TRANSFORMACIÓN DE ESQUEMAS DE BASES DE DATOS RELACIONALES AL LENGUAJE DE ONTOLOGÍAS WEB

CARLOS ALBEIRO BRAVO LLANOS
CARLOS ALBERTO SUAREZ MUÑOZ

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
PROGRAMA DE INGENIERÍA DE SISTEMAS
POPAYÁN
2013

TRANSFORMACIÓN DE ESQUEMAS DE BASES DE DATOS RELACIONALES AL LENGUAJE DE ONTOLOGÍAS WEB

CARLOS ALBEIRO BRAVO LLANOS
CARLOS ALBERTO SUAREZ MUÑOZ

Trabajo de grado para optar al título de Ingeniero de Sistemas

Directora
PhD. CAROLINA GONZALEZ SERRANO

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
PROGRAMA DE INGENIERÍA DE SISTEMAS
POPAYÁN
2013

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

1 PRESENTACIÓN DEL PROYECTO

1.1 INTRODUCCIÓN

La Web Semántica es una extensión de la web existente que proporciona un marco de trabajo común para hacer contenidos web entendibles por las máquinas (Hazber, Yang, & Jin, 2010), así, la representación de los contenidos existentes en la web tradicional como contenidos ricos en semántica ha sido uno de los temas que ha ganado importancia en los últimos años. Para esto, el W3C (World Wide Web Consortium) ha recomendado varios formatos que permiten representar la información en la Web Semántica incluyendo RDF (Resource Definition Framework), RDFS (Resource Definition Framework Schema) y OWL (Ontology Web Languaje). Estos lenguajes han sido desarrollados para representar semántica entendible por las máquinas y para facilitar mejores formas de procesamiento de la información (Hazber et al., 2010).

El éxito de la Web Semántica depende de la masiva creación de información semántica (Hazber et al., 2010), por lo anterior es necesario migrar la información que se encuentra actualmente en la web a los formatos recomendados por el W3C. La mayor parte de la información de la web tradicional está almacenada en RDB (Relational Database) (Gherabi, Addakiri, & Bahaj, 2012), en (Choi & Kim, 2012) se asegura que tres cuartos de la información de la web actual proviene de RDB, es por esto que las RDB son vistas como una fuente muy importante de información para la Web Semántica, por lo cual se han iniciado esfuerzos que buscan formas de transformar las RDB en formatos que puedan ser entendidos por las máquinas.

Ya que la estructura de la información de las RDB no es entendible por las máquinas, el W3C ha reconocido la necesidad de cambiar la estructura de los RDBMS (Relational DataBase Management System) en estructuras de datos compatibles con la Web Semántica (Vavliakis, Grollios, & Mitkas, 2012). Mientras los RDBMS evolucionan de manera que faciliten la interacción de las maquinas a través de la web, uno de los retos fundamentales de la Web Semántica es que las

aplicaciones puedan recuperar automáticamente información semántica de las RDB ya existentes (Yang & Wu, 2010). Por lo anterior, las Bases de Datos Relacionales deben ser representadas usando una estructura formal que permita selección, procesamiento y almacenamiento de información con alto contenido semántico. En este sentido, las ontologías son consideradas el mecanismo de representación de conocimiento más adecuado para proveer semántica a la información y contribuir con la evolución de la Web actual hacia la denominada Web Semántica.

Una Ontología es una representación formal de un conjunto de conceptos en un dominio y las relaciones entre esos conceptos (Myroshnichenko & Murphy, 2009). Esta Ontología debe ser representada en algún lenguaje formal, que permita hacerla compatible y procesable por las maquinas, como el OWL que fue recomendado por el W3C como lenguaje formal para la creación de ontologías web (Santoso, Haw, & Abdul-Mehdi, 2011). El OWL fue diseñado para ser usado por aplicaciones que necesitan procesar el contenido de la información en lugar de solo presentarla a los humanos. OWL proporciona a las máquinas mayor interpretabilidad del contenido web que el soportado por XML (Extensible Markup Language), RDF y RDFS, aportando vocabulario adicional junto con una semántica formal (Web-Ontology Working Group, 2004).

En los últimos años se han desarrollado trabajos de investigación que buscan representar el contenido y las estructuras de las RDB en ontologías OWL, en este intento han surgido varios enfoques para la transformación de RDB a ontologías OWL. Según (Myroshnichenko & Murphy, 2009) los tres mejores enfoques son: (i) la transformación directa de esquemas relacionales a ontologías, (ii) la transformación de esquemas relacionales en combinación con semánticas de dominio adicionales a ontologías y (iii) la transformación de esquemas entidad-relación (ER) a ontologías. En los dos primeros enfoques se toma como punto de partida para la transformación el RDBS.

En este trabajo de grado se presenta el planteamiento, ejecución y los resultados del proyecto de investigación titulado *Transformación de Esquemas de Bases de Datos Relacionales al Lenguaje de Ontologías Web*, por medio del cual se elaboró un prototipo de servicio para la transformación de esquemas de bases de datos relacionales al lenguaje de ontologías web llamado RDBS2OWL2 que permite transformar un esquema de base de datos MySQL a una Ontología OWL2 mediante una conexión remota.

Este trabajo ha sido desarrollado a partir de la caracterización de un conjunto de trabajos de investigación relacionados con el tema y que fueron seleccionados bajo criterios específicos definidos para este propósito. Los trabajos seleccionados fueron analizados con el fin de identificar los elementos de las bases de datos relacionales que han sido considerados en las propuestas de transformación e identificar falencias sobre las cuales se hicieron aportes. Este análisis permitió identificar la propuesta que se consideró más adecuada y que fue tomada como base para el servicio propuesto, y permitió establecer la necesidad de incluir algunas restricciones relacionadas con las relaciones de herencia.

1.2 PLANTEAMIENTO DEL PROBLEMA

1.2.1 Descripción

Según (Choi & Kim, 2012), la mayoría de los enfoques están basados en la transformación directa de RDB a ontologías usando reglas de mapeo. Muchos de ellos no hacen una definición formal de las reglas de transformación, incluyen en la ontología conceptos o elementos definidos por los autores que no son presentados formalmente o que no cumplen con el estándar OWL definido por el W3C. Algunos enfoques como los de (Hazber et al., 2010) y (Chen, Liu, & Shang, 2012) analizan ejemplos particulares de RDBSs, definen reglas que son válidas para casos particulares encontrados en el esquema que analizaron y las presentan como reglas que pueden ser aplicadas en cualquier escenario que cuente con una estructura similar al ejemplo analizado. Sin embargo, en ocasiones las reglas que se basan en ejemplos particulares no pueden ser aplicadas en todos los casos que

presentan estructuras similares y de ser aplicadas en un contexto para el que no son válidas, afectarían la ontología proporcionándole un significado erróneo.

En los enfoques de (Astrova, 2009), (Sequeda, Tirmizi, Corcho, & Miranker, 2011) y (Choi & Kim, 2012) se definen reglas de transformación para la herencia. Todos definen reglas para identificar y transformar el caso más general de herencia que se puede presentar, pero ninguno de estos enfoques define reglas para representar y diferenciar la herencia parcial, total, solapada, exclusiva, ni las combinaciones que se pueden formar entre estas. Por lo anterior, en todas las ontologías generadas por estos enfoques las relaciones de herencia identificadas quedan implícitamente representadas como herencias de tipo parcial y solapada, algo que puede no ser correcto en el contexto de los diferentes RDBS. Además, los conceptos relacionados con la herencia son conceptos propios de los MER (Modelo Entidad-Relación) que no se ven reflejados en los RDBS ya que se pierden en el proceso de transformación de un MER a un MR (Modelo Relacional) pero que continúan implícitos en el universo del discurso.

Las propuestas que definen reglas para representar las relaciones de herencia, solo establecen una relación de subclase entre las relaciones implicadas, indicando que una es subclase de la otra, y no establecen ningún tipo de restricción sobre la relación de herencia formada. Por lo cual, esta relación de herencia queda representada como una herencia parcial y solapada, es decir una instancia del súper tipo puede no ser una instancia de algún subtipo (parcialidad) y a su vez una misma instancia del súper tipo puede ser instancia de varios subtipos (solapamiento).

Existen dos restricciones asociadas a la herencia que proporcionan significados contrarios a los anteriores. La restricción de totalidad, contrario a la parcialidad, exige que una instancia del súper tipo sea instancia de al menos un subtipo, y la restricción de exclusividad, contrario al solapamiento, restringe al súper tipo de manera que solo pueda ser instancia de un único subtipo. Estas restricciones no son consideradas en las propuestas analizadas, por lo cual deben ser expresadas

como reglas formales que definan la estructura OWL que represente su significado en una ontología.

Las reglas que son la base para la transformación de varias propuestas, deben ser totalmente claras, evitando la generación de confusiones o ambigüedades. Estas reglas deben ser expresadas formalmente y deben poder ser aplicadas siempre que se cumplan las condiciones definidas para estas. Un concepto en un RDBS identificado por una regla siempre debe ser el mismo y la estructura OWL que resulta de la aplicación de la regla, siempre debe tener el mismo significado.

Por lo anterior, el presente proyecto plantea la siguiente pregunta de investigación: ¿Cómo incluir restricciones para representar los diferentes tipos de herencia en la transformación de RDBS a ontologías OWL?

1.2.2 Justificación

Las propuestas relacionadas con la transformación directa de RDB a OWL que definen reglas de transformación, presentan un conjunto de reglas aisladas, cada una definida para transformar un componente o elemento del RDBS, y no presentan un mecanismo que articule dichas reglas para realizar una transformación completa del RDBS como un todo.

Según (Astrova, 2009) existe la necesidad automatizar la transformación de las RDB a ontologías debido a que la transformación manual es difícil de hacer y toma demasiado tiempo. Por lo anterior un conjunto de reglas, como los presentados en algunas propuestas, no es suficiente ya que cada regla solo define las condiciones que se deben cumplir en el RDBS para formar un segmento OWL. De esta manera los conjuntos de reglas se convierten en guías para la construcción de una ontología a partir de un RDBS. Esto implica que la transformación debe ser realizada por una persona con amplios conocimientos en bases de datos, en ontologías y en el lenguaje OWL.

Por lo anterior, se hace evidente la necesidad de complementar las propuestas de transformación directa de RDB a OWL que existen, con reglas de transformación

formales que permitan representar las restricciones asociadas a los diferentes tipos de herencia, que se ajusten estrictamente al estándar OWL definido por el W3C y que representen casos generales de manera que puedan ser aplicadas en cualquier contexto. Además, es necesario articular el conjunto de reglas en un mecanismo que permita realizar una transformación completa de un RDBS, que pueda ser implementado y puesto a disposición de los usuarios que requieran transformar RDBS a OWL, y del público en general para futuras correcciones, modificaciones y/o extensiones.

Con el presente proyecto se pretende ofrecer un servicio que permita transformar directamente un RDBS a una ontología OWL, que este basado en reglas formales de transformación, que cumpla con el estándar OWL y cuya información, tanto teórica como técnica, sea accesible con el fin de facilitar su estudio en proyectos posteriores.

1.3 OBJETIVOS

1.3.1 Objetivo general

Proponer un servicio semiautomático para la transformación de RDBSs a ontologías OWL que considerando la mayor cantidad de elementos de los RDBSs permita generar ontologías con un adecuado nivel de completitud y precisión.

1.3.2 Objetivos específicos

- Caracterizar los mecanismos de transformación existentes con el fin de identificar los elementos de los RDBS necesarios para el desarrollo del servicio propuesto.
- Diseñar un servicio semiautomático para la transformación de RDBS a ontologías OWL que incluya elementos identificados en la caracterización.
- Implementar el servicio semiautomático propuesto en un prototipo.

- Evaluar la funcionalidad del servicio semiautomático propuesto mediante la implementación de un prototipo que incluye elementos identificados en la caracterización[1].

1.4 MARCO METODOLÓGICO

Para caracterizar los mecanismos de transformación existentes se diseñó un protocolo de caracterización que permitió buscar, tomando como referencia la revisión sistemática realizada en (Pino, García, & Piattini, 2007), trabajos relacionados con la transformación de RDBS a OWL, y seleccionar los trabajos que cumplen los criterios de selección definidos para identificar las propuestas de mayor interés para el presente proyecto.

Para la construcción del prototipo de servicio propuesto se utilizó la metodología de desarrollo ágil XP (Wells, 2009), la cual permitió diseñar e implementar un prototipo con todas las funcionalidades requeridas por el servicio propuesto. En el diseño del prototipo propuesto se adaptó el conjunto de reglas propuesto por (Choi & Kim, 2012) y se plantearon nuevas reglas de transformación que permiten representar las restricciones asociadas a los diferentes tipos de herencia.

La evaluación se llevó a cabo mediante la definición de un Proceso para la Evaluación del Prototipo. El proceso de evaluación fue diseñado para verificar si las transformaciones realizadas por el prototipo cumplen el conjunto de reglas de transformación, verificando los atributos de calidad del software completitud y precisión, y fue ejecutado para transformar cuatro RDBS diferentes. La información relacionada con esta evaluación se encuentra en el Capítulo 5.

1.5 ESTRUCTURA DEL DOCUMENTO

El documento está organizado de la siguiente manera:

Capítulo 1. Presentación del Proyecto: en este capítulo se muestra la importancia de la transformación de los Esquemas de Bases de Datos Relacionales

[1] Se considerarán pruebas de transformación que verifican la completitud y precisión de las ontologías resultantes.

al Lenguaje de ontologías Web, se plantea el problema que dio origen al proyecto de investigación, se presentan los objetivos del proyecto y el marco metodológico empleado en el desarrollo del mismo.

Capítulo 2. Marco Teórico y Estado del Arte: en este capítulo se presentan los estudios relacionados con la transformación de Bases de Datos al Lenguaje de Ontologías Web y se definen todos los conceptos involucrados en el proyecto de investigación.

Capítulo 3. Caracterización de los Mecanismos de Transformación: en este capítulo se presenta la definición y ejecución de un protocolo de caracterización, que permitió identificar y analizar las propuestas de mayor interés para el presente proyecto, mediante la definición de una serie de criterios de selección que debían cumplir las propuestas estudiadas.

Capítulo 4. Diseño e Implementación del Prototipo: en este capítulo se presenta la adaptación de un conjunto de reglas seleccionado a partir de la caracterización, la definición de nuevas reglas para representar los diferentes tipos de herencia, un algoritmo que articula el conjunto de reglas seleccionado y las reglas presentadas en el presente proyecto, y el prototipo de un servicio para la transformación de RDBS a OWL.

Capítulo 5. Evaluación del Prototipo: en este capítulo se presentan los criterios de evaluación del prototipo, el proceso definido para la evaluación y los resultados de la ejecución del proceso de evaluación.

Capítulo 6. Resultados, Conclusiones y Trabajo Futuro: en este capítulo se presentan los resultados y las conclusiones derivadas de la ejecución del proyecto de investigación, y se plantean problemas abiertos que merecen ser tratados en proyectos de investigación posteriores.

2 MARCO TEÓRICO Y ESTADO DEL ARTE

2.1 INTRODUCCIÓN

En la actualidad la mayoría de los sitios Web existentes están respaldados por bases de datos escritas en SQL (Structured Query Language) lo que expone en la Web gran cantidad de datos, los cuales se encuentran dispersos en diferentes sistemas. Para lograr compartir y reusar los datos expuestos en la red, la Web Semántica ha considerado como uno de sus principales objetivos el permitir la integración de datos en toda la Web (Sequeda et al., 2011). Con este objetivo propuesto el W3C (World Wide Web Consortium) quien se encarga de la estandarización de la Web Semántica, ha recomendado varios formatos para la representación de datos en la Web como lo son RDF, RDFS y OWL. Estos lenguajes han sido desarrollados para representar Semántica entendible por las máquinas y para facilitar mejores formas de procesamiento de la información (Hazber et al., 2010).

Se espera que la Web Semántica se convierta en la principal y más accesible corriente para usuarios comunes, por lo que la necesidad de integrar bases de datos a la Web Semántica se incrementará (Sequeda et al., 2011). Además ya para el 2012, tres cuartos de los datos que circulan en la red provienen de RDB (Choi & Kim, 2012). Por lo anterior, el presente proyecto propone la transformación de RDBS a OWL con el objetivo de lograr trasladar la Semántica contenida en un RDBS a un documento OWL.

En el presente capítulo se abordan los conceptos utilizados para la trasformación de RDBS a OWL, al igual que los trabajos o proyectos relacionados con el tema, de los que se hace una descripción mencionando los aspectos más relevantes para el presente proyecto. Este capítulo se organiza en dos secciones que son: (i) marco teórico y (ii) estado del arte.

2.2 MARCO TEÓRICO

En la trasformación de RDBS a OWL intervienen una gran variedad de conceptos y formalidades que permiten realizar el proceso de asignación entre elementos RDBS a OWL. Los conceptos y formalidades más relevantes son: Web Semántica, transformación, RDBS y OWL. Estos conceptos permiten entender con mayor claridad la solución propuesta en este proyecto.

2.2.1 Web Semántica

El W3C además de guiar la construcción de los clásicos *documentos Web* también está ayudando a construir un conjunto de tecnologías para soportar los *datos Web*, que son clases de datos que se pueden encontrar en las bases de datos. El término *Web Semántica* hace referencia a la visión de la Web como datos enlazados (W3C, 2013) de manera que, se considera una extensión de la Web existente, que proporciona un marco de trabajo común que hace contenidos Web entendibles por las máquinas. Lo anterior, permite que los datos sean compartidos y reusados a través de aplicaciones empresariales, para esto la W3C ha recomendado varios formatos entre ellos RDF, RDF Schema y OWL, lenguajes que permiten representar la semántica entendible para la maquina (Hazber et al., 2010).

2.2.2 Transformación

Existen dos enfoques para integrar las bases de datos relacionales con la Web Semántica. El primer enfoque es llamado *direct mapping,* donde el esquema SQL de la base de datos es transformado mecánicamente a una Ontología y los datos relacionales son expuestos como instancias de la Ontología generada. El segundo enfoque supone la existencia de una Ontología de dominio y apunta a *envolver* la base de datos de tal manera que su contenido aparece como instancias de esa Ontología. Los sistemas de *envoltura* normalmente proveen un lenguaje de transformación usado para detallar las correspondencias de los nombres de esquemas de bases de datos a conceptos y propiedades ontológicas (Sequeda et

al., 2011). La *transformación* que se menciona durante el desarrollo del presente trabajo hace alusión al primer enfoque mencionado.

2.2.3 RDBS

Los *Esquemas de Bases de Datos Relacionales* describen un conjunto de esquemas de tablas y un conjunto de restricciones de integridad, donde un esquema de tabla es un conjunto de columnas, y estas columnas tienen asociadas un tipo de dato predefinido (Choi & Kim, 2012).

2.2.4 OWL

El W3C ha creado un grupo de trabajo dedicado exclusivamente a desarrollar el tema de *Lenguaje de Ontología Web* (OWL), el cuál ha sido creado para ser usado por aplicaciones que necesitan procesar el contenido de la información en lugar de solo presentar la información a los humanos (Horrocks, Ruttenberg, Hawke, & Herman, 2012). Debido a que este lenguaje es usado para el modelado de Ontologías en la Web Semántica (Ouyang, Cui, & Ye, 2010) es llamado Lenguaje de la Web Semántica el cual está diseñado para representar conocimiento rico y complejo acerca de entidades, grupos de entidades y relaciones entre ellas. A los documentos escritos en OWL se les conoce como Ontologías, las cuales se pueden publicar en la *World Wide Web,* estas pueden ser referenciadas o referenciar a otras Ontologías OWL publicadas en la Web (Hitzler, Krötzsch, Parsia, F. Patel-Schneider, & Rudolph, 2012).

El lenguaje OWL está basado en un lenguaje de lógica computacional, lo cual permite que programas de computación puedan razonar, verificar la consistencia del conocimiento e inferir conocimiento explicito e implícito en una Ontología (Hitzler et al., 2012).

Existen dos ventajas de las Ontologías respecto a los documentos escritos en el Esquema XML (eXtensible Markup Language). La primera de estas, es que la

Ontología realiza una representación del conocimiento a diferencia del Esquema XML, el cual realiza una representación en formato de mensaje que en combinación de especificaciones de protocolos, conducen a una semántica operacional, que no está diseñada para soportar el razonamiento fuera del contexto en la que se creó. Y la segunda ventaja consiste en la disponibilidad de herramientas que pueden razonar sobre Ontologías, herramientas que proporcionan apoyo genérico, que no es específico del domino de un tema en particular, que sería el caso si se fuera a construir un sistema para razonar sobre un esquema XML estándar especifico de la industria (Smith, Welty, & McGuinness, 2004).

El Lenguaje OWL provee tres sublenguajes que cada vez incrementan su expresividad, para uso por comunidades específicas de implementadores y usuarios (Smith et al., 2004):

- OWL Lite: Apoya a aquellos usuarios que necesitan principalmente una jerarquía de clasificación y características de restricciones simples.

- OWL DL: Apoya a usuarios que desean la máxima expresividad sin perder integridad computacional y decidibilidad[2] de razonamiento de sistemas. OWL DL es llamado así debido a su correspondencia con lógicas descriptivas. Fue diseñado para apoyar la descripción de segmentos de negocio en Lógica Descriptiva y tiene propiedades computacionales deseables para los sistemas de razonamiento.

