Extensión del framework OWLAPI para la administración y razonamiento sobre grandes ontologías

Manuel E. Puebla-Martínez[1], José M. Perea-Ortega[2], and Alfredo Simón-Cuevas[3]

[1] Universidad de las Ciencias Informáticas, La Habana, Cuba
mpuebla@uci.cu
[2] Universidad de Extremadura, Badajoz, España
jmperea@unex.es
[3] Universidad Tecnológica de La Habana "José Antonio Echeverría", Cujae La Habana, Cuba
asimon@ceis.cujae.edu.cu

Abstract. Nowadays, managing and reasoning on large ontologies is considered a resource-intensive task, mainly due to the complexity of the reasoning algorithms involved. In addition, most of the existing tools to manage large ontologies require high-performance computers, which could make some users hard to find a feasible solution. This paper presents an approach to manage and reason on OWL2 large ontologies by using the OWLAPI framework. The main novelty focuses on the development of a software solution that allows managing large ontologies, together with the development of specific search procedures in OWLAPI that would require the use of reasoners in the case of this type of ontologies. The proposed approach was tested by generating a spatial ontology on the region of Marianao (Cuba) and launching several queries on it. The results show the great performance of the approach developed.

Keywords: Large Ontologies, Knowledge Representation, Reasoning on Large ontologies, OWLAPI, OWL2

Resumen. En la actualidad, la administración y el razonamiento en ontologías grandes se considera una tarea de uso intensivo de recursos, principalmente debido a la complejidad de los algoritmos de razonamiento involucrados. Además de este problema, la mayoría de las herramientas existentes para administrar grandes ontologías requieren computadoras de alto rendimiento, lo que podría hacer que para algunos usuarios fuese difícil encontrar una solución viable. En esta investigación se presenta un enfoque para gestionar y razonar sobre ontologías grandes utilizando el *framework* OWLAPI. La principal novedad se centra en el desarrollo de una solución de software que permite administrar ontologías grandes, unido al desarrollo de procedimientos de búsqueda específicos en OWLAPI que necesitarían el uso de razonadores para el caso de ontologías con características similares a las descritas en este trabajo. El enfoque propuesto fue probado generando una ontología espacial en la

región de Marianao (Cuba) y lanzando varias consultas al respecto. Los resultados muestran el rendimiento del enfoque desarrollado.

Palabras claves: Ontologías grandes, Representación del conocimiento, Razonando sobre ontologías grandes, OWLAPI, OWL2

1. Introducción

El uso de ontologías es cada vez más requerido por aquellos sistemas de información que necesitan procesar la semántica asociada al contenido, como por ejemplo los sistemas de Recuperación de Información Geográfica (GIR, siglas en inglés). En este ámbito, las ontologías resultan muy útiles para describir la semántica y relacionar los datos espaciales, generar nuevo conocimiento espacial y mejorar la toma de decisiones en este dominio. Sin embargo, el uso efectivo de las ontologías no solo requiere su codificación en un lenguaje formal, sino también requiere de un soporte adecuado de herramientas para su administración, que posibiliten el razonamiento automático o la inferencia a partir del conocimiento que representan. La presente investigación se desarrolla en el marco de un proyecto que tiene por objetivo el desarrollo de un GIR soportado en el uso de una ontología geográfica. La ontología se genera de forma semiautomática usando un método que aprovecha la diversidad de fuentes de información y ofrece la posibilidad de obtener una ontología que conceptualiza cualquier lugar geográfico.

