen un dominio[36] y un rango[37] de clases en que esa propiedad puede actuar (se muestran en las dos ventanas inferiores de la pantalla). Esta propiedad se utiliza como "axioma" en el razonamiento. La segunda pestaña con la propiedad datatype (señalada con un rectángulo verde) relaciona instancias de clase con instancias de datos. Este tipo de propiedad define características adicionales de las clases a nivel de tipo de datos. En la figura 5.4 se muestran las

[36] El dominio y el rango restringen la relación entre clases. Un dominio de la propiedad reduce el número de individuos a los que puede aplicarse la propiedad. Si una propiedad relaciona un individuo con otro individuo, y la propiedad tiene una clase como uno de sus dominios, entonces el individuo debe pertenecer a esa clase.

[37] El rango de una propiedad reduce los individuos que una propiedad puede tener como su valor. Si una propiedad relaciona un individuo con otro individuo, y ésta tiene a una clase como su rango, entonces el otro individuo debe pertenecer a dicha clase.

características de esta propiedad. Las propiedades OWL de tipo ObjectProperty, además, poseen características que enriquecen su significado:

- Funcional: Si una propiedad es funcional para un determinado individuo, puede haber a lo sumo un individuo que se relaciona con ese individuo a través de la propiedad. Veamos el caso de la relación **tieneMadreBiológica** (una persona puede tener solo una madre biológica). Si se dan las relaciones que el individuo **Juan tieneMadreBiológica María** y el individuo **Juan tieneMadreBiológica Margarita**, y si suponemos que **tieneMadreBiológica** es funcional, se puede inferir que María Margarita es el mismo individuo.
- Funcional Inversa: Si una propiedad es funcional y tiene inversa, significa que la propiedad es funcional inversa. Por ejemplo, la inversa funcional de **tieneMadreBiológica** es **esMadreBiologicaDe**; así, si **María esMadreBiologicaDe Juan** entonces **Juan tieneMadreBiológica María**.
- Transitiva: Una propiedad es transitiva cuando el individuo **a** se relaciona con el individuo **b** y este se relaciona con el individuo **c**, implica que **a** se relaciona con **c**.
- Simétrica: Una propiedad es simétrica cuando relaciona al individuo **a** con el individuo **b**, pero además la misma propiedad relaciona el individuo **b** con el individuo **a**.

En la figura 5.3 se muestra una *propiedad inversa*, en la ventana izquierda señalada en letras grises, y en la ventana derecha en la parte inferior, así como las *características de las propiedades* (funcional, inversa funcional, simétrica y transitiva), las propiedades *datatype* (rectángulos verdes), las *ObjectProperty* (rectángulo azul) y el *dominio* y *rango* para la propiedad seleccionada.

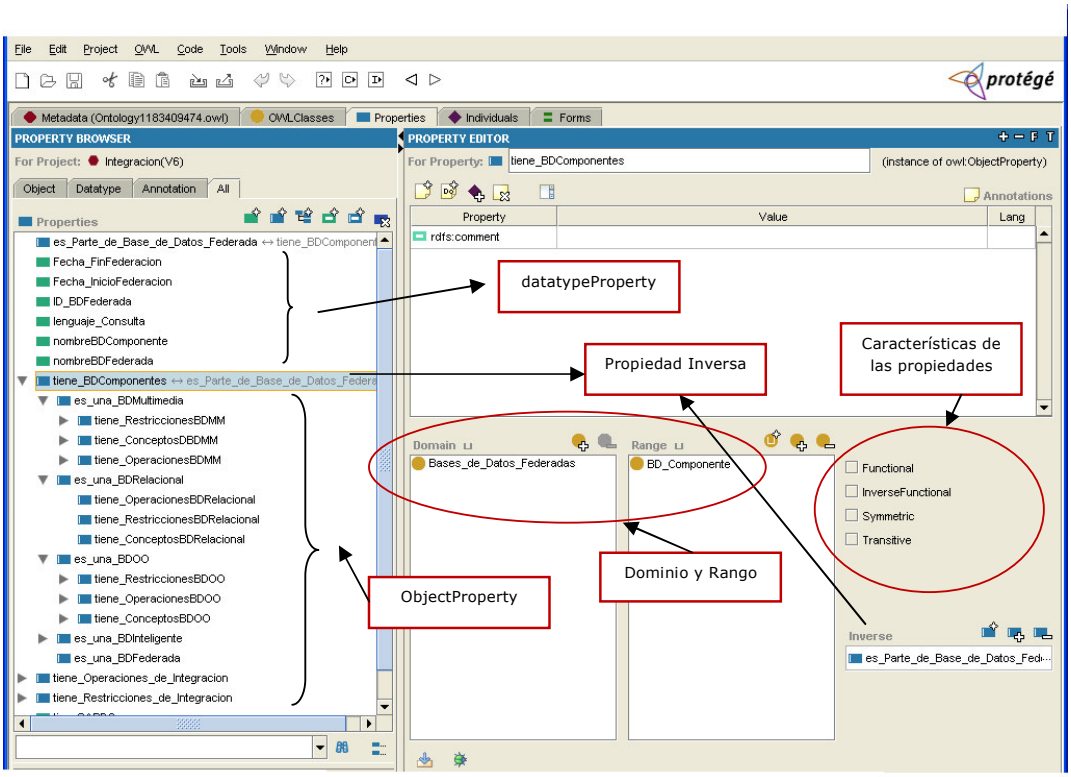

Figura 5.3. Pestaña a través de la cual se definen las propiedades objeto (ObjectProperty) del modelo.

En la figura 5.4 se muestran las características de las propiedades datatype, en ellas se especifica el dominio de acuerdo a la clase a la que pertenecen (en este ejemplo para el datatype: Fecha_FinFederacion su dominio es la clase Bases_de_Datos_Federadas), como se muestra en la ventana derecha señalada como Dominio (ver figura 5.4). Por otro lado, el Rango para esta propiedad está definido por los valores que puede tomar como variable, en este caso es un tipo de dato fecha (date). Además, a esta propiedad se le pueden especificar ciertos tipos de valores a través de la ventana *valores permitidos*. Este tipo de propiedad solo puede tener la propiedad funcional.

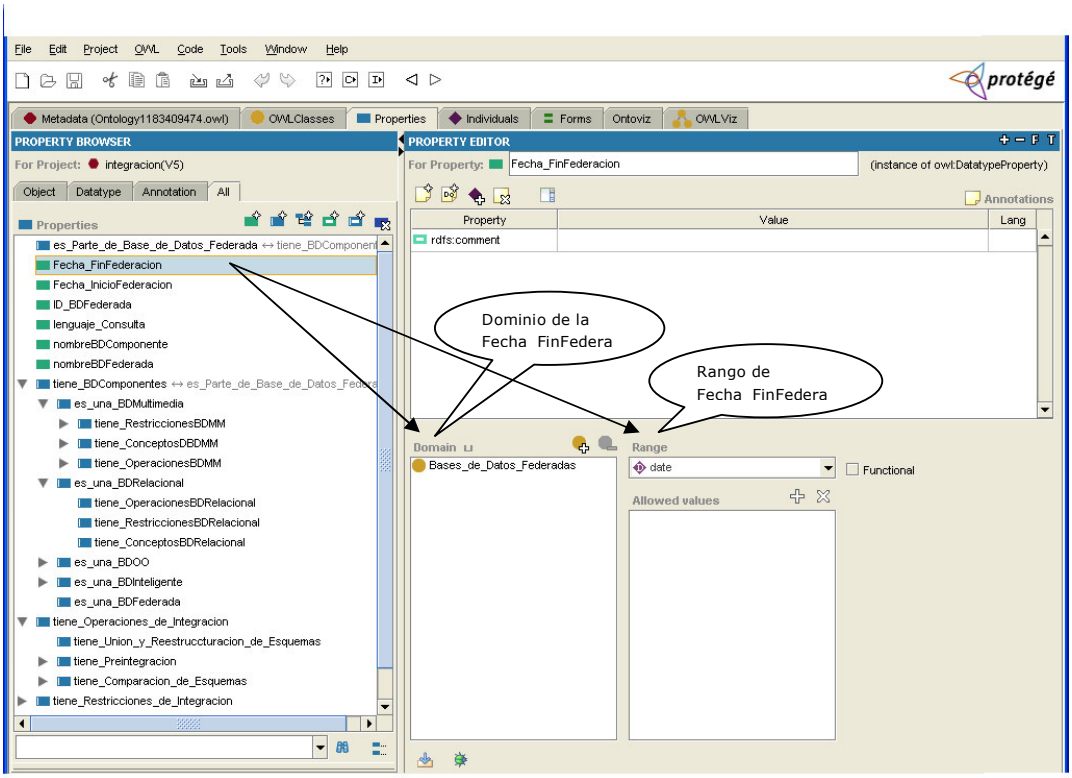

Figura 5.4. Propiedades datatype

Las propiedades objectProperty y datatype validan la completitud debido a que definen de manera explícita el tipo de relación entre clases, lo que permite los procesos de inferencia de conocimiento; así como también la consistencia de la ontología porque a través de las objectproperty se definen las restricciones que establecen el comportamiento permitido de la ontología que garantiza generar conclusiones no contradictorias. En cuanto a la concisión, estas propiedades al estar definidas no permiten que existan redundancias entre términos ni almacenar definiciones innecesarias.

5.3 Formalización de la Ontología utilizando Protégé-OWL

A continuación se muestra la ontología para la integración de bases de datos federadas formalizado a través de Protégé-OWL.

El corazón del modelo de integración está conformado por los elementos que se muestran en la taxonomía para los conceptos, operaciones y restricciones de las bases de datos federadas (figuras 3.2 3.4 y 3.5). Las definiciones ontológicas allí descritas modelan el comportamiento de las bases de datos federadas según el modelo de Shet & Larson [Shet, & Larson, 1990] para bases

de datos relacionales; a este modelo se le agregaron las bases de datos orientadas a objetos, bases de datos multimedia y bases de datos inteligentes.

Para evaluar la consistencia, completitud y concisión se utilizó Protégé-OWL, que al ser utilizado en la especie OWL DL realiza estás evaluaciones utilizando de las propiedades de descripción lógica descritas en la sección 5.2.1.1. Esta herramienta tiene limitaciones en cuanto a la representación de las restricciones, ya que solo pueden representarse las restricciones sobre las relaciones entre conceptos "is-a", "part-Of" y "has". En nuestra taxonomía, además de las relaciones is_a, Part_Of y has, existen otro tipo de relaciones como: "forma", "incorpora", "sucede" y "genera" que deben ser definidas en OWL. Al no estar definidas en esta herramienta, esta parte de nuestra ontología no pudo ser implementada.

En la figura 5.5 se muestra la taxonomía de la ontología. Esta taxonomía se genera a través de un plugin en Protégé-OWL llamado OWLViz (generado a través de la aplicación graphViz conectada a Protégé-OWL) que permite visualizar la taxonomía. Se puede observar que la ontología posee más de 200 conceptos.

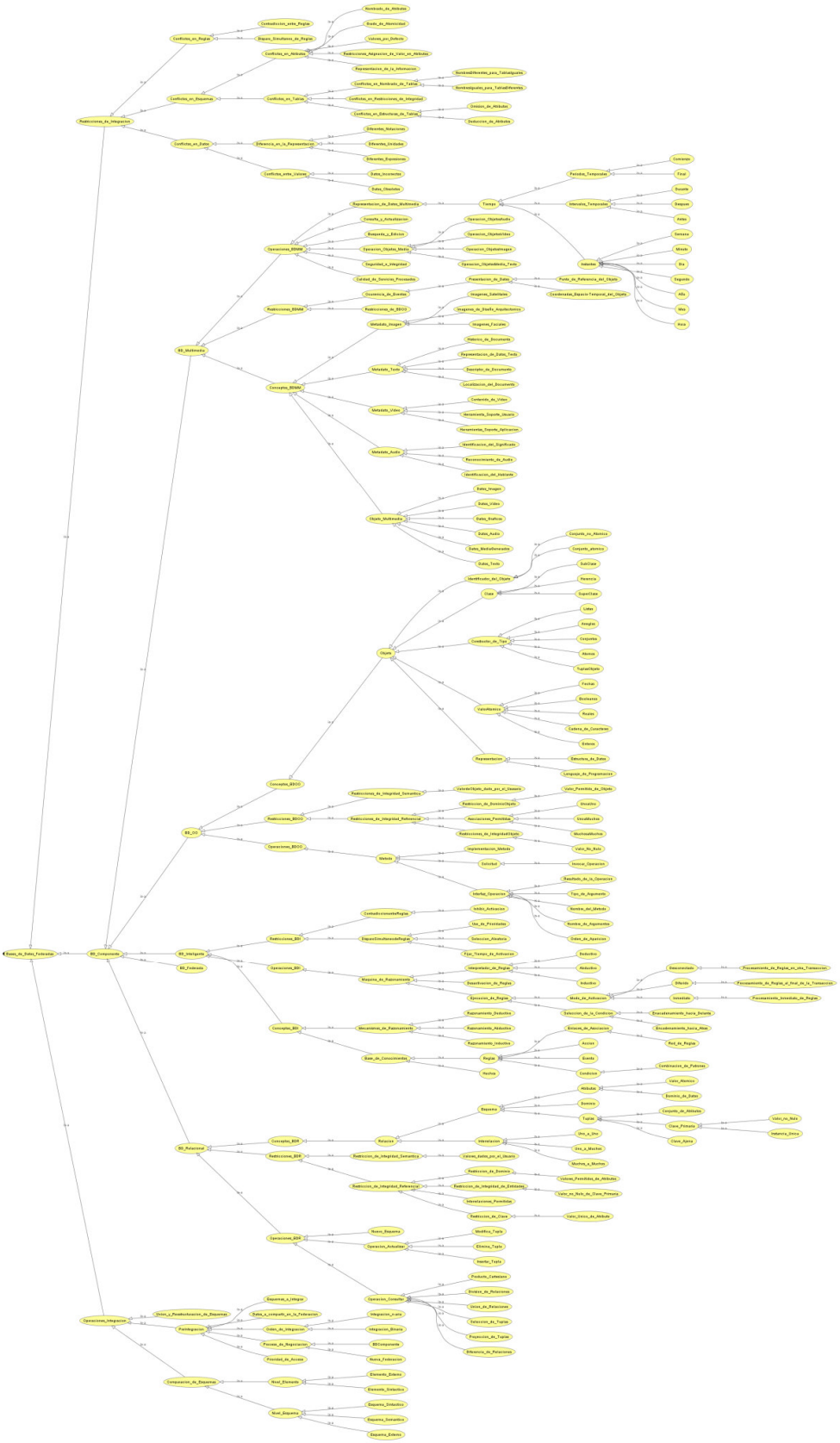

Figura 5.5. Taxonomía de la Ontología integración de bases de datos federadas

5.3.1 Modelo de integración validado en Protégé-OWL

A continuación se muestran los conceptos, propiedades y definiciones (restricciones) del modelo de integración en Protégé-OWL. Se describe cada uno de los elementos que conforman la ontología y sus propiedades. En la medida de que se vaya implementando en Protégé-OWL la ontología, se va realizando la validación de las clases y subclases (conceptos) de la taxonomía. A través de las definiciones (descritos en las tablas de axiomas del modelo en el capítulo 3) se establecen las restricciones de cada una de las clases y sus propiedades

5.3.1.1 Clases, sub-Clases y Propiedades de la Ontología

Como se puede observar en la Figura 5.6, la clase principal es Bases_de_Datos_Federadas, y sus subclases son: la subclase "*BD_Componente*", la subclase "*Operaciones_Integración*" y la subclase "*Restricciones_de_Integracion*"; estas son disjuntas entre sí, es decir, una *BD_Componentes* no puede ser una *Operación_de_Integracion* ni una *Restricción_de_Integracion*.

Las subclases de *BD_Componentes* son: *BD_Federada*, *BD_Inteligente*, *BD_MM*, *BD_OO* y *BD_Relacional,* que son disjuntas entre ellas. Las subclases de *Operaciones_Integración* son: *Preintegración*, *Comparacion_de_Esquemas* y *Union_y_Reestructuracion_de_Esquemas,* y son disjuntas entre ellas. Las subclases de *Restricciones_de_Integracion* son: *Conflictos_en_Datos, Conflictos_en_Esquemas y Conflictos_en_Reglas,* y son disjuntas entre ellas.

Dentro de las subclase BD_Inteligente se encuentran las subclases: Conceptos_BDI, Operaciones_BDI, Restricciones_BDI; la subclase BD_MM tiene a su vez las subclases: Conceptos_BDI, Operaciones_BDI y Restricciones_BDI, y así sucesivamente para cada uno de las subclases de las BD_Componentes.

De igual manera, la subclase Operaciones_de_Integracion tiene las subclases: Comparacion_de_Esquemas, Pre_Integracion y Union_y_Reestructuracion_de_Esquemas.

Finalmente, la subclase Restricciones_de_Integracion tiene las siguientes subclases: Conflictos_en_Datos, Conflictos_en_Esquemas y Conflictos_en_Reglas.

Con estas definiciones de clases y subclases, así como con la disyunción entre ellas, se está validando que la ontología es completa, ya que los conceptos están explícitamente establecidos, y no existen redundancias pues sus nombres son únicos y tienen significados diferentes.