- OWL FULL: Está dirigido a usuarios que desean la máxima expresividad y la libertad sintáctica de RDF sin garantías de cómputo, lo que permite a una Ontología aumentar el significado del vocabulario predefinido en RDF u OWL.

[2] En lógica, el término decidible se refiere a la existencia de un método efectivo para determinar si un objeto es miembro de un conjunto de fórmulas.

El desarrollador es quien toma la elección de cual sublenguaje se debe usar. Esta decisión es tomada considerando que sublenguaje posee las características que más se ajusten a las necesidades presentes (Smith et al., 2004).

Actualmente el grupo de trabajo de OWL, ha producido OWL 2 el cual está en su segunda edición, siendo OWL 2 una recomendación del W3C que perfecciona y extiende OWL (Horrocks et al., 2012).

OWL 2 tiene una estructura muy similar a OWL, ya que casi todos los componentes básicos de OWL 2 están presentes en OWL, aunque posiblemente con nombres diferentes. La compatibilidad hacia atrás con OWL es, para todo los efectos prácticos, completo, ya que todas las Ontologías OWL siguen siendo válidas, con deducciones idénticas en OWL 2 para todos los casos prácticos. También proporciona una nueva sintaxis llamada *OWL 2 Manchester* (W3C OWL Working, 2012).

Además, OWL 2 adiciona un mayor poder expresivo de las propiedades, soporte extendido para los tipos de datos, capacidades de meta modelo simples, capacidades de anotación extendidas y llaves (Golbreich & Wallace, 2012). Al igual que OWL tiene sublenguajes, OWL 2 define *perfiles* que comúnmente son llamados fragmentos o sublenguajes en lógica computacional (Motik, Grau, et al., 2012). Estos *perfiles* son sublenguajes (subconjuntos sintácticos) de OWL 2 que brindan importantes ventajas en escenarios de aplicaciones particulares, se definen para poner restricciones en las estructuras de las Ontologías OWL 2. Estas restricciones sintácticas pueden ser específicas para modificar la gramática de la sintaxis en el estilo funcional y posiblemente obtener restricciones globales adicionales. Los tres *perfiles* diferentes *que* se definen son: OWL 2 EL, OWL 2 QL y OWL RL (W3C OWL Working, 2012).

- OWL 2 EL: Es particularmente usado en aplicaciones que emplean Ontologías con un gran número de propiedades y/o clases. Este perfil captura la capacidad expresiva usada por muchas de estas Ontologías y desarrolla problemas de

razonamiento estándar en un tiempo polinomial con respecto al tamaño de la Ontología. El acrónimo refleja que el perfil se basa en la familia de lógicas descriptivas que solo proporcionan cuantificación existencial.

- OWL 2 QL: Se dirige a aplicaciones que utilizan volúmenes muy grandes en datos de instancias y aplicaciones donde responder la consulta es la tarea de razonamiento más importante. La capacidad expresiva de este lenguaje es bastante limitada aunque si incluye la mayoría de las características principales de los modelos conceptuales, tales como diagramas de clases UML y diagramas ER. El acrónimo QL refleja el hecho que, consultar en este perfil se puede implementar mediante la reescritura de consultas en un lenguaje de consulta relacional estándar.

- OWL 2 RL: Está dirigido a aplicaciones que requieren razonamiento escalable sin sacrificar demasiada capacidad expresiva. Los sistemas de razonamiento para este perfil, pueden ser implementados usando motores de razonamiento basados en reglas. El acrónimo RL refleja el hecho que el razonamiento en este perfil se puede implementar usando un lenguaje estándar.

Estos perfiles son independientes cada uno del otro, así los usuarios pueden saltar las descripciones de perfiles que no son de su interés. Escoger que *perfil* usar en la práctica, dependerá de la estructura de las Ontologías y las tareas de razonamiento manejadas. Una Ontología en cualquier perfil puede escribirse en un documento de Ontología usando cualquiera de las sintaxis de OWL 2.

2.3 ESTADO DEL ARTE

En la búsqueda para consolidar el estado actual del conocimiento en la trasformación de RDBS a Ontologías OWL, inicialmente se realizó una revisión de bibliografía científica, que incluye principalmente aspectos relacionados con la trasformación de RDBS a OWL. Para obtener los artículos de la revisión, se llevó a cabo un proceso de investigación documental en librerías especializados tales

como: IEEE (Institute of Electrical and Electronics Engineers), ACM (Association for Computing Machinery), LNCS (Lecture Notes in Computer Science), Science Direct y Elsevier. Esta búsqueda se efectuó estableciendo en primer lugar la pregunta de búsqueda, para transformaciones de RDB a OWL *¿Qué propuestas existen para la transformación de Esquemas de Bases de Datos Relacionales a Ontologías OWL?* Y luego se estableció la pregunta de búsqueda para transformaciones de ER a OWL *¿Qué propuestas existen para la transformación de Esquemas Entidad Relación a Ontologías OWL?*

Cada una de las preguntas de búsqueda se utilizó para identificar las palabras claves más comunes que pueden dar respuesta a estas preguntas. Con estas palabras se elaboró una cadena de búsqueda por cada pregunta, y en las cadenas se incluyeron todas las posibles combinaciones de dichas palabras clave. Se usaron las cadenas de búsqueda, para hacer la exploración en las librerías especializadas. En la búsqueda de las transformaciones de RDB a OWL se limitó el año de publicación de los artículos, estableciendo una ventana de cuatro (4) años, del 2009 hasta 2012. En la búsqueda de transformaciones de ER a OWL no se limitó el año de publicación debido a la poca cantidad de propuestas que usan este enfoque.

2.3.1 Trabajos Relacionados

A continuación se presenta una breve descripción y apreciación de cada uno de los artículos obtenidos como resultado del proceso de investigación documental, los cuales se han clasificado en tres enfoques distintos: (i) Transformación directa de esquemas relacionales a Ontologías, (ii) Trasformación de esquemas relacionales en combinación de semánticas de dominio adicionales, (iii) Transformación de esquemas entidad-relación (ER) a Ontologías, y (iv) Otros.

i. Transformación directa de esquemas relacionales a Ontologías

Un esquema de transformación directa, mapea términos directamente a otros términos. Estos términos pueden tener una o muchas correspondencias, de tal manera que pueden mapearse directamente a uno o a muchos términos (Barker & Alhajj, 2006). De esta forma, se realizan las correspondencias pertinentes de términos de *Esquemas Relacionales* a términos usados por *Lenguaje de Ontología Web* generando como resultado una Ontología, cuyo proceso de trasformación se realiza de forma automática. Los estudios que están bajo este enfoque son:

En (Barker & Alhajj, 2006) se presenta un framework que usa la infraestructura de la Web semántica para conducir la interoperabilidad entre los sistemas RDB. La aproximación propuesta usa un algoritmo formalmente descrito, que toma los metadatos de las RDB y las restricciones estructurales para construir la Ontología OWL. La Ontología generada es descrita bajo un conjunto de vocabularios definidos OWL-RDBO (OWL Relational DataBase Ontology), los cuales describen el sistema RDB en la Web, por lo que la Ontología resultante contiene elementos del vocabulario OWL-RDBO, estos elementos no se encuentran definidos en el estándar OWL de la W3C.

En (Astrova, 2009) se propone la transformación de RDB escritas en SQL a Ontologías escritas en OWL. El autor define reglas de transformación para elementos RDB a OWL las cuáles relacionan las construcciones de un modelo relacional a construcciones de un modelo Ontológico con algunas excepciones. El trabajo propuesto no evidencia una implementación de las reglas descritas para el proceso de transformación, sin embargo, el autor sugiere la automatización del mismo debido a la gran cantidad de tiempo que implicaría realizarlo manualmente.

En (Hazber et al., 2010) se presenta un enfoque para transformar las DB (Data Base), definidas en DDL (Data Definition Language), a Ontologías, usando vocabulario RDFS (Resource Description Framework Schema) y XSD (XML Schema Definition). Los autores consideran diferentes elementos de los RDBS

como tablas, relaciones, columnas simples, columnas compuestas, tipos de datos, y restricciones, que son mapeados a la Ontología, y registros de instancias de RDBS que son mapeados a instancias de la Ontología. Se presentan, nuevas reglas de transformación de bases de datos relacionales a Ontologías de la Web Semántica de forma directa y automática. La investigación deja abierta la posibilidad de extender dicha transformación a otros casos presentes en las RDBs. Cabe anotar, que aunque los autores consideran la importancia de incluir una implementación de los diferentes enfoques de transformación existentes, no presentan un prototipo funcional que permita demostrar el uso y efectividad de su propuesta.

En (Būmans, 2010) se presenta un enfoque de transformación simple de bases de datos relacionales a Ontologías OWL, en el que se establece una correspondencia entre elementos de un esquema RDB y Ontologías OWL. Lo anterior es realizado a través de un Esquema de Asignación, el cual está diseñado para generar triples RDF basados en SQL, que permiten la correspondencia de fuentes RDB a destinos OWL y poblar la Ontología resultante de forma automática. Aunque en (Būmans, 2010) se muestra un ejemplo de implementación, no se presenta una descripción clara y detallada del algoritmo y las reglas usadas para llevar a cabo el proceso de transformación.

En (Vavliakis et al., 2012) se describen las características, ventajas y usos del framework RDOTE, Sin embargo no se detalla como el framework propuesto realiza el proceso de transformación de RDB a OWL. No hay evidencia de la creación de reglas y su vinculación a un algoritmo que permita generar una Ontología.

En (Choi & Kim, 2012) se propone un nuevo conjunto de reglas de mapeo para transformar RDBS e instancias a Ontologías correspondientes representadas en OWL, este enfoque comprende tres fases:

- Fase 1: Se realiza la correspondencia de los elementos y propiedades del RDBS a OWL 2, aplicando las reglas de transformación las cuales están presentadas en la sintaxis funcional de OWL2.

- Fase 2: Se realiza la correspondencia de las instancias del RDBS a las instancias de la Ontología generada en la fase anterior. Para esto, esto se aplican las reglas de trasformación apropiadas las cuales también son presentadas en la sintaxis funcional de OWL2.

- Fase 3: Esta fase es el escenario para interpretar el razonamiento de la Ontología OWL compuesta, construida en las fases anteriores.

Estas fases indican que se realiza una transformación directa. Además (Choi & Kim, 2012) soporta no sólo la generación física de Ontologías, sino también, la generación virtual de ellas. Las Ontologías generadas por este enfoque satisfacen OWL 2 DL[3]. Comparado con otros enfoques, el trabajo propuesto es aplicable a un mayor número de DBs debido a que solo basta con que la DB esté en primera forma normal para que esta transformación sea aplicada (lo cual no es posible con la mayoría de enfoques que requieren bases de datos bien normalizadas generalmente en tercera forma normal). Este estudio no describe un desarrollo o implementación de las reglas propuestas, lo cual no permite evidenciar adecuadamente su funcionalidad.

En (Sequeda et al., 2011) se estudia uno de los enfoques existentes para la integración de bases de datos relacionales con la Web Semántica, el cual consiste en transformar directamente el contenido de las bases de datos y los esquemas a lenguajes de la Web Semántica. Se revisan algunos enfoques existentes que automáticamente exponen bases de datos relacionales a la Web Semántica, bajo el enfoque de transformación directa de esquemas SQL a OWL. De esta revisión resulta

[3] OWL 2 DL es un lenguaje de Ontologías para la Web Semántica que define formalmente el significado. Puede ser usado junto con información escrita en RDF, y son intercambiados principalmente como documentos RDF.

la propuesta de un sistema consolidado de reglas, para realizar el proceso de transformación de DB a OWL. La propuesta se implementa y se prueba, con un caso de estudio, del cual se obtiene resultados que permiten concluir que, sí las restricciones no están explicitas en el Esquema Relacional, un mapeo directo no será tan eficaz. Sin embargo sí la aplicación de DB es desarrollada cuidadosamente bajo herramientas de Entidad Relación o métodos de modelado UML, la transformación será sencilla y la Ontología resultante será significativa para el dominio. Aunque se menciona la implementación de las reglas en (Sequeda et al., 2011), no hay evidencia de proceso y/o el algoritmo utilizado para desarrollar la implementación.

ii. Transformación de esquemas relacionales en combinación con semánticas de Ontologías de dominio

En este enfoque se encuentran los estudios que dentro de su proceso para lograr la transformación, utilizan una Ontología de dominio adecuada al contexto en el cual se desarrolló el Esquema Relacional. Es importante anotar, que para esta labor siempre es necesario la inclusión de un experto en el proceso de trasformación, el cual debe tener un amplio conocimiento del dominio y de la creación de Ontologías ya que de este conocimiento depende la calidad de la Ontología resultante.

Los estudios que están bajo este enfoque son:

En (Barker & Alhajj, 2006) se describe una herramienta llamada RDB2ONT para generar y publicar Ontologías OWL a partir de RDB, presenta un algoritmo detallado paso a paso, que usa los metadatos y las restricciones estructurales para construir la Ontología. La Ontología generada es descrita usando y ajustándose a un conjunto de vocabularios definidos en una Ontología que describe el sistema de RDB en la Web. Usando este conjunto de vocabularios se garantiza que las aplicaciones sobre la Web pueden trabajar con instancias de datos que se ajustan a un conjunto de estructuras y vocabularios conocidos.

En (Levshin, 2009) se presenta un enfoque para la traducción de información, esquema y restricciones de las bases de datos a la Web Semántica. Se consideran las restricciones más importantes (posibles llaves primarias y foráneas) así como restricciones CHECK (incluyendo NOT NULL) de las bases de datos. Crea en primer lugar una terminología que representa la RDB para obtener la estructura ontológica correspondiente y luego usa SPARQL[4] para representar las restricciones y para la verificación de las mismas. Además presenta una implementación en Java para probar el concepto.

iii. Transformación de esquemas entidad-relación (ER) a OWL

En (Xu, Cao, Dong, & Su, 2004) se presenta una propuesta formal y una herramienta automática llamada ER2OW, para la transformación de esquemas ER a OWL. Esta herramienta lee el código XML de un esquema ER producido por una herramienta ER CASE como PowerDesign y con el uso de reglas formalmente definidas realiza la transformación a Ontologías en sintaxis OWL DL. Sin embargo, en las reglas de transformación solo se considera la herencia en general y no sus diferentes tipos.

En (Fahad, 2008) se presenta un framework para la transformación de los artefactos de diseño y análisis estructural de ERD (Extended ER Diagram) a OWL-DL. Debido a que ERD tiene un mayor dominio que ER, se detallan las características de ERD durante la descripción del framework de trasformación. Las reglas descritas para la transformación se han probado sobre un análisis estructurado y un ejemplo de diseño. Además, se implementó el conjunto de reglas en un prototipo llamado ER2OWL2 y se realizó una comparación con otra herramienta de transformación, lo que indico que ER2OWL arrojo Ontologías más completas y sin errores de completitud. En cuanto a las reglas definidas para la

[4] SPARQL es un lenguaje de consultas para RDF y puede ser usado para expresar consultas a través de diversas fuentes de información, si la información esta almacenada nativamente como RDF o mostrada como RDF por medio de un middleware.

herencia y sus tipos, (Fahad, 2008) solo considera un tipo de herencia ya que solo se hace la trasformación de la relación *Es-Un* a una propiedad de OWL.

En (Belhadef, Ouafek, & Kholladi, 2008) se propone un conjunto de reglas entre el modelo conceptual ER y su esquema de bases de datos relacional correspondiente hacia las Ontologías. Se definen tres conjuntos de reglas, un conjunto para los conceptos del modelo ER, uno para las relaciones del modelo ER y un conjunto de reglas para las instancias, tomadas del RDBS y las tablas de la BD. Estas reglas están divididas en dos categorías que se complementan entre sí para construir una Ontología que representa una nueva vista de la realidad percibida.

Las reglas propuestas en (Belhadef et al., 2008) permiten obtener un conjunto de conceptos y relaciones entre los conceptos que pueden ser usados para definir una Ontología. Sin embargo estas reglas no están definidas para la elaboración de Ontologías en un lenguaje determinado. Además, las reglas definidas para la herencia, hacen referencia a la herencia en general y no se hace distinción o diferenciación entre tipos de herencia.

En (Myroshnichenko & Murphy, 2009) se presenta una solución al problema del mapeo automático de esquemas ER bien formados a Ontologías OWL Lite equivalentes semánticamente. El esquema ER bien formado es un subconjunto del modelo Entidad Relación y tiene una semántica relacional simple (de primer orden lógico). Este estudio presenta un conjunto de reglas de mapeo que capturan completamente la semántica de un esquema ER bien formado y define una tabla en la que se relacionan los elementos de esquemas ER bien formados con su correspondencia en OWL Lite. Sin embargo, la aproximación propuesta no considera ningún tipo de herencia.

En (Dong, Sun, & Wu, 2010) se presenta un método de transformación de modelo ER a OWL, basado en lógica de descripción. Se utilizan los beneficios de la lógica descriptiva para simplificar la lógica de descripción de los diagramas ER. Lo anterior, debido a la semántica formal que presentan los diagramas, y a que OWL

se basa en lógica descriptiva, lo que ocasiona que el proceso de la transformación de diagramas ER a OWL sea factible. Se definen reglas para la transformación de conjuntos de entidades y atributos relacionales, aunque las reglas definidas no consideran las relaciones de herencia que se pueden presentar en un diagrama ER.

En (Gherabi et al., 2012) se presenta un enfoque para la extracción de la RDB a partir del modelo conceptual de la BD para generar una Ontología OWL. Este modelo pretende incrementar el nivel de representación de la DB, extrayendo de la base de datos relacional un modelo conceptual el cual se utiliza como modelo intermedio para generar la Ontología OWL. Presenta un prototipo soportado en un caso de estudio y establece un algoritmo que permite obtener el modelo intermedio. Sin embargo no se muestran las reglas usadas en el proceso de transformación.

En (Yajai & Sriharee, 2012) se presenta la herramienta EERtoOWL2, que tiene dos funcionalidades principales. La primera consiste en la transformación de la base de datos relacional en información ontológica representada en OWL2. Este proceso requiere el esquema de la base de datos, un modelo ER extendido llamado EER (Enhanced-Entity Relationship) y los registros de la BD. Y en segundo lugar EERtoOWL2, soporta la validación de la información transformada, usando razonamiento ontológico. Adicionalmente, genera reportes de la información que no es válida para la Ontología, reportes que son presentados a los usuarios. Respecto a la herencia, los autores consideran las restricciones de totalidad y parcialidad en los supertipos, y las restricciones de unión y disyunción en los subtipos. Aunque las restricciones de unión y disyunción están asociadas a elementos OWL, las restricciones de totalidad y parcialidad están asociadas a “conceptos superiores” del modelo EER, que son elementos descritos por este modelo, cuyas reglas no son expuestas en el estudio.

iv. Otros

Los estudios que se muestran a continuación, son aquellos que utilizan algún enfoque diferente a los mencionados en las secciones anteriores. Algunos de los enfoques usados en estos artículos son: combinación manual y automática (Chen et al., 2012), uso de grafos (Yang & Wu, 2010), minería de datos (Santoso et al., 2011) entre otros.

En (Chen et al., 2012) se muestra una estrategia hibrida para construir instrumentos científicos ontológicos de modelos de base de datos relacional. Esta estrategia combina la construcción automática y la construcción manual, lo que se muestra en los tres pasos que la componen. En primer lugar se realiza la construcción manual del instrumento científico OWL, luego la extracción del modelo relacional usando el JDBC (Java Database Connectivity) y por último se obtiene el esquema de transformación de acuerdo a las reglas de asignación descritas en el estudio.

En (Santoso et al., 2011) se presenta una estrategia para extraer Ontologías sobre RDBS. Esta propuesta combina la información presente en el esquema de bases de datos y el valor de las instancias, para tomar la ventaja de ambos en el proceso de construcción de la Ontología. Esto se hace usando el concepto de jerarquía como conocimiento básico para seleccionar un conjunto de datos específicos y relevantes para el tipo de conocimiento que se va a extraer en la Ontología.

En (Yang & Wu, 2010) se propone un lenguaje de modelado intermedio que se basa en grafos, el cual crea un puente en entre RDB y la Ontología. Este lenguaje abstrae semiautomáticamente información semántica de las instancias RDB. La trasformación se hace primero expresando el RDBS y las instancias de la DB en forma de grafos, usando el lenguaje formal basado en grafos. Este grafo es posteriormente transformado automáticamente en una Ontología escrita en OWL DL.

En (Vavliakis, Symeonidis, Karagiannis, & Mitkas, 2011) se describe la herramienta llamada Iconomy *y* su uso. También se muestran algunos de sus procesos para realizar la transformación así como algoritmos de algunos procesos. A pesar de esto no se muestra en detalle las reglas de trasformación usadas para generar OWL a partir de un RDB.

2.4 APORTES

En los artículos presentados anteriormente se considera el proceso de transformación de esquemas RDBS a Ontologías OWL. Sin embargo, la mayoría de los enfoques usados para realizar la transformación, presentan problemas comunes como: (i) ausencia de implementación que demuestren con evidencia la funcionalidad de la solución propuesta, (ii) descripción incompleta o inexistente de algoritmos y reglas usados durante el proceso de transformación y (iii) consideración de solo algunos tipos de herencia.

Por lo anterior, se evidencia la necesidad de proponer una solución que incluya un algoritmo de transformación de esquemas RDBS a Ontologías OWL, que presente una adecuada descripción de las reglas utilizadas y su incorporación en un algoritmo propuesto junto con su implementación. La aproximación propuesta deberá ser accesible a una comunidad de investigadores y/o desarrolladores interesados en el tema para su reusó, adaptación y/o mejoramiento además de considerar diferentes tipos de herencia.