En este contexto, fue utilizado dicho método para obtener una ontología geográfica de uno de los municipios de La Habana (Cuba), específicamente Marianao, la cual contenía más de cuatro mil individuos, 76 millones de propiedades de objetos, y más de 100 mil propiedades de datos; resultando ser una ontología grande. Si se construyese la ontología de toda Cuba o del Continente Americano, los valores antes señalados crecerían de manera exponencial pudiendo llegar a alcanzar los 30 GB de memoria o más. Esto condujo a plantearse el siguiente problema: ¿cómo administrar y razonar sobre ontologías con tales dimensiones? La mayor parte de los editores de ontologías hacen un uso intensivo de la memoria principal y no son particularmente adecuados para el razonamiento sobre ontologías con millones de instancias, como las que a menudo son necesarias para las aplicaciones que requieren representar y describir semánticamente datos espaciales del mundo real. La integración de las Bases de Datos (DB, siglas en inglés) parece ser la solución, ya que estas aprovechan las ontologías para incrementar su semántica, mientras que las ontologías se benefician de las DB para estructurar y almacenar grandes cantidades de instancias. Como resultado, muchas áreas de investigación están surgiendo y los trabajos resultantes de esta complementariedad son enormes [12].

OWLAPI (*Ontology Web Language* API, en inglés), es el *framework* utilizado para administrar la ontología a ser utilizada por el sistema GIR que se desarrolla. A favor de OWLAPI se dirá que es el *framework* utilizado por el editor

de ontologías más utilizado actualmente: *Protégé*[1]. Sin embargo, este *framework* no soporta ontologías como las que se generan en esta investigación, al menos con un hardware estándar, pues el mismo carga la información del fichero OWL en memoria RAM. Algo similar sucede con los razonadores que trabajan con memoria interna: *Pellet*, *FaCT++*, *HermiT*, *TrOWL*, *RACER*, entre otros. Estos, además de cargar la ontología, necesitan generar nuevo conocimiento, lo cual implica la ejecución de complejos algoritmos que deben verificar una gran variedad de restricciones debido al alto grado de expresividad del lenguaje OWL2. El editor de ontologías más popular, *Protégé*, tampoco es capaz de cargar una ontología con las características descritas en este artículo, utilizando el framework OWLAPI para administrar la ontología. Todos ellos generan el conocido error "*Out of Memory*" cuando se intenta cargar una ontología grande.

En este trabajo se presenta una extensión al *framework* OWLAPI para administrar y razonar sobre grandes ontologías, basada en los principios de un sistema OBDB (*Ontology Based Data Base*, en inglés). De este modo, una ontología se considera grande cuando su ABOX supere o iguale los valores obtenidos y ya descritos en la ontología generada de forma automática para este trabajo.

El ABOX se considera uno de los tres componentes en los que se divide conceptualmente una ontología, y contiene afirmaciones de rol entre individuos de la ontología (por ejemplo, *hasChild (John, Mary)*) y afirmaciones de pertenencia (por ejemplo, *(John:Man)*). La solución presentada se apoya en el *framework* OWLAPI y permite administrar y satisfacer las necesidades de razonamiento del futuro sistema GIR a desarrollar.

Por otro lado, el futuro sistema GIR a desarrollar necesitará trabajar con el lenguaje ontológico OWL2, y no con RDF, OWL, OWL2-QL o OWL2-RL; los cuales son lenguajes menos expresivos y utilizados en las herramientas encontradas en el análisis del estado del arte.

2. Trabajo relacionado

OWLAPI es una API de alto nivel para trabajar con ontologías OWL2, por lo que está estrechamente alineada con la especificación estructural OWL2. Soporta el análisis gramatical y la traducción en las sintaxis definidas en la especificación W3C (sintaxis funcional, RDF/XML, OWL/XML y la sintaxis de Manchester OWL); manipulación de estructuras ontológicas; y el uso de motores de razonamiento. La implementación de referencia de la OWLAPI, escrita en Java, incluye *validadores* para los distintos perfiles OWL 2 QL, OWL2 EL y OWL2 RL. El OWLAPI tiene un uso extendido en una variedad de herramientas y aplicaciones [9]. La mayor limitante de OWLAPI está en la necesidad de cargar el fichero OWL en memoria RAM, la cual es muy limitada si comparamos los valores medios actuales con el tamaño de las ontologías grandes.