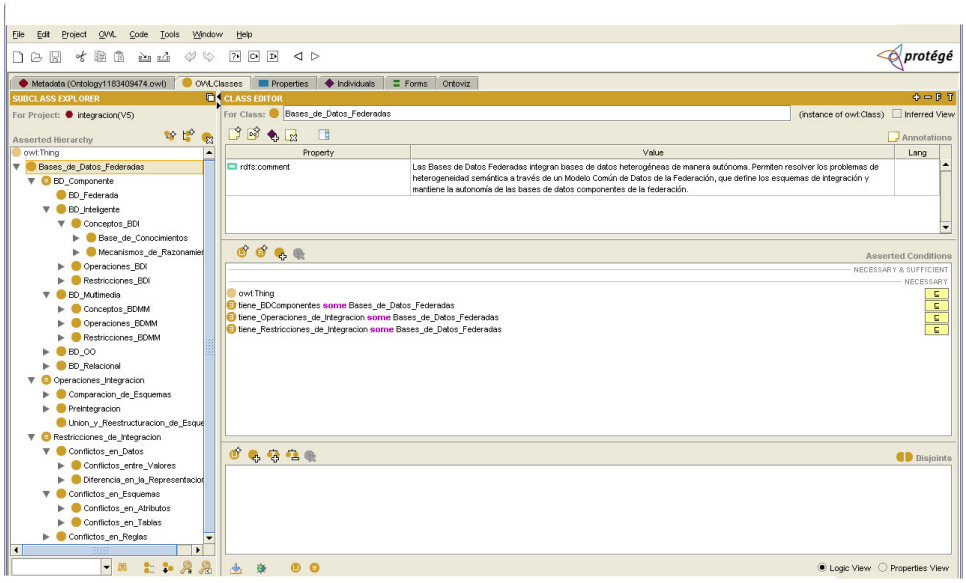

Figura 5.6 Clases, subclases y restricciones de la clase Base_de_Datos_Federadas

A continuación, en la figura 5.7 se muestran las propiedades de la clase Base_de_Datos_Federadas. Allí se muestran las propiedades objectProperty y datatype de esta clase, y lo que se muestra en gris son las propiedades heredadas. Se observan señaladas por un rectángulo verde las propiedades datatype (por ejemplo Fecha_Fin_de_Federación (que puede contener múltiples datos, especificados entre paréntesis) e ID_BDFederada (especificado como single string, que indica que solo puede contener un valor único, esto ocurre al asignarle a este datatype la propiedad funcional), y señaladas por un rectángulo azul las propiedades objeto (por ejemplo: tiene_BDComponetes, tiene_Operaciones_de_Integración y tiene_restricciones_de_Integracion, que poseen valores múltiples, lo que indica que estas propiedades no son funcionales).

Estas propiedades validan la completitud al definir el comportamiento de cada una de las clases y las relaciones con otras clases de manera formal.

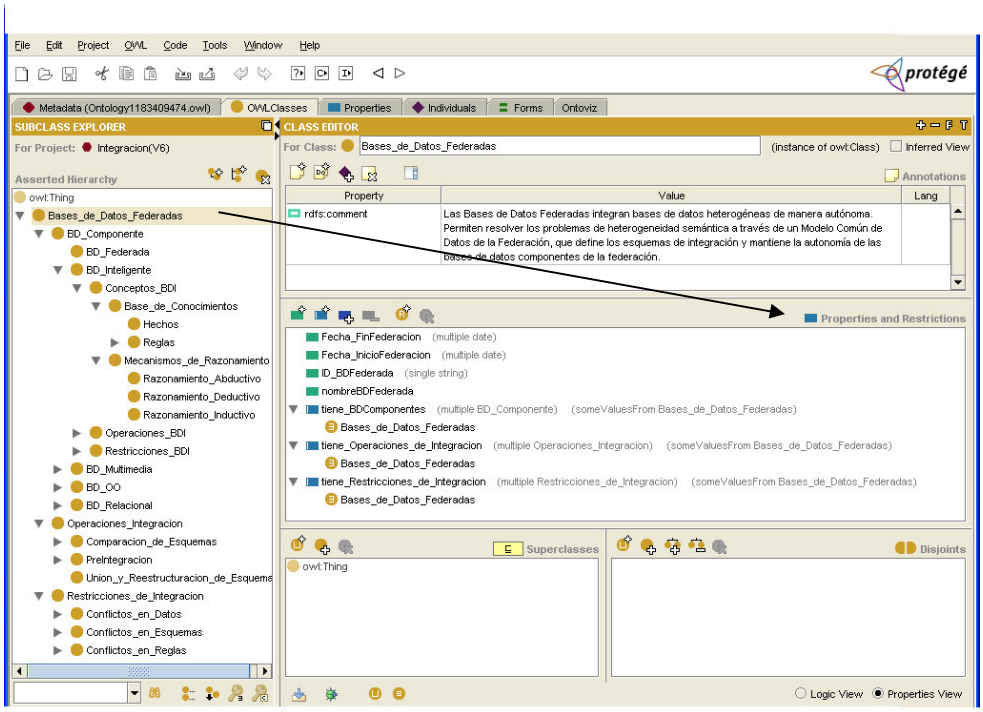

Figura 5.7. Propiedades y Restricciones de la clase Base_de_Datos_Federadas

En las figuras 5.8 y 5.9 se muestra la subclase BD_Componente en su vista lógica y vista de propiedades (se visualiza en la parte inferior derecha de la pantalla que muestran las figuras 5.8 y 5.9).

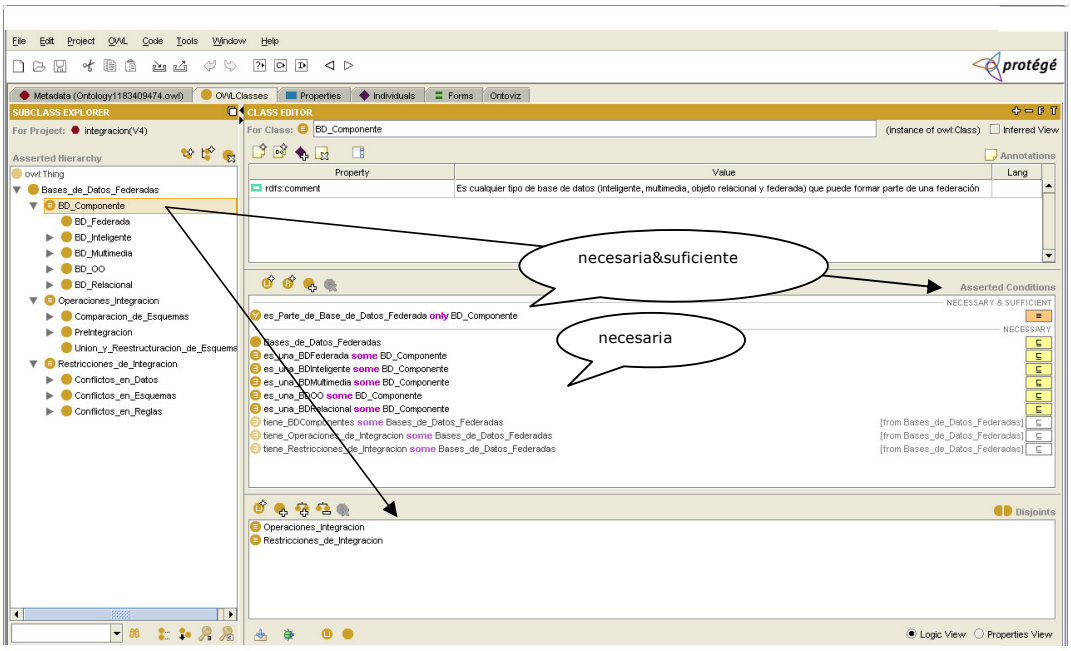

Figura 5.8. Vista lógica para las bases de datos componentes en Protégé-OWL

En la figura 5.8 se muestra las subclases de las BD_Componentes que son: BD_Federada, BD_Inteligentes, BD_MM, BD_OO y BD_Relacional. Se muestran las restricciones de las BD_Componentes (estos son los axiomas de la tabla 1 Capitulo 3) que son las condiciones *necesarias* o *necesaria y suficiente* que deben cumplir las bases de datos componentes. La única condición *necesaria y suficiente* que tiene es: es_Parte_De_Base_de_Datos_Federada, para indicar que debe satisfacer esa condición para ser miembro de esta clase (BD_Componentes). Finalmente, se muestra que la clase BD_Componente es disjunta con las clases Operacion_Integracion y Restricción_Integracion.

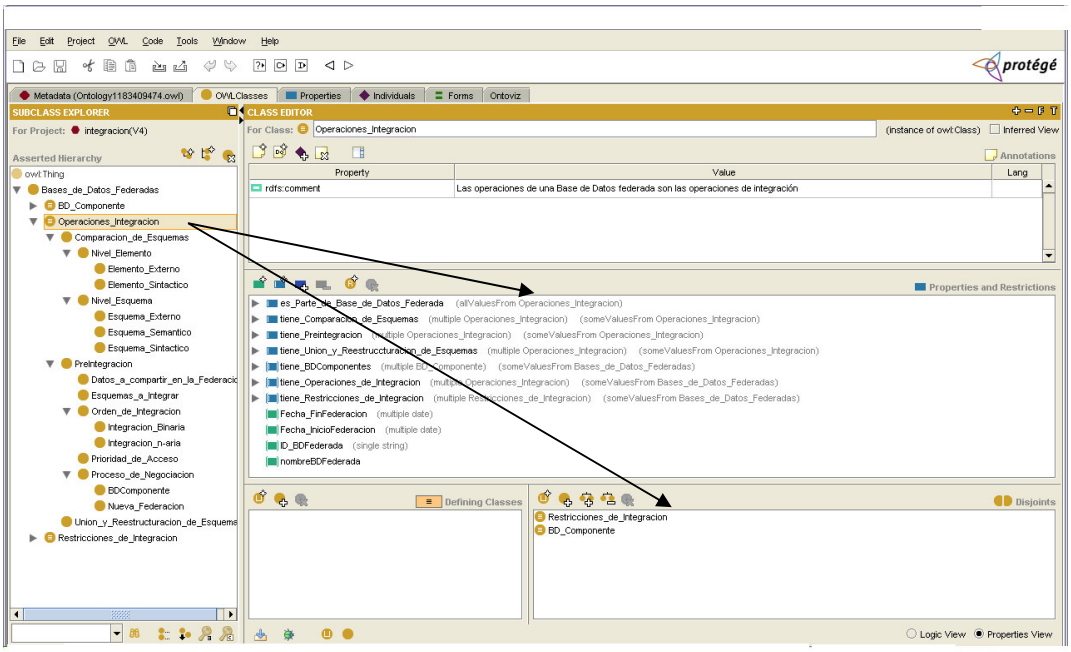

Figura 5.9. Vista de propiedades para las bases de componentes en Protégé-OWL

En la figura 5.9 se observan las propiedades ObjectProperty y datatypeProperty que tienen las BD_Componentes. Las objectProperty son: *es_una_BDFederada, es_una_BDInteligente, es_una_BDMM, es_una_BDOO, es_una_BDR.* Los datatypeProperty son heredados de su superclase Base_de_Datos_Federadas.

En la figura 5.10 se muestra la vista de propiedades para las operaciones de integración.

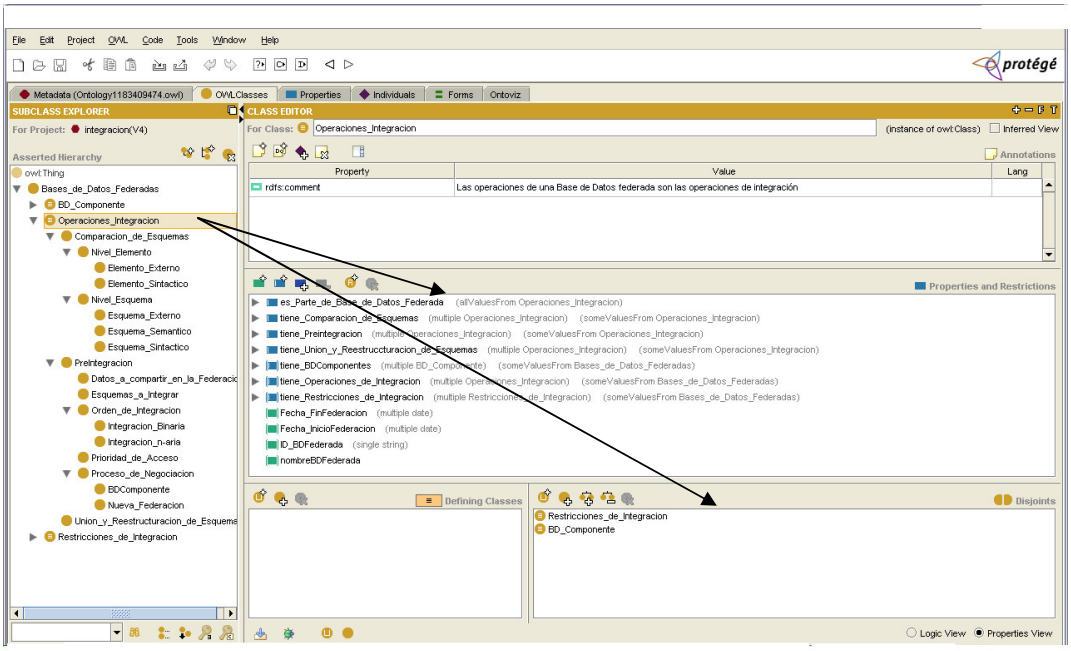

Figura 5.10. Vista de propiedades para las operaciones de integración en Protégé OWL

En esta figura se observa que la clase Operaciones_Integracion está conformada por las subclases: Preintegracion, Comparacion_Esquemas, Union_y_Reestructuracion_Esquemas. Estas tienen a su vez otras subclases; por ejemplo, Preintegración tiene las subclases: Datos_a_Compartir_en_la_Federacion, Esquemas_a_Integrar, Orden_deIntegracion, Prioridad_de_Acceso y Proceso_de_Negociacion. La subclase Comparación_de_Esquemas tiene Nivel_Elemento y Nivel_Esquema. Allí se puede observar que algunas de esas subclases tienen otras subclases, como por ejemplo, la subclase Nivel_Elemento tiene las subclases Elemento_Externo y Elemento_Sintactico.

La clase Operaciones_Integracion es disjunta con las clases BD_Componente y Restricciones_Integracion.

En la siguiente figura se muestra la vista lógica de la clase Operaciones_de_Integracion. Esta vista presenta la definición de la clase a través de las propiedades descritas por medio de las condiciones necesarias que tiene esta clase.

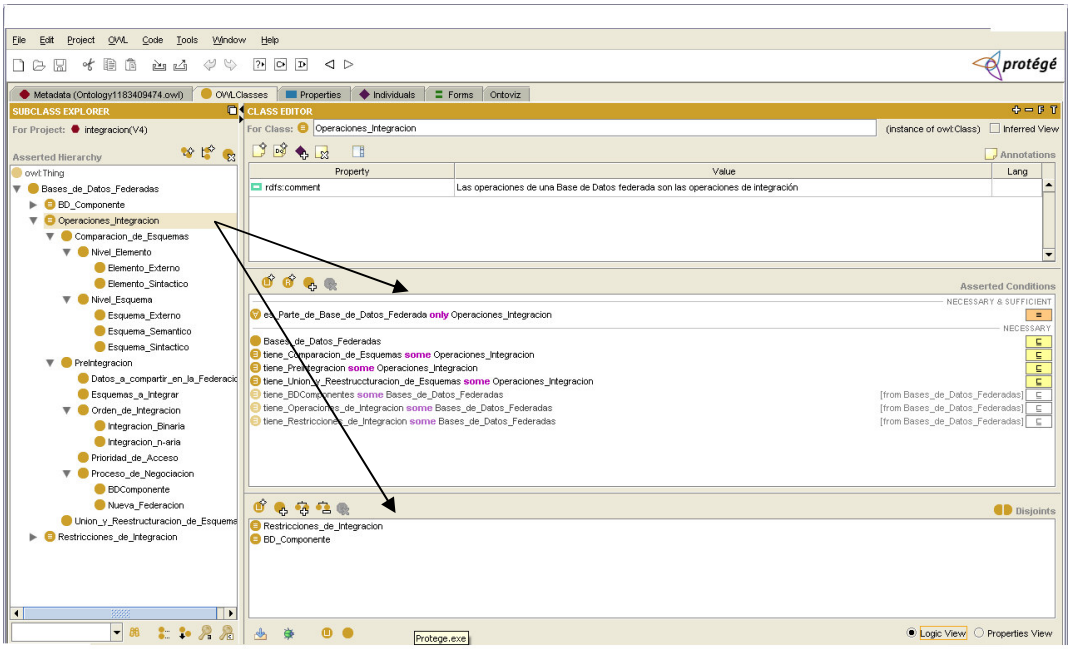

Figura 5.11. Vista lógica para las operaciones de integración en Protégé OWL

Se señalan las restricciones que posee la clase Operaciones_Integracion de la misma manera como se hizo para la clase BD_Componente anteriormente explicada. Estas restricciones validan que la ontología sea completa y consistente, pues se establecen las definiciones válidas para la clase Operaciones_de_Integración de la ontología.

A continuación, en las figuras 5.12, 5.13, y 5.14 se muestran las propiedades y subpropiedades de la clase BD_Componentes de la ontología. Estas propiedades son ObjectProperty, lo que indica que son relaciones entre individuos.