2.5 ALCANCE

El alcance del presente proyecto se limita al proceso de trasformación de los RDBS a Ontologías OWL, que es un paso necesario en el intento de lograr la interoperabilidad con significado de la Web. La transformación se aborda desde el punto de vista sintáctico y no se profundiza en la semántica de las Ontologías resultantes. El propósito fundamental es complementar las iniciativas existentes relacionadas con dicha trasformación, incluyendo restricciones que no han sido

consideradas. Lo anterior se hace con el fin de que este trabajo pueda servir como insumo para investigaciones que se vienen adelantando o que se realicen en el futuro y cuyo propósito sea contribuir al logro de niveles de interoperabilidad semántica en la Web.

3. CARACTERIZACIÓN DE LOS MECANISMOS DE TRANSFORMACIÓN

3.1 INTRODUCCIÓN

Para elaborar una propuesta que permita transformar los RDBS a OWL que incluya la mayor cantidad de elementos del RDBS y restricciones para representar los diferentes tipos de herencia, se realizó una caracterización de las propuestas de transformación existentes, con el fin de identificar los elementos de los RDBS y los tipos de herencia que han sido considerados en dichas propuestas de transformación.

La caracterización fue realizada en dos partes. En la primera se consideraron los mecanismos basados en el enfoque *direct mapping* presentado en la sección 2.1.2, lo que permitió identificar elementos de los RDBS que han sido considerados en las propuestas de transformación existentes, y seleccionar la propuesta más adecuada que sirvió de base para el servicio propuesto en el presente proyecto. En la segunda parte se consideraron los mecanismos que usan el enfoque de transformación de ER a OWL. Esta caracterización se realizó con el propósito de profundizar más en la transformación de las relaciones de herencia. Debido a que los conceptos asociados a este tema son propios de los modelos ER, se hizo necesario analizar más propuestas que pudieran incluir reglas de transformación relacionadas con este tema y que pudieran ser adaptadas para el presente proyecto.

Los estudios fueron caracterizados siguiendo un protocolo definido formalmente en el que se consideraron criterios establecidos de acuerdo al propósito de cada caracterización. Lo anterior, permitió identificar y estudiar las propuestas más relevantes para cada una de las caracterizaciones.

El presente capitulo describe la forma como se buscaron, seleccionaron y caracterizaron las propuestas de transformación, y los resultados de dicha caracterización. El capítulo está organizado en cuatro secciones: i) Definición del

protocolo de caracterización, ii) Caracterización de las propuestas que usan el enfoque Direct Mapping, iii) Caracterización de las propuestas que usan el enfoque ER to OWL, y iv) Conclusiones.

3.2 DEFINICIÓN DEL PROTOCOLO DE CARACTERIZACIÓN

Con el fin de estudiar las propuestas relacionadas con el presente trabajo, se definió un protocolo de caracterización que permitió buscar, seleccionar y analizar atributos particulares de cada uno de los trabajos relacionados con el fin de identificar aquellos que pudieran brindar aportes de valor para la construcción del servicio propuesto.

El protocolo está dividido en tres etapas que se describen a continuación:

1) Búsqueda de estudios

Una manera de garantizar la confiabilidad de la caracterización es analizar la mayor cantidad de estudios relacionados con el tema de investigación. Para esto se tomó como referencia las tres primeras fases de la revisión sistemática realizada en (Pino et al., 2007) de manera que se pudieran hallar los estudios más relevantes para el tema de investigación del presente proyecto.

Las tres fases tomadas como referencia son:

- **Formulación de la pregunta de búsqueda:** donde se define la pregunta que se busca resolver con los estudios obtenidos de la búsqueda y se identifican las palabras claves relacionadas con la pregunta definida.

- **Selección de fuentes:** donde se forma una cadena de búsqueda a partir de las palabras clave, que represente de la mejor manera la pregunta de búsqueda. Se seleccionan las fuentes donde se realizará la búsqueda y se definen los criterios de búsqueda necesarios.

- **Elección de estudios:** donde se eligen, de los resultados arrojados por las fuentes seleccionadas, los estudios que se consideran más adecuados y de mayor valor para estudiarlos en las siguientes etapas de la caracterización.

La **¡Error! No se encuentra el origen de la referencia.** representa las fases de la búsqueda.

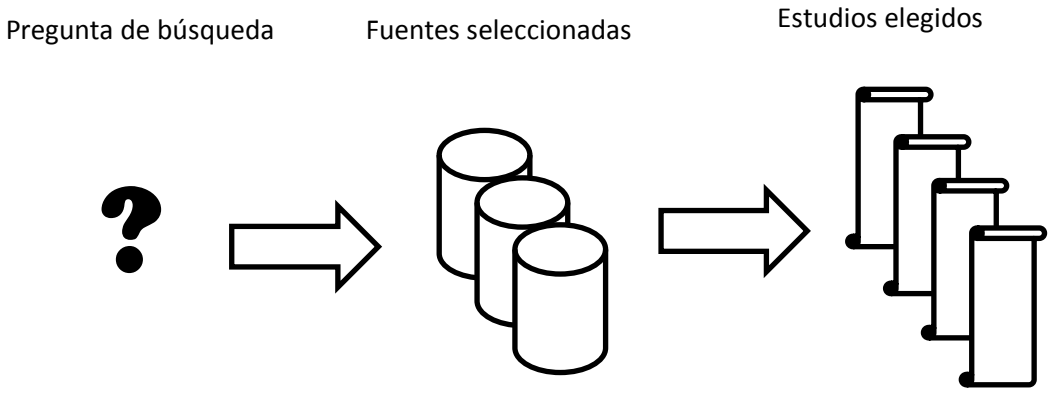

Figura 1. Fases de la búsqueda

2) Selección de estudios

En esta etapa se definen los criterios de selección que deben cumplir las propuestas que serán analizadas en la etapa de caracterización y se realiza un análisis riguroso sobre todos los estudios obtenidos de la etapa de búsqueda con el fin de determinar cuáles son los estudios que cumplen dichos criterios.

3) Caracterización

En esta etapa se define el propósito de la caracterización y se analizan los estudios que resultaron de las etapas anteriores, centrando el análisis en los aspectos de mayor relevancia presentando los resultados obtenidos.

3.3 CARACTERIZACIÓN DE LAS PROPUESTAS QUE USAN EL ENFOQUE DIRECT MAPPING

3.3.1 Búsqueda de estudios

Con el fin de estudiar la mayor cantidad de propuestas relacionadas con la transformación de RDB a OWL, se hizo una búsqueda que permitió identificar los estudios que abordan la transformación de RDB a OWL usando el enfoque *direct mapping,* en el que el esquema SQL de las DBs es transformado mecánicamente a una ontología (Sequeda et al., 2011), estudios cuya ontología generada está regida estrictamente por el estándar OWL definido por el W3C, y que presentan las reglas usadas para la transformación.

Formulación de la pregunta de búsqueda

La pregunta de búsqueda formulada para esta caracterización fue: ¿Qué propuestas existen para la transformación de Bases de Datos Relaciones a ontologías OWL? Las palabras claves que pueden haber sido usadas en los estudios relacionados con la transformación de RDB a OWL, usando el enfoque Direct Mapping, son: *mechanism, tool, approach, approximation, transformation, mapping, assignment, relational database, RDB, ontology, ontology web language* y *OWL*. Estas palabras claves fueron seleccionadas debido a que son muy comunes en las diferentes secciones de los documentos relacionados con el tema de investigación del presente proyecto.

Selección de fuentes

Las palabras clave mencionadas en la Formulación de la pregunta de búsqueda fueron enlazadas usando los conectores lógicos AND y OR con el fin de formar una cadena de búsqueda que pudiera ser usada en los motores de búsqueda de las revistas indexadas y que representara de la mejor manera la pregunta de búsqueda

formulada anteriormente. A continuación se muestra la cadena formada:

(mechanism OR tool OR approach OR approximation) AND (transformation OR mapping OR assignment) AND ("relational database" OR RDB) AND (ontology OR "ontology web language" OR OWL)

Figura 2. Cadena de búsqueda de estudios con enfoque Direct Mapping

Esta cadena de búsqueda fue ingresada en los motores de búsqueda de las revistas IEEE, LNCS y ACM[5]. Con el propósito de obtener los trabajos de investigación más recientes, los resultados de la búsqueda fueron limitados a los estudios publicados entre el año 2009 y el año 2012.

Elección de estudios

Los estudios arrojados por los motores de búsqueda, inicialmente fueron analizados con base en el título, resumen y palabras clave, con el propósito de identificar aquellos estudios que podían brindar respuesta a la pregunta de búsqueda definida. A continuación se listan los estudios que se consideró podían responder la pregunta de búsqueda formulada:

- (Chen et al., 2012)
- (Vavliakis et al., 2011)
- (Santoso et al., 2011)
- (Gherabi et al., 2012)
- (Choi & Kim, 2012)
- (Yang & Wu, 2010)
- (Levshin, 2009)
- (Būmans, 2010)
- (Vavliakis et al., 2012)
- (Astrova, 2009)

[5] Debido a que los motores de busque de las revista indexadas funcionan de manera diferente, la cadena de búsqueda fue adaptada de manera que pudiera ser usada en cada uno de ellos.

- (Sequeda et al., 2011)
- (Hazber et al., 2010)

3.3.2 Selección de estudios

Definición de los criterios de selección

Los estudios obtenidos de la búsqueda presentada anteriormente, fueron analizados con rigurosidad para seleccionar aquellos que cumplieran los siguientes criterios:

i) La ontología generada cumple el estándar OWL definido por el W3C

Algunas de las propuestas existentes para la transformación de RDB a OWL incluyen, en la ontología resultante, elementos de un lenguaje propio definido por los autores que son usados para complementar la ontología. Este es el caso de (Barker & Alhajj, 2006), como se aprecia en la **¡Error! No se encuentra el origen de la referencia.** incluye etiquetas como “<rdbo></rdbo>” que son usadas para apoyar la interoperabilidad con otros sistemas y aplicaciones.

```
...
<rdbo:Relation rdf:ID="Employee">
  ...
  <owl:Restriction>
    <owl:onProperty rdf:resource="#ID" />
    <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0</owl:minCardinality>
  </owl:Restriction>
</rdbo:Relation>
...
<rdbo:Attribute rdf:ID="ID">
  <owl:sameAs rdf:resource="http://localhost/dbOne/Dependent/Dependent.owl#EID" />
  <rdbo:isNullable>false</rdbo:isNullable>
  <rdbo:type>integer</rdbo:type>
</rdbo:Attribute>
...
```

Figura 3. Ejemplo de ontología con elementos adicionales

Ya que el servicio propuesto en el presente trabajo se rige por el estándar OWL establecido por el W3C, los estudios que incluyen elementos que no se encuentran

definidos en el estándar OWL, no fueron considerados para su análisis en la caracterización.

ii) Uso del enfoque *Direct Mapping*

De acuerdo a (Myroshnichenko & Murphy, 2009) los tres mejores enfoques para extraer ontologías de esquemas de bases son: (i) la transformación directa de esquemas relacionales a ontologías, (ii) la transformación de esquemas relacionales en combinación con semánticas de dominio adicionales a ontologías, y (iii) la transformación de esquemas entidad-relación (ER) a ontologías. Para (Sequeda et al., 2011) existen dos enfoques arquitecturales para integrar RDB con la Web Semantica. En el primer enfoque llamado *direct mapping* el esquema SQL de las DBs es transformado mecánicamente a una ontología y los datos relacionales son expuestos como instancias de la ontología generada. En el segundo enfoque se asume que existe una ontología de dominio que tiene como objetivo envolver la DB de manera que su contenido aparezca como instancias de esa ontología.

Se puede deducir que los dos enfoques arquitecturales a los que se refiere (Sequeda et al., 2011) son los mismos enfoques (i) y (ii) de los que (Myroshnichenko & Murphy, 2009) consideran como parte de los mejores enfoques para la extracción de ontologías a partir de esquemas de DB. Estos enfoques tienen en común que ambos parten de un esquema relacional y culminan con una ontología.

El servicio propuesto en el presente trabajo busca realizar una transformación directamente desde un RDBS a una ontología OWL, es decir sigue el enfoque *direct mapping.* Por esta razón, se seleccionaron los estudios que siguen el mismo enfoque con el propósito de identificar propuestas más adecuadas que pudieran servir de base para el servicio propuesto.

iii) Presentación de las reglas de transformación usadas.

Con el fin de elaborar una propuesta para la transformación de RDBS a OWL que esté soportada en reglas de transformación definidas formalmente, se seleccionaron los estudios que describen las reglas utilizadas para la transformación, de manera que se pudiera elegir un conjunto de reglas que sirviera de base para la construcción del servicio propuesto en el presente proyecto.

Análisis de los estudios

Los estudios obtenidos de la búsqueda presentada en la sección anterior, fueron analizados con rigurosidad para identificar aquellos que cumplieran los criterios definidos anteriormente. La Tabla 1 muestra los criterios que cumple cada estudio.

Tabla 1. Criterios cumplidos por los estudios con enfoque Direct Mapping

	Estándar OWL	Enfoque Direct Mapping	Reglas
(Chen et al., 2012)	✓	✓	✓
(Vavliakis et al., 2011)	✓	X	X
(Santoso et al., 2011)	✓	X	✓
(Gherabi et al., 2012)	✓	X	X
(Choi & Kim, 2012)	✓	✓	✓
(Yang & Wu, 2010)	✓	X	✓
(Levshin, 2009)	X	X	X
(Būmans, 2010)	✓	✓	X

(Vavliakis et al., 2012)	✓	✓	X
(Astrova, 2009)	✓	✓	✓
(Sequeda et al., 2011)	✓	✓	✓
(Hazber et al., 2010)	✓	✓	✓

Resultado de la selección

A continuación se listan las propuestas que cumplieron los tres criterios de selección, por lo cual fueron analizadas en la etapa de caracterización:

- (Chen et al., 2012)
- (Choi & Kim, 2012)
- (Sequeda et al., 2011)
- (Hazber et al., 2010)
- (Astrova, 2009)

3.3.3 Caracterización

Las propuestas que cumplieron los criterios de selección y que fueron seleccionadas en el apartado anterior, fueron estudiadas en detalle, más exactamente el análisis se centró en los conjuntos de reglas. Cada regla fue analizada con el fin de identificar el o los elementos del RDBS involucrados en esta y así identificar todos los elementos de los RDBS considerados por cada conjunto de reglas. Además el análisis permitió identificar algunas reglas que resultan confusas o ambiguas cuando se intenta interpretarlas.

Por otra parte el estudio realizado sobre las propuestas mencionadas permitió identificar el tipo de OWL (Lite, DDL, Full u OWL2) soportado por cada una y la existencia o no de implementaciones de las propuestas.

Elementos de los RDBS considerados en las propuestas.

La Tabla 2 muestra la relación entre las reglas de cada propuesta y los elementos RDBS:

Tabla 2. Elementos RDBS transformados por las propuestas caracterizadas

	(Chen et al., 2012)	(Choi & Kim, 2012)	(Astrova, Irina, 2009)	(Hazber, Mohamed,Yang, Jincai,Jin, Qian,2010)	(Sequeda et al., 2011)
Tablas	R1a, R1b, R1c	R1, R18, R25, R27	R1a, R1b, R1c	Ra1, Ra2	R1a, R2a, R3a
Columnas	R2a, R2b	R4, R5, R6, R8, R9, R22, R24	R2	Rb1, Rb2	R5a, R5b, R5c
Tipos de datos	R3	R9		Rc	R5a, R5b, R5c
Primary key	R1b, R4a	R2, R3, R7, R13, R16, R17, R19, R21, R25, R27	R3.3	Rd3	R6a
Foreign	R4b	R2, R3, R7, R13, R15, R16,	R3.4a, R3.4b,	Rd4.1, Rd4.2, Rd4.3, Rd4.4,	R4a, R4b, R4c, R4d,

key		R17, R19, R21, R26	R3.4c	Rd4.5, Rd4.6, Rd4.7	R6a
unique	R4c	R2, R3, R7, R13, R14, R16, R17, R19, R21, R27	R3.1	Rd2	R4c, R4d
Not null	R4d	R20, R23	R3.2	Rd1	R4b, R4d, R5b
Check	R4e	R11	R3.5a	Rd5.1, Rd5.2, Rd5.3, Rd5.5	
Enum	R4e	R10	R3.5b	R5.4	R5c

La tabla anterior no muestra las propuestas que consideran reglas para la herencia ya que este concepto no está representado en los RDBS por un elemento propio. Pero en (Astrova, 2009), (Sequeda et al., 2011) y (Choi & Kim, 2012) se definen reglas para identificar y representar este concepto en OWL.

La regla R3.4c de (Astrova, 2009) y la regla R18 de (Choi & Kim, 2012) emplean el mismo principio para identificar las relaciones de herencia. Estas reglas definen una relación de herencia cuando *toda* la llave primaria de una tabla (la tabla hija) está referenciando la llave primaria de otra tabla (la tabla padre). Por su parte la regla R6 de (Sequeda et al., 2011) define una relación de herencia cuando *parte* de la llave primaria de una tabla (la tabla hija) está referenciando la llave primaria de otra tabla (la tabla padre). Las tres reglas representan de la misma forma la herencia en la ontología OWL, sin embargo la regla de (Sequeda et al., 2011) es

susceptible de errores ya puede llegar a considerar relaciones de herencia a simples asociaciones entre tablas.

Las reglas Rd4.6 y Rd4.7 de (Hazber et al., 2010) son confusas por lo cual no se pudo obtener una interpretación exacta de estas. Por otra parte la regla 1b de (Chen et al., 2012) y la regla 27 de (Choi & Kim, 2012) son ambiguas, es decir permiten diferentes interpretaciones.

La tabla presentada anteriormente muestra que en la propuesta de (Choi & Kim, 2012) varias reglas están relacionadas con más de un elemento del RDBS, por lo cual cada elemento del RDBS tiene más reglas en comparación con las otras propuestas, especialmente las reglas relacionadas con las llaves primarias, foráneas y únicas. También se puede observar que la propuesta de (Astrova, 2009) no define ninguna regla para los tipos de datos, al igual que en la propuesta de (Sequeda et al., 2011) donde no se definen reglas para las restricciones check. Los conjuntos de reglas presentados en cada propuesta se pueden ver en el Anexo 1.

Versión de OWL soportado y existencia de implementaciones.

La Tabla 3 relaciona cada estudio con la versión de OWL soportado por la propuesta e indica en cuales de estos se menciona la existencia de una implementación.

Tabla 3. Versión de OWL soportado por los estudios caracterizados

	Versión de OWL	Implementación
(Choi & Kim, 2012)	OWL 2	X
(Chen et al., 2012)	OWL 1	✓
(Sequeda et al., 2011)	OWL 1	✓

(Hazber et al., 2010)	OWL 1	X
(Astrova, 2009)	OWL 1	X

Como se puede observar en la tabla anterior, solo la propuesta de (Choi & Kim, 2012) soporta la última recomendación del W3C (OWL 2). En el documento los autores manifiestan que su propuesta satisface el perfil OWL 2 DL. Las otras propuestas están elaboradas sobre OWL 1. Ninguna de ellas especifica que perfil de OWL 1 satisfacen.

Por otra parte solo en los estudios de (Chen et al., 2012) y (Sequeda et al., 2011) se manifiesta la existencia de una implementación de sus propuestas, pero ninguno de estos ofrece la posibilidad de acceder a sus implementaciones ni presentan detalles sobre estas.

3.4 CARACTERIZACIÓN DE LAS PROPUESTAS QUE USAN EL ENFOQUE ER to OWL

3.4.1 Búsqueda de estudios

Con el fin de estudiar la mayor cantidad de propuestas que puedan aportar insumos para la transformación de las relaciones de herencia, se hizo una búsqueda que permitió identificar los estudios que abordan la transformación de DB a OWL usando el enfoque *ER to OWL,* en el que la transformación es realizada a partir de Esquemas Entidad Relación, estudios cuya ontología generada está expresada en el lenguaje OWL definido por el W3C, y que presentan las reglas usadas para la transformación.

Formulación de la pregunta de búsqueda

La pregunta de búsqueda formulada para esta caracterización fue: ¿Qué propuestas existen para la transformación de esquemas Entidad Relación a

ontologías OWL? Las palabras claves que pueden haber sido usadas en los estudios relacionados con la transformación de DB a OWL, usando el enfoque ER to OWL, son: *mechanism, tool, approach, approximation, transformation, mapping, assignment, entity relationship, ER, ontology, ontology web language* y *OWL*. Estas palabras claves fueron seleccionadas debido a que son muy comunes en las diferentes secciones de los documentos relacionadas con el tema de investigación del presente proyecto.

Selección de fuentes

Las palabras clave mencionadas en la sección anterior fueron enlazadas usando los conectores lógicos AND y OR con el fin de formar una cadena de búsqueda que pudiera ser usada en los motores de búsqueda de las revistas indexadas y que representara de la mejor manera la pregunta de búsqueda formulada anteriormente. A continuación se muestra la cadena formada:

(mechanism OR tool OR approach OR approximation) AND (transformation OR mapping OR assignment) AND ("entity relationship" OR ER) AND (ontology OR "ontology web language" OR OWL)

Figura 4. Cadena de búsqueda de estudios con enfoque ER to OWL

Esta cadena de búsqueda fue ingresada en los motores de búsqueda de las revistas IEEE, LNCS y ACM[6]. En este caso los resultados de la búsqueda no fueron limitados a un rango de tiempo específico ya que el enfoque ER to OWL no es muy usado, por lo cual son pocas las propuestas que existen relacionadas con este enfoque, y en el presente proyecto se buscaba obtener el mayor número de estudios posible.