Los sistemas DBBO (*DataBase Based on Ontologies*, en inglés), también conocidos como OBDA (*Ontology-Based Data Access*, en inglés), se han convertido en un popular paradigma para acceder a una o varias fuentes de datos

[1] `http://protege.stanford.edu`

mediante el uso de ontologías. Estos sistemas aprovechan las ontologías para incrementar su capacidad de administrar información semántica. En los sistemas DBBO, los usuarios acceden a los datos a través de una capa conceptual (abstracción de los aspectos específicos relacionados con las fuentes de datos), que proporciona un cómodo vocabulario de consulta. La capa conceptual es representada generalmente mediante una ontología en RDF u OWL, y se conecta a las bases de datos relacionales subyacentes utilizando asignaciones R2RML. Cuando se realiza una consulta SPARQL sobre la ontología, el sistema DBBO explora las asignaciones representadas para recuperar elementos de las fuentes de datos y construir las respuestas [4]. Los sistemas DBBO no permiten hacer cambios en la DB a través de la ontología, debido a que en los casos generales donde hay asignaciones complejas arbitrarias entre la ontología y la DB, este problema no admite una solución general y constituye un problema de investigación abierto hoy día. Esto es conocido en bases de datos como el "*view update problem*" [7]. La razón es que, en general, debido a las asignaciones, no hay una manera única para propagar una actualización específica del nivel de la ontología a la base de datos subyacente. Otra limitación de los sistemas DBBO es que no están diseñados para soportar ontologías arbitrarias en OWL2, debido a que su función principal es acceder a grandes DB a través de una ontología, por lo que en general se usan lenguajes menos expresivos como es el caso de OWL2-QL. Algunos ejemplos de sistemas DBBO son Ontop[2], Optique [3], GraphDB[3], RDFox[4] y OntoDB [5].

Por otra parte, en los últimos años han surgido los sistemas OBDB (Ontology Based DataBase, en inglés), un modelo que permite almacenar y consultar ontologías con una gran cantidad de instancias. Los sistemas OBDB también aprovechan los beneficios de las funcionalidades ofertadas por los Sistemas de Administración de Bases de Datos Relacionales (RDBMs, siglas en inglés), como son: rendimiento en las consultas, almacenamiento eficiente de los datos, administración de transacciones, entre otras. Ejemplos de estos sistemas son: *Sesame Database Manager*[5], DLDB-OWL [13], OWLIM [11], *InstanceStore* [10], *Minerva* [17], DBOWL [15], *OntoMinD* [1], *OwlOntDB* [6] y FGOLD [2].

En [17] se reporta una comparación entre *Minerva* y *Sesame Database Manager*, DLDB-OWL, OWLIM e *InstanceStore*, concluyéndose como ventajas del primero: 1) Soporta ontologías en OWL-DL con una inferencia completa sobre el TBOX, pero para ontologías en OWL-Lite con un máximo de tripletas RDF de 2.200.000. 2) El proceso de razonamiento y evaluación que se lleva a cabo como parte de las consultas se realizan en memoria externa, materializando todos los resultados de inferencia en la DB, lo que la hace una herramienta más adecuada para manejar grades ontologías. 3) La alta escalabilidad y optimización de consultas, tanto para Minerva como para DLDB-OWL. Sin embargo, Min-

[2] `http://ontop.inf.unibz.it`
[3] `http://ontotext.com/products/graphdb`
[4] `http://www.cs.ox.ac.uk/isg/tools/RDFox`
[5] `http://www.sesamedatabase.com`

erva no soporta OWL2, lo cual limita su nivel de expresividad. Tampoco soporta expresiones de clases en las consultas de usuarios.