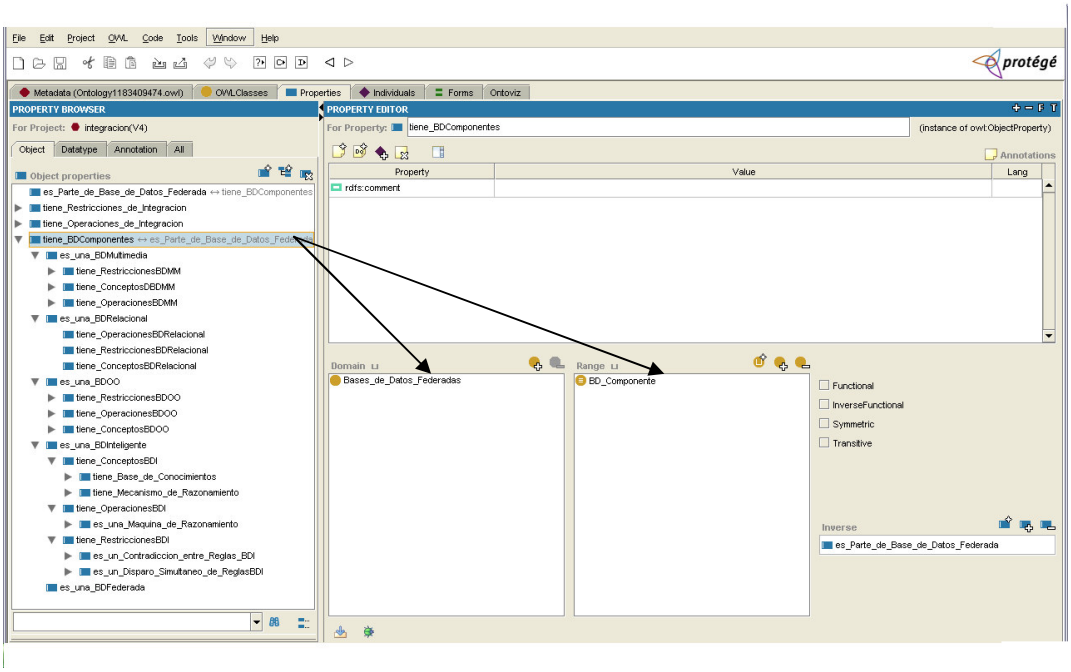

Figura 5.12. Propiedades y subpropiedades de la ontología para las bases de datos componentes

Entre las propiedades ObjectProperty definidas para la clase Bases_de_Datos_Federadas está las siguientes:

- *tiene_BDComponentes*, propiedad que define las bases de datos que pueden conformar una federación, su propiedad inversa es *es_parte_de_BDFederada (se muestra en letra gris a la derecha de la propiedad)*. Sus subpropiedades son:
    - *es_una_BDFederada*
    - *es_una_BDInteligente*, propiedad que define las bases de datos inteligentes, y sus subpropiedades son:
        - *tiene_ConceptosBDI* con su inversa *es_parte_deBDI*
        - *tiene_OperacionesBDI* con su inversa *es_parte_deBDI*
        - *tiene_RestriccionesBDI* con su inversa *es_parte_deBDI*
    - *es_una_BDMM,* propiedad que define las bases de datos multimedia, y sus subpropiedades son:
        - *tiene_ConceptosBDMM* con su inversa *es_parte_deBDMM*
        - *tiene_OperacionesBDMM* con su inversa *es_parte_deBDMM*
        - *tiene_RestriccionesBDMM* con su inversa *es_parte_deBDMM*
    - *es_una_BDOO,* propiedad que define las bases de datos orientadas a objetos, y sus subpropiedades son:
        - *tiene_ConceptosBDOO* con su inversa *es_parte_deBDOO*
        - *tiene_OperacionesBDOO* con su inversa *es_parte_deBDOO*
        - *tiene_RestriccionesBDOO* con su inversa *es_parte_deBDOO*
    - *es_una_BDR,* propiedad que define las bases de datos relacionales, y sus subpropiedades son:
        - *tiene_ConceptosBDR* con su inversa *es_parte_deBDR*

- *tiene_OperacionesBDR* con su inversa *es_parte_deBDR*
- *tiene_RestriccionesBDR* con su inversa *es_parte_deBDR*

Todas las objectProperty pueden estar definidas sobre un dominio, por ejemplo, en la figura 5.12 se muestra que el Dominio para la propiedad **tiene_BD_Componente** es la clase Bases_de_Datos_Federadas, y el Rango es la clase BD_Componente. De la misma manera, se muestra el dominio y rango de las propiedades tiene_Operaciones_de_Integracion y tiene_Restricciones_de_Integracion en las figuras 5.13 y 5.14.

En la figura 5.13 se muestran las propiedades ObjectProperty para las Operaciones de Integración de una base de datos federada (llamadas *tiene_Operaciones_de_Integracion)*

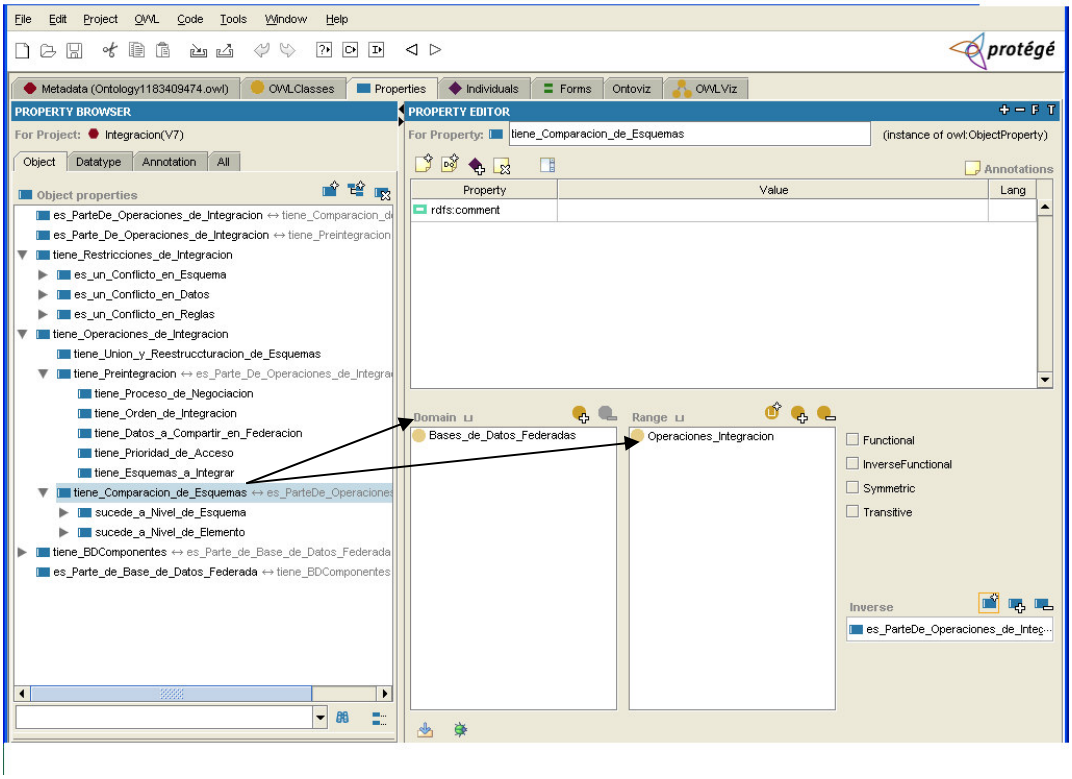

Figura 5.13. Propiedades y subpropiedades de la ontología para las operaciones de integración

tiene_Operaciones_de_Integracion es una propiedad que define las operaciones que realiza una base de datos federada, no tiene propiedad inversa. Sus subpropiedades son las siguientes:

- *tiene_Preintegracion* con su inversa *es_parte_de_Operaciones_de_Integracion*
- *tiene_Comparacion_de_Esquemas* con su inversa *es_parte_de_Operaciones_de_Integracion*
- *tiene_Union_y_Reestructuracion_de_Esquemas* con su inversa *es_parte_de_Operaciones_de_Integracion*

En la siguiente figura se muestran las propiedades ObjectProperty para las Restricciones de Integración en las bases de datos federadas (llamada *tiene_Restricciones_de_Integracion*).

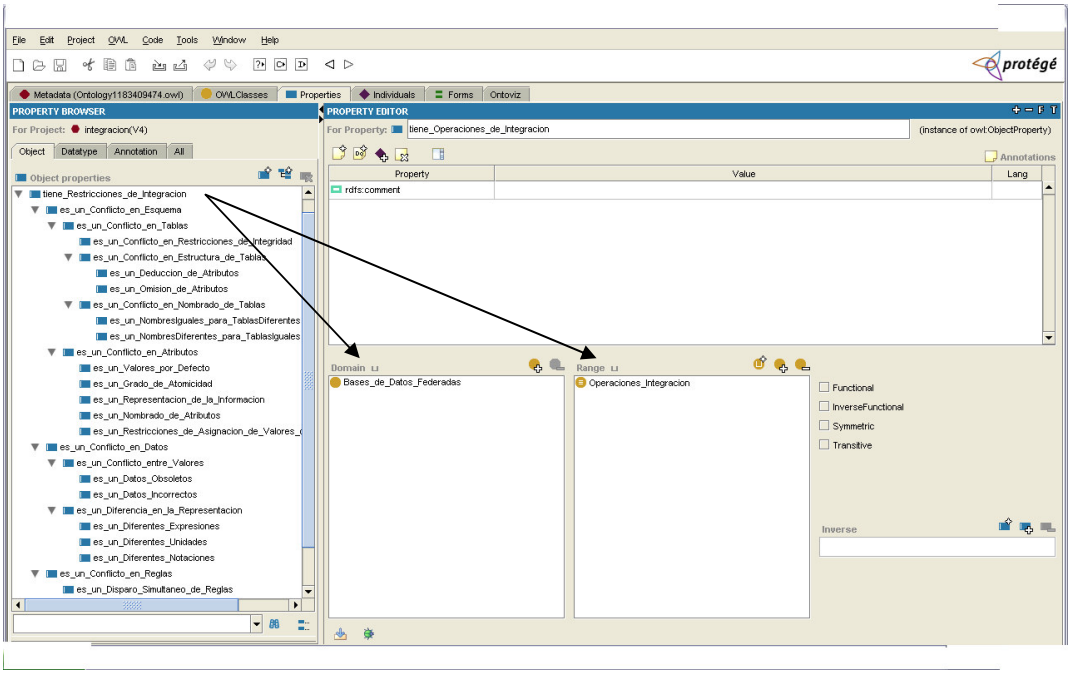

Figura 5.14. Propiedades y subpropiedades de la ontología para las restricciones de integración

tiene_Restricciones_de_Integracion es una propiedad que define las restricciones que tienen las Bases de Datos Federadas, no tiene inversa. Sus subpropiedades son las siguientes:

- *tiene_conflictos_en_Datos* con su inversa *es_parte_de_Restricciones_de_Integracion*
- *tiene_conflictos_en_Esquemas* con su inversa *es_parte_de_Restricciones_de_Integracion*
- *tiene_conflictos_en_Reglas* con su inversa *es_parte_de_Restricciones_de_Integracion*

Las bases de datos componentes también tienen definidas todas sus clases y subclases de acuerdo a las taxonomías definidas en el capítulo 3. A continuación se muestran las clases y subclases, así como sus propiedades y restricciones, para las bases de datos inteligentes. Se definieron de igual manera para el resto de las bases de datos componentes. Las figuras 5.15, 5.16 y 5.17 muestran la vista lógica para las clases Conceptos_BDI, Operaciones_BDI y Restricciones_BDI. En ellas se pueden observar las clases y sus subclases, las restricciones y las clases disjuntas.

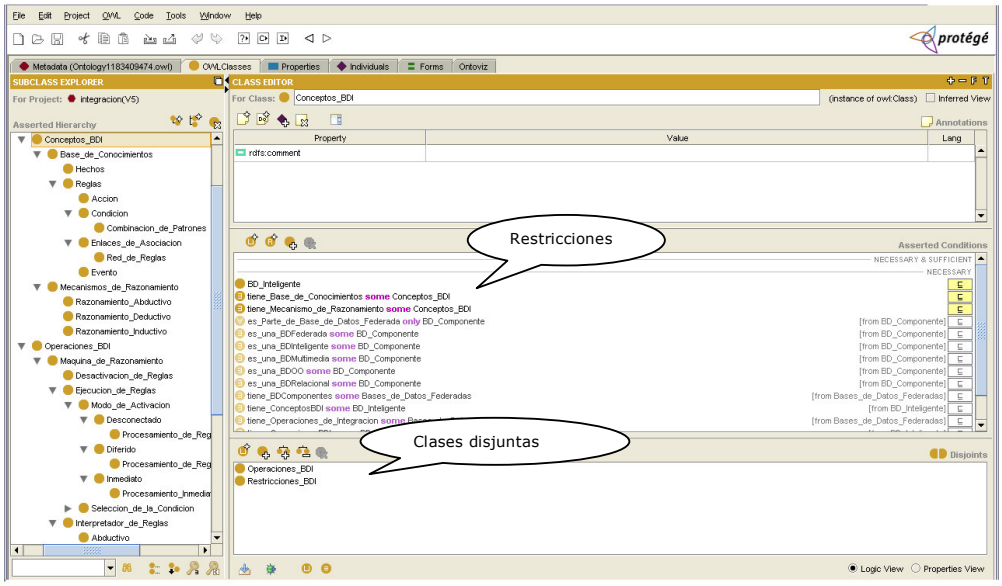

Figura 5.15. Vista lógica para la clase Conceptos_BDI en Protégé OWL

Esta figura muestra la clase Conceptos_BDI, con sus definiciones dadas en las condiciones necesarias. Es una clase disjunta con Operaciones_BDI y Restricciones_BDI.

En la figura 5.16 se muestran la clase Operaciones_BDI, sus subclases, sus restricciones dadas por las condiciones necesarias, y sus clases disjuntas Conceptos_BDI y Restricciones_BDI.

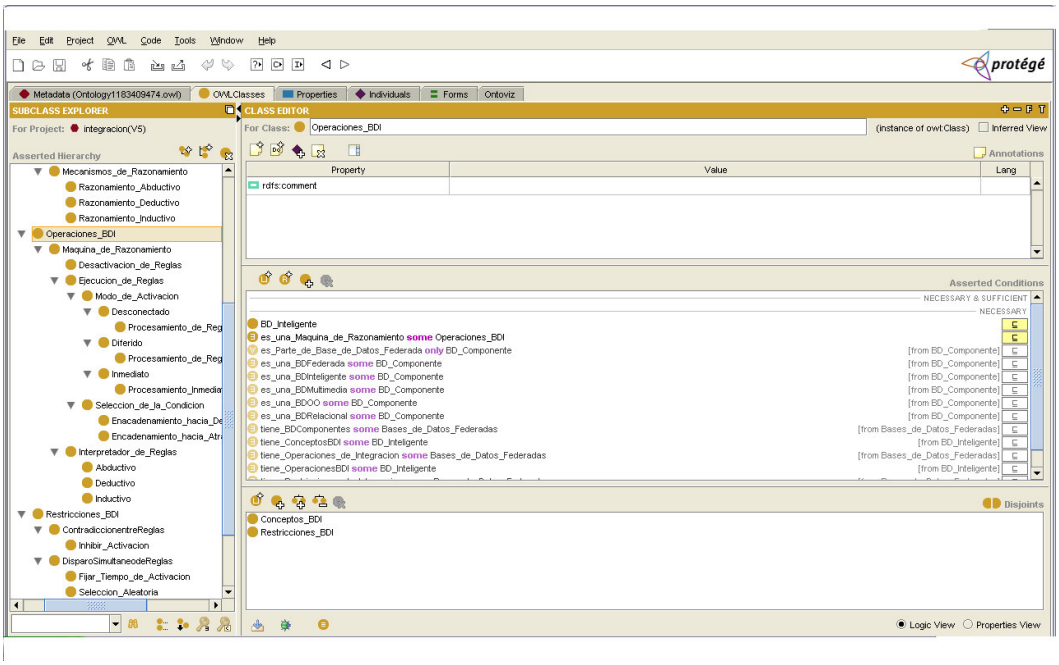

Figura 5.16. Vista lógica para la clase Operaciones_BDI en Protégé OWL

En la figura 5.17 se muestra la clase Restricciones_BDI, sus subclases, sus restricciones dadas por las condiciones necesarias, y sus clases disjuntas Conceptos_BDI y Operaciones_BDI.

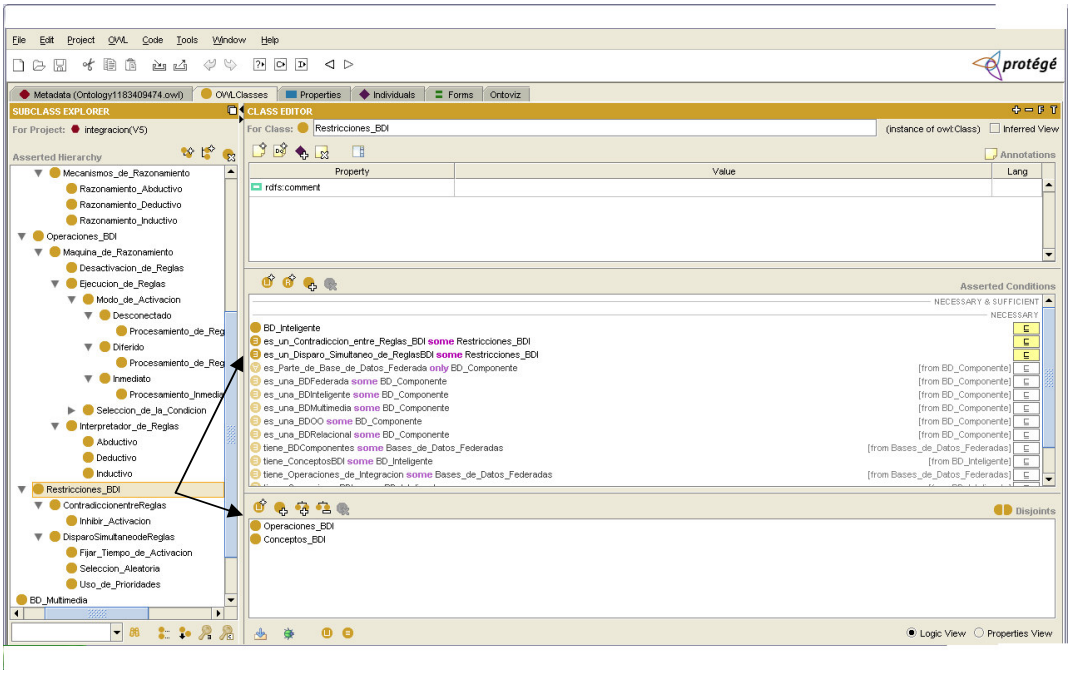

Figura 5.17. Vista lógica para clase Restricciones_BDI en Protégé OWL

A continuación, en la siguiente figura se chequea si el sublenguaje de OWL en el que se encuentra la ontología es OWL DL, que permite validar otro tipo de consistencia llamada "*subsumption*"[38], la cual garantiza que toda clase de la ontología es subclase de otra de la misma ontología (inicialmente, todas las clases son subclases de owl:Thing, pero además, después se debe dar una relación de jerarquía entre las clases (superclases/subclases) para la definición de ellas).

[38]Al generar la ontología en OWL DL, se usa la descripción de las clases para determinar si existen relaciones entre superclases/subclases.