[6] Debido a que los motores de busque de las revista indexadas funcionan de manera diferente, la cadena de búsqueda fue adaptada de manera que pudiera ser usada en cada uno de ellos.

Elección de estudios

Los estudios arrojados por los motores de búsqueda, inicialmente, fueron analizados con base en el título, resumen y palabras clave, con el propósito de identificar aquellos estudios que podían brindar respuesta a la pregunta de búsqueda definida. A continuación se listan los estudios que se consideró podían responder la pregunta formulada:

- (Belhadef et al., 2008)
- (Yajai & Sriharee, 2012)
- (Myroshnichenko & Murphy, 2009)
- (Fahad, 2008)
- (Xu et al., 2004)
- (Dong et al., 2010)

3.4.2 Selección de estudios.

Definición de los criterios de selección

Los estudios obtenidos de la búsqueda presentada anteriormente, fueron analizados con rigurosidad para seleccionar aquellos que cumplieran los siguientes criterios:

i) La ontología generada esta expresada en el lenguaje OWL.

Para esta caracterización, no se descartaron las propuestas que no cumplieran estrictamente con el estándar OWL definido por el W3C. Para este caso era suficiente con que la ontología generada estuviera escrita en OWL, independiente mente de que siguieran o no el estándar, ya que el análisis, en la fase de caracterización, solo se centraría en las reglas relacionadas con la herencia. Pero solo se consideraron las propuestas que generaban ontologías en OWL, ya que existen propuestas como la de (Belhadef et al., 2008) en la que las reglas de

transformación están definidas de forma general y no están expresadas para un lenguaje determinado.

ii) Uso del enfoque *ER to OWL*

Con el presente proyecto se pretende definir restricciones que permitan representar los diferentes tipos de herencia en las ontologías obtenidas de una transformación, como los conceptos relacionados con la herencia hacen parte de los esquemas ER y no de los RDBS, se seleccionaron las propuestas que usan el enfoque de transformación ER to OWL con el fin de identificar los avances que se han realizado con respecto a la transformación de las relaciones de herencia y/o las falencias que existen en relación a este tema, sobre las cuales se pudieran hacer aportes.

iii) Presentación de las reglas de transformación usadas

Con el fin de elaborar una propuesta para la transformación de RDBS a OWL que esté soportada en reglas de transformación definidas formalmente, se seleccionaron los estudios que describen las reglas utilizadas para la transformación, de manera que se pudiera adaptar las reglas relacionadas con la herencia para la construcción del servicio propuesto en el presente proyecto.

Análisis de los estudios

Los estudios obtenidos de la búsqueda presentada en la sección anterior, fueron analizados con rigurosidad para identificar aquellos que cumplieran los criterios definidos anteriormente. La Tabla 4 muestra los criterios que cumple cada estudio:

Tabla 4. Criterios cumplidos por los estudios con enfoque ER to OWL

	OWL	Enfoque ER to OWL	Reglas
(Myroshnichenko & Murphy,	✓	✓	✓

2009)			
(Belhadef et al., 2008)	X	✓	✓
(Xu et al., 2004)	✓	✓	✓
(Dong et al., 2010)	✓	✓	✓
(Yajai & Sriharee, 2012)	✓	✓	X
(Fahad, 2008)	✓	✓	✓

Resultado de la selección

A continuación se listan las propuestas que cumplieron los tres criterios de selección, por lo cual fueron analizadas en la etapa de caracterización:

- (Myroshnichenko & Murphy, 2009)
- (Xu et al., 2004)
- (Dong et al., 2010)
- (Fahad, 2008)

3.4.3 Caracterización

Las propuestas que usan el enfoque ER to OWL, que fueron seleccionadas en la sección anterior, fueron analizadas con el propósito de identificar los avances que se han realizado en relación a la transformación de las relaciones de herencia. Específicamente en esta caracterización se buscó identificar reglas relacionadas con la transformación de relaciones de herencia y reglas que representarán las restricciones asociadas a los diferentes tipos de herencia.

La Tabla 5 muestra los resultados del análisis realizado a las propuestas que resultaron de la búsqueda y selección.

Tabla 5. Estudios que presentan reglas para la herencia

	Herencia	Tipos de herencia
(Myroshnichenko & Murphy, 2009)	X	X
(Fahad, 2008)	✓	X
(Xu et al., 2004)	✓	X
(Dong et al., 2010)	X	X

Como se puede observar en la tabla anterior, solo en las propuestas de (Fahad, 2008) y (Xu et al., 2004) se define reglas para representar la herencia. En cada propuesta se define una regla que transforma las relaciones de tipo *is-a* al elemento OWL *subClassOf* en el que se relaciona la superclase y la subclase. Pero ninguna de las propuestas define reglas para representar las restricciones asociadas a los diferentes tipos de herencia.

3.5 CONCLUSIONES

En la caracterización realizada a las propuestas que usan el enfoque Direct Mapping se pudo establecer que solo tres estudios definen reglas para la identificación y representación en OWL de las relaciones de herencia. En una de las propuestas, la regla definida para la herencia es susceptible de errores ya que puede considerar como relación de herencia a una relación simple entre dos tablas. Cada una de estas propuestas define solo una regla para la herencia y en ninguna de estas se definen reglas para representar los diferentes tipos de herencia.

De acuerdo a la caracterización realizada sobre los estudios que usan el enfoque ER to OWL, desde esta perspectiva tampoco se han definido reglas para la representación de las restricciones asociadas a los diferentes tipos de relaciones de herencia. Por lo anterior, se hace evidente la necesidad de definir reglas formales de transformación que permitan representar este tipo de restricciones de manera que sirvan como complemento para los conjuntos de reglas presentados en algunas propuestas.

El conjunto de reglas de (Choi & Kim, 2012) tiene más reglas asociadas con cada elemento del RDBS que las demás propuestas, es una de las propuestas que define una regla adecuada para la identificación y representación de las relaciones de herencia, y es la única propuesta que soporta OWL 2 que es la última recomendación del W3C. Por lo anterior, el conjunto de reglas definido en este estudio fue tomado como base la para la construcción del servicio propuesto en el presente proyecto.

De esta caracterización también se pudo establecer que no existen implementaciones de propuestas que usen el enfoque Direct Mapping, que cumplan con el estándar OWL definido por el W3C y que soporten la transformación en reglas formales. Por lo anterior se puede evidenciar la necesidad de una implementación que realice transformaciones de RDBS a OWL, que cuente con las características mencionadas anteriormente, que esté disponible para su uso, correcciones, modificaciones y/o extensiones.

4 DISEÑO E IMPLEMENTACIÓN DEL PROTOTIPO

4.1 INTRODUCCIÓN

Durante el desarrollo del presente proyecto se definieron nuevas reglas de transformación para representar las restricciones asociadas a los diferentes tipos de herencia y se diseñó un algoritmo que adapta el conjunto de reglas propuestas por (Choi & Kim, 2012) y las articula con las nuevas reglas para transformar un RDBS a una ontología OWL2. Este algoritmo fue implementado en un prototipo de servicio que permite transformar un RDBS de Mysql a una ontología OWL2, el cual ofrece dos operaciones. La primera permite obtener la información de las relaciones de herencia identificadas en un RDBS Mysql, en dicha información se encuentran relacionadas las tablas padre con sus respectivas tablas hijas. La segunda permite obtener una ontología en OWL2 que resulta de la transformación de un RDBS MySql, la cual es generada por el algoritmo propuesto.

Además se diseñó e implementó un cliente del servicio, el cual consume las dos operaciones ofrecidas por el servicio y presenta los resultados.

El desarrollo del prototipo de servicio y el cliente fueron realizados tomando como referencia la metodología de desarrollo ágil XP (eXtreme Programming) (Wells, 2009), la cual permitió diseñar y construir en dos iteraciones los prototipos funcionales tanto del servicio como del cliente.

El presente capitulo está organizado en cinco secciones. En la primera sección se presenta el conjunto de reglas propuesto en (Choi & Kim, 2012) que fue tomado como base para la construcción del servicio. En la segunda sección se presenta la redefinición de las reglas ambiguas. En la tercera sección se presenta la definición formal de las reglas para la herencia propuestas en el presente proyecto. En la cuarta sección se presenta el algoritmo diseñado. Y en la quinta sección se presenta el desarrollo del servicio propuesto.

4.2 CONJUNTO DE REGLAS BASE

En la sección 3.5 se defino que el conjunto de reglas presentado en (Choi & Kim, 2012) seria usado como base para el servicio propuesto en el presente trabajo. A continuación se presenta el conjunto de reglas propuesto en dicho estudio:

Regla 1: Una tabla T_i crea una *clase* cls_{Ti}.
Declaration(**Class**(*:*cls_{Ti}))

Regla 2: Si una columna $T_i.C_k$ es $T_i.C_k$ *E Key(*T_i*)*, la $T_i.C_k$ crea una clase cls_{Ti}.
Declaration(**Class**(*:*$cls_{Ti.ck}$))

Regla 3: Si una columna $T_i.C_k$ es $T_i.C_k$ *E Key(*T_i*)*, la $T_i.C_k$ crea una *propiedad de objeto* $op_{Ti.ck}$.
Declaration(**ObjectProperty**(*:*$op_{Ti.ck}$))

Regla 4: Una columna $T_i.C_k$ crea una *propiedad de dato* $dp_{Ti.ck}$.
Declaration(**DataProperty**(*:*$dp_{Ti.ck}$))

Regla 5: La *propiedad de objeto* $op_{Ti.ck}$ define a la case cls_{Ti} como su *dominio*.
ObjectPropertyDomain(**:***op*$_{Ti.ck}$ *:cls*$_{Ti}$)

Regla 6: Una *propiedad de objeto* $op_{Ti.ck}$ define a la case cls_{Ti} como su *rango*.
ObjectPropertyRange(**:op**$_{Ti.ck}$ *:cls*$_{Ti.ck}$)

Regla 7: Si una columna $T_i.C_k$ es $T_i.C_k$ *E Key(*T_i*)*, la *propiedad de dato* $dp_{Ti.ck}$ derivada de la columna $T_i.C_k$ define a la clase $cls_{Ti.ck}$ como su *rango*.
DataPropertyDomain(**:dp**$_{Ti.ck}$ *:cls*$_{Ti.ck}$)

Regla 8: Si una columna $T_i.C_k$ es $T_i.C_k$ *∉ Key(*T_i*)*, la *propiedad de dato* $dp_{Ti.ck}$ derivada de la columna $T_i.C_k$ define a la clase $cls_{Ti.ck}$ como su *dominio*.
DataPropertyDomain(**:dp**$_{Ti.ck}$ *:cls*$_{Ti}$)

Regla 9: Si una columna $T_i.C_k$ es $T_i.C_k$ *∉ check(*T_i*)* y $T_i.C_k$ *∉ enum(*T_i*)*, la *propiedad de dato* $dp_{Ti.ck}$ derivada de la columna $T_i.C_k$ define el *tipo de dato* $dt_{Ti.ck}$ correspondiente al *type(*$T_i.C_k$*),* como su *rango*.
DataPropertyRange(**:dp**$_{Ti.ck}$ *dt*$_{Ti.ck}$)

Regla 10: Si una columna $T_i.C_k$ es $T_i.C_k$ *E enum(*T_i*),* la *propiedad de dato* $dp_{Ti.ck}$ derivada de la columna $T_i.C_k$, define la *enumeración de literales,* correspondiente a la *restricción check con enumeración* para la $T_i.C_k$, como su *rango.*
DataPropertyRange(**:dp**$_{Ti.ck}$ **DataOneOf**(*lt*$_1$ *… lt*$_n$))

Regla 11: Si una columna $T_i.C_k$ es $T_i.C_k$ *E check*(T_i), la *propiedad de dato* $dp_{Ti.Ck}$ derivada de la columna $T_i.C_k$ define la restricción de *tipo de dato,* correspondiente a la restricción *check* para la $T_i.C_k$, como su *rango.*

DataPropertyRange(:$dp_{Ti.Ck}$ **DatatypeRestriction**(f_1 lt_1 … f_n lt_n))

Regla 12: Si la clase cls_{Ti} derivada de la T_i define las *propiedades de objeto* derivadas de cada columna $T_i.C_k$ en *pKey*(T_i) como sus llaves. Cuando *pKey*(T_i) = { $T_i.C_1$, … , $T_i.C_m$ }, m ≥ *1.*

HasKey(:cls_{Ti} (:$op_{Ti.C1}$ … :$op_{Ti.Cm}$) ())

Regla 13: Si una columna $T_i.C_k$ es $T_i.C_k$ *E key*(T_i), la *clase* $cls_{Ti.Ck}$ derivada de la $T_i.C_k$ define la *propiedad de dato* $dp_{Ti.Ck}$, derivada de la $T_i.C_k$, como su *llave*.

HasKey(:$cls_{Ti.Ck}$ () (:$dp_{Ti.Ck}$))

Regla 14: Si una columna $T_i.C_k$ es $T_i.C_k$ *E unique*(T_i), la *propiedad de objeto* $op_{Ti.Ck}$ derivada de la columna $T_i.C_k$ es definida como una *propiedad funcional inversa.*

InverseFunctionalObjectProperty(:$op_{Ti.Ck}$)

Regla 15: Si una columna $T_i.C_k$ es $T_i.C_k$ *E fkey*(T_i), la *clase* $cls_{Ti.Ck}$ derivada de la columna $T_i.C_k$ es definida como *subclase* de la *clase* $cls_{ref(Ti.Ck)}$ derivada de la columna referenciada *ref*($T_i.C_k$) por la columna $T_i.C_k$.

SubClassOf(:$cls_{Ti.Ck}$:$cls_{ref(Ti.Ck)}$)

Regla 16**:** Si una columna $T_i.C_k$ es $T_i.C_k$ *E key*(T_i), la *clase* $cls_{Ti.Ck}$ derivada de la columna $T_i.C_k$ es definida como una *subclase* de expresión de *clase universal* **DataAllValuesFrom**(:$dp_{Ti.Ck}$ $dr_{Ti.Ck}$), donde $dr_{Ti.Ck}$ es el rango de la *propiedad de dato* $dp_{Ti.Ck}$.

SubClassOf(:$cls_{Ti.Ck}$ **DataAllValuesFrom**(:$dp_{Ti.Ck}$ $dr_{Ti.Ck}$))

Regla 17: Si una columna $T_i.C_k$ es $T_i.C_k$ *E key*(T_i), la *clase* $cls_{Ti.Ck}$ derivada de la columna $T_i.C_k$ es definida como una *subclase* de la expresión de clase *cardinalidad exacta* **DataExactCardinality**(:$dp_{Ti.Ck}$ $dr_{Ti.Ck}$), donde $dr_{Ti.Ck}$ es el rango de la *propiedad de dato* $dp_{Ti.Ck}$.

SubClassOf(:$cls_{Ti.Ck}$ **DataExactCardinality**(:$dp_{Ti.Ck}$ $dr_{Ti.Ck}$))

Regla 18: Si existe la tabla padre *parent*(T_i) de una tabla T_i, la *clase* cls_{Ti} derivada de la tabla T_i es definida como una *subclase* de la *clase* $cls_{parent(Ti)}$ derivada de la tabla *parent*(T_i) referenciada.

SubClassOf(:cls_{Ti} :$cls_{parent(Ti)}$)

Regla 19: Si una columna $T_i.C_k$ es $T_i.C_k$ *E key*(T_i), la *clase* cls_{Ti} derivada de la tabla T_i es definida como una *subclase* de la expresión de *clase universal* **ObjectAllValuesFrom**(:$op_{Ti.Ck}$:$cls_{Ti.Ck}$).

SubClassOf(:cls_{Ti} **ObjectAllValuesFrom**(:$op_{Ti.Ck}$:$cls_{Ti.Ck}$))

Regla 20: Si una columna $T_i.C_k$ *E key*(T_i) y $T_i.C_k$ *E notnull*(T_i), la *clase* cls_{Ti} derivada de la tabla T_i es definida como una *subclase* de la expresión de *clase* **ObjectExactCardinality**(**1 :op**$_{Ti.Ck}$ **:cls**$_{Ti.Ck}$).

SubClassOf(**:cls**$_{Ti}$ **ObjectExactCardinality**(**1 :op**$_{Ti.Ck}$ **:cls**$_{Ti.Ck}$))

Regla 21: Si una columna $T_i.C_k$ *E key*(T_i) y $T_i.C_k$ *∉ notnull*(T_i), la *clase* cls_{Ti} derivada de la tabla T_i es definida como una *subclase* de la expresión de *clase* **ObjectMaxCardinality**(**1 :op**$_{Ti.Ck}$ **:cls**$_{Ti.Ck}$).

SubClassOf(**:cls**$_{Ti}$ **ObjectMaxCardinality**(**1 :op**$_{Ti.Ck}$ **:cls**$_{Ti.Ck}$))

Regla 22: Si una columna $T_i.C_k$ es $T_i.C_k$ *∉ key*(T_i), la *clase* cls_{Ti} derivada de la tabla T_i es definida como una *subclase* de la expresión de *clase universal* **DataAllValuesFrom**(**:dp**$_{Ti.Ck}$ **dr**$_{Ti.Ck}$), donde **dr**$_{Ti.Ck}$ es el rango de la *propiedad de dato* **dp**$_{Ti.Ck}$.

SubClassOf(**:cls**$_{Ti}$ **DataAllValuesFrom**(**:dp**$_{Ti.Ck}$ **dr**$_{Ti.Ck}$))

Regla 23: Si una columna $T_i.C_k$ es $T_i.C_k$ *∉ key*(T_i) y $T_i.C_k$ *E notnull*(T_i), la *clase* cls_{Ti} derivada de la tabla T_i es definida como una *subclase* de la expresión de *clase* **DataExactCardinality**(**1 :dp**$_{Ti.Ck}$ **dr**$_{Ti.Ck}$), donde **dr**$_{Ti.Ck}$ es el rango de la *propiedad de dato* **dp**$_{Ti.Ck}$.

SubClassOf(**:cls**$_{Ti}$ **DataExactCardinality**(**1 :dp**$_{Ti.Ck}$ **dr**$_{Ti.Ck}$))

Regla 24: Si una $T_i.C_k$ es $T_i.C_k$ *∉ key*(T_i) y $T_i.C_k$ *∉ notnull*(T_i), la *clase* cls_{Ti} derivada de la tabla T_i es definida como *subclase* de la expresión de clase **DataMaxCardinality**(**1 :dp**$_{Ti.Ck}$ **dr**$_{Ti.Ck}$), donde **dr**$_{Ti.Ck}$ es el rango de la *propiedad de dato* $dp_{Ti.Ck}$.

SubClassOf(**:cls**$_{Ti}$ **DataMaxCardinality**(**1 :dp**$_{Ti.Ck}$ **dr**$_{Ti.Ck}$))

Regla 25: Si existe la tabla padre *parent*(T_i) de una tabla T_i y una columna $T_i.C_k$ es $T_i.C_k$ *E pkey*(T_i), la *propiedad de objeto* $op_{Ti.Ck}$ derivada de la columna $T_i.C_k$ es definida como una *subpropiedad* de la *propiedad de objeto* $op_{ref(Ti.Ck)}$ derivada de la columna *ref*($T_i.C_k$) referenciada por la columna $T_i.C_k$.

SubObjectPropertyOf(:op$_{Ti.Ck}$ **:op**$_{ref(Ti.Ck)}$**)**

Regla 26: Si una columna $T_i.C_k$ es $T_i.C_k$ *E fkey*(T_i), la *propiedad de dato* $dp_{Ti.Ck}$ derivada de la columna $T_i.C_k$ es definida como una *subpropiedad* de la *propiedad de dato* $dp_{ref(Ti.Ck)}$ derivada de la columna *ref*($T_i.C_k$) referenciada por la columna $T_i.C_k$.

SubDataPropertyOf(**:dp**$_{Ti.Ck}$:**dp**$_{ref(Ti.Ck)}$)

4.3 REDEFINICIÓN DE REGLAS AMBIGUAS

En la sección 3.3.3.1 se mencionó que la regla 27 de la propuesta de (Choi & Kim, 2012) es ambigua. Como el conjunto de reglas de esta propuesta es la base para el

servicio propuesto en el presente trabajo, se hizo necesario redefinir esta regla expresando el significado que se consideró más adecuado para contribuir a la transformación de un RDBS a OWL2.

Para el presente trabajo, la regla 27 de (Choi & Kim, 2012) se redefinió así:

Si no existe la tabla parent(T_i) y una columna $T_i.C_k$ es $T_i.C_k$ E fkey(T_i),y $T_i.C_k$ E pkey(T_i) o $T_i.C_k$ E unique(T_i)[7], todas las clases derivadas de T_i y $T_i.C_k$ son definidas como clases disjuntas por parejas. Para $1 \leq k \leq n$, la declaración para esta restricción es la siguiente:

DisjointClasses(:cls$_{Ti}$:cls$_{Ti.C1}$ … :cls$_{Ti.Cn}$)

Esta restricción impide que una instancia de la clase derivada de una columna sea instancia de la clase derivada de la tabla a la cual pertenece dicha columna.