Por otro lado, *OntoMinD* comparte con *Minerva* la segunda ventaja enunciada en el párrafo anterior, pero no deja claro si soporta OWL2. DBOWL es un razonador escalable, que soporta razonamiento completo OWL-DL para ontologías con ABOX bien grandes (billones de instancias). Sin embargo, a pesar de que está licenciado con licencia GNU-GPL según el sitio oficial de la universidad de Manchester, no se ha encontrado la manera de descargarlo y utilizarlo de manera local. Los autores solo brindan la posibilidad de utilizarlo a través de servicios web. *OwlOntDB* provee una completa cobertura de razonamiento sobre ontologías en OWL2-RL, no así sobre ontologías en OWL2, diferenciándose del resto de las herramientas analizadas. Sin embargo, no se ha logrado localizar la herramienta para evaluar su utilización. En [14] se considera la ontología SUMO (*Suggested Upper Merged Ontology*, en inglés) como una ontología grande, sin embargo la misma solo ocupa 36 MB en memoria. En [16] se propone el razonador *Chainsaw* para grandes ontologías, sin embargo, los propios autores reconocen que dicho razonador no es capaz de razonar sobre ontologías con las características mencionadas en este trabajo, debido a que no está preparado para dividir grandes ontologías sin ser cargadas previamente en memoria interna. En [8] se evalúan los sistemas *Sesame Database Manager* y DLDB-OWL con la utilización de "*Large OWL DataSets*". Los conjuntos de datos pequeños son de 15 ficheros OWL y un total de 8 MB, y los grandes conjuntos de datos poseen 999 y 583 MB. En [10] se propone cierto grado de escalabilidad con más de 100.000 individuos, especificando que el resto de las aplicaciones existentes fallan con esa cantidad. Solo el caso de DBOWL (billones de individuos) parece ser comparable con las ontologías abordadas en este trabajo en cuanto a cantidad de individuos, no así en cuanto a tamaño del fichero OWL, pues en la evaluación de DBOWL realizada en [15] solo utilizan ontologías de 100 y 200 MB.

Después de analizar las propuestas de otros autores, se concluye que las *grandes ontologías* de la mayoría de las investigaciones actuales no son comparables en cuanto a su dimensión con las ontologías que se presentan en este trabajo, las cuales superan en tamaño a las encontradas en el estado del arte. En el análisis del estado del arte sobre los sistemas DBBO se identificaron dos limitaciones fundamentales: 1) La imposibilidad de hacer cambios en las DB a través de la ontología generada (sólo se permite consultar la DB a través de la ontología). 2) No están diseñados para soportar ontologías arbitrarias en OWL2. Esta última limitación también es identificada en los sistemas OBDB, unido a la imposibilidad de trabajar con el concepto de ontología grande dado en esta investigación.

La solución propuesta en este artículo resuelve las limitaciones identificadas, aunque en cuanto a las potencialidades de razonamiento, solo satisface las condiciones de diseño y necesidades del futuro sistema GIR. Satisfacer todas las posibilidades de razonamiento sobre una ontología OWL2 con las características descritas, es un trabajo mucho más extenso y profundo, aunque viable en una futura investigación y parcialmente abordado en la solución OwlOntDB [6].

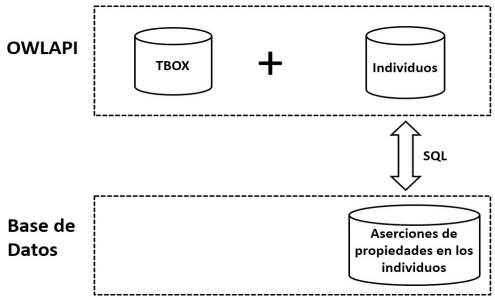

Fig. 1: Asignación de responsabilidades entre el *framework* OWLAPI y la Base de Datos