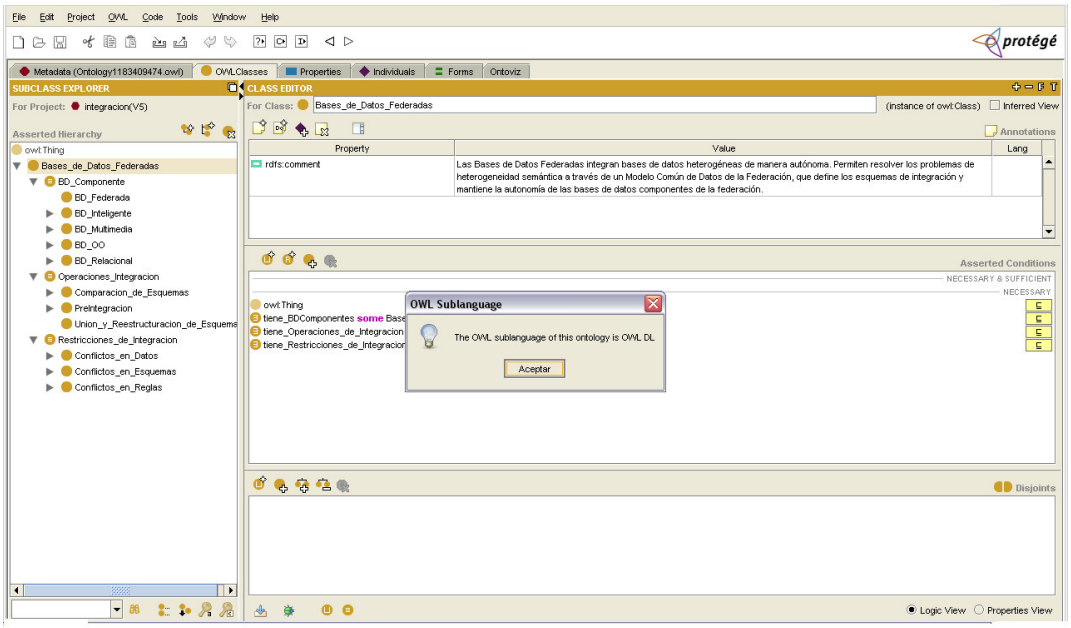

Figura 5.18. La ontología se encuentra en OWL DL

5.4 Instancias para el Modelo de Integración en Protégé-OWL

A continuación se muestra una instancia para la ontología de integración de bases de datos federadas basada en el ejemplo del capitulo 4. Protégé-OWL genera las instancias para las Bases de Datos Federadas de acuerdo a las clases, propiedades y restricciones ya descritas en la sección anterior. Esta instancia es una base de datos federada que integra dos bases de datos: una base de datos inteligente, llamada Sistema de Inscripción Inteligente, y una base de datos multimedia, llamada Sistema multimedia de Control de Cursos. Los cuadros señalados en rojo son los individuos que se generan debido a las definiciones establecidas para la clase BD_Federada, como son: tiene_BDComponentes, tiene_Operaciones_de_Integracion y tiene_Restricciones_de_Integracion.

En la figura 5.19 se muestra un individuo de la clase BD_Federada llamado Inscripcion_Clases (definida en el capitulo anterior). En el lado derecho de la pantalla se observan las características que el individuo puede tener. Se observan las propiedades datatype, como ID_BDFederada (BDF_01), Fecha_Inicio_Federación (2008-03-03). Estas se generan de acuerdo a las propiedades datatype que posee la clase Bases_de_Datos_Federadas mostradas en la figura 5.7. Además, vemos las propiedades objectProperty, que en este caso vienen dadas por las restricciones establecidas en la figura 5.6 que se interpretan de la siguiente manera: BD_Federada tiene_BD_Componente; BD_Federada_tiene_Operacione_de_Integración y BD_Federada_tiene_Restricciones_de_Integracion. Esto restringe los valores que los individuos de las bases de datos federadas pueden tener. Se valida la consistencia al solicitar para cada individuo entradas válidas, por ejemplo el tipo

de dato de la fecha de creación de la base de datos federada; la completitud al permitir incluir individuos de cualquiera de las clases e inferir nuevo conocimiento desde las restricciones definidas para cada clase; y la concisión al no permitir redundancias de los individuos ya que se generan de acuerdo a las propiedades existentes de las clases que no permiten nombres iguales.

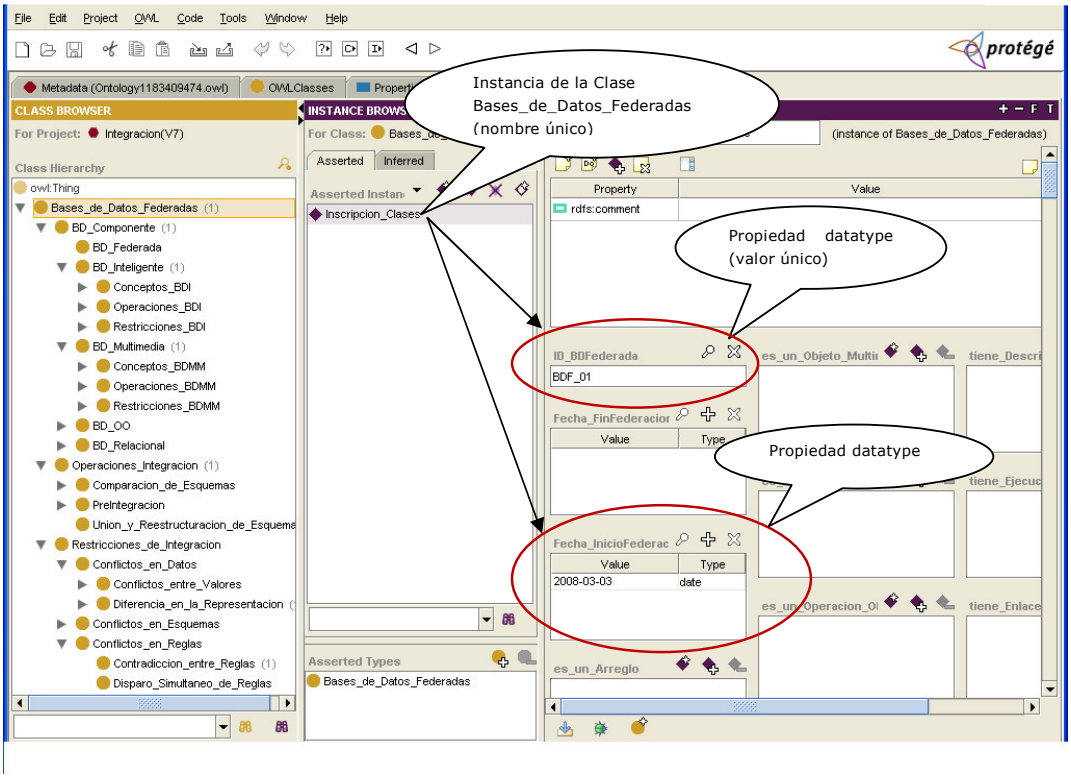

Figura 5.19. Instancia de la Base de Datos Federada Inscripción_Clase

En la siguiente figura se muestra la pantalla del individuo Inscripcion_Clase de la clase Base_de_Datos_Federada. Debido a las definiciones planteadas para la clase, como se muestran en la figura 5.6, aparecen los recuadros rojos para instanciar los detalles definidos para ella. Ejemplos de los detalles a introducir son las propiedades tiene_Operaciones_de_Integración y tiene_BDComponente, que fueron formulados en la figura 5.6 para la clase Base_de_Datos_Federadas.

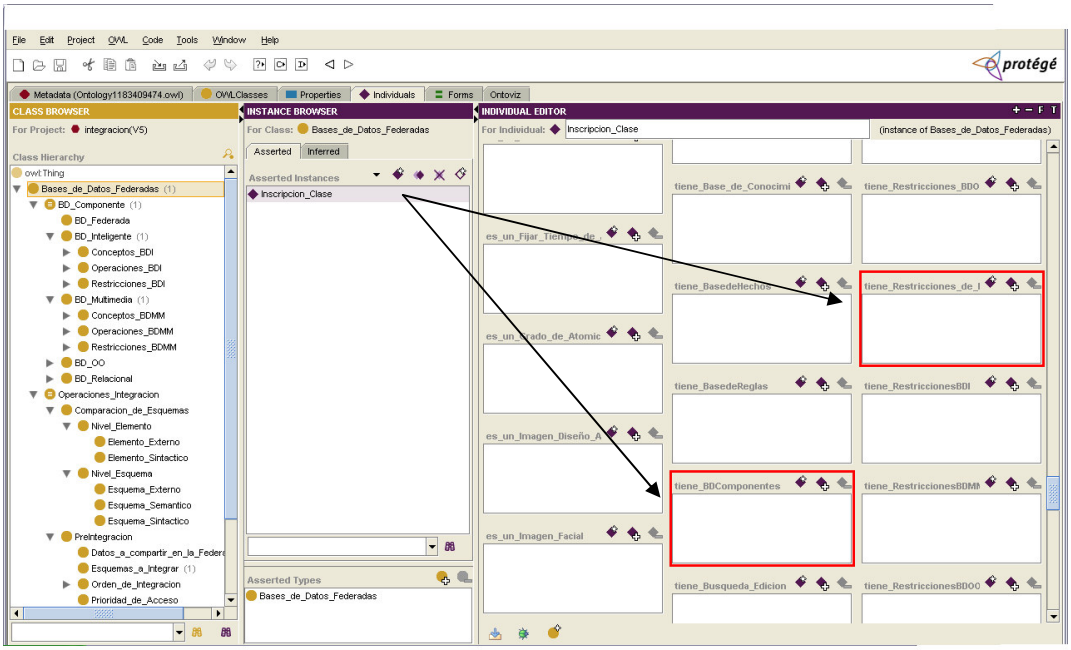

Figura 5.20. Individuo Inscripción_Clase de la clase Base_de_Datos_Federada Inscripcion_Clase

En la figura 5.21 se muestra como se va instanciando la Base de Datos Federada Inscripcion_Clase, indicando que es una base de datos que tiene componentes de la federación, y tiene restricciones de integración. En general, en la figura 5.21 están señaladas las definiciones que puede tener el individuo bases de datos federadas Inscripción_Clase.

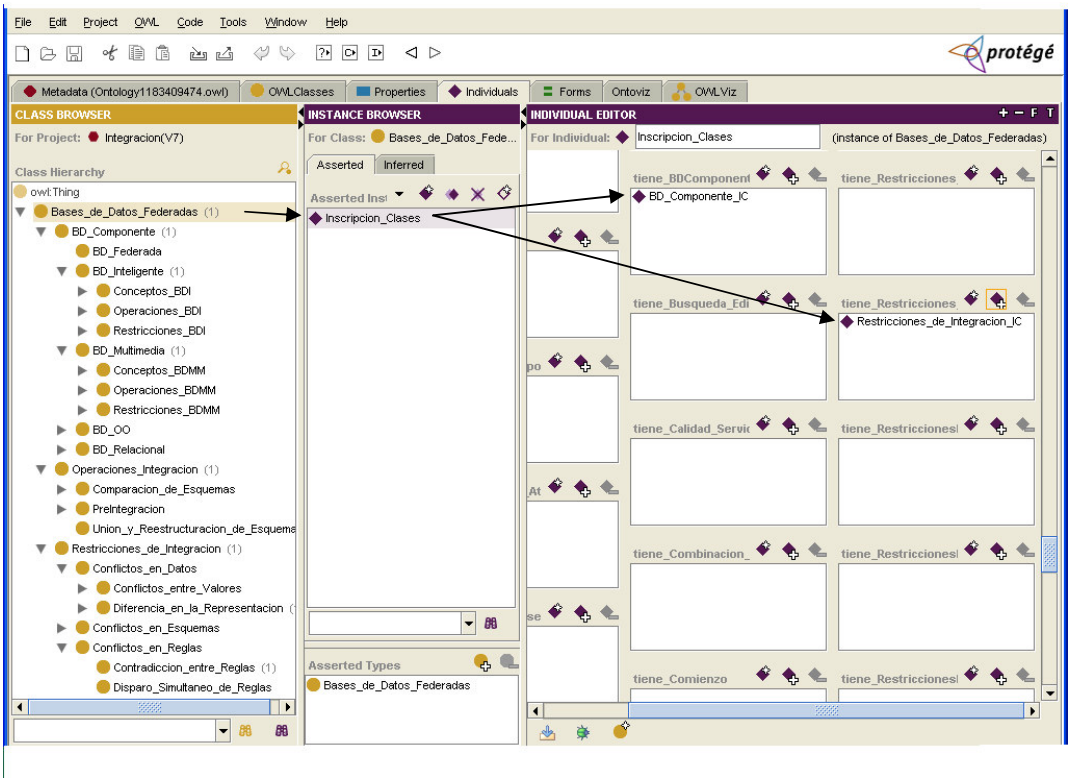

Figura 5.21 Instancias de las propiedades tiene_BDComponente y tiene_Restricciones_de_Integridad de la base de datos federada Inscripcion_Clase

En el Capitulo 4 se describe que la Base de Datos Federada está conformada por dos bases de datos: una multimedia y una inteligente. En la siguiente figura se muestra cómo ocurre ésta instanciación. En la definición **tiene_BDComponentes** se pueden observar las instancias bases de datos componentes de la federación, que es una base de datos inteligente y una base de datos multimedia. Esto se indica en la figura 5.22, con los círculos en rojo. Los recuadros en rojo no se llenan, pues son para las instancias bases de datos orientadas a objeto y bases de datos relacionales, que no son utilizadas en nuestro ejemplo.

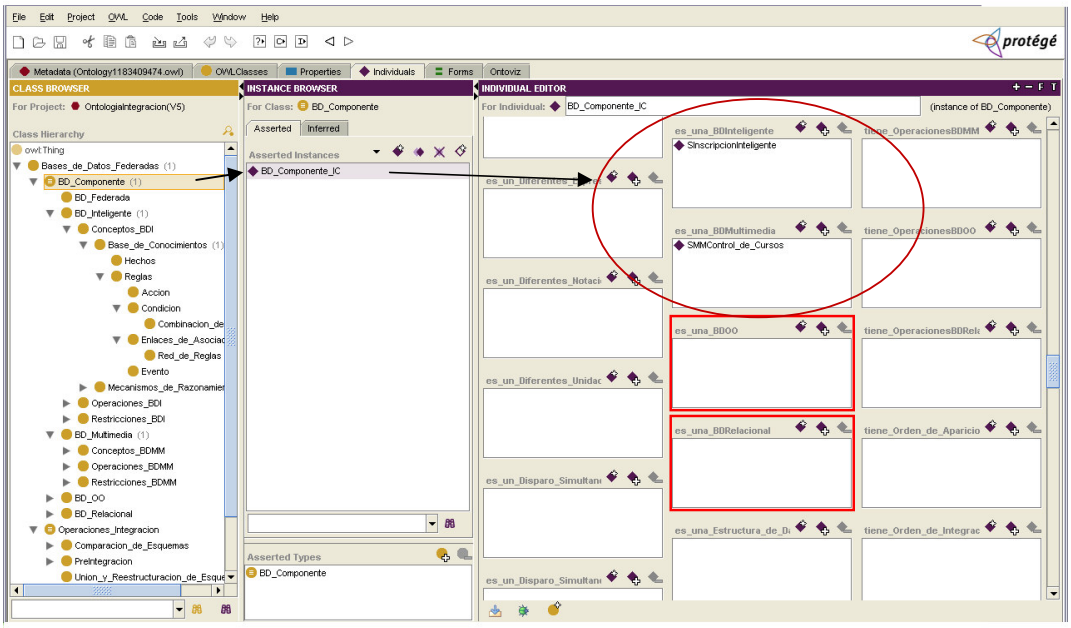

Figura 5.22. Instancias de la Clase BD_Componentes para la BD Federada Inscripcion_Clase

Cada una de las bases de datos componentes de la federación debe indicar qué esquemas van a compartir. Esto se realiza a través de las instancias, como se indica en el capítulo 4 en los axiomas de la tabla 24. En la figura 23, al instanciar la base de datos Inteligente SInscripciónInteligente, mostramos ese proceso. En esta figura se puede observar en la ventana del centro que contiene los esquemas que van a conformar la federación de la base de datos inteligente. Cuando se instancia la base de hechos, automáticamente se instancian los esquemas de la base de datos componentes debido a las propiedades y definiciones que se establecieron para la clase base de datos componentes. Esto se observa en el recuadro con la llamada que dice "se genera".

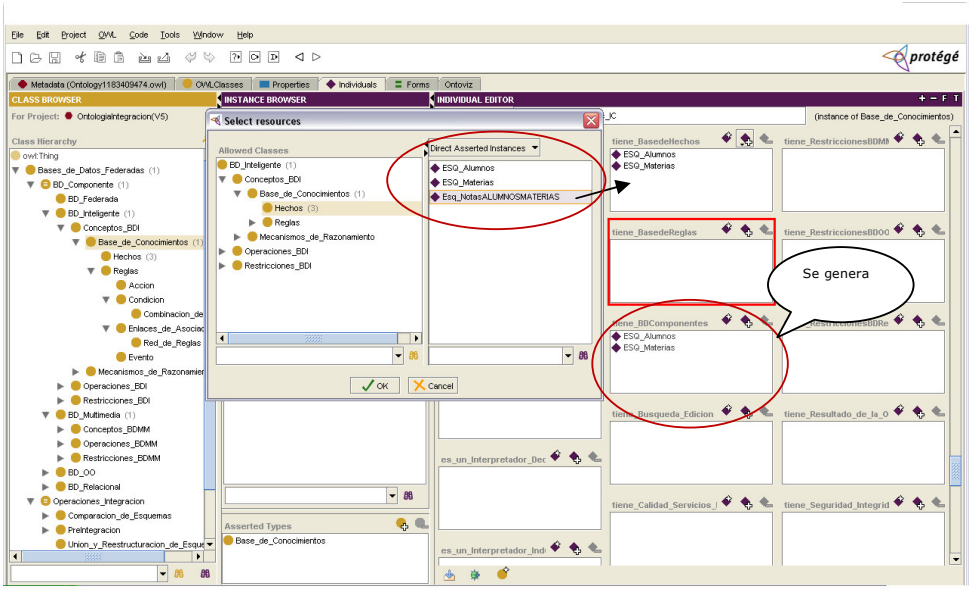

Figura 5.23. Instancia de la Base de Datos Inteligente Inscripción

A continuación, en la figura 5.24 se muestra la instancia de la Base de Conocimientos para la Base de Datos Inteligente SInscripcionInteligente. Estas instancias ocurren de acuerdo a las propiedades y restricciones establecidas para la clase base de datos inteligente.