4.4 DEFINICIÓN DE REGLAS PARA LA HERENCIA

Como se mencionó en la sección 3.5, la caracterización de las propuestas de transformación permitió evidenciar la necesidad de una definición formal de reglas de transformación que representen las restricciones asociadas a los diferentes tipos de herencia. Por lo anterior, en el presente proyecto se definieron nuevas reglas que permiten representar la herencia total, exclusiva y total-exclusiva.

En la propuesta de (Choi & Kim, 2012) además del conjunto de reglas, se definen una serie de funciones que permiten expresar las reglas en una forma más compacta y fácil de entender. Entre estas se encuentra la función *parent(T_i)* que retorna, si existe, la tabla *Padre* de la tabla T_i.

Para expresar las reglas, que se definieron en la presente propuesta, de una manera que resulte simple y fácil de entender, se definió la función *childs(T_i)* que

[7] Las funciones fkey(T_i), pkey(T_i) y unique(T_i) son funciones definidas en (Choi & Kim, 2012).

fue empleada en la definición formal de las reglas que se presentan más adelante. *childs(T_i)* fue definida así:

childs(T_i): retorna un conjunto de tablas hijas de T_i.

Debido a que la regla para la herencia total se deriva de la definición de la regla para la herencia total-exclusiva, primero se presenta la regla para la herencia exclusiva, seguido de la regla para la herencia total-exclusiva y finalmente se presenta la regla para la herencia total. Las reglas son presentadas en el orden mencionado con el fin de facilitar su presentación y comprensión por parte del lector.

Regla para la herencia exclusiva

Una herencia exclusiva indica que una instancia de la superclase no puede ser instancia de varias subclases, o lo que es lo mismo, una instancia de la superclase solo puede ser instancia de una subclase. Para representar esta restricción en OWL2, se empleó el axioma de clase[8] DisjointClasses(CE_1 ... CE_n) que indica que todas las expresiones de clase[9] CE_j, con $1 \leq j \leq n$, son disjuntas por parejas, es decir, ningún individuo puede ser a la vez una instancia de CE_i y CE_j, donde $i \neq j$ (Motik, F. Patel-Schneider, & Parsia, 2012).

De acuerdo a lo anterior la regla para expresar la restricción asociada a la herencia exclusiva se definió así:

Si existe la tabla T_i tal que childs(T_i) ≠ Ø, y la relación entre la tabla T_i y las tablas childs(T_i) es una relación de herencia exclusiva, las clases derivadas de las tablas childs(T_i) son definidas como clases disjuntas. Cuando childs(T_i) = {T_1,..,T_n} donde $n \geq 1$, la declaración para esta restricción es la siguiente:

[8] Un axioma es una declaración que indica lo que es verdad en el dominio y un axioma de clase permite establecer relaciones entre expresiones de clase (Motik, F. Patel-Schneider, et al., 2012).

[9] Una expresión de clase representa un conjunto de individuos mediante la especificación formal de condiciones sobre las propiedades de los individuos (Motik, F. Patel-Schneider, et al., 2012).

DisjointClasses(:cls$_{T1}$ ... :cls$_{Tn}$)

Esta declaración permite restringir las instancias de las subclases, impidiendo que un mismo individuo pueda ser instancia de varias subclases. Esto puede ser notado en el siguiente ejemplo:

SubClassOf(:SuperComputador :Computador)	Un SuperComputador es un computador
SubClassOf(:MicroComputador :Computador)	Una MicroComputador es un computador
DisjointClasses(:SuperComputador :MicroComputador)	Un SuperComputador no puede ser un MicroComputador
ClassAssertion(:SuperComputador :C1)	C1 es un SuperComputador

El anterior ejemplo representa un fragmento de una ontología. Si se agrega la siguiente línea, la ontología seria inconsistente.

ClassAssertion(:MicroComputador a:C1)

Regla para la herencia total-exclusiva

Una herencia total indica que toda instancia de la superclase debe estar representada en *al menos una* instancia de las subclases. La restricción solo indica la mínima representación que debe tener la superclase en las subclases y no establece un máximo. Si a esta restricción le adicionamos la restricción de la herencia exclusiva que se presentó anteriormente se tiene: una herencia total-exclusiva indica que toda instancia de la superclase debe estar representada en *una y solo una* instancia de las subclases, es decir una instancia de la superclase debe estar representada en exactamente una instancia de las subclases. Para representar esta restricción en OWL2, se empleó el axioma de clase DisjointUnion(C CE_1 ... CE_n), el cual indica que cada instancia de C es una

instancia de exactamente una CE_i, y cada instancia de CE_i es una instancia de C (Motik, F. Patel-Schneider, et al., 2012).

De acuerdo a lo anterior la regla para expresar la restricción asociada a la herencia total-exclusiva se definió así:

Si existe la tabla T_i tal que childs(T_i) ≠ Ø, y la relación entre la tabla T_i y las tablas childs(T_i) es una relación de herencia total-exclusiva, la clase derivada de T_i es definida como una unión disjunta de las clases derivadas de las tablas childs(T_i). Cuando childs(T_i) = {T_1,..,T_n} donde n ≥ 1, la declaración para esta restricción es la siguiente:

DisjointUnion(:cls$_{Ti}$:cls$_{T1}$ ... :cls$_{Tn}$)

De acuerdo a (Motik, F. Patel-Schneider, et al., 2012) la declaración anterior puede ser vista como una abreviación de los axiomas siguientes:

EquivalentClasses(:cls$_{Ti}$ ObjectUnionOf(:cls$_{T1}$ ... :cls$_{Tn}$))

DisjointClasses(:cls$_{T1}$ ... :cls$_{Tn}$)

Cualquiera de las declaraciones anteriores, permite establecer la representación máxima y mínima de las superclases respecto a las subclases en una sola instancia. Esto puede ser notado en el siguiente ejemplo:

SubClassOf(:Hombre :Persona)	Un hombre es una persona
SubClassOf(:Mujer :Persona)	Una mujer es una persona
DisjointUnion(:Persona :Hombre :Mujer)	Cada hombre es una persona, cada mujer es una persona y nadie pude ser hombre y mujer
ClassAssertion(:Persona :Juan)	Juan es una Persona

Hasta el momento Juan puede ser un hombre o una mujer, y la ontología está incompleta. La siguiente declaración indica que Juan es un hombre, por lo cual ya no podría ser una mujer.

ClassAssertion(:Hombre :Juan)

Si se incluyera la siguiente declaración la ontología seria inconsistente:

ClassAssertion(:Mujer :Juan)

Regla para la herencia total

En la *Regla para la herencia exclusiva* se mostró como el axioma de clase DisjointClasses(CE_1 ... CE_n) puede ser usado para representar la herencia exclusiva. En la *Regla para la herencia total-exclusiva*, presentada anteriormente, se definió la herencia total como una restricción que define que toda instancia de la superclase debe estar representada en *al menos una* instancia de las subclases y se mostró como el axioma de clase DisjointUnion(C CE_1 ... CE_n) puede ser usado para representar esta restricción. En la misma regla también se indicó que el axioma de clase DisjointUnion(C CE_1 ... CE_n) es una representación corta del significado expresado por los axiomas EquivalentClasses(C ObjectUnionOf(CE_1 ... CE_n)) y DisjointClasses(CE_1 ... CE_n), es decir las dos expresiones representan lo mismo.

Teniendo en cuenta lo anterior, en el presente proyecto se consideró que al remover el axioma DisjointClasses(CE_1 ... CE_n), que fue empleado para representar la herencia exclusiva, de la representación en forma alargada de la herencia total-exclusiva, se conservaría una restricción que puede ser usada para representar la herencia total. Por esta razón para representar dicha restricción en OWL2, se empleó la expresión de clase ObjectUnionOf(CE_1 … CE_n) y el axioma de clase EquivalentClasses(CE_1 … CE_n).

La expresión de clase ObjectUnionOf(CE_1 … CE_n) permite representar todos los individuos que son instancias de al menos una de las expresiones de clase CE_i donde $1 \leq i \leq n$. El axioma de clase EquivalentClasses(CE_1 … CE_n) indica que todas las expresiones de clase CE_i donde $1 \leq i \leq n$, son semánticamente equivalentes una de otra. En este caso el axioma de clase usado tiene la forma EquivalentClasses(C CE), donde C es una clase y CE es una expresión de clase. Esta forma de axioma de clase con frecuencia es llamada *definición* porque define la clase C en términos de la expresión de clase CE (Motik, F. Patel-Schneider, et al., 2012).

De acuerdo a lo anterior la regla para representar la restricción asociada a una herencia total se definió así:

Si existe la tabla T_i tal que childs(T_i) ≠ Ø, y la relación entre la tabla T_i y las tablas childs(T_i) es una relación de herencia total, la clase derivada de T_i es definida como clase equivalente a la expresión de clase ObjectUnionOf(C_1 … C_n), donde C_1 … C_n son las clases derivadas de las tablas childs(T_i). Cuando childs(T_i) = {T_1,..,T_n} donde n ≥ 1, la declaración para esta restricción es la siguiente:

EquivalentClasses(:cls$_{Ti}$ ObjectUnionOf(:cls$_{T1}$ … :cls$_{Tn}$))

A continuación se presenta un ejemplo que emplea esta declaración:

SubClassOf(:Profesor :Persona)	Un profesor es una persona
SubClassOf(:Estudiante :Persona)	Una estudiante es una persona
EquivalentClasses(:Persona : Profesor :Estudiante)	Cada profesor es una persona, cada estudiante es una
ClassAssertion(:Profesor :Juan)	Juan es una profesor
ClassAssertion(:Estudiante :Juan)	Juan es una estudiante

Las últimas dos declaraciones establecen que Juan es un estudiante y un profesor, además de establecer a Juan como una persona. En este caso la superclase

pertenece a más de una subclase, lo cual es correcto porque la declaración no restringe lo contrario.

4.5 ALGORITMO

Las propuestas que definen reglas para la transformación de RDB a OWL, que fueron caracterizadas en la sección 3, presentan un conjunto de reglas aisladas, es decir, cada regla define las condiciones que debe cumplir un elemento del RDBS para generar un segmento OWL que lo represente. Ninguna de las propuestas presenta una estrategia para articular el conjunto de reglas de manera que se pueda realizar la transformación del RDBS como un todo.

Con el fin de diseñar e implementar un servicio que permita transformar un RDBS a OWL2, que esté soportado en el conjunto de reglas propuesto por (Choi & Kim, 2012), incluyendo las reglas para representar las restricciones asociadas a las relaciones de herencia que fueron definidas anteriormente y la regla redefinida en la misma sección, se diseñó un algoritmo que acopla todas las reglas mencionadas para transformar un RDBS por completo. Las figuras 5 y 6 describen el algoritmo propuesto.

4.6 DESARROLLO DEL PROTOTIPO

El prototipo fue desarrollado siguiendo las seis fases de la metodología XP presentadas en (Sánches, 2004), las cuales son presentadas a continuación:

4.6.1 Fase de exploración

En esta fase se elaboraron dos historias de usuario que representan las dos funcionalidades requeridas por el prototipo. Una funcionalidad consiste en mostrar las relaciones de herencia identificadas en un RDBS y la otra consiste en la transformación del RDBS a OWL2. Las historias de usuario pueden ser vistas en el Anexo 7.

```
Inicio
        Para cada tabla de la BD hacer:
                Declaration(Class(:cls_tabla)) //R1

                Si tabla tiene hijos
                        segun tipo de herencia hacer:
                                caso esclusiva:
                                        DisjointClasses(:cls_Child1 ... :cls_Childn) //RN1
                                caso total y exclusiva
                                        DisjointUnion(:cls_tabla :cls_Child1 ... :cls_Childn) // RN2
                                caso total:
                                        EquivalentClasses(:tabla ObjectUnionOf(:cls_Child1 ... :cls_Childn)) // RN3
                                otro
                                        no hacer nada
                        fin segun
                fin si
                Si tiene Padre entonces
                        SubClassOf(:cls_tabla :cls_Parent) //R18
                fin si
                Para cada columna de la tabla hacer:
                        Declaration(DataProperty(:dp_columna)) //R4

                        Si no tiene Enum ni tiene Check
                                DataPropertyRange(:dp_columna :tipoColumna) //R9
                        Fin si

                        Si tiene Enum
                                DataPropertyRange(:dp_columna DAtaOneOf(valor1 ... valorn)) //R10
                        Fin si

                        Si tiene Check
                                DataPropertyRange(:dp_columna DatatypeRestriction(restriccion)) //R11
                        Fin si

                        Si es llave primaria, foranea o unica entonces
                                Declaration(Class(:cls_columna)) //R2
                                Declaration(ObjectProperty(:op_columna))  //R3
                                ObjectPropertyDomain(:op_columna :cls_tabla)  //R5
                                ObjectPropertyRange(:op_columna :cls_columna)  //R6
                                DataPropertyDomain(:dp_columna :cls_columna)  //R7

                                Si es llave primaria y tiene padre
                                        SubObjectPropertyOf(:op_columna :op_columna_padre) //R25
                                Fin si
```

Figura 5. Algoritmo - Primera parte

```
                                        HasKey(:cls_columna ()(:dp_columna)) //13
                                        Si es lave unica
                                                InverseFuncionalObjectProperty(:op_columna) //14
                                        Fin si

                                        Si la llave es foranea
                                                SubClassOf(:cls_columna :cls_columna_referenciada) //15
                                                SubDataPropertyOf(:dp_columna :dp_columna_referenciada) //R26
                                        Fin si

                                        SubClassOf(:cls_columna DataAllValuesFrom(:dp_columna dpr_columna)) //R16. dpr_columna es el
rango de la columna
                                        SubClassOf(:cls_columna DataExactCardinality(:dp_columna dpr_columna)) //R17. dpr_columna es el
rango de la columna
                                        SubClassOf(:cls_tabla ObjectAllValuesFrom(:op_columna :cls_columna)) //R19

                                        Si la columna no puede ser nula
                                                SubClassOf(:cls_tabla ObjectExactCardinality(1 :op_columna :cls_columna)) //R20
                                        Si no
                                                SubClassOf(:cls_tabla ObjectMaxCardinality(1 :op_columna :cls_columna)) //R21
                                        Fin si
                                Si no
                                        DataPropertyDomain(:dp_columna :cls_tabla)  //R8
                                        SubClassOf(:cls_tabla DataAllValuesFrom(:dp_columna dpr_columna)) //22. dpr_columna es el rango de
la columna

                                        Si la columna no puede ser nula
                                                SubClassOf(:cls_tabla DataExactCardinality(1 :dp_columna dpr_columna)) //R23. dpr_columna
es el rango de la columna
                                        Si no
                                                SubClassOf(:cls_tabla DataMaxCardinality(1 :dp_columna :cls_columna)) //R24. dpr_columna
es el rango de la columna
                                        Fin si
                                Fin si
                        Fin para

                        HasKey(:cls_tabla (op_columna1 ... op_columnan)()) //R12. donde op_columnaj es llave primaria

                        Si la tabla no tiene padre y alguna columna que no es llave foranea es llave primaria o unica
                                DisjointClasses(:cls_tabla :cls_columna1 ... :clas_columnan) //R27 cls_columnaj son las clases derivadas de todas
las columnas de la tabla
                        Fin si
                Fin para
Fin
```

Figura 6. Algoritmo - Segunda parte

Se estableció que el RDBS seria extraído de la BD a través de una conexión remota y que el servicio propuesto estaría disponible como un servicio web. Por esta razón se diseñó la arquitectura inicial del sistema como se muestra en la **¡Error! No se encuentra el origen de la referencia.**.

Considerando que una futura herramienta basada en el servicio propuesto requerirá ofrecer un adecuado nivel de seguridad para el manejo de la conexión remota con la BD, se estableció que el desarrollo del prototipo se haría utilizando tecnologías de la plataforma Java EE (Java Enterprise Edition) como JAX-WS (Java API for XML Web Service) para el desarrollo del servicio propuesto y de un cliente para dicho servicio, y JDBC (Java Database Connectivity) para la conexión con la BD. Además se decidió que se usaría el entorno de desarrollo de NetBeans para facilitar las actividades de codificación.

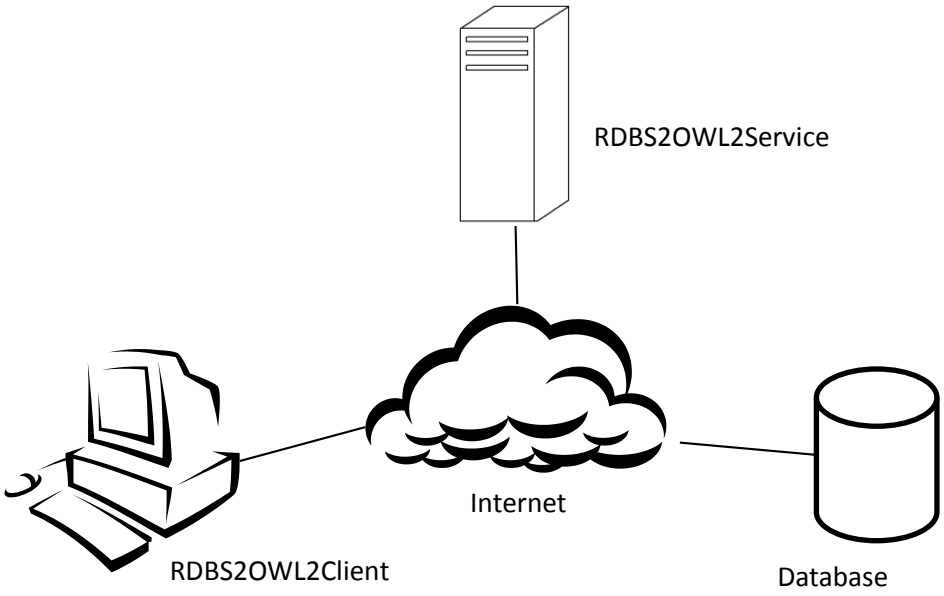

Figura 7. Arquitectura inicial del sistema

4.6.2 Fase de planificación de la entrega

En esta fase se determinó que la prioridad en el desarrollo seria la historia de usuario HU-2 debido a que esta representa la mayor parte de la funcionalidad del servicio propuesto. También se definió que el prototipo seria desarrollado en dos iteraciones. En la primera iteración se desarrollaría las operaciones ofrecidas por el servicio y en la segunda iteración se desarrollaría el cliente del servicio y se haría la integración del sistema.

4.6.3 Fase de iteraciones

Iteración I – Implementación del servicio

i) Diseño

Arquitectura del sistema

La **¡Error! No se encuentra el origen de la referencia.** muestra la arquitectura física del sistema.

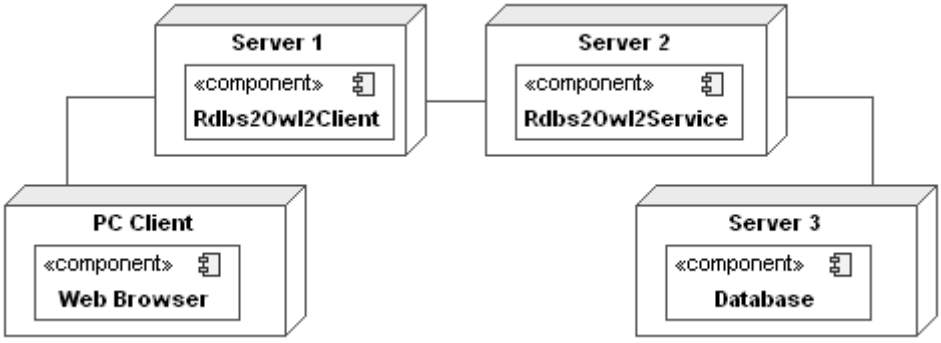

Figura 8. Diagrama de despliegue del sistema

Definición del servicio

El servicio propuesto ofrece dos operaciones que requieren enviar o recibir una lista de listas. Debido a que una lista no puede ser descrita en WSDL (Web Service

Description Language), se adoptó un formato especial de cadena para representar una lista donde sus elementos están delimitados por los caracteres “[]” y cada elemento de la lista está separado por una coma. Así, la cadena “[1,2,3]” representa una lista con tres elementos y la cadena “[[1,2][3,4,5]]” representa una lista que contiene dos listas, la primera con dos elementos y la segunda con tres elementos.

La primera operación *getInheritance* retorna un listado de las relaciones de herencia identificadas en el RDBS. El retorno es una lista de listas (en el formato que se describió anteriormente) donde cada lista contiene las tablas implicadas en una relación de herencia, siendo el primer elemento de la lista la tabla padre y los demás elementos las tablas hijas. Esta operación recibe cuatro parámetros: la url donde se encuentra la DB, el nombre de la DB, el usuario y contraseña de la DB.

La segunda operación *getOntology* retorna la ontología que se obtiene de la transformación del RDBS. Esta operación recibe cinco parámetros: la url donde se encuentra la DB, el nombre de la DB, el usuario y contraseña de la DB, y una lista de listas (en el formato que se describió anteriormente) donde cada lista contiene tres elementos. El primero corresponde a la tabla padre, el segundo indica si la relación es exclusiva o no, y el tercero indica si la relación es total o no. Los dos últimos solo pueden tomar los valores “true” o “false”. Las figuras 9 y 10 muestran la descripción del servicio.

Diagrama de paquetes y clases

La estructura del servicio propuesto está organizada como lo muestra la **¡Error! No se encuentra el origen de la referencia.**.