3. Solución propuesta

3.1. Administración de la ontología

Para lograr que el *framework* OWLAPI soportara la administración de grandes ontologías fue necesario extraer algunos datos del fichero OWL y almacenarlo en una DB. En la solución propuesta se decidió extraer del fichero OWL gestionado por OWLAPI, algunos datos pertenecientes al ABOX y mantener íntegramente el TBOX de la ontología, de forma similar a los sistemas OBDB analizados. Esto se hizo con el objetivo de poder utilizar los razonadores en memoria interna en algunas tareas específicas y así mantener la capacidad de razonamiento sobre la ontología. Los razonadores existentes están diseñados para realizar procesos de inferencia solo sobre lo incluido en el fichero OWL de la ontología, fundamentalmente sobre el TBOX. Los datos del ABOX almacenados en la DB fueron las relaciones de objeto expresadas entre pares de individuos (*Object Property Assertion Axioms*) y la asignación de valores a las propiedades de datos (*Data Property Assertion Axioms*). La Figura 1 ilustra de forma gráfica la asignación de responsabilidades entre el *framework* OWLAPI y la DB, en el momento de gestionar los componentes de la ontología.

Al menos para la ontología geográfica generada en este trabajo, esa modificación fue suficiente para lograr que el *framework* OWLAPI y el editor *Protégé* cargaran la información restante en el fichero OWL. Sin embargo, ninguno de los razonadores de memoria interna antes mencionados fue capaz de razonar sobre la ontología restante en el fichero OWL, al menos con el hardware utilizado en esta investigación, debido a que los individuos continúan siendo almacenados en el fichero OWL gestionado por OWLAPI. Debido a la simplicidad de la información a almacenar en memoria externa, inicialmente se utilizó como medio de almacenamiento en memoria externa los ficheros. El fichero donde se almacenaron las relaciones de objeto expresadas entre pares de individuos y la asignación de valores a las propiedades de datos obtuvo un tamaño de 10 GB. Al hacer varias pruebas, los autores notaron que el tiempo de acceso a la información era demasiado elevado y la frecuencia con la que se debía acceder a los

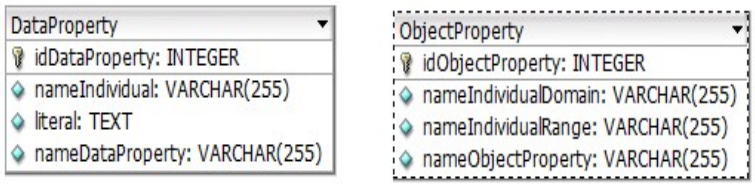

Fig. 2: Modelo físico de la DB utilizada

mismos también. Por tal motivo, se decidió utilizar una DB, cuyo modelo físico se ilustra en la Figura 2. La solución desarrollada consta de los mecanismos necesarios para que las instancias de propiedades puedan ser integradas nuevamente en su forma tradicional en el fichero OWL, siempre y cuando la memoria interna del hardware soporte el crecimiento de la ontología.

3.2. Razonando sobre grandes ontologías desde OWLAPI

Una vez cargada la ontología grande en el *framework* OWLAPI era necesario hacer algunas operaciones sobre ella que exigían el uso de los razonadores, lo cual continuaba siendo una limitación a pesar de las modificaciones realizadas. En este sentido, se implementaron las operaciones necesarias sin el uso de un razonador y solo apoyándose en los recursos brindados por OWLAPI. Las operaciones de inferencia desarrolladas fueron:

1. Buscar las subclases directas e indirectas de una clase.
2. Identificar los individuos directos e indirectos de una clase, teniendo en consideración la equivalencia entre clases y entre individuos.
3. Identificar las *clases topes*, es decir, las clases que no son subclases de ninguna clase.
4. Identificar todas las clases equivalentes a una clase, considerando que la relación de equivalencia es transitiva.
5. Identificar el conjunto de superclases de una clase.
6. Obtener el conjunto de individuos que están relacionados con otro a través de una propiedad de objeto, teniendo en consideración las características de la propiedad de objeto expresada en OWL: simetría, funcional, transitividad, inversa, equivalencia.
7. Obtener el conjunto de literales asociados a una propiedad de datos para un individuo en particular.

El Algoritmo 1 muestra el pseudocódigo implementado para obtener la operación de inferencia número uno, formado por dos métodos pertenecientes a la clase *Ontology* construida para encapsular el *framework* OWLAPI.