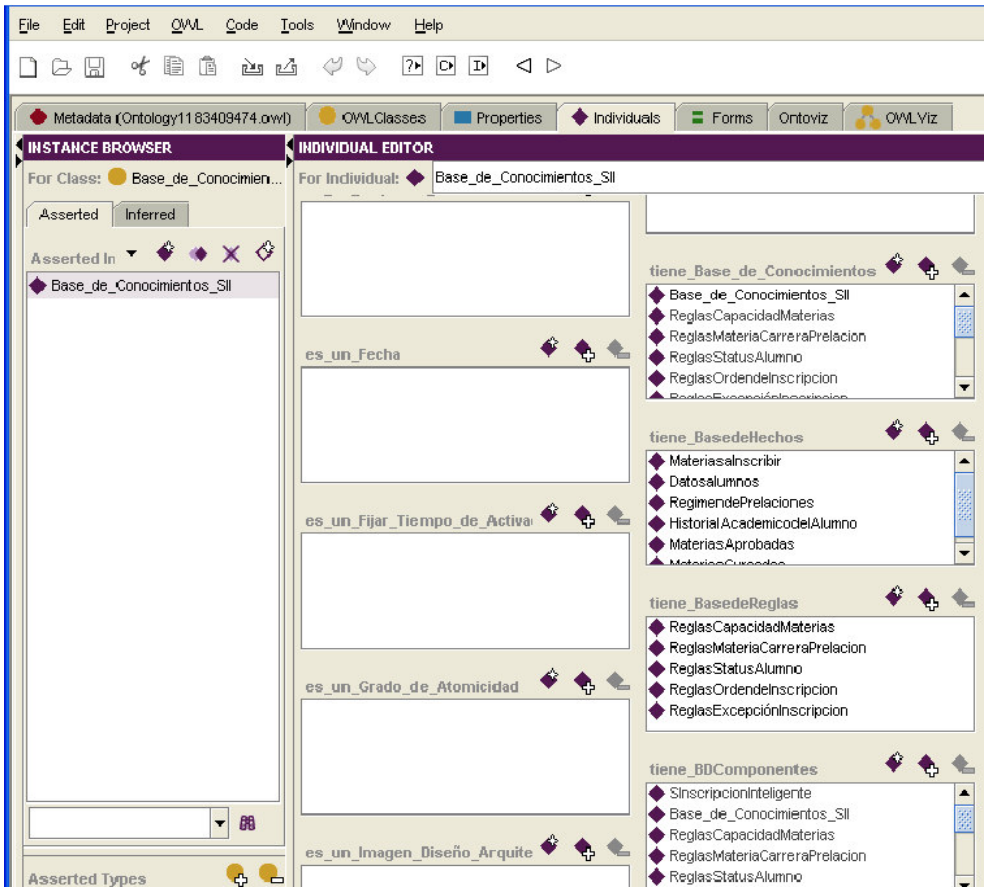

Figura 5.24. Instancia de la Base de Conocimientos de la Base de Datos Inteligente SInscripcionInteligente

Se muestra que esta base de datos inteligente tiene la instancia base de hechos y la instancia base de reglas. Esta base de Hechos está conformada por los esquemas: DatosAlumnos, HistorialAcademicodelAlumno, MateriasaInscribir, MAteriasAprobadas, MateriasCursadas, RegimendePrelaciones; y las reglas: ReglasCapacidadMaterias, ReglasExcepcionInscripcion, ReglasMateriaCarreraPrelacion, ReglasOrdendeInscripcion, ReglasStatusAlumno, tomadas del ejemplo del capítulo 4.

A continuación, en la figura 5.25 se muestra la instancia base de datos multimedia Control de Cursos.

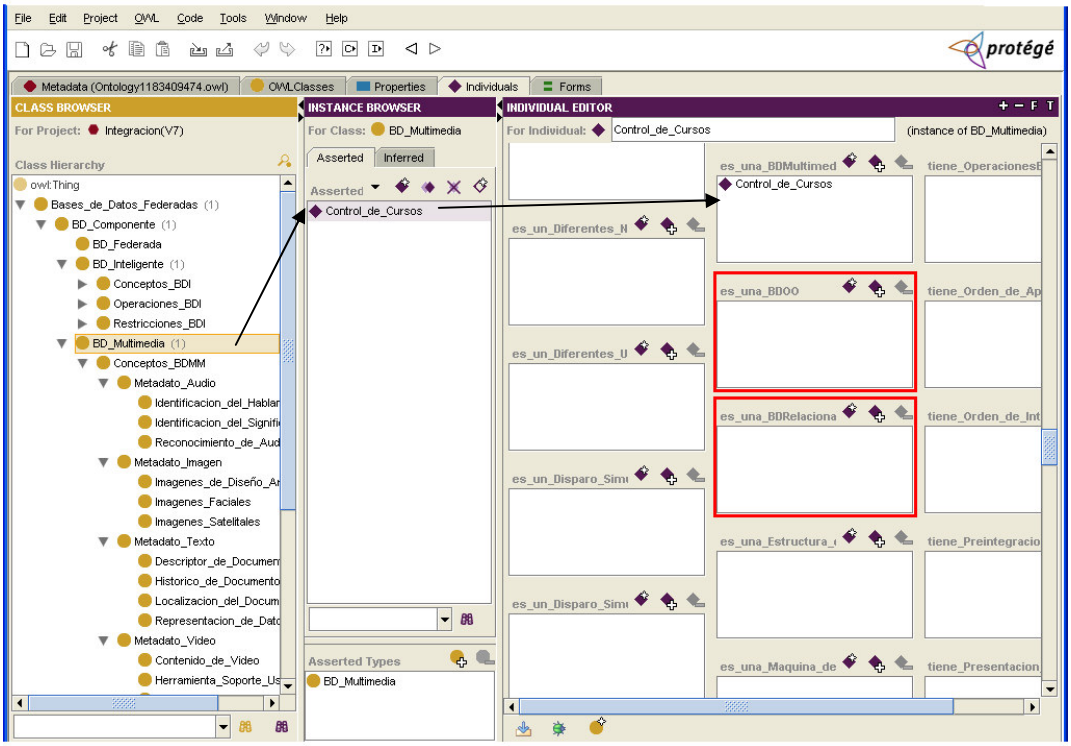

Figura 5.25. Instancia de la Base de Datos Multimedia llamada Control_de_Cursos

En la siguiente figura se muestra la instancia de los metadatos de audio, video, texto e imagen, que posee el Objeto de Aprendizaje Gestión de Conocimiento que se encuentra en la base de datos multimedia Control_de_Cursos.

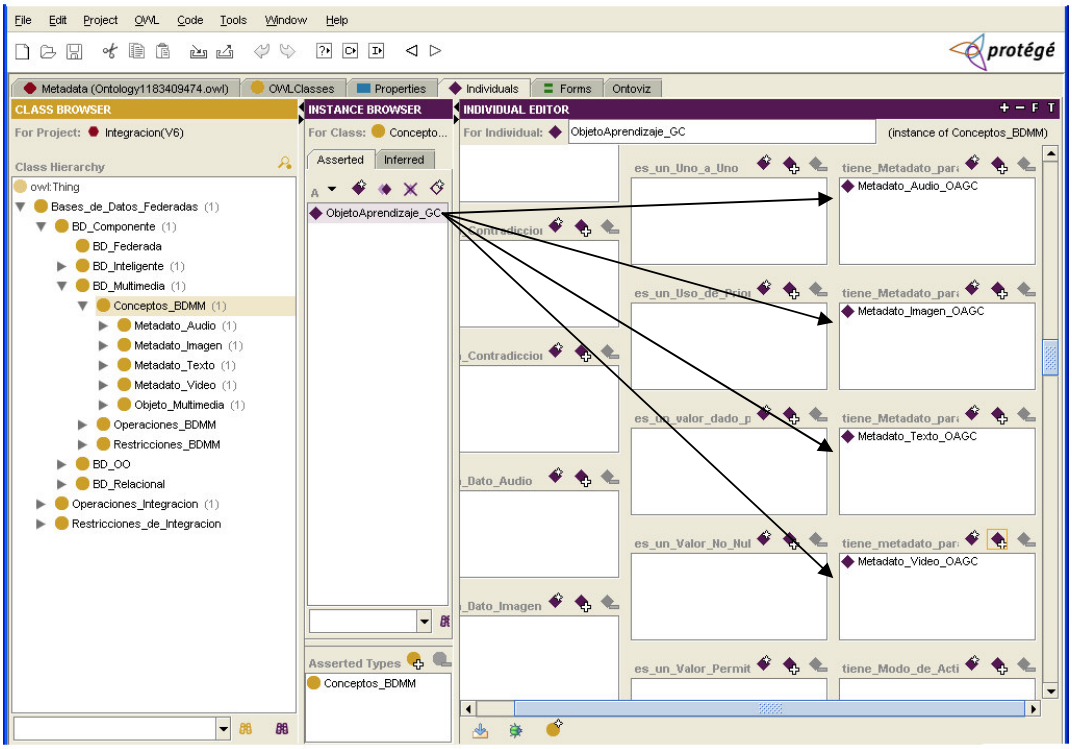

Figura 5.26. Instancia del Objeto de Aprendizaje Gestión de Conocimiento (OAGC) y las instancias de los metadatos que contiene

5.5 Validaciones adicionales

En la siguiente figura se muestra el resultado de correr el test de validación de la ontología. Este test revisa y rectifica si existen errores. La lista de validación muestra los parámetros que son verificados a través de las propiedades de OWL. El espacio blanco de fondo significa que no existen errores.

Las propiedades de completitud, concisión y consistencia se validan a través de los parámetros listados dentro del Sanity Test. Estos parámetros se listan a continuación:

- Las propiedades de Dominio y Rango validan completitud, concisión y consistencia, estas propiedades son las tres primeras y dos últimas dentro del Sanity Test.

  Domain of a property should not be empty
  Domain of a property should not contain redundant classes
  Domain of subproperty can only narrow superproperty
  Range of a property should not contain redundant classes
  Range of a property can only narrow superproperty

- Las propiedades de las relaciones entre clases (ObjectProperty y dataProperty) y sus características validan completitud y consistencia:

InverseOfFunctionalMustBeInverseFunctional
InverseOfInverseFunctionalMustBeFunctional
InverseOfSubpropertyMustBeSubpropertyOfInverseOfSuperproperty
InverseOfSymmetricPropertyMustBeSimmetrycProperty
InverseOfTopLevelPropertyMustBeTopLevelProperty
InverseOfTransitivePropertyMustBeTransitiveProperty
InversePropertyMustHaveMatchingRangeAndDomain

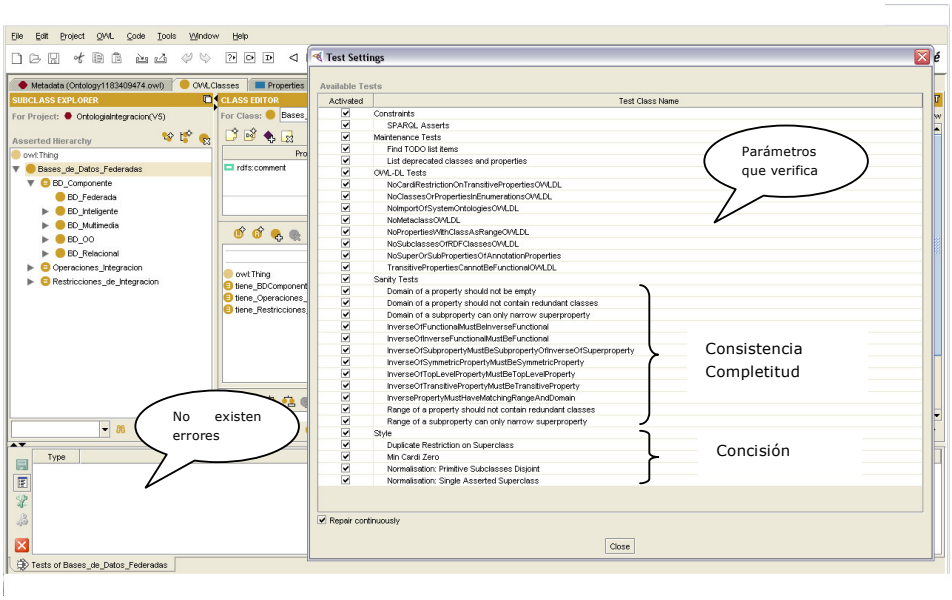

Figura 5.27. Test de validación de las propiedades de la ontología

En la siguiente figura se muestra el test de validación para el individuo Inscripcion_Clase. Las propiedades de Dominio y Rango para el individuo Inscripción_Clase validan completitud, concisión y consistencia, a través del Sanity Test de la misma manera que para la ontología. Las propiedades de las relaciones entre individuos (ObjectProperty y dataProperty) y sus características validan completitud y consistencia del individuo Inscripcion_Clases.

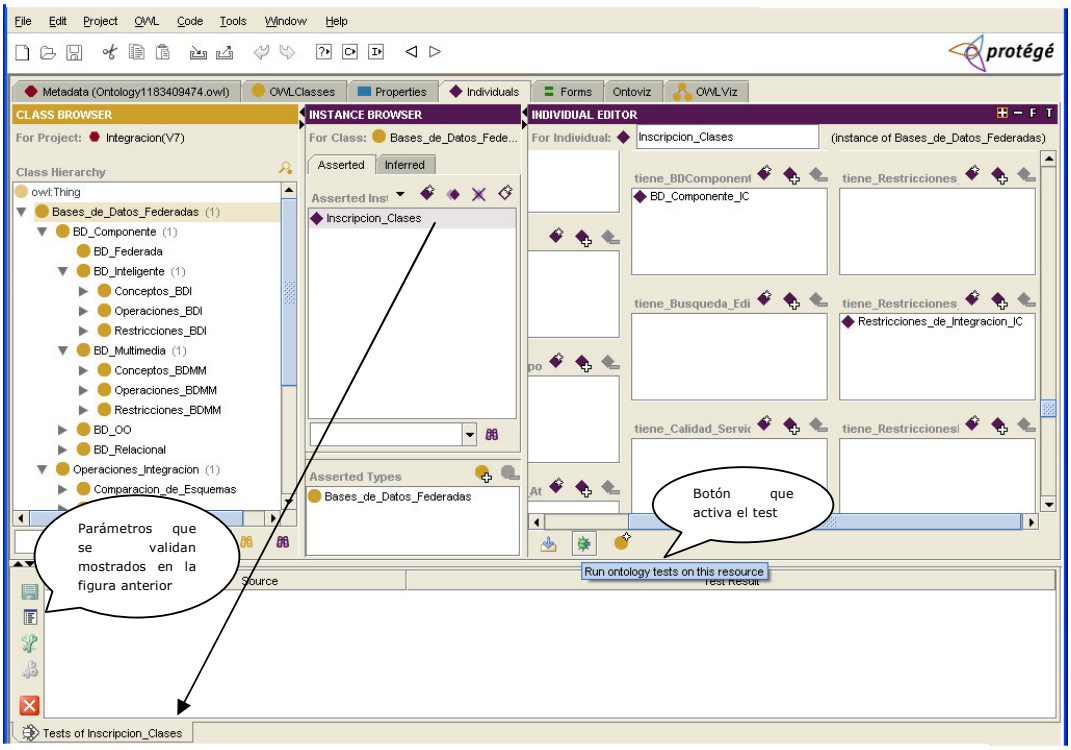

Figura 5.28. Test de ontología para el individuo Inscripcion_Clase

En la figura 5.29 se muestra una parte del código OWL para la ontología que genera la herramienta Protégé-OWL, indicando las clases, subclases, restricciones y propiedades de clases disjuntas, como se observa en la pantalla.

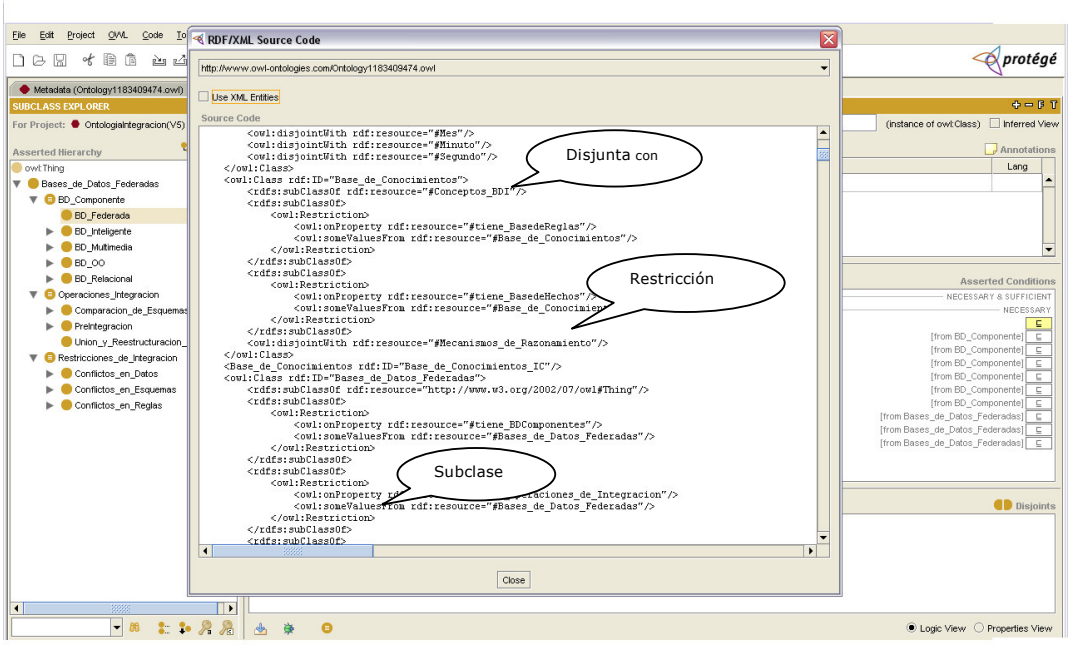

Figura 5.29. Código OWL generado por el editor Protégé-OWL para la ontología

Finalmente, en la figura 5.30 se visualiza la ontología con sus propiedades. Esta visualización es una página Web que puede funcionar como un diccionario o una página de búsquedas, en caso de que existan direcciones URL de las bases de datos que se utilizan como individuos. En este trabajo no se puede visualizar direcciones url, pues las bases de datos son un ejemplo no implementado.