A continuación se presenta una breve descripción de cada paquete:

RDBS2OWL2: contiene todos los paquetes que ofrecen las funcionalidades necesarias para la transformación de un RDBS a OWL2. Estos paquetes son:

Database: contiene las clases necesarias para extraer la información del RDBS requerida por el algoritmo.

```
<definitions targetNamespace="http://service/" name="rdbs2owl2WebService">
	<types>
		<xs:schema version="1.0" targetNamespace="http://service/">
			<xs:element name="getInheritance" type="tns:getInheritance"/>
			<xs:element name="getInheritanceResponse" type="tns:getInheritanceResponse"/>
			<xs:element name="getOntology" type="tns:getOntology"/>
			<xs:element name="getOntologyResponse" type="tns:getOntologyResponse"/>
			<xs:complexType name="getOntology">
				<xs:sequence>
					<xs:element name="server" type="xs:string" minOccurs="0"/>
					<xs:element name="database" type="xs:string" minOccurs="0"/>
					<xs:element name="user" type="xs:string" minOccurs="0"/>
					<xs:element name="password" type="xs:string" minOccurs="0"/>
					<xs:element name="inheritance" type="xs:string" minOccurs="0"/>
				</xs:sequence>
			</xs:complexType>
			<xs:complexType name="getOntologyResponse">
				<xs:sequence>
					<xs:element name="return" type="xs:string" minOccurs="0"/>
				</xs:sequence>
			</xs:complexType>
			<xs:complexType name="getInheritance">
				<xs:sequence>
					<xs:element name="server" type="xs:string" minOccurs="0"/>
					<xs:element name="database" type="xs:string" minOccurs="0"/>
					<xs:element name="user" type="xs:string" minOccurs="0"/>
					<xs:element name="password" type="xs:string" minOccurs="0"/>
				</xs:sequence>
			</xs:complexType>
			<xs:complexType name="getInheritanceResponse">
				<xs:sequence><xs:element name="return" type="xs:string" minOccurs="0"/>
				</xs:sequence>
			</xs:complexType>
		</xs:schema>
	</types>
	<message name="getInheritance">
		<part name="parameters" element="tns:getInheritance"/>
	</message>
	<message name="getInheritanceResponse">
		<part name="parameters" element="tns:getInheritanceResponse"/>
	</message>
	<message name="getOntology">
		<part name="parameters" element="tns:getOntology"/>
	</message>
```

Figura 9. Definición del servicio - Primera parte

```
	<message name="getOntologyResponse">
		<part name="parameters" element="tns:getOntologyResponse"/>
	</message>
	<portType name="rdbs2owl2WebService">
		<operation name="getInheritance">
			<input wsam:Action="http://service/rdbs2owl2WebService/getInheritanceRequest" message="tns:getInheritance"/>
			<output wsam:Action="http://service/rdbs2owl2WebService/getInheritanceResponse"
message="tns:getInheritanceResponse"/>
		</operation>
		<operation name="getOntology">
			<input wsam:Action="http://service/rdbs2owl2WebService/getOntologyRequest" message="tns:getOntology"/>
			<output wsam:Action="http://service/rdbs2owl2WebService/getOntologyResponse"
message="tns:getOntologyResponse"/>
		</operation>
	</portType>
	<binding name="rdbs2owl2WebServicePortBinding" type="tns:rdbs2owl2WebService">
		<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
		<operation name="getInheritance">
			<soap:operation soapAction=""/>
			<input>
				<soap:body use="literal"/>
			</input>
			<output>
				<soap:body use="literal"/>
			</output>
		</operation>
		<operation name="getOntology">
			<soap:operation soapAction=""/>
			<input>
				<soap:body use="literal"/>
			</input>
			<output>
				<soap:body use="literal"/>
			</output>
		</operation>
	</binding>
	<service name="rdbs2owl2WebService">
		<port name="rdbs2owl2WebServicePort" binding="tns:rdbs2owl2WebServicePortBinding">
			<soap:address location="http://localhost:8080/rdbs2owl2/rdbs2owl2WebService"/>
		</port>
	</service>
</definitions>
```

Figura 10. Definición del servicio - Segunda parte

OWL: contiene las clases necesarias para crear los elementos OWL2 requeridos por el algoritmo y generar la ontología.

Logic: contiene la lógica necesaria para transformar un RDBS a OWL2. Hace uso de los paquetes Database y OWL para cumplir su objetivo.

Service: contiene una clase que actúa como interfaz entre el cliente y el servicio.

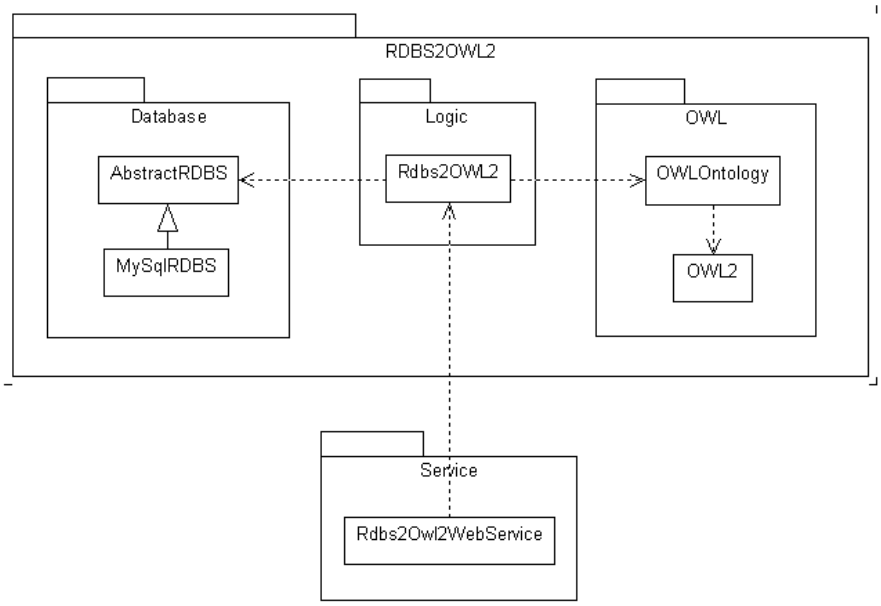

Figura 11. Diagrama de paquetes del servicio

A continuación se presenta una breve descripción de las clases:

AbstractRDBS: es una clase abstracta que define los métodos con los parámetros y retornos necesarios para el correcto funcionamiento del algoritmo. Esta clase permite extender el prototipo para transformar bases de datos diferentes a MySql.

MySqlRDBS: extiende la clase AbstractRDBS e implementa sus métodos abstractos para extraer información del esquema de una DB MySql.

OWLOntology: crea y organiza la estructura de la ontología generada por el algoritmo.

OWL2: ofrece las funciones necesarias para crear los elementos OWL2 requeridos por la ontología.

Rdbs2Owl2: implementa la lógica del algoritmo propuesto en el presente proyecto.

Rdbs2Owl2WebService: contiene funciones que responden a las solicitudes del cliente.

ii) Implementación

La **¡Error! No se encuentra el origen de la referencia.** muestra los componentes del servicio.

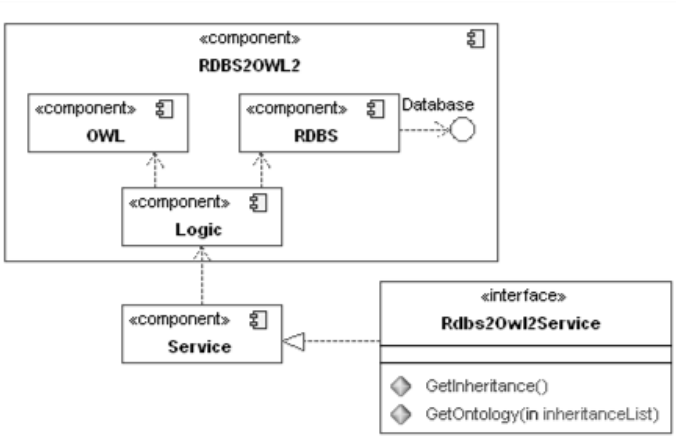

Figura 12. Diagrama de componentes del servicio

iii) Pruebas Unitarias

Las pruebas unitarias del servicio se realizaron a medida que se implementaba cada regla de transformación. Una vez se implementaba una regla, se ejecutaba el código suministrándole una base de datos en la que existieran elementos que cumplieran la regla, y se verificaba si el resultado era correcto o no. Estas pruebas unitarias estaban centradas en la verificación de la correcta estructuración de los elementos OWL2 que se generaban en la transformación, así como la correcta estructuración de la ontología que se obtenía al finalizar una trasformación.

Además se ejecutó el proceso de evaluación del prototipo. Los resultados de estas pruebas son presentados en el siguiente capítulo en la sección 5.2.

Iteración II – Implementación del cliente

i) Diseño

Diagrama de paquetes y clases

La estructura del cliente está organizada como lo muestra la **¡Error! No se encuentra el origen de la referencia.**.

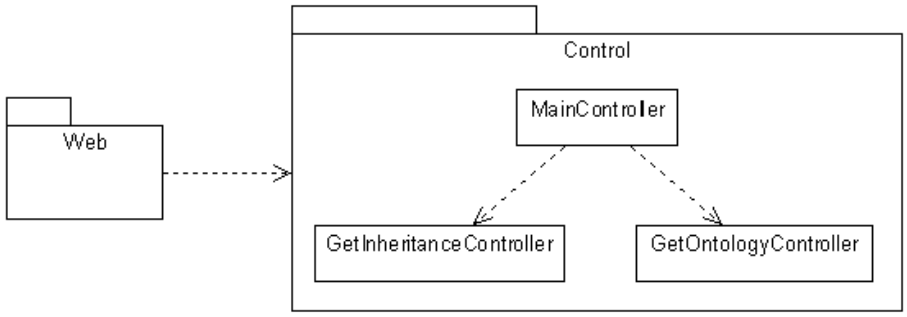

Figura 13. Diagrama de paquetes del cliente

A continuación se presenta una breve descripción de cada paquete:

Web: contiene las vistas de la página web.

Controller: contiene las clases necesarias para gestionar el flujo de mensajes entre las vistas del cliente y las operaciones del servicio.

A continuación se presenta una breve descripción de las clases:

MainController: recibe los mensajes del cliente, verifica que la información sea correcta e invoca los métodos correspondientes a la petición del cliente.

GetInheritanceController: invoca la operación *getInheritance* del servicio enviando los parámetros necesarios, recibe la respuesta y la pasa a las vistas del cliente.

GetOntologyController: invoca la operación *getOntology* del servicio enviando los parámetros necesarios, recibe la respuesta y la pasa a las vistas del cliente.

ii) Implementación

La **¡Error! No se encuentra el origen de la referencia.** muestra los componentes del cliente.

Figura 14. Diagrama de componentes del cliente

iii) Pruebas unitarias

Se realizaron pruebas para garantizar la correcta navegabilidad entre las vistas de la página y se realizaron pruebas para verificar la correcta conversión de la información de las relaciones de herencia al formato requerido por el servicio, el cual fue presentado en la *Definición del servicio*.

iv) Pruebas de integración

Se realizaron pruebas para verificar que el cliente y el servicio podían comunicarse enviando y recibiendo mensajes entre sí, y para verificar que las respuestas del servicio correspondieran a las peticiones solicitadas por el cliente. Estas pruebas fueron realizadas sobre un entono común, es decir tanto cliente como servicio desplegados en un mismo servidor.

4.6.4 Fase de producción

En esta fase el sistema fue desplegado en un entorno distribuido para verificar su correcto funcionamiento, para lo cual el cliente y el servicio fueron desplegados en servidores diferentes, y se realizaron transformaciones con bases de datos remotas para comprobar el correcto flujo de mensajes entre las partes.

4.6.5 Fase de mantenimiento

En esta fase se mejoraron las interfaces graficas del cliente adicionando hojas de estilos en casca (CSS, por su sigla en inglés) y se incluyeron algunas interfaces estáticas que contienen información acerca del proyecto y de sus autores.

4.6.6 Fase de muerte

En esta fase se elaboró el manual de usuario donde se detalla la funcionalidad ofrecida por el cliente y su forma de uso, y el manual técnico donde se detallan los aspectos relacionados con la arquitectura del servicio. El manual de usuario puede ser visto en el Anexo 2 y el manual técnico en el Anexo 3. Con la elaboración de estos documentos se terminó la construcción del prototipo.

5 EVALUACIÓN DEL PROTOTIPO

5.1 INTRODUCCIÓN

Durante el desarrollo del presente trabajo se construyó un prototipo que implementa el algoritmo propuesto en la sección 4.3, que adapta el conjunto de reglas propuesto en (Choi & Kim, 2012) y las nuevas reglas definidas en la misma sección. Con el fin de comprobar la validez del prototipo desarrollado en relación al cumplimiento del conjunto de reglas, se hizo necesario diseñar un proceso que permite evaluar la precisión y completitud de las ontologías generadas por el prototipo.

En el presente trabajo se entiende por precisión el *"grado en que un producto o sistema proporciona los resultados correctos con el grado de precisión necesario"* y por completitud el *"grado en que el conjunto de funciones cubre todas las tareas especificadas y los objetivos de usuario"* (ISO/IEC 25010, 2011).

El presente capítulo está dividido en tres secciones. En la primera sección se describe en detalle el proceso de evaluación definido para evaluar el prototipo, en la segunda sección se presentan los resultados obtenidos de la ejecución del proceso de evaluación, y finalmente, en la tercera sección se presentan las conclusiones.

5.2 DEFINICIÓN DEL PROCESO DE EVALUACIÓN

Para evaluar la precisión y la completitud del prototipo, se definió un proceso de evaluación que permite determinar si el prototipo cumple con las reglas de transformación (precisión) y si todos los elementos del RDBS, considerados en las reglas, son transformados a OWL (completitud). La **¡Error! No se encuentra el origen de la referencia.** representa el proceso de evaluación definido.

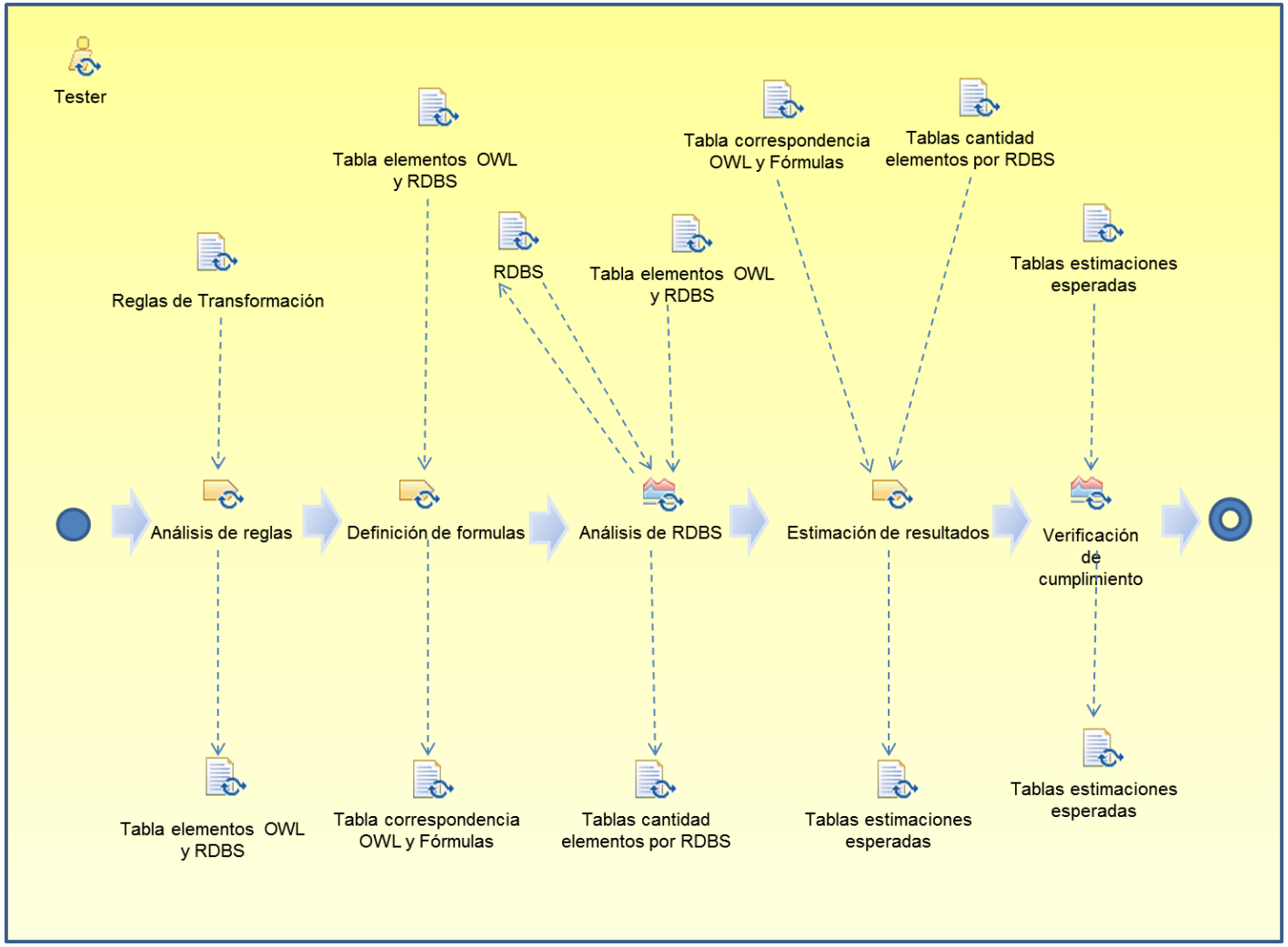

Figura 15. Diagrama del proceso de evaluación

A continuación se detalla el proceso de evaluación:

5.2.1 Análisis de reglas

En esta fase se analiza cada una de las reglas con el fin de identificar que elementos del RDBS y que elementos de OWL están asociados a cada regla. Esto permitirá identificar que elementos de OWL se deben generar por cada elemento del RDBS.

Para presentar esta información se utilizó el formato presentado en la Tabla 6.

Tabla 6. Formato para los datos del Análisis de Reglas

Elemento RBDS	**Elemento OWL**	**Regla**
Elemento_RDBS_1	*Elemento_OWL_1*	*R1*
	Elemento_OWL_2	*R2*
Elemento_RDBS_2	*Elemento_OWL_1*	*R3*
	Elemento_OWL_4	*R4*
	Elemento_OWL_3	*R5*
Elemento_RDBS_3	*Elemento_OWL_2*	*R6*

Cada elemento de la columna *Elemento RDBS* debe ser descrito claramente de manera que se identifique y se diferencie de los demás.

5.2.2 Definición de formulas

En esta fase se debe identificar, tomando como base la tabla obtenida en la fase anterior, que elementos del RDBS generan un mismo elemento de OWL y de esta manera establecer las fórmulas que permiten determinar la cantidad que se espera obtener de cada elemento OWL al aplicar el conjunto de reglas.

Para presentar esta información se utilizó el formato presentado en la Tabla 7.

Tabla 7. Formato para la Definición de Formulas

Elementos OWL	Formula
Elemento_OWL_1	*#Elemento_RDBS_1 + #Elemento_RDBS_2*
Elemento_OWL_2	*#Elemento_RDBS_1 + # Elemento_RDBS_3*
Elemento_OWL_3	*#Elemento_RDBS_2*
Elemento_OWL_4	*#Elemento_RDBS_2*

5.2.3 Análisis de los RDBS

En esta fase se define el número de RDBS que serán empleados en la prueba y se analizan por separado con el fin de establecer el número de ocurrencias de cada uno de los elementos RDBS que fueron identificados en la fase de *Análisis de reglas***.**

Para presentar esta información se utilizó el formato presentado en la Tabla 8.

Tabla 8. Formato para el Análisis de los RDBS

Elementos RBDS	**Cantidad**
Elemento_RDBS_1	*# Elemento_RDBS_1*
Elemento_RDBS_2	*# Elemento_RDBS_2*
Elemento_RDBS_3	*# Elemento_RDBS_3*

Nota: Se debe llenar un formato por cada RDBS.

5.2.4 Estimación de resultados esperados

En esta fase se hace una estimación de la cantidad de cada uno de los elementos OWL que se espera obtener en la ontología generada por el prototipo. Esta estimación se hace aplicando las formulas obtenidas en la fase 2 con los datos de la Tabla 8.

Para presentar esta información se utilizó el formato presentado en la Tabla 9.

Tabla 9. Formato para las estimaciones

Elemento OWL	Cantidad esperada	Cantidad obtenida	% Error
Elemento_OWL_1			
Elemento_OWL_2			
Elemento_OWL_3			
Elemento_OWL_4			
Total			

Nota: En esta fase solo se llena la columna *Cantidad esperada*, con los resultados de las formulas. La columna *Cantidad obtenida* se llena en la siguiente fase al igual que la columna *% Error*.

5.2.5 Verificación del cumplimiento de las reglas

En esta fase se transforman los RDBS usando el prototipo y se analizan los elementos que conforman la ontología generada con el fin de contrastar el resultado obtenido de la transformación con los resultados esperados que fueron identificados en la fase anterior.