Algoritmo 1: Algoritmo de la operación de inferencia 1

Entrada: La clase a la cual se le buscarán las subclases (owlClass) y la condición para saber si las subclases a buscar serán directas o

```
 indirectas (direct).
Salida: Un conjunto con las subclases del parámetro owlClass.

Metodo1:
Set<OWLClass> GetSubclassNotReasoner (OWLClass owlClass, boolean direct)
Inicio
 1) Crear un conjunto vacío de objetos clases (owlClasses)
 2) Invocar y retornar el resultado del método GetSubclassNotReasoner
 que está sobrecargado, con los parámetros owlClass, owlClasses y direct
Fin

Metodo2 (sobrecarga al Método 1):
GetSubclassNotReasoner (OWLClass OwlClass, Set<OWLClass> listResult,
   boolean direct)
Inicio
 if (OwlClass != null) {
   - Guardar en list todos todos los axiomas de tipo subclass para la
          superclase OwlClass.
   - Para cada objeto presente en list hacer:
     Añadir a listResult la subclase directa que está en cada axioma
          almacenado en list.
 }
 if (!direct){  //es decir, si piden las subclases indirectas
  Para cada subclase directa de OwlClass hacer:
   - Invocar recursivamente al método
     GetSubclassNotReasoner(la subclase, listResult, direct)
 }
 Retornar listResult
Fin
```

4. Resultados

A través de la solución expuesta se logró administrar una ontología geográfica que conceptualiza los lugares geográficos del municipio Marianao, cuya composición y características se muestran en la Tabla 1. El fichero OWL (sin las relaciones de objeto expresadas entre pares de individuos) se ha dejado disponible para la comunidad científica[6].

Clases	722
Clases equivalentes	12
Relaciones taxonómicas o subclases	954
Individuos	4.665
Relaciones de objeto entre pares de individuos	76.398.888
Propiedades de datos con valor en individuos	102.755
Pares de individuos similares	13.321
Anotaciones	728
Tamaño en memoria externa (sin aserciones de propiedades)	15.310 KB

Table 1: Caracterización de la ontología geográfica del municipio Marianao

[6] http://sinai.ujaen.es/wp-content/uploads/2017/12/OntoMarianao-OWLfile-2MB.rar

En la Tabla 2 se muestran los resultados de las funcionalidades de inferencia desarrolladas sobre el *framework* OWLAPI para la ontología de Marianao. En dicha tabla, la columna *Tipo* hace referencia al número de operación de razonamiento o inferencia mostrada en la Sección 3.2.

Id	**Consulta**	**Información recuperada**	**Tipo**
Q1	Subclases directas e indirectas de la clase *aeroway*	1) *aerodrome* 2) *aiport* En la ontología la clase *aiport* es subclase directa de *aerodrome* y esta a su vez subclase directa de *aeroway*	1
Q2	Instancias directas e indirectas de la clase *aeroway*	1) OSM-24047147-26-08 2) OSM-252369011-Hangar 3) OSM-252369012-Hangar 4) OSM-259849768-Aerodromo-Ciudad-Libertad 5) Geo-8554333-Ciudad-Libertad-Airport 6) DB-osm_new_aeroways-2 7) DB-osm_new_aeroways-3 Además de lo expuesto en Q1, hay que considerar que la clase *aeroway* es equivalente a *osm_new_aeroways*. Los 4 primeros resultados son instancias directas de *aeroway*. El quinto es instancia directa de *airport*, y las dos últimas, instancias directas de *osm_new_aeroways*	2 y 4
Q3	Clases *Topes* en la ontología	1) *SpatialObject*	3
Q4	Calles que cruzan la localidad de *Los Pocitos*	1) Avenida 51 2) 130 3) 116 4) 128 5) 120 6) 118 ... Se obtuvieron 36 resultados (se muestran 6 ejemplos)	6
Q5	Objetos espaciales en *Los Pocitos*	1) Arroyo Bañabuey 2) Policlínico Docente 27 de noviembre 3) Plaza de Marianao 4) Los-Pocitos 5) Plaza-de-Marianao ...	6
Q6	Superclases de *Administrative-Boundary*	1) *boundary* 2) *GeonameConcept* 3) *SpatialObject* La clase *AdministrativeBoundary* es subclase de *boundary* y de *GeonameConcept*, las cuales a su vez son subclases de *SpatialObject*	5