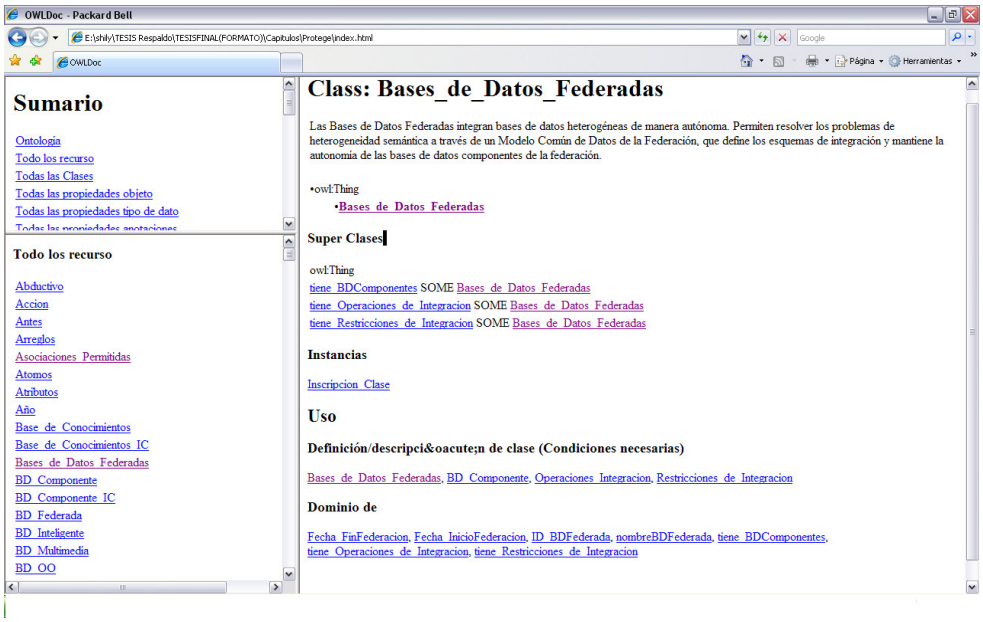

Figura 5.30. Ontología generada como una página Web

CONCLUSIONES

En este trabajo de tesis se consiguió modelar el proceso de integración de bases de datos heterogéneas utilizando ontologías. Para ello, se tomó como referencia la arquitectura de bases de datos federadas de Shet & Larson presentada en [Shet, A., & Larson, J. 1990]. El trabajo planteó la extensión de la arquitectura de Shet&Larson para bases de datos federadas, ya que dicha arquitectura sólo integra semánticamente bases de datos relacionales. Particularmente, los ajustes que se le hicieron a dicha arquitectura, que a su vez permitieron cubrir los objetivos planteados en la tesis, fueron:

- Extensión de la arquitectura de Bases de Datos Federadas de Shet&Larson para soportar la integración de diferentes tipos de Bases de Datos, como son las Bases de Datos Multimedia, Bases de Datos Orientadas a Objeto y Bases de Datos Inteligentes. Esto se lleva a cabo en nuestro modelo cuando se plantea que las bases de datos federadas tienen diferentes tipos de bases de datos componentes, y le agregamos operaciones y restricciones de integración que rigen el comportamiento y permiten considerar las características de dichas bases de datos componentes.
- Generación de un Modelo Ontológico[39] para el proceso de Integración de las Bases de Datos Federadas. Este modelo permite realizar la integración semántica de los datos que conforman las bases de datos federadas. Por la forma en que se hizo el modelo, nuevos tipos de bases de datos pueden ser incluidas.
- Generación de los Modelos Ontológicos para representar cada uno de los siguientes tipos de Bases de Datos: Bases de Datos Relacionales, Bases de Datos Multimedia, Bases de Datos Inteligentes, y Bases de Datos Orientados a Objeto. Estos modelos son necesarios para la federación de bases de datos heterogéneas vía modelos ontológicos. Estos modelos pueden ser extendidos por expertos en el área de bases de datos, para mejorar su calidad en cuanto a las descripciones y características de cada una de las bases de datos. Dichas extensiones pueden ser incluidas sin problemas en los modelos respectivos.

Así, en este trabajo se presentó un enfoque semántico para facilitar el proceso de integración de datos e interoperabilidad entre bases de datos, basado en ontologías. La ontología es utilizada como un modelo conceptual para la representación de los dominios particulares de cada tipo de base de datos, como del proceso de integración de bases de datos federadas. La integración resuelve los problemas de interoperabilidad que genera la heterogeneidad en las bases de datos en puntos como: modelo de datos, esquemas de datos, tipos de relaciones, atributos y datos.

[39] Un modelo ontológico comprende atributos, relaciones entre atributos, y axiomas

Por medio de las ontologías se hizo el control semántico del proceso de integración de las bases de datos, tomando en cuenta el significado de los elementos que conforman las bases de datos a integrar. Llamamos control semántico a la capacidad de nuestro sistema de resolver de manera inteligente los problemas semánticos (por ejemplo, nombres iguales para tablas diferentes o nombres diferentes para tablas iguales) que surgen en el proceso de integración de bases de datos heterogéneas. Utilizando la ontología se maneja el conocimiento existente, a través de reglas y condiciones, a tomar en cuenta durante el proceso de integración.

Un comentario a resaltar es que la interoperabilidad sintáctica se resuelve utilizando estándares como XML y RDF en la descripción de datos y metadatos del sistema, pero la interoperabilidad semántica depende de las necesidades particulares de cada dominio. Particularmente, nuestro modelo de integración para bases de datos heterogéneas a través de ontologías plantea una propuesta para resolver el problema de la interoperabilidad semántica, al resolver los conflictos de integración desde el punto de vista de esquemas y tipos de datos, para aquellas bases de datos que quieran trabajar juntas sin perder la autonomía en sus operaciones.

En general, el modelo ontológico permite mostrar el funcionamiento y el significado de los conceptos de las bases de datos que conforman una federación, así como del proceso de integración, a través de operaciones y restricciones. Además, con el modelo ontológico del proceso de integración para bases de datos federadas se pueden realizar búsquedas inteligentes, ya que se pueden realizar consultas entre sistemas diferentes de manera sencilla, sin que los sistemas tengan que modificar la representación interna de su información. Específicamente, las consultas podrían definirse de manera universal, sin que los sistemas tuvieran que modificar la representación interna de su información, con la posibilidad de enriquecer semánticamente el proceso de búsqueda federada.

Uno de los motivos básicos por los que la ontología ha cobrado tanta importancia y se ha extendido su uso es por su *reusabilidad* (esto es, una misma ontología se puede reutilizar en diversas aplicaciones de forma individual, o combinada con otras), y la posibilidad de *compartir conocimiento* (esto es, el conocimiento que ella describe es compartido por una determinada comunidad). Estos dos aspectos son cubiertos por nuestra propuesta.

El modelo se probó en un caso de estudio que tenía tres partes: el modelado de una base de datos inteligente y una base de datos multimedia usando nuestros modelos respectivos, y el modelado del proceso de integración. Esa prueba se hizo en dos fases: primero se instancio en la descripción en lógica de predicado de primer orden de nuestro modelo ontológico, y después se implementó a traves de la herramienta Protégé-OWL.

La herramienta Protégé-OWL permite validar la consistencia, completitud y concisión de nuestra ontología. Esto es debido a que dicha herramienta, al estar basada en OWL, utiliza un modelo lógico que permite definir y describir los conceptos, y las relaciones que se dan entre ellos, chequeando automáticamente si las mismas cumplen con esos tres criterios de validación (por ejemplo, si se mantiene la jerarquía de conceptos correctamente). En la creación de la ontología con la herramienta Protégé-OWL se presentaron limitaciones en cuanto a las relaciones entre conceptos (o clases, como se llaman en la herramienta) que Protégé-OWL permite representar y validar, ya que ella sólo permite relaciones entre clases del tipo: "is-a", "part_of" y "has", por lo que algunas de las relaciones definidas en nuestra ontología, aún cuando se describen en la herramienta, no pudieron ser validadas por Protégé-OWL.

Nuestra propuesta ontológica para conseguir la integración requiere de elementos tecnológicos adicionales. Estos elementos son una *máquina de aprendizaje* que permiten aprender de la dinámica operacional para ir enriqueciendo el modelo ontológico, y un *mecanismo de razonamiento* para ser utilizado en procesos de inferencia, como se muestra en el capitulo 3 en la figura 3.1. Además, herramientas de *minería de datos* serían de gran utilidad para tareas de mantenimiento y consulta a la federación. Esta ontología, con todas estas características, actuaría como un elemento inteligente de integración.

Algunos posibles proyectos futuros, además de lo comentado en el párrafo anterior, son:

- Diseñar el Sistema Multiagente de nuestro modelo de integración, que realice el proceso de búsquedas e integración de diferentes bases de datos. De los Sistemas Multiagente podríamos explotar sus características distribuidas y autonómicas.
- Construir APIs que permitan a un usuario no experto integrar las bases de datos de manera sencilla. Por ejemplo, en el área de Bioinformática existen muchas bases de datos que poseen información de genes. Esta aplicación se colocaría encima de las ontologías de las bases de datos, le solicitaría al usuario la información que requiere consultar, identificaría los tipos de bases de datos a integrar, para luego realizar el proceso de integración usando nuestro modelo.
- Diseñar una Base de Conocimiento que permita registrar las experiencias y errores que surjan durante el proceso de integración, y aprender a la máquina de aprendizaje. Cuando la ontología realice el proceso de integración, debe quedar registrado cada experiencia del proceso de integración. Un ejemplo de conocimientos a registrar serian los conflictos resueltos y no resueltos.

REFERENCIAS

Aberer, K. 2003; "*Special issue on peer to peer data management"*. SIGMOD Record 32(3), pp. 29-33

Aditya Kalyanpur, Bijan Parsia, Evren Sirin, Bernardo Cuenca-Grau, James Hendler 2005; "*Swoop: A 'Web' Ontology Editing Browser*". Journal of Web Semantics 4(2), pp. 144-153

Álvarez F., Lavariega J.C. 2004, "*Una Arquitectura Basada en Componentes de Software de un Sistema de Bases de Datos Distribuidas con una Estructura Cliente-Servidor de 3 Capas".* Memorias del XXXIV Congreso de Investigación y Extensión del Sistema Tecnológico de Monterrey. ACM International Conference Proceeding Series, pp. 501–506.

Arpirez JC, Corcho O, Fernandez-López M, Gómez-Pérez A, 2003, "*WebODE in a scalable ontological engineering workbench".* ACM press, pp. 6-13

Batini, C.; Lenzerini, M.; and Navathe, S. 1986; "*A comparative analysis of methodologies for database schema integration"*. ACM Computing Survey 18(4), pp. 323-364.

Becarevié D., Roantree M. 2004; "A Metadata Approach to Multimedia Database federations", Information and Software Technology 46(3), pp. 195-207

Bechhofer S., Carr L., Goble C., Hall W., 2002; *Conceptual Open Hypermedia = The semantic Web?.* Articulo presentado en The Second International Workshop on the Semantic Web, pp. 44—50.

Bechhofer S., Horrocks I., Goble C., Stevens R., 2001; "*OilEd: a reasonable ontology beditor for the semantic Web"*. Lecture Notes in Artificial intelligence LNAI 2174, Springer Verlag, pp. 396-408

Bernstein, P. 2003; "*Applying model management to classical meta data problems"*. In Proceedings of the Conf. on Innovative Database Research (CIDR), pp. 209-220

Bertino E., Catania B., Zarri Gian P. 2001; Intelligent Database System, Addison-Wesley

Brickley D., Guha RV. 2003; "*RDF Vocabulary Descrption Language 1.0: RDF Schema"*, W3C Working Draft. http:// www.w3.org/TR/PR-rdf-schema

Cattell R., Barry D. 2000; The Object Data Standard: ODMG 3.0; Morgan Kaufmann Publishers,

Chaudri VK, Farqhuar A., Fikes R., Karp PD., Rice JP., 1998; "*Open Knowledge Base Connectivity 2.0.3".* Technical report http://www.ai.sri.com/0kbc/okbc-2-0-3.pdf

Collins S, Shamkant N., Mark Leo 2002; "*XML schema mappings for heterogeneous database access*", Information and Software Technology 44(4), pp. 251-257.

Corcho Oscar 2005; A Layered Declarative Approach to Ontology Translation with Knowledge Preservation. IOS Press

Corcho O., Fernandez-López M., Gomez-Perez A. 2003; "*Methodologies, tools and languages for building ontologies. Where is their meeting point?*". Data & Knowledge Engineering (46), pp. 41-64.

Date C. J. 2005; Database in depth: relational theory for practitioners. O' REILLY.

Date C. J, H Darwen, 2007; "Databases, Types, and the Relational Model: The Third Manifesto", Addison-Wesley.

Dean M., Schreiber G., 2003; "OWL *Web Ontology Language Reference*". W3C Working Draft. http://www.w3.org/TR/OWL-REF/

Dejing Dou, Paea LePendu., 2006; "*Ontology-based Integration for Relational Database*". Memorias 2006 ACM symposium on applied computing, pp. 461-466.

Domingue J. 1998; "*Tadzebao and WebOnto: Discussing, Browsing and editing Ontologies on the Web*". Memoria Workshop on Knowledge Acquisition, Modeling and Management KM4 pp 1-20

Eberhart A 2004; **"***Ontology-based Infrastructure for Intelligent Applications*", CiteSeer.ISTcientific Literature Digital Library

Elmagarmid, A., and Pu, C. 1990; "*Introduction to the special issue on heterogeneous databases*". ACM Computing Survey 22(3), pp. 175-178.

Rahm, E., and Bernstein, P. 2001; "*On matching schemas automatically*". VLDB Journal 10 (4). pp. 22.

Farquhar A, Fikes R, Rice J., 1997; "*The Ontolingua Server: A tool for collaborative Ontology Construction*", International Journal of Human Computers Studies 46(6) pp. 707-727.

Eun-Hye Choi, Tatsuhiro Tsuchiya and Tohru Kikuno., 2006; "*Model Checking Active Database Rules under Various Rule Processing Strategies*", IPSJ Digital Courier (2), pp. 826-839

Fernandez L, Gomez-Perez A., Jurista N., 1997; "*METHONTOLOGY: From Ontological Arts Towards Ontological Enngineering*", Symposium on Ontological Engineering of AAAI Stanford University, pp. 33-40

Fridman N., Hafner C. 1997; "*The state of art in ontology design. A survey and comparative review*". American Association for Artificial Intelligence AI Magazine, Fall, pp. 53–74.

García F. M. 2005, "Estrategia para integrar bases de datos basada en el registro y extracción de correspondencias semánticas", Revista Transferencia Año 18 Nro 69.

Garcia-Molina, H.; Papakonstantinou, Y.; Quass, D.; Rajaraman, A.; Sagiv, Y.; Ullman, J.; and Widom, J., 1997; "*The TSIMMIS project: Integration of heterogeneous information sources"*. J. System Integration 7 (3-4), pp. 381-407.

Genesereth MR, Fikes RE (1992); "*Knowledge Interchange Format. Versión 3.0 Reference Manual*". Technical report Logic 92-1. Computer Science Department. Stanford University.

Gómez-Pérez A., Fernández-López M., Corcho O., 2004; Ontological Engineering. Springer-Verlag.

Gray, J., 2004; "*The next database revolution"*. Proceedings of the 2004 ACM SIGMOD international conference on Management of data.

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

Grüninger M., Fox MS., 1995; "*Methodology for the design and evaluation of ontologies*", Skuce D (ed) IJCAI95 Workoshop on Basic Ontological issues in Knowledge Sharing, http:\\isye.gatech.edu

Huimin Zhao, Sudha Ram., 2005; "*Entity identification for heterogeneous database integration a multiple classifier system approach and empirical evaluation*". Information Systems 30, pp. 119-132.

Horrocks I., van Harmelen F., 2001; "*Reference Description of the DAML+OIL". Ontology Markup Language*. Technical Report. http://www.daml.org/

Ives, Z.; Florescu, D.; Friedman, M.; Levy, A. and Weld, D. 1999; "*An adaptive query execution system for data integration*". ACM SIGMOD Record 28(2), pp. 299-310.

Kactus 1996; "*The KACTUS booklet version 1.0*". Esprit project 8145 KACTUS http://www.swi.psy.uva.nl/projects/NewKACTUS/Reports.html

Kholer J, Stephan Philippi and Matthias Lange., 2003; "*SEMEDA: ontology based integration of biological databases*"; Bioinformatics 19(18) pp. 2420-2427

Kogut P., Holmes W., 2001; "*AeroDAML: Appliying Information Extraction to Generate DAML Annotation from Web Pages".* First International Conf. on Knowledge Capture (K-CAP'01).

Kifer M., Lausen G., Wu J., 1995; "*Logical foundations of Object-Oriented and Framed-Based Languages*". Journal of the ACM 42(4) pp. 741-843

Knoblock, C.; Minton, S.; Ambite, J.; Ashish, N.; Modi, P.; Muslea, I.; Philpot, A. and Tejada, S., 1998; "*Modeling web sources for information integration"*. In Proc. of the National Conference on Artificial Intelligence (AAAI), pp 211-218.

Heflin JD, Hendler JA., 2001; "*A portrait of the Semantic Web in Action*". IEEE Intelligent Systems 16(2), pp. 54-59.

Holger Knublauch, Ray W. Fergerson, Natalya F. Noy and Mark A. Musen., 2004; "*The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications*", Third International Semantic Web Conference, pp 229-243.

Karp PD, Chaudri V., Thomere J., 1999; "*XOL: An XML-Based ontology Exchange Language*". Version 0.3 Technical Report. http://www.ai.sri.com/pkarp/xol/xol.html

Kosch H. and M. Döller., 2005; "*Multimedia database systems: Where are we now?".* IASTED DBA Konferenz, Innsbruck, Österreich. Disponible en: www.itec.uni-klu.ac.at/˜harald/MMDBoverview.pdf

Kosh Harold., 2004; Distributed Multimedia DB Technologies Supported by MPEG-7 and MPEG-21. CRC Press

Lambrecht, E.; Kambhampati, S. and Gnanaprakasam, S., 1999; "*Optimizing recursive information gathering plans*". Journal of Intelligent Information Systems 22(2), pp. 119-163.

Lassila O, McGuinness D., 2001; "*The Role of Frame-Based Representation on the Semantic Web*". Technical report KSL-01-02. Knowledge System Laboratory Stanford University.

Lassila O., Swick R., 1999; "*Resource Description Framework (RDF) Model and Syntax Specification*" W3C Recommendation. http://www.w3.org/TR/REC-rdf-syntax/

Lenat DB, Guha RV., 1990; Building Large Knowledge-based System: Representation and Inference in the Cyc Project. Addison-Wesley, Boston, Masacchusets.

Levy, A. Y.; Rajaraman, A. and Ordille, J., 1996; "*Querying heterogeneous information sources using source descriptions*". In Proc. of VLDB´96, pp. 251-262.

Linthincum D., 2000; Enterprise Application Integration. Addison-Wesley Information Technology Series, United States, 3rd edition.