Esta fase es iterativa y culmina cuando una misma versión del prototipo haya transformado exitosamente todos los RDBS o se hayan realizado el número máximo de iteraciones definido previamente. Si se alcanza el número máximo de iteraciones sin la transformación exitosa de todos los RDBS, se toma la versión del prototipo con la que se obtuvo el mejor resultado. El prototipo cambia de versión cuando se hace algún cambio sobre el mismo y se considera una iteración a todas las transformaciones realizadas con una misma versión del prototipo.

Las siguientes tareas deben realizarse por cada RDBS analizado en la fase 3.

Transformación del RDBS usando el prototipo y análisis de resultados obtenidos

Se realiza la transformación del RDBS usando la última versión del prototipo. Una vez se obtiene la ontología generada por el prototipo, se procede a establecer la cantidad de cada uno de los elementos OWL que están presentes en la ontología y con esta información se llena la columna *Cantidad obtenida* de la Tabla 9.

Calculo del porcentaje de error

En este paso se calcula el error para cada uno de los elementos OWL obtenidos en la ontología y se llena la columna *% Error* de la Tabla 9. El porcentaje de error se obtiene aplicando la formula ((Cantidad esperada – Cantidad Obtenida) / 100).

Si se presenta un error mayor a 0 en alguno de los elementos OWL, se debe determinar, con ayuda de la Tabla 7, que elementos del RDBS están asociados a dicho elemento OWL con el fin de detectar los posibles orígenes de la falla que está produciendo el error. Una vez identificado el origen del error se deben hacer los cambios pertinentes al prototipo y volver a ejecutar las tareas de la fase 5 sobre el actual RDBS.

Cuando se superen todos los errores para el actual RDBS se obtiene una nueva versión del prototipo. Con la nueva versión del prototipo, se ejecuta nuevamente la fase 5 del proceso con todos los RDBS transformados anteriormente y con los que falten por transformar. Esto se hace con el fin de asegurar que los cambios realizados al prototipo no hayan afectado la transformación de elementos RDBS diferentes a los que produjeron el error.

Si no se presentan errores y hay más RDBS para probar, se debe ejecutar nuevamente las tareas de la fase 5 con el siguiente RDBS, de lo contrario terminan las iteraciones y termina el proceso de evaluación.

5.3 EJECUCIÓN DEL PROCESO DE EVALUACIÓN

5.3.1 Análisis de reglas

En la Tabla 10 se relaciona cada elemento del RDBS con los elementos OWL que se obtienen de una transformación, de acuerdo al conjunto de reglas propuesto por (Choi & Kim, 2012) y a las nuevas reglas propuestas en el presente proyecto. La Tabla 10 muestra, para el caso del elemento RDB *Tablas*, que según las reglas R1 y R12, se deben generar por cada tabla los elementos OWL *class* y *HasKey con ObjectProperty* respectivamente. El resto de la tabla debe ser interpretado de la misma manera.

Tabla 10. Resultado del Análisis de Reglas

Elemento RBD	**Elementos OWL**	**Regla**
Tablas	Class	R1
	HasKey con ObjectProperty	R12
Campos	DataProperty	R4
Campos llave	Class	R2
	ObjectProperty	R3
	ObjectProperyDomain	R5
	ObjectPropertyRange	R6
	DataPropertyDomain	R7
	HasKey con DataProperty	R13
	SubClassOf con DataAllValuesFrom	R16
	SubClassOf con DataExactCardinality	R17
	SubClassOf con ObjectAllValuesFrom	R19
Campos llave y no nulos	SubClassOf con ObjectExactCardinality	R20
Campos llave y nulos	SubClassOf con ObjectMaxCardinality	R21
Campos llave	SubObjectPropertyOf	R25

primaria y tabla tiene padre[10]		
Campos llave foránea	SubClassOf	R15
	SubDataPropertyOf	R26
Campos únicos	InverseFunctionalObjectProperty	R14
Campos no llave	DataPropertyDomain	R8
	SubClassOf con DataAllValuesFrom	R22
Campos no llave y no nulos	SubClassOf con DataExactCardinality	R23
Campos no llave y nulos	SubClassOf con DataMaxCardinality	R24
Campos check	DataPropertyRange con DatatypeRestriccion	R11
Campos enum	DataPropertyRange con DataOneOf	R10
Tabla tiene Padre	SubClassOf	R18
Campos no check y no enum	DataPropertyRange	R9
Campos no llave foránea y, llave primaria o único, y tabla no tiene padre	DisjointClasses	R27
Relaciones herencia exclusiva	DisjointClasses	RN1
Relaciones herencia total-exclusiva	DisjointUnion	RN2

[10] En (Choi & Kim, 2012) se define una tabla padre de la siguiente forma: Si Ti y Tj son tablas con i ≠ j y existe una correspondencia de uno a uno entre sus llaves primarias, entonces Tj es llamada padre de la tabla Ti.

Relaciones herencia total	EquivalentClasses con ObjectUnionOf	RN3

En la Tabla 11 se describe cada elemento RDBS mencionado en la tabla anterior.

Tabla 11. Descripción de los elementos RDBS considerados por las reglas

Elemento RBD	Descripción
Tablas	Son las tablas del RDBS.
Campos	Son los campos o columnas de cada tabla del RDBS.
Campos llave	Son campos que han sido definidos como primary key, foreign key o unique key.
Campos llave y no nulos	Son campos que han sido definidos como primary key, foreign key o unique key, y que no pueden ser nulos.
Campos llave y nulos	Son campos que han sido definidos como primary key, foreign key o unique key, y que pueden ser nulos.
Campos llave primaria y tabla tiene padre	Son campos definidos como primary key y pertenecen a una tabla que tiene padre.
Campos llave foránea	Son campos definidos como foreign key.
Campos únicos	Son campos definidos como unique key.
Campos no llave	Son campos que no han sido definidos como primary key, foreign key o unique key.
Campos no llave y no nulos	Son campos que no han sido definidos como primary key, foreign key o unique key, y que no pueden ser nulos.
Campos no llave y nulos	Son campos que no han sido definidos

	como primary key, foreign key o unique key, y que pueden ser nulos.
Campos check	Son campos que tienen una restricción check.
Campos enum	Son campos que tienen una restricción enum.
Tabla tiene padre	(Choi & Kim, 2012) define una tabla padre de la siguiente forma: Si Ti y Tj son tablas con i ≠ j y existe una correspondencia de uno a uno entre sus llaves primarias, entonces Tj es llamada padre de la tabla Ti.
Campos no check y no enum	Son campos que no tienen restricción check ni enum.
Campos no llave foránea y, llave primaria o único, y tabla no tiene padre	Son campos que no han sido definidos como foreign key, que han sido definidos como primery key o unique key, y pertenecen a una tabla que no tiene padre.
Relaciones herencia exclusiva	Son relaciones de herencia entre tablas que representan herencia de tipo exclusivas.
Relaciones herencia total-exclusiva	Son relaciones de herencia entre tablas que representan herencia de tipo total y exclusiva.
Relaciones herencia total	Son relaciones de herencia entre tablas que representan herencia de tipo total.

5.3.2 Definición de formulas

La Tabla 12 muestra las fórmulas que se obtuvieron a partir de la información contenida en la Tabla 10, que permitieron establecer la cantidad de cada elemento OWL que se esperaba obtener en la trasformación de los RDBS.

Tabla 12. Resultado de la Definición de Fórmulas

Elementos OWL	**Fórmulas**
Class	#Tablas + #campos llave
ObjectProperty	#campos llave
DataProperty	#campos
ObjectProperyDomain	#campos llave
ObjectPropertyRange	#campos llave
DataPropertyDomain	#campos
HasKey con ObjectProperty	#Tablas
HasKey con DataProperty	#campos llave
SubClassOf con DataAllValuesFrom	#campos
SubClassOf con DataExactCardinality	#campos llave + #Campos no llave y no nulos
SubClassOf con ObjectAllValuesFrom	#campos llave
SubClassOf	#campos llave foránea + #Tabla tiene padre
SubDataPropertyOf	#campos llave foránea
InverseFunctionalObjectProperty	#campos únicos
DataPropertyRange con DatatypeRestriccion	#campos check
DataPropertyRange con DataOneOf	#campos enum
DataPropertyRange	#campos no check y no enum
SubClassOf con	#campos llave y no nulos

ObjectExactCardinality	
SubClassOf con ObjectMaxCardinality	#campos llave y nulos
SubClassOf con DataMaxCardinality	#campos no llave y nulos
SubObjectPropertyOf	#campos llave primaria y tiene padre
DisjoinClasses	# campos no llave foránea y, llave primaria o único, y tabla no tiene padre + #relaciones herencia exclusiva
DisjointUnion	# relaciones herencia total-exclusiva
EquivalentClasses con ObjectUnionOf	# Relaciones herencia total

Nota: los elementos OWL *DataPropertyDomain* y *SubClassOf con DataAllValuesFrom*, de acuerdo al análisis de la Tabla 12, tienen como fórmula (#campos llave + #Campos no llave) que es equivalente a #campos.

5.3.3 Análisis de los RDBS

Para verificar el cumplimiento de las reglas se emplearon cuatro RDBS compuestos por diferentes elementos de RDB y que en conjunto abarcan todos los casos tratados por el conjunto de reglas propuesto por (Choi & Kim, 2012) y las nuevas reglas propuestas en el presente proyecto. Para garantizar la consistencia de los RDBS usados en la evaluación, se tomó como referencia algunos ejemplos presentados en (Cuadrada et al., 2008).

A continuación se presenta una breve descripción de cada RDBS:

RDBS 1 - Observación de aves: base de datos que contiene información sobre las observaciones realizadas a distintas especies de aves en la Península Ibérica.

RDBS 2 - Proyectos de investigación: base de datos que almacena la información concerniente a los proyectos de investigación tanto actuales como pasados en los que trabajan los profesores de una universidad.

RDBS 3 - Alojamientos rurales: base de datos que guarda información sobre los alojamientos rurales que existen en una comunidad.

RDBS 4 - Gestión de proyectos informáticos: base de datos para gestionar los proyectos informáticos que desarrolla una empresa de consultoría para sus empresas clientes.

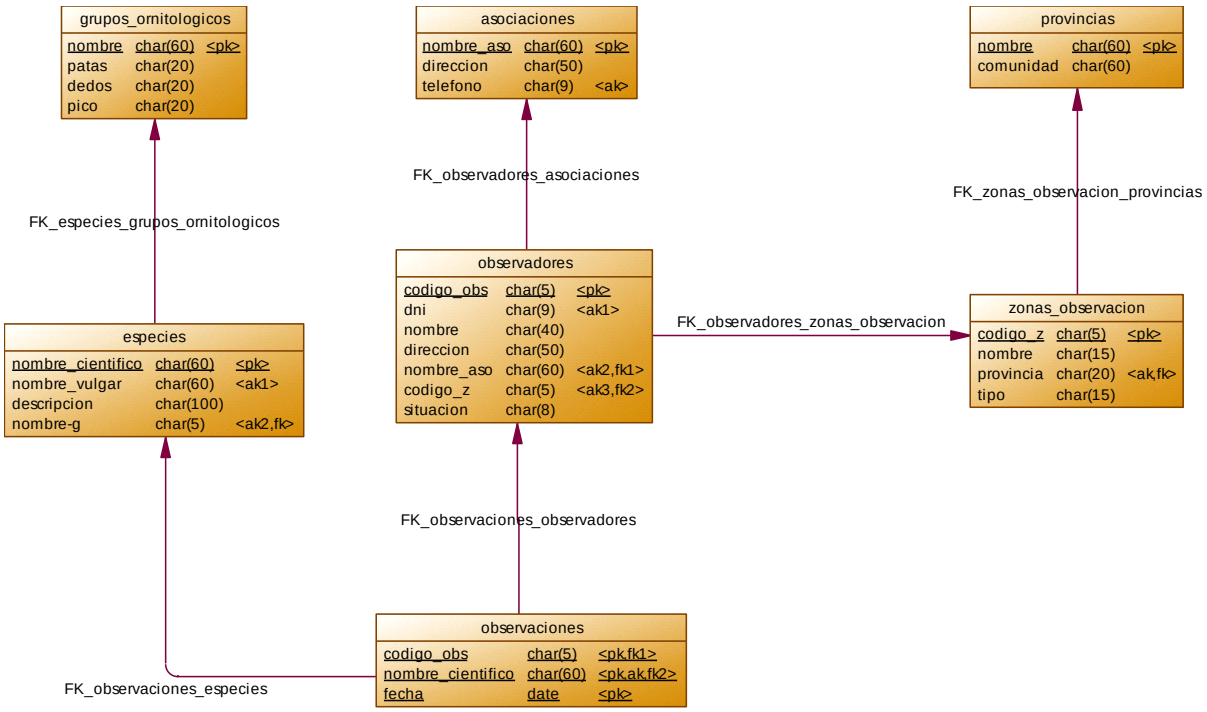

Figura 16. Esquema relacional de la DB Observación de Aves

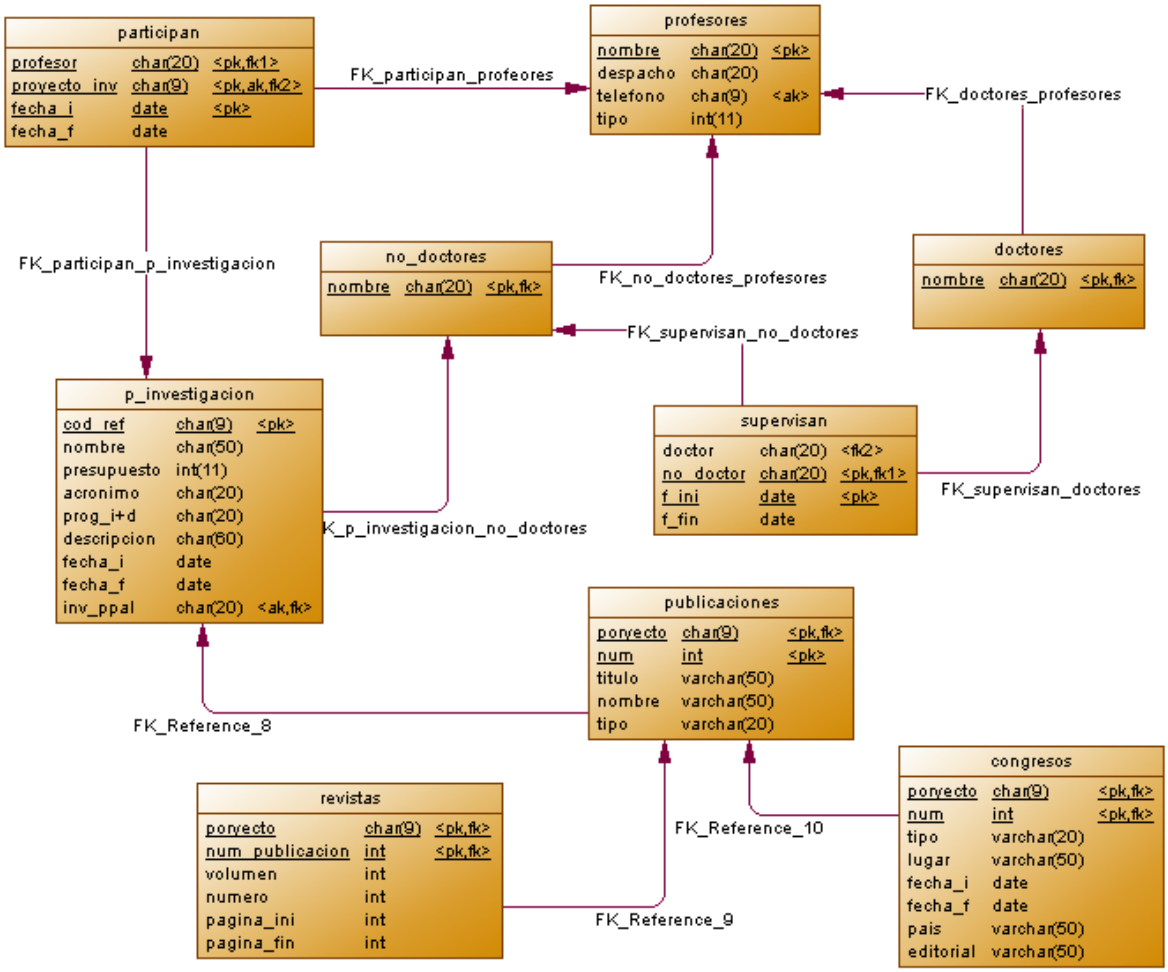

Figura 17. Esquema relacional de la DB Proyectos de Investigación

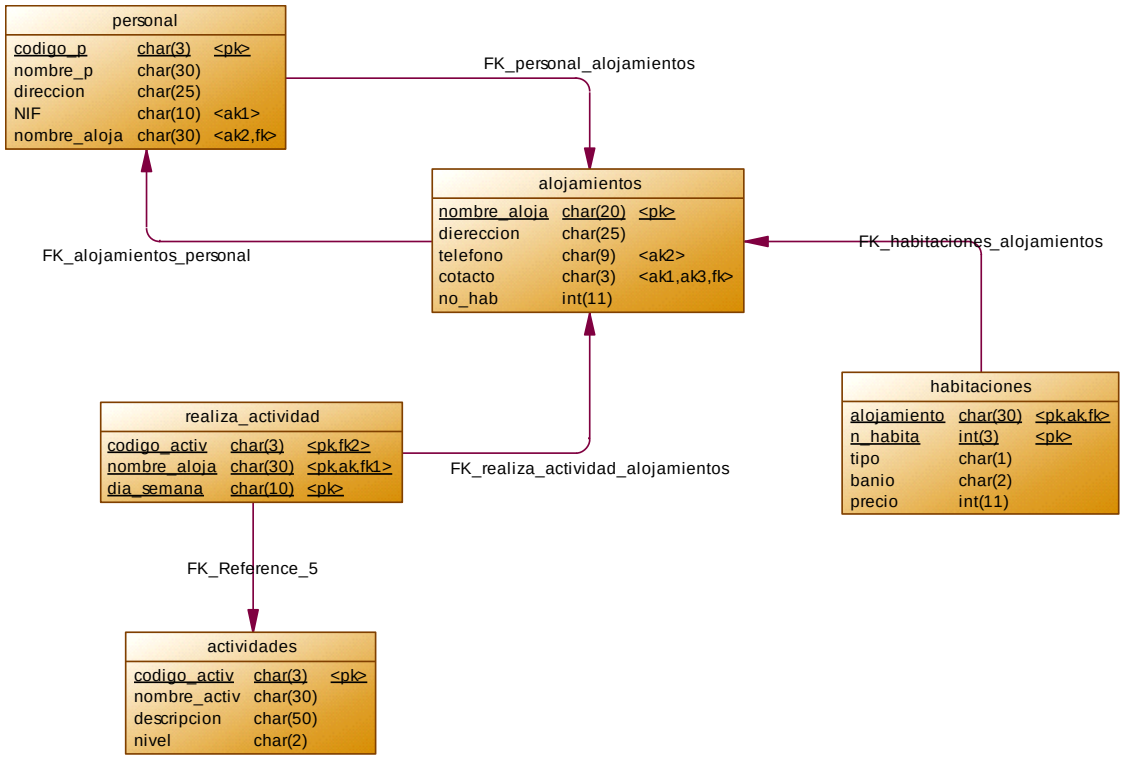

Figura 18. Esquema relacional de la DB Alojamientos Rurales

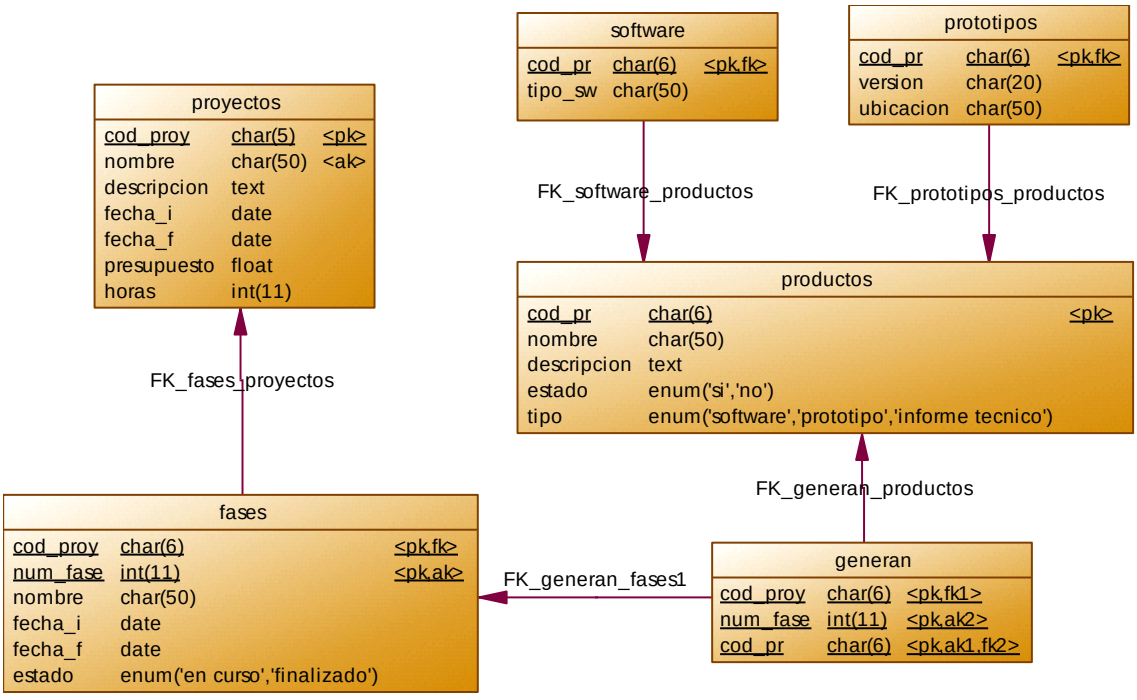

Figura 19. Esquema relacional de la DB Gestión de Proyectos Informáticos

Los scripts con los RDBS de las bases de datos presentadas anteriormente pueden verse en el Anexo 4**.**

La Tabla 13 muestra los resultados del análisis de los RDBS presentados anteriormente, indicando la cantidad de cada uno de los elementos RDBS identificados en la fase1.

Tabla 13. Resultados del Análisis de los RDBS

Elemento RBDS	RDBS 1	RDBS 2	RDBS 3	RDBS 4
Tablas	7	9	5	6
Campos	27	42	22	26
Campos llave	16	17	12	10
Campos llave y no nulos	14	17	11	10
Campos llave y nulos	2	0	1	0
Campos llave primaria y tabla tiene padre	0	6	0	2
Campos llave foránea	6	10	5	6
Campos únicos	3	1	3	1
Campos no llave	11	25	10	16
Campos no llave y no nulos	8	25	9	16
Campos no llave y nulos	3	0	1	0
Campos check	0	0	0	0
Campos enum	0	0	0	3
Tabla tiene padre	0	4	0	2
Campos no check y no enum	27	42	22	23
Campos no llave foránea y, llave primaria o único, y tabla no tiene padre	7	5	0	3
Relaciones herencia exclusiva	0	1	0	0

Relaciones herencia total-exclusiva	0	1	0	0
Relaciones herencia total	0	0	0	1

5.3.4 Estimación de resultados esperados

La Tabla 14 presenta la cantidad esperada de elementos OWL para cada RDBS. Estos valores fueron obtenidos aplicando las formulas presentadas en la Tabla 12.

Tabla 14. Resultados de las estimaciones

Elemento OWL	**RDBS 1**	**RDBS 2**	**RDBS 3**	**RDBS 4**
Class	23	26	17	16
ObjectProperty	16	17	12	10
DataProperty	27	42	22	26
ObjectProperyDomain	16	17	12	10
ObjectPropertyRange	16	17	12	10
DataPropertyDomain	27	42	22	26
HasKey con ObjectProperty	7	9	5	6
HasKey con DataProperty	16	17	12	10
SubClassOf con DataAllValuesFrom	27	42	22	26
SubClassOf con DataExactCardinality	24	42	21	26
SubClassOf con ObjectAllValuesFrom	16	17	12	10
SubClassOf	6	14	5	8
SubDataPropertyOf	6	10	5	6
InverseFunctionalObjectProperty	3	1	3	1
DataPropertyRange con	0	0	0	0

DatatypeRestriccion				
DataPropertyRange con DataOneOf	0	0	0	3
DataPropertyRange	27	42	22	23
SubClassOf con ObjectExactCardinality	14	17	11	10
SubClassOf con ObjectMaxCardinality	2	0	1	0
SubClassOf con DataMaxCardinality	3	0	1	0
SubObjectPropertyOf	0	6	0	2
DisjoinClasses	7	6	5	3
DisjointUnion	0	1	0	0
EquivalentClasses con ObjectUnionOf	0	0	0	1
Total	283	385	222	233

5.3.5 Verificación del cumplimiento de reglas

Para la verificación del cumplimiento de reglas se estableció que se realizarían máximo seis iteraciones de las cuales solo se realizaron cuatro, debido a que en la cuarta se logró transformar exitosamente todos los RDBS. En las cuatro iteraciones se ejecutó un total de ocho transformaciones, con los respectivos análisis de resultados obtenidos y cálculo del porcentaje de error. Como resultado de las iteraciones se obtuvo 4 versiones del prototipo, de las cuales la versión 4 transformó exitosamente todos los RDBS.

A continuación se presenta un resumen de las actividades realizadas y los resultados obtenidos en cada iteración:

Iteración I

El RDBS 1 (Observación de Aves) fue transformado usando la versión 1 del prototipo. Durante esta transformación se presentó un error en la ejecución producida por un error de implementación que fue corregido y se obtuvo la versión 2 del prototipo, con la cual se ejecutó una nueva transformación. En esta transformación no se presentó ningún error entre los resultados esperados y los resultados obtenidos, por lo cual se procedió con la transformación del RDBS 2 usando la misma versión del prototipo, la versión 2.

Iteración II

El RDBS 2 (Proyectos de Investigación) fue transformado usando la versión 2 del prototipo. En esta transformación no se presentó ningún error entre los resultados esperados y los resultados obtenidos, por lo cual se procedió con la transformación del RDBS 3 usando la versión 2 del prototipo.

El RDBS 3 (Alojamientos rurales) fue transformado usando la versión 2 del prototipo. En esta transformación no se presentó ningún error entre los resultados esperados y los resultados obtenidos, por lo cual se procedió con la transformación del RDBS 4 usando la versión 2 del prototipo.

El RDBS 4 (Gestión de proyectos informáticos) fue transformado usando la versión 2 del prototipo. En esta transformación se presentó errores en los resultados de los elementos OWL *SubClassOf* y *SubObjectPropertyOf*. El porcentaje de Error Total obtenido fue 0.04. Estos errores fueron corregidos hasta que el error total se hizo igual a cero, obteniendo la versión 3 del prototipo, con el cual se ejecutó una nueva iteración, transformando los RDBS anteriores.

Iteración III

El RDBS 1 (Observación de Aves) fue transformado usando la versión 3 del prototipo. En esta transformación se presentó errores en los resultados de los

elementos OWL *SubClassOf, SubObjectPropertyOf y DisjoinClasses.* El porcentaje de Error Total obtenido en esta transformación fue de 0.06. Estos errores fueron corregidos hasta que el error total se hizo igual a cero y se obtuvo la versión 4 del prototipo, con la cual se ejecutó una nueva iteración, transformando los RDBS anteriores.

Iteración IV

El RDBS 2 (Proyectos de Investigación) fue transformado usando la versión 4 del prototipo. En esta transformación no se presentó ningún error entre los resultados esperados y los resultados obtenidos, por lo cual se procedió con la transformación del RDBS 3 usando la versión 4 del prototipo.

El RDBS 3 (Alojamientos rurales) fue transformado usando la versión 4 del prototipo. En esta transformación no se presentó ningún error entre los resultados esperados y los resultados obtenidos, por lo cual se procedió con la transformación del RDBS 4 usando la versión 4 del prototipo.

El RDBS 4 (Gestión de proyectos informáticos) fue transformado usando la versión 4 del prototipo**.** En esta transformación no se presentó ningún error entre los resultados esperados y los resultados obtenidos.

Con esta transformación se terminaron las iteraciones debido a que la versión 4 del prototipo transformó exitosamente los cuatro RDBS destinados para la prueba, obteniendo un error total de cero (0) en cada transformación. Los resultados obtenidos en cada transformación y los cálculos del porcentaje de error respectivo pueden verse en el Anexo 5.

5.4 CONCLUSIONES

El resultado obtenido en la última iteración de la fase 5, durante la ejecución del proceso de evaluación del prototipo, donde se obtuvo un error de cero (0%), permitió determinar que el prototipo desarrollado en el presente trabajo satisface, con un alto nivel de precisión y completitud, el conjunto de reglas propuesto por

(Choi & Kim, 2012) y las nuevas reglas propuestas en el capítulo anterior. Lo anterior, debido a que las transformaciones realizadas por la versión 4 del prototipo transformaron todos los elementos presentes en los RDBS que son considerados por las reglas (completitud) y los elementos OWL obtenidos en las ontologías corresponden con los resultados esperados para cada transformación (precisión).

Sin embargo, no se puede garantizar que la versión 4 del prototipo, que para el presente trabajo se considera la versión final, transforme correctamente cualquier RDBS, debido a que el porcentaje de error obtenido, el cual es nulo, resultó de una serie de modificaciones realizadas que surgieron de errores identificados durante la ejecución del proceso de evaluación.

6. CONCLUSIONES

En este trabajo de grado se ha descrito un servicio para la transformación de RDBS a OWL, que es el resultado del desarrollo del proyecto de investigación denominado “Transformación de Esquemas de Bases de Datos Relacionales al Lenguaje de Ontologías Web” y cuyos objetivos corresponden a: i) Caracterización de los mecanismos de transformación existentes, ii) Diseño de un servicio semiautomático, iii) Implementación del servicio en un prototipo y iv) Evaluación de la funcionalidad del prototipo.

El cumplimiento del primer objetivo permitió identificar los elementos de los RDBS que han sido considerados en las propuestas que define reglas formales de transformación, que cumplen el estándar OWL y que usan el enfoque Direct Mapping. Además permitió evidenciar la necesidad de definir reglas formales para la representación de los diferentes tipos de herencia.

El cumplimiento del segundo objetivo permitió obtener nuevas reglas de transformación, definidas formalmente, para la representación en OWL2 de los diferentes tipos de herencia (total, exclusiva y total-exclusiva). Además, se obtuvo un algoritmo para transformar un RDBS, en el que se adapta y articula el conjunto de reglas propuesto en (Choi & Kim, 2012) con las reglas propuestas en el presente proyecto.

Con el cumplimiento del tercer y cuarto objetivo se logró obtener un prototipo de servicio para la transformación de un RDBS Mysql a OWL2, que implementa el algoritmo propuesto en el presente proyecto y que alcanzó, durante la evaluación realizada, un nivel alto de precisión y completitud.

Este capítulo está dividido en dos secciones. En la primera sección se presentan las conclusiones del trabajo de investigación, conclusiones generadas a partir de los resultados obtenidos a través del cumplimiento de cada uno de los objetivos. En la segunda sección se plantea una serie de recomendaciones que pueden ser

útiles para investigadores que aborden la temática relacionada con el presente proyecto, y se plantea una serie de trabajos futuros que permitirían consolidar los resultados obtenidos.

6.1 CONCLUSIONES DEL TRABAJO DE INVESTIGACIÓN

La caracterización realizada a las propuestas que usan el enfoque Direct Mapping permitió establecer que solo tres de estas definen reglas para la identificación y representación en OWL de las relaciones de herencia. En una de las propuestas, la regla definida para la herencia es susceptible de errores ya que puede considerar como relación de herencia a una relación simple entre dos tablas. Cada una de estas propuestas define solo una regla para la herencia y en ninguna de estas se definen reglas para representar los diferentes tipos de herencia. Así mismo, la caracterización realizada sobre los estudios que usan el enfoque ER to OWL, permitió establecer que desde esta perspectiva tampoco se han definido reglas para la representación de las restricciones asociadas a los diferentes tipos de herencia.

La caracterización también permitió establecer que no existen implementaciones de las propuestas que usan el enfoque Direct Mapping, que cumplan con el estándar OWL definido por el W3C y que soporten la transformación en reglas definidas formalmente.

Por lo anterior, en el presente proyecto se definieron nuevas reglas formales de transformación que permitieron representar las restricciones asociadas a los diferentes tipos de herencia y que sirvieron para complementar el conjunto de reglas propuesto en (Choi & Kim, 2012), el cual fue tomado como base para la construcción del prototipo de servicio para la transformación de RDBS a OWL2.

El prototipo desarrollado fue evaluado mediante la definición y ejecución de un proceso de evaluación que permitió obtener una versión del prototipo que satisface, con un alto nivel de precisión y completitud, el conjunto de reglas propuesto en

(Choi & Kim, 2012) y adaptado para el servicio propuesto. El alto nivel de completitud se debe a que el prototipo transformó todos los elementos considerados por las reglas de transformación y que estaban presentes en los RDBS, y el alto nivel de precisión se debe a que los elementos OWL obtenidos en las ontologías corresponden con los resultados esperados para cada transformación.

Sin embargo, no se puede garantizar que la versión final del prototipo, obtenida del proceso de evaluación, transforme correctamente cualquier RDBS, debido a que el porcentaje de error obtenido, el cual es nulo, resultó de una serie de modificaciones realizadas que surgieron de errores identificados durante la ejecución del proceso de evaluación. Por esta razón se propone que el porcentaje de error promedio obtenido en el presente proyecto, el cual es 0.013, sea tomado como referencia para determinar si una transformación realizada por el prototipo desarrollado en el presente proyecto, cuenta con un nivel alto de precisión y completitud, es decir que una transformación cuyo porcentaje de error sea menor o igual a 0.013 puede ser considerada como confiable, de lo contrario no se puede garantizar su confiabilidad.

6.2 RECOMENDACIONES Y TRABAJO FUTURO

Como recomendación para los investigadores que aborden la transformación de RDBS a ontologías OWL, se sugiere que durante la investigación se estudien propuestas cuyos aportes relacionados con la transformación estén soportados en reglas formales. De la misma manera se sugiere que los futuros aportes a la transformación sean expresados en forma de reglas, que deben ser totalmente claras, evitando la generación de confusiones o ambigüedades. Es importante que las reglas propuestas en futuros proyectos de investigación sean basadas en estructuras generales de un RDBS y no en casos particulares asociados a un contexto, es decir que las reglas no sean producto del contexto representado por la base de datos.

Como trabajo futuro se propone verificar la validez de las reglas propuestas en el presente proyecto y de las ontologías generadas por el prototipo sometiéndolas al juicio de expertos. Esto con el fin de garantizar que los conceptos relacionados con los diferentes tipos de herencia, son representados adecuadamente por las estructuras de OWL2 empleadas en cada una de las reglas presentadas.

Se propone evaluar el prototipo realizando transformaciones con RDBS de mayor tamaño, es decir con RDBS que tengan un mayor número de elementos, con el fin de analizar y mejorar el comportamiento del prototipo bajo este tipo de situaciones, en términos de los tiempos de respuesta.

Finalmente, se propone como trabajo futuro extender el prototipo de manera que soporte transformaciones para otros tipos de motores de bases de datos relacionales y por medios diferentes a conexiones remotas, por ejemplo a través de un script. Y extender el prototipo para transformar los registros de las bases de datos a instancias de la ontología generada, para lo cual pueden ser usar las mismas reglas de transformación de instancias propuestas en (Choi & Kim, 2012).

BIBLIOGRAFIA

Astrova, I. (2009). Rules for Mapping SQL Relational Databases to OWL Ontologies. *Springer Link*, 415–424.

Barker, K., & Alhajj, R. (2006). RDB2ONT: A Tool for Generating OWL Ontologies From Relational Database Systems. *Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services (AICT-ICIW'06)*, 170–170. doi:10.1109/AICT-ICIW.2006.159

Belhadef, H., Ouafek, N., & Kholladi, M. K. (2008). A set of mapping rules E/R and relational schema database towards an ontology. *2008 First International Conference on the Applications of Digital Information and Web Technologies (ICADIWT)*, (1), 289–295. doi:10.1109/ICADIWT.2008.4664361

Būmans, G. (2010). RDB2OWL : a Practical Approach for Transforming RDB Data into RDF / OWL, 1–3.

Chen, A., Liu, L., & Shang, J. (2012). A Hybrid Strategy to Construct Scientific Instrument Ontology from Relational Database Model. *2012 International Conference on Computer Distributed Control and Intelligent Environmental Monitoring*, 25–33. doi:10.1109/CDCIEM.2012.14

Choi, J. W., & Kim, M. H. (2012). Generating OWL Ontology from Relational Database. *2012 Third FTRA International Conference on Mobile, Ubiquitous, and Intelligent Computing*, 53–59. doi:10.1109/MUSIC.2012.17

Cuadrada, D., Castro, E., Iglesisa, A. M., Martínez, P., Calle, F. J., De Pablo, C., … Moreno, L. (2008). Desarrollo de Bases de Datos: casos practicos desde el diseño a la implementación. In Alfaomega Grupo Editor (Ed.), (1st ed., pp. 212–327). Mexico city: RA-MA Editotial.

Dong, H., Sun, J., & Wu, Q. (2010). The semantic description and logical proof on ER diagram based on the OWL language. *2010 5th International Conference on Computer Science & Education*, 1813–1815. doi:10.1109/ICCSE.2010.5593802

Fahad, M. (2008). ER2OWL : Generating OWL Ontology from, *288*, 28–37.

Gherabi, N., Addakiri, K., & Bahaj, M. (2012). Discovering new technique for mapping relational database based on semantic web technology. *2012 International Conference on Communications and Information Technology (ICCIT)*, 27–32. doi:10.1109/ICCITechnol.2012.6285808

Golbreich, C., & Wallace, E. K. (2012). OWL 2 Web Ontology Language New Features and Rationale (Second Edition). Retrieved October 29, 2013, from http://www.w3.org/TR/2012/REC-owl2-new-features-20121211/

Hazber, M., Yang, J., & Jin, Q. (2010). Towards Integration Rules of Mapping from Relational Databases to Semantic Web Ontology. *2010 International Conference on Web Information Systems and Mining*, 335–339. doi:10.1109/WISM.2010.21

Hitzler, P., Krötzsch, M., Parsia, B., F. Patel-Schneider, P., & Rudolph, S. (2012). OWL 2 Web Ontology Language Primer (Second Edition). Retrieved October 17, 2013, from http://www.w3.org/TR/2012/REC-owl2-primer-20121211/#Appendix:_The_Complete_Sample_Ontology

Horrocks, I., Ruttenberg, A., Hawke, S., & Herman, I. (2012). OWL Working Group. Retrieved October 17, 2013, from http://www.w3.org/2007/OWL/wiki/OWL_Working_Group

ISO/IEC 25010. (2011). Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models.

Levshin, D. V. (2009). Mapping Relational Databases to the Semantic Web with Original Meaning, 5–16.

Motik, B., F. Patel-Schneider, P., & Parsia, B. (2012). OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax (Second Edition). Retrieved November 02, 2013, from http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/

Motik, B., Grau, B. C., Horrocks, I., Wu, Z., Fokoue, A., & Lutz, C. (2012). OWL 2 Web Ontology Language Profiles (Second Edition). Retrieved October 29, 2013, from http://www.w3.org/TR/2012/REC-owl2-profiles-20121211/

Myroshnichenko, I., & Murphy, M. C. (2009). Mapping ER Schemas to OWL Ontologies. *2009 IEEE International Conference on Semantic Computing*, 324–329. doi:10.1109/ICSC.2009.61

Ouyang, D., Cui, X., & Ye, Y. (2010). Mapping integrity constraint ontology to relational databases. *The Journal of China Universities of Posts and Telecommunications*, *17*(6), 113–121. doi:10.1016/S1005-8885(09)60534-3

Pino, F. J., García, F., & Piattini, M. (2007). Software process improvement in small and medium software enterprises: a systematic review. *Software Quality Journal*, *16*(2), 237–261. doi:10.1007/s11219-007-9038-z

Sánches, C. (2004). Ciclo de vida de un proyecto XP. Retrieved November 25, 2013, from http://oness.sourceforge.net/proyecto/html/ch05s02.html

Santoso, H. A., Haw, S.-C., & Abdul-Mehdi, Z. T. (2011). Concept hierarchy as background knowledge. *Knowledge-Based Systems*, *24*(3), 457–464. doi:10.1016/j.knosys.2010.11.003

Sequeda, J. F., Tirmizi, S. H., Corcho, O., & Miranker, D. P. (2011). *Survey of directly mapping SQL databases to the Semantic Web*. *The Knowledge Engineering Review* (Vol. 26, pp. 445–486). doi:10.1017/S0269888911000208

Smith, M. K., Welty, C., & McGuinness, D. L. (2004). OWL Web Ontology Language Guide. Retrieved October 25, 2013, from http://www.w3.org/TR/owl-guide/

Vavliakis, K. N., Grollios, T. K., & Mitkas, P. a. (2012). RDOTE - Publishing Relational Databases into the Semantic Web. *Journal of Systems and Software*, 1–11. doi:10.1016/j.jss.2012.07.018

Vavliakis, K. N., Symeonidis, A. L., Karagiannis, G. T., & Mitkas, P. a. (2011). An integrated framework for enhancing the semantic transformation, editing and querying of relational databases. *Expert Systems with Applications*, *38*(4), 3844–3856. doi:10.1016/j.eswa.2010.09.045

W3C. (2013). SEMANTIC WEB. Retrieved October 18, 2013, from http://www.w3.org/standards/semanticweb/

W3C OWL Working, G. (2012). OWL 2 Web Ontology Language Document Overview (Second Edition). Retrieved October 29, 2013, from http://www.w3.org/TR/2012/REC-owl2-overview-20121211/

Web-Ontology Working Group. (2004). OWL Web Ontology Language Overview. Retrieved August 23, 2013, from http://www.w3.org/TR/2004/REC-owl-features-20040210/

Wells, D. (2009). Extreme Programming: A gentle introduction. Retrieved September 10, 2013, from http://www.extremeprogramming.org/index.html

Xu, Z., Cao, X., Dong, Y., & Su, W. (2004). Formal Approach and Automated Tool for Translating ER Schemata into OWL Ontologies, 464–475.

Yajai, A., & Sriharee, G. (2012). EERtoOWL2: A Tool for Transforming RDB Data to OWL2 for Data Validation. *2012 IEEE 24th International Conference on Tools with Artificial Intelligence*, 970–975. doi:10.1109/ICTAI.2012.137

Yang, S., & Wu, J. (2010). Mapping Relational Databases into Ontologies through a Graph-based Formal Model. *2010 Sixth International Conference on Semantics, Knowledge and Grids*, 219–226. doi:10.1109/SKG.2010.33