MacGregor R., 1991; "*Retrospective on LOOM*", Technical Report Information Science Institute. University of Southern California. http://www.isi.edu/isd/LOOM/papers/mac_gregor/Loom_Retrospective.html

Luke S., Heflin JD., 2000; "*SHOE 1.01 Proposed Specification, Technical Report*". Parallel Understanding Systems Group. Department of Computer Science. University of Maryland. http://www.cs.umd.edu/projects/plus/SHOE/spec1.01.htm

Maedche A., Motik B., Stojanovic L., Studer R., Volz R., 2003; "*Ontologies for Enterprise Knowledge Management"*. IEEE Intelligent Systems 18(2), pp. 26-33

Mahesh K., 1996; "*Ontology development for machine translation: Ideology and Methodology*". Technical report MCCS-96-292. Computing Research Laboratory New Mexico State University. http://citeseer.nj.nec.com/mahesh96ontology.html

Miller, R.; Haas, L. and Hernandez, M., 2000; "*Schema mapping as query discovery*". In Proc. of VLDB´2000, pp. 77-88.

Mota H. Laura., 2002; Generación de reglas restauradoras de la consistencia en esquemas relacionales con vistas. Tesis de Doctorado en Informática, Universidad Politécnica de Valencia España.

Motta E., 1999; Reusable Components for Knowledge Modeling: Principles and Case Studies in Parametric Design. IOS Press, Amsterdam.

Muñoz A., Aguilar J., 2007; "*Ontological Model for Intelligent Database*", Computers Science and Technology, Proc. 11th WSEAS International Conference on Computers, pp. 1-6.

Muñoz A, Aguilar J. 2007;" Esquema Ontológico para Bases de Datos Multimedia", Sometido a revisión en Revista Ciencia Ingeniería.

Muñoz A, Aguilar J., Martinez R., 2006; "*Integration Ontology for Distributed Database*". *Advanced Software Engineering: Expanding the Frontiers of Software Technology (Ed. S.Ochoa, G. Roman),* IFIP, Springer, pp. 85-93

Muñoz A, Aguilar J., Martínez R., 2005, "*Modelo Inteligente para Bases de Datos Distribuidas*". *Revista Gerencia Tecnológica Informática* 4(10), pp. 91-116.

Muñoz A., Aguilar J., 2005a; "*Architecture for Distributed Intelligent Databases*". *IEEE, 13th Euromicro Conference on Parallel, Distributed and Network-based Processing*, Euromicro-PDP 2005, pp. 322-328.

Neches R., Fikes RE, Finin T., Gruber TR, Senator t, Swartout WR., 1991; "*Enabling technology for knowledge sharing*". AI Magazine 12(3), pp. 36-56.

Ouksel, A., and Seth, A. P., 1999; "*Special issue on semantic interoperability in global information systems*". ACM SIGMOD Record 28(1), pp. 5-12.

Parent, C. and Spaccapietra, S., 1998; "*Issues and approaches of database integration*". Communications of the ACM 41(5), pp. 166-178.

Paton, N. W., 1998. Active rules in databases systems. Springer.

Piatinni M., 2000; "*Líneas de Evolución de las Bases de Datos*". Novatica 145, pp. 100 -102.

Pottinger, R. A., and Bernstein, P. A., 2003; "*Merging models based on given correspondences*". Very Large Data Bases(Vldb) Series Proceedings of the 29th international conference on Very large data bases - Volume 29, pp 862-873

Prabhakaran B., 1997; Multimedia Database Management Systems. Kluwer Academic Publisher Boston/London/Dordrecht.

Rahm, E. and Bernstein P., 2001; "*A survey of approaches to automatic schema matching*". VLDB Journal 10(4), pp. 334-350.

Reeve L., Hyoil Han., 2005; "*Survey of semantic annotation platforms*". Proceedings of the 2005 ACM symposium on Applied computing, pp. 1634 - 1638

Sharma Chakravarty, 2005; Fundamentals of Active Database Systems, Morgan Kaufmann Publishers Inc.

Shet A. P., 1998; "C*hanging Focus on Interoperability in Information Systems: from Systems, Syntax, Structure to Semantics*". Interoperating Geographic Information Systems. M.F. Goodchild, M. J. Egenhofer, R. Fegeas and C.A. Kottman (eds) Kluwer.

Shet, A., & Larson, J., 1990; "*Federated database systems for managing distributed, heterogeneous, and autonomous databases*". ACM Computing Survey 22(3) pp. 183-236.

Sowa JF., 1999; Knowledge Representation: Lógical, Philosophical and Computational Foundations. Brooks Cole Publishing Co., Pacific Grove, California.

Staab, S., Maedche, A. and Handschuh, S., 2001; "*An Annotation Framework for the Semantic Web*". Proc. 1 Int. Workshop on MultiMedia Annotaion, Tokyo.

Stevens R, Baker P, Bechhofer S, Ng G, Jacoby A, Paton NW, Goble CA, Brass A 2001; "*Transparent Access to Multiple Bioinformatics Informations Source (TAMBIS)*". Bioinformatics 16(2), pp. 184-5

Swartout W., Ramesh P., Knigth K., Russ T., 1997; "*Toward Distributed Use of Large-Scale Ontologies*". AAAI Springer symposium on Ontological Engineering, Stanford University, pp138-148

Uschold M., King M., 1995; "*Towards a Methodology for Building Ontologies*", Skuce D (eds) IJCAI'95 Workshop on Basic Ontological Issues in Knowledge Sharing, pp 6-10

Van Heijst et al., 1996; "*Using Explicit Ontologies in KBS Development*" International Journal of Human and Computer Studies. 46(2-3) pp. 183-292, Elsevier.

Wache H., Vögele T., Visser U., Stuckenschmidt H., Schuster G., Neumann H. and Hübner S., 2001; "*Ontology-Based Integration of Information A Survey of Existing Approaches*". Intelligent Systems Group, Center for Computing Technologies, University of Bremen Germany. Workshop: Ontologies and Information Sharing

Weigand, H**.,** 1997; "Multilingual *Ontology-Based Lexicon for News Filtering –The TREVI Project*". En K. Mahesh, pp. 138-159.

W3C, 2004; *Web Services Architecture,* W3C URL: http://www.w3.org/TR/2004/NOTE-wsarch-20040211/

ANEXO

PROTÉGÉ-OWL

A.1 Ontologías OWL

Las ontologías OWL son utilizadas para capturar conocimiento en un dominio de interés. Una ontología describe los conceptos en el dominio así como también las relaciones entre estos conceptos. Diferentes lenguajes ontológicos proporcionan diferentes facilidades. El desarrollo más reciente en estándares de lenguajes ontológicos es el OWL del World Wide Web Consortium (W3C). El Protégé OWL permite describir conceptos y además proporciona nuevas facilidades, posee un conjunto adicional de operadores (ej. or y negación). Está basado en diferentes modelos lógicos lo que permite definir y describir los conceptos. Se pueden construir conceptos complejos a partir de la definición de conceptos simples. Además el modelo lógico permite el uso de razonadores que verifica cuándo las instrucciones y definiciones de la ontología son mutuamente consistentes y puede también reconocer cuáles conceptos se ajustan a las definiciones. El razonador puede ayudar a mantener la jerarquía correctamente y es utilizado en los casos en que las clases tienen más de un padre.

A.2 Las tres especies de OWL

Las ontologías OWL están categorizadas en tres especies o sub-lenguajes: OWL-Lite, OWL-DL y OWL-Full. La característica que define a cada uno de ellos es su expresividad; OWL-Lite es el sub-lenguaje menos expresivo y OWL-Full es el más expresivo. La expresividad de OWL-DL se encuentra entre el OWL-Lite y el OWL-Full.

OWL-Lite es un sub-lenguaje sintácticamente simple, es utilizado en situaciones donde se requiera jerarquía simple de clases y solo sean necesarias restricciones simples. Por ejemplo se considera que OWL-Lite proporciona una vía rápida de migración para tesauros y otras jerarquías simples.

OWL-DL es mucho más expresivo que OWL-Lite y está basado en *Descripciones Lógicas*. La descripción lógica es un fragmento de Lógica de Primer Orden lo que permite razonamiento automático. Por lo tanto es posible calcular automáticamente la jerarquía de clasificación y chequear las inconsistencias en una ontología OWL-DLK.

OWL-Full es el más expresivo sub-lenguaje OWL. Es utilizado en situaciones donde es más necesaria la alta expresividad que la completitud del lenguaje. Por lo tanto OWL-Full no realiza razonamiento automático

A.3 Componentes de las Ontologías OWL

La ontología OWL tiene componentes similares a las ontologías basadas en Protégé Frames, sin embargo la terminología utilizada para describir estos componentes es un poco diferente. Una ontología OWL consiste de Individuos, Propiedades y Clases, lo cual puede corresponder a Protégé Frames Instancias, Slots y Clases.

Los ***Individuos*** representan objetos en el dominio de interés. Una diferencia importante entre Protégé y OWL es que OWL no usa Asunción de Nombre Unico (UNA, por sus siglas en inglés, Unique Name Assumption). Esto significa que para dos nombres diferentes se puede referir a un mismo individuo. Por ejemplo, "Queen Elizabeth", "The Queen" y "Elizabeth Windsor" se refieren todas al mismo individuo. En OWL se puede establecer explícitamente que los individuos son los mismos o que son diferentes. LA figura A1 muestra una representación de individuos en un dominio (aquí se representan los individuos como un rombo). Los individuos también son llamados instancias, pueden ser referenciados como "instancias de clases".

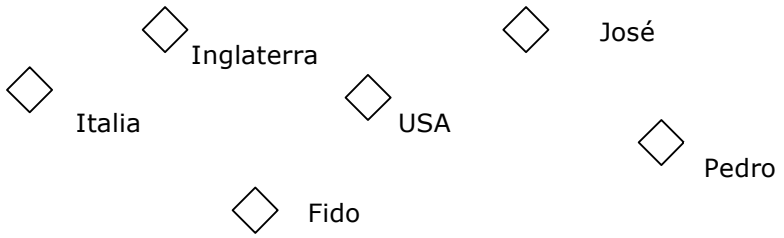

Figura A1. Representación de Individuos

Las ***propiedades*** son relaciones binarias entre individuos, es decir las propiedades enlazan dos individuos. Por ejemplo la propiedad **tieneHermano** puede enlazar los individuos **José** con el individuo **Nancy**, o la propiedad **tieneHijo** puede enlazar el individuo **Silvia** con el individuo **José**. Las propiedades pueden tener inversa. Por ejemplo la inversa de **tieneDueño** es **esDueñode**. Las propiedades se pueden limitar a tener un valor simple, esto es son *funcionales*. También pueden ser *Transitivas* o *Simétricas*. Las propiedades son conocidas como roles en descripción lógica y relaciones en UML y otras nociones orientadas a objeto. En la figura A2 se muestra una representación de algunas de las propiedades.

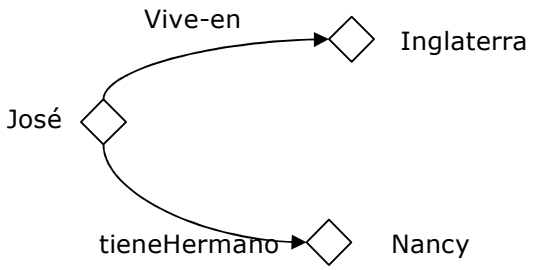

Figura A2. Representación de Propiedades

Propiedades OWL

Las propiedades OWL representan relaciones entre dos individuos. Hay dos tipos de propiedades *Propiedades Object* y *Propiedades Datatype*. Las propiedades Object enlazan individuos con individuo. Las propiedades Datatype enlazan un individuo con un XML Schema Datatype (vocabulario XML, elementos y atributos XML). La propiedad Annotation se utiliza para agregar información (metadatos) a las clases, individuos y las propiedades object/datatype

En la figura A3 se muestra un ejemplo de estas propiedades.

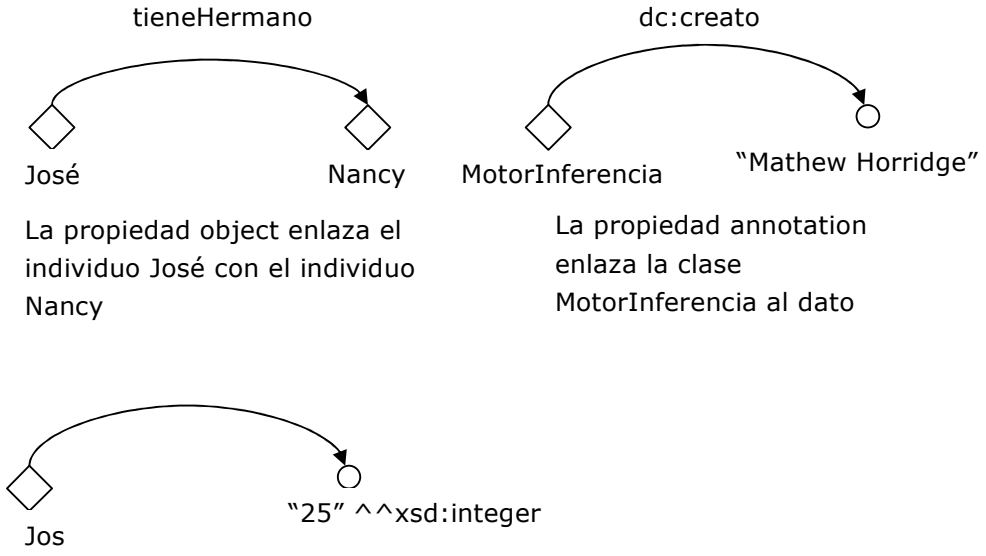

Figura A3. Propiedades Object, Datatype y Annotation

Propiedad Inversa

Toda propiedad Object puede tener propiedad inversa. Si una propiedad enlaza el individuo **a** con el individuo **b** entonces la propiedad inversa enlaza el individuo **b** con el individuo **a**. Por ejemplo, la figura 4 muestra la propiedad

tienePadre y su propiedad inversa tieneHijo (Si **Carlos tienePadre Luis** entonces por la propiedad inversa se puede inferir que **Luis tieneHijo Carlos**).

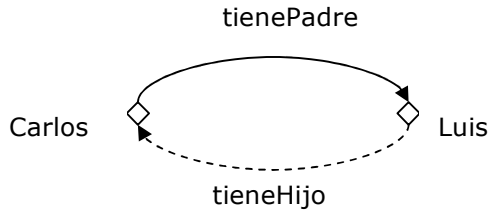

Figura A4. Un ejemplo de propiedad inversa: tieneHijo es la propiedad inversa de tienePadre

Características de las Propiedades OWL.

OWL enriquece el significado de las propiedades a través del uso de las "*características de las propiedades*". Estas son:

Propiedad Funcional. Una propiedad es funcional cuando para un individuo dado existe más de un individuo que está relacionado a él a través de la propiedad. Esta propiedad es llamada también *valuada simple* y también *características*.

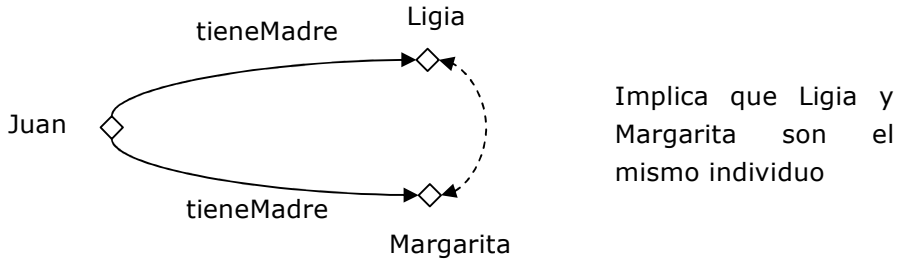

Figura A5. Un ejemplo de propiedad Funcional: tieneMadre

Propiedad Funcional Inversa. Una propiedad es funcional inversa si la propiedad inversa es funcional. En la figura A6 se muestra un ejemplo de esta.

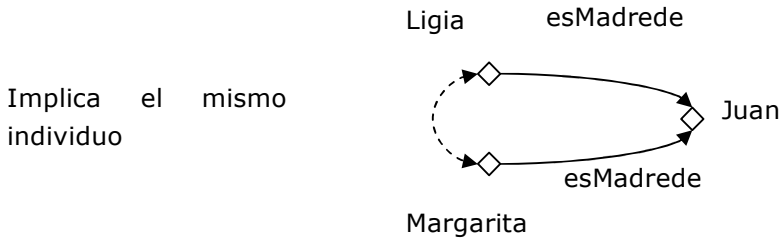

Figura A6. Ejemplo de propiedad inversa Funcional: esMadrede

Propiedad Transitiva. Una propiedad es transitiva cuando una propiedad P que relaciona al individuo **a** con el individuo **b** y el individuo **b** se relaciona con el

individuo **c** y se puede inferir que el individuo **a** se relaciona con el individuo **c** a través de esta propiedad **P**. Por ejemplo la propiedad transitiva **tieneAncestro**, Si el individuo Marcos tiene un ancestro que es Luís y Luís tiene un ancestro que es José, entonces se puede inferir que Marco tiene un ancestro que es José, esto se indica con líneas punteadas.

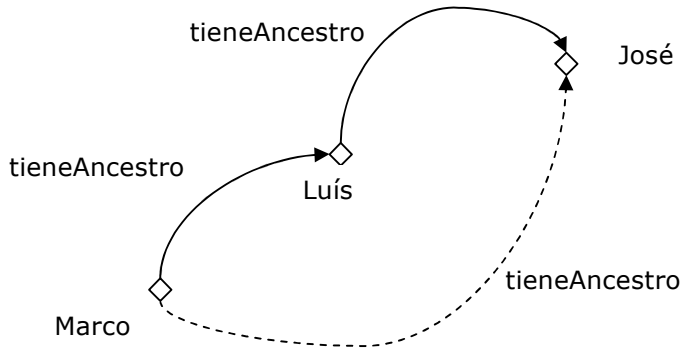

Figura A7. Ejemplo de propiedad transitiva: tieneAncestro

Propiedad Simétrica. Una propiedad P es simétrica cuando relaciona al individuo a con el individuo b y a su vez al individuo b con el individuo a. En la siguiente figura se muestra un ejemplo de la propiedad simétrica **tieneHermano**.

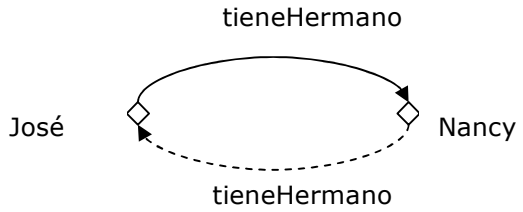

Figura A8. Un ejemplo de Propiedad Simétrica: tieneHermano

Propiedades de Rango y Dominio. Las propiedades pueden tener un dominio y un rango específico. Las propiedades enlazan un individuo desde un dominio a individuos de un rango. El dominio y rango de las propiedades son utilizados como axiomas en el razonador.

Clases

Las clases OWL son interpretadas como un conjunto que contiene individuos. Se describen usando descripciones formales (matemáticas) que establecen de manera precisa los requerimientos de membrecía de la clase. Por ejemplo la clase **Gato** contiene todos los individuos que son gatos en el dominio de interés. Las clases están organizadas en jerarquías de superclases y subclases. Por ejemplo, la clase **Animal** y **Gato**, donde **Gato** es una subclase de **Animal** y **Animal** es una superclase de **Gato**. Esto significa que: "Todos los Gatos son Animales", "Todos los miembros de la clase **Gato** son miembros de la clase **Animal**", "Ser un Gato

implica que es un Animal" y "**Gato** está incluido en **Animal**". Una de las características claves de OWL-DL es que las relaciones subclase superclase (relación de inclusión) es calculada automáticamente por el razonador. La figura 5 muestra una representación de algunas clases que contienen individuos, las clases están representadas por círculos u óvalos de manera similar a los conjuntos de los diagramas de Venn. La palabra concepto es utilizada en lugar de clase. Las clases son una representación concreta de un concepto.

En OWL las clases se construyen a través de descripciones que especifican las condiciones que deben satisfacer los individuos para ser miembro de la clase.

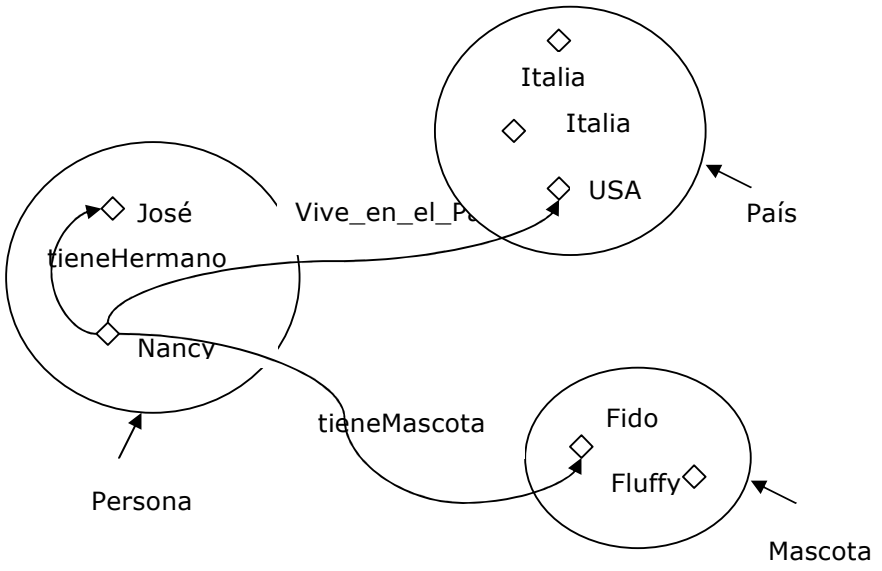

Figura A9. Representación de Clases (contiene individuos)

Uso de un Razonador

La característica clave de las ontologías que usan OWL-DL es que pueden ser procesadas por un razonador. El principal deservicio de un razonador es que prueba cuando una clase es subclase de otra. Otro servicio que ofrece el razonador es el chequeo de consistencia, basado en las descripciones de una clase el razonador puede chequear cuando es posible o no que una clase tenga instancias. Una clase es inconsistente si no puede tener instancias.

Existen varios razonadores que pueden ser utilizados en Protégé-OWL, ejemplo de estos son: FacTC++, Pellet, RACER y Jess.

A.4 Características del lenguaje OWL DL

A continuación se describen las características del lenguaje OWL relacionadas con RDFSchema. Estas propiedades permiten validar el comportamiento deductivo de una ontología editada en Protégé OWL. Las propiedades para las diferentes clases de OWL se describen a partir del OWL Lite y se amplían para el OWL DL y OWL Full.

Características Generales.

- ***Clases (Class)***: una clase define un grupo de individuos que permanecen juntos porque comparten propiedades. Por ejemplo, Deborah y Frank son miembros de la clase Persona. Las clases estan organizadas en una jerarquía de especialización usando ***(subClassOf)subClaseDe***. Existe una clase general llamada ***Thing*** que es la superclase de todas las clases OWL.
- ***rdfs:subClaseDe(subClassOf)***. Se crean jerarquías de clases a través de instrucciones que establecen que una clase es una subclase de otra clase. Por ejemplo, la clase persona se puede establecer como subclase de mamífero. De aquí un razonador puede deducir que si un individuo es persona, entonces es un mamífero.
- ***rdfs:Property***. Las propiedades se utilizan para establecer relaciones entre individuos o de individuos con datos. Ejemplo de propiedades son tieneHijos, tieneHermanos, tieneParientes y tieneEdad. Las tres primeras pueden ser utilizadas para relacionar una instancia de una clase Persona con otra instancia de la clase Persona (y son, por lo tanto, las ocurrencias de la propiedad objeto (ObjectProperty)), y la última tieneEdad puede ser usada para relacionar una instancia de la clase Persona con una instancia de un tipo de dato entero (y es, por lo tanto, una ocurrencia de la propiedad datatype (DatatypeProperty)). Tanto owl:ObjectProperty y owl:DatatypeProperty son subclases de la clase RDF rdf:Property.
- ***rdfs:subPropiedadDe (rdfs:subPropertyOf)***. Se pueden crear jerarquías de subpropiedades estableciendo que una propiedad es subpropiedad de otras propiedades. Por ejemplo tieneHermanos se puede establecer como una subpropiedad de tieneParientes. De esto un razonador puede deducir que si un individuo está relacionado con otro a través de la propiedad tieneHermanos, entonces también está relacionada al otro a través de la propiedad tieneParientes.
- ***rdfs:dominio (rdfs:domain)***. Un dominio de una propiedad limita los individuos a los cuales se les aplica la propiedad. Si una propiedad relaciona un individuo con otro individuo, y la propiedad tiene una clase como uno de sus dominios, entonces los individuos pertenecen a la clase. Por ejemplo, la propiedad tieneHijos se establece que tiene el dominio de los Mamíferos. De esto el razonador puede deducir que si Frank tieneHija Ana, entonces Frank es un Mamífero. rdfs:dominio es una restricción global porque esta asociada no solo con la propiedad sino también con una clase.
- ***rdfs:rango (rdfs:range).*** El rango de una propiedad limita los individuos que pueden tomar el valor de la propiedad. Si una propiedad relaciona un individuo con otro individuo, y la propiedad tiene una clase como su rango, entonces el otro individuo pertenece a la clase rango. Por ejemplo, la propiedad tieneHijos tiene como rango Mamíferos. De esto un razonador puede deducir que si Luisa está relacionada a Debora por la propiedad tieneHijos, (Debora es la hija de Luisa), entonces Debora es un Mamífero. El Rango también es una restricción global.
- ***Individuos (Individual)***. Los individuos son instancias de clases, y las propiedades se usan para relacionar un individuo con otro. Por ejemplo, un individuo llamado Debora se describe como una instancia de la clase Persona y

la propiedad tieneEmpleador puede ser utilizada para relacionar el individuo Debora con el individuo UniversidaddeLosAndes.

Igualdades y Desigualdades OWL

Las siguientes características OWL están relacionadas con las igualdades y desigualdades.

- ***equivalentClass****.* Dos clases son equivalentes si tienen la misma instancia. La igualdad puede ser usada para crear clases sinónimo. Por ejemplo, Carro es una *equivalentClass* a Automóvil. De esto el razonador puede deducir que cualquier individuo que sea una instancia de Carro es también una instancia de Automóvil.
- ***equivalentProperty.*** Dos propiedades son equivalentes si relacionan un individuo al mismo conjunto de otros individuos. La igualda puede ser utilizada para crear propiedades sinónimo. Por ejemplo, la propiedad tieneLider se puede afirmar que es *equivalentProperty* de tieneMaestro. De esto un razonador puede deducir que si X está relacionado a Y por la propiedad tienenLider, X también está relacionada con la propiedad Y tieneMaestro y viceversa. Un razonador también puede deducir que tieneLider es una subpropiedad de tieneMaestro y tieneMaestro es una subpropiedad de tieneLider.
- ***sameAs***. Dos individuos pueden ser los mismos. Este constructo se usa para crear nombres diferentes que se refieren al mismo individuo. Por ejemplo, se puede establecer que Debora es el mismo individuo que DeboraRugeles.
- ***DiferenteDe (differentFrom).*** Un individuo puede ser diferente de otro individuo. Por ejemplo, el individuo Frank es diferente del individuo Debora. Así, si los individuos Frank y Debora son a la vez valores para una propiedad funcional (la propiedad tiene al menos un valor), entonces hay contradicción. Explícitamente establecer que los individuos son diferentes es importante cuando se usan lenguajes como OWL que no asume que los individuos tienen un solo nombre.
- ***todosDiferentes (AllDifferent)***. Se puede establecer que un grupo de individuos son mutuamente distintos con una declaración AllDiferent. Por ejemplo, Frank, Debora y Juan se establece que son diferentes usando el constructo AllDifferent.

Características de las propiedades OWL

Existen identificadores OWL que se usan para proporcionar información relacionada con las propiedades y sus valores.

- ***inversaDe (inverseOf)***. Una propiedad se puede establecer como la inversa de otra propiedad. Si la Propiedad P1 es la inversa de la propiedad P2, entonces si X está relacionada con Y a través de P2, entonces Y está relacionada a X por la propiedad P1. Por ejemplo, si tieneHijos es la inversa de tienePadres y Debora tienePadres Luisa, entonces el razonador deduce que Luisa tieneHijos Debora.
- ***PropiedadTransitiva (TransitiveProperty)***. Una propiedad es transitiva, si el par (x,y) es una instancia de la propiedad transitiva P, y el par (y,z) es una

instancia de P, entonces el par (x,z) es también una instancia de P. Por ejemplo si se establece que ancestros es transitiva, y si Sara es un ancestro de Luisa ((Sara,Luisa) es una instancia de la propiedad ancestro) y Luisa es un ancestro de de Debora ((Sara,Debora) es una instancia de la propiedad ancestro). En OWL Lite y OWL DL al imponer la condición de transitividad de la propiedad (y sus superpropiedades) no puede tener la restricción de maxCardinality 1.

- ***Propiedad Simetrica (SymmetrycProperty)***. Una propiedad es simétrica si dado el par (x,y) es una instancia de la propiedad simétrica P, entonces el par (y,x) tambien es una instancia de P. Por ejemplo, se establece que amigo es una propiedad simétrica, entonces un razonador que tiene establecido que Frank es amigo de Debora deduce que Debora es amiga de Frank.
- ***Propiedad Funcional (FunctionalProperty)***. Esta propiedad establece que tiene un valor único. Si una propiedad es FunctionalProperty, entonces no tiene más de un valor para cada individuo (puede no tener valores para un individuo). FunctionalProperty establece que la propiedad de cardinalidad mínima es cero y su cardinalidad máxima es 1. Por ejemplo, tieneEmpleadorPrimario se establece que es FunctionalProperty, de esto el razonador deduce que ningún individuo puede tener más de un empleador primario. Esto no implica que todas las Personas tengan al menos un empleador primario.
- ***PropiedadInversaFuncional (InverseFunctionalProperty)***. Si una propiedad es inversa funcional entonces la inversa de la propiedad es funcional. Por lo tanto la inversa de la propiedad tiene al menos un valor para cada individuo. Por ejemplo, tieneCedula (identificador único) e inversa funcional, la inversa de esta propiedad (esCedulaDe) tiene como máximo un valor para cualquier individuo en la clase de números de cedula. De esto un razonador deduce que si dos instancias de Persona tienen el mismo número de cedula, entonces estas dos instancias se refieren al mismo individuo.

Restricciones OWL

OWL Lite permite colocar restricciones en cómo pueden utilizar las propiedades las instancias de una clase. Estos tipos son utilizados dentro del contexto de una owl:Restriction. Los elementos de owl:onProperty indican la propiedad restringida. Las dos siguientes restricciones limitan qué valores pueden ser usados mientras que las restricciones restantes limitan cuántos valores pueden ser usados.

- ***todoslosValoresDe (allValuesFrom)***. Se establece para una propiedad con respecto a una clase. Esto significa que esta propiedad de esta clase particular tiene una restricción de rango local asociada. Entonces si una instancia de la clase esta relacionada con la propiedad a un segundo individuo, el segundo individuo puede ser inferido que es una instancia de la clase con restricción de rango local. Por ejemplo, la clase Persona puede tener una propiedad llamada tieneHija restringida a tener allValuesFrom de la clase Mujer. Esto significa que si un individuo de persona Luisa está relacionado está relacionado por la propiedad tieneHija al individuo Debora, entonces de esto un razonador puede deducir que Debora es una instancia de la clase Mujer.

- ***algunosValoresDe (someValuesFrom)***. La restricción someValuesFrom establece una propiedad con respecto a una clase. Una clase puede tener una restricción en una propiedad de que al menos un valor para esta propiedad es de un cierto tipo. Por ejemplo, la clase ArticuloSemanticaWeb puede tener una restricción somevaluesFrom en la propiedad tienePalabraClave que establece que algunos valores de la propiedad tienePalabraClave puede ser una instancia de la clase TopicoSemanticaWeb. Esto permite la opción de tener múltiples palabras clave y tantas como una o mas sean instancias de la clase TopicoSemanticaWeb, entonces el artículo será consistente con la restricción someValuesFrom. Si miArticulo es una instancia de la clase ArticuloSemanticaWeb, entonces miArticulo está relacionado por la propiedad tienePalabraClave con al menos una instancia de la clase TopicoSemanticaWeb.

OWL Lite incluye una forma limitada de restricciones de cardinalidad. Las restricciones de cardinalidad OWL son llamadas restricciones locales, ya que son establecidas con respecto a una clase particular.

- ***cardinalidadMinima (minCardinality)***. Establece una propiedad con respecto a una clase particular. Si se establece la minCardinality 1 en una propiedad con respecto a una clase, entonces cualquier instancia de la clase estará relacionada con al menos un individuo. Esta restricción es otra manera de decir que la propiedad requiere tener un valor para todas las instancias de la clase. Por ejemplo la clase Persona no tendría restricciones de cardinalidad mínima establecida en la propiedad tieneHijos puesto que no todas las personas tienen hijos. La clase Padres sin embargo puede tener la cardinalidad mínima de 1 en la propiedad tieneHijos. Si un razonador conoce que Luisa es una Persona, no puede deducir acerca de la cardinalidad de la propiedad tieneHijos. Pero si se descubre que Luisa es una instancia de Padres entonces el razonador puede deducir que Luisa está relacionada con al menos un individuo por la propiedad tieneHijos.
- ***Cardinalidadmaxima (maxCardinality)***. La cardinalidad se establece en una propiedad con respecto a una clase. Si se establece la maxcardinality de 1 en una propiedad con respecto a una clase, entonces cualquier instancia de esta clase estará relacionada como máximo con un individuo por esta propiedad.
- ***cardinalidad (cardinality)***. La cardinalidad se proporciona como una conveniencia cuando es útil establecer que una propiedad en una clase tiene *minCardinality* 0 y *maxCardinality* 0 o *minCardinality* 1 y *maxCardinality* 1. Por ejemplo, la clase Persona tiene exactamente un valor para la propiedad tieneMadredeNacimiento. De esto el razonador deduce que no hay dos individuos distintos de la clase Madre que puedan ser valores de la propiedad tieneMadredeNacimiento para una misma persona.

OWL datatypes

OWL usa mecanismos XML Schema Datatype[40] y RDF[41] para los valores de los datos.

Descripciones adicionales del lenguaje OWL para OWL DL y OWL Full

Tanto OWL DL como OWL Full usan el mismo vocabulario, pero OWL DL está sujeto a algunas restricciones. OWL DL requiere separación de tipos (una clase no puede ser un individuo o propiedad, una propiedad no puede ser un individuo o una clase). Esto implica que las restricciones no pueden ser aplicadas a los elementos del lenguaje OWL (en OWL Full si lo permite). Además, OWL DL requiere que las propiedades sean ObjectProperties o DatatypeProperties. DatatypeProperties son relaciones entre instancias de clases y literales RDF y XML Schema datatypes, mientras que las propiedades ObjectProperties son relaciones entre instancias de dos clases. A continuación se describe el vocabulario que extiende el OWL Lite a OWL DL y OWL Full.

- ***UnoDe (oneOf)*** (clases enumeradas). Las clases se pueden describir por la enumeración de los individuos que conforman la clase. Los miembros de la clase son exactamente el conjunto de individuos enumerados; ni más ni menos. Por ejemplo la clase diasdelaSemana se puede describir enumerando los individuos: Lunes, Martes, Miércoles, Jueves, Viernes, Sábado, Domingo. De esto un razonador puede deducir la cardinalidad máxima (7) de toda propiedad que tenga diasdelaSemana como restricción allValuesFrom. tieneValor (hasValue) (valores propiedad). Una propiedad puede requerir tener un cierto individuo como un valor (esto muchas veces es llamado valores propiedad (property values)). Por ejemplo, la instancia de la clase de ciudadanosVenezolanos está caracterizada como personas que tienen Venezolano como un valor de su nacionalidad. (El valor nacionalidad, Venezolano, es una instancia de la clase nacionalidad.
- ***disjointWith***. Las clases pueden ser disjuntas. Por ejemplo, Hombre y Mujer son clases disjuntas, de la declaración disjointWith un razonador puede deducir que existe una inconsistencia cuando un individuo establece que es una instancia de ambos; similarmente un razonador puede deducir que si A es una instancia de Hombre entonces A no es una instancia de Mujer.
- ***unionOf, complementOf, intersectionOf*** (Combinaciones Booleanas). OWL DL y OWL Full permiten combinaciones booleanas arbitrarias de clases y restricciones: unionOf, complementOf, e intersectionOf. Por ejemplo usando unionOf se puede establecer que una clase contiene cosas que son Venezolanos o Daneses. Usando el complementOf se puede establecer que los niños *no* son CiudadanosMayores. La ciudadanía de la Unión Europea se puede describir como la unión de la ciudadanía de todos los miembros.
- ***minCardinality, maxCardinality, cardinality (full cardinalidad)***. Mientras que en OWL Lite la cardinalidad está restringida al menos, o máximo o exactamente 1 o 0, OWL full permite establecer cardinalidad para enteros no negativos.

[40] Ver http://www.w3.org/TR/xmlschema~2/ ver en XML Schema Datatype
[41] RDF= Resource Description Framework. Ver http://www.w3.org/TR/RDF~primer/