Table 2: Operaciones realizadas sobre la ontología geográfica generada

Todos los experimentos expuestos en este trabajo fueron realizados sobre un hardware con un CPU *Mobile DualCore Intel Core* i5-2430M, 2800 MHz (28 x 100) y 4 GB de memoria RAM.

5. Conclusiones

En este trabajo se presenta una extensión del *framework* OWLAPI que permite la administración y el razonamiento sobre grandes ontologías. Las pruebas realizadas demuestran la viabilidad de la solución para satisfacer las necesidades de información de un futuro sistema de Recuperación de Información Geográfica.

El resultado obtenido en este trabajo puede ser utilizado en cualquier ámbito donde se necesite administrar y razonar sobre ontologías con las características descritas. Como trabajo futuro, se podría aumentar el número de funcionalidades implementadas sin la necesidad de un razonador, así como trasladar los individuos de la ontología hacia la DB.

Agradecimientos

Los autores desean agradecer, de manera especial, los criterios y recomendaciones aportadas por el profesor Dr. C. Ignazio Palmisano, de la Universidad de Manchester, el cual sugirió la realización de este artículo a través de uno de sus comentarios. Igualmente, al profesor Dr. C. Diego Calvanese, de la Facultad de Ciencias de la Computación de la Universidad de Bozen-Bolzano, por sus intercambios relacionados con la herramienta Ontop y los sistemas DBBO.

Este trabajo también ha sido parcialmente financiado por el Ministerio de Economía y Competitividad del Gobierno de España, proyecto REDES (TIN2015-65136-C2-1-R).

References

1. Al-Jadir, L., Parent, C., Spaccapietra, S.: Reasoning with large ontologies stored in relational databases: The OntoMinD approach. Data Knowl. Eng. 69(11), 1158–1180 (2010)
2. Bakhtouchi, A.: FGOLD: A Flexible and Graphical Ontology-based Database Ready for Use. International Journal on Human Machine Interaction 2(2), 32–49 (2015)
3. Calvanese, D., Giese, M., Haase, P., Horrocks, I., Hubauer, T., Ioannidis, Y., Jiménez-Ruiz, E., Kharlamov, E., Kllapi, H., Klüwer, J., Koubarakis, M., Lamparter, S., Möller, R., Neuenstadt, C., Nordtveit, T., Özcep, O., Rodríguez Muro, M., Roshchin, M., Ruzzi, M., Savo, F., Schmidt, M., Soylu, A., Waaler, A., Zheleznyakov, D.: Optique: OBDA Solution for Big Data. In: Poster track of the Extended Semantic Web Conference (2013)
4. Calvanese, D., Cogrel, B., Komla-Ebri, S., Kontchakov, R., Lanti, D., Rezk, M., Rodriguez-Muro, M., Xiao, G.: Ontop: Answering SPARQL queries over relational databases. Semantic Web 8(3), 471–487 (2017)

5. Dehainsala, H., Pierra, G., Bellatreche, L.: Ontodb: An ontology-based database for data intensive applications. In: Ramamohanarao, K., Krishna, P.R., Mohania, M.K., Nantajeewarawat, E. (eds.) DASFAA. Lecture Notes in Computer Science, vol. 4443, pp. 497–508. Springer (2007)
6. Faruqui, R.U., MacCaull, W.: OwlOntDB: A Scalable Reasoning System for OWL 2 RL Ontologies with Large ABoxes. In: Weber, J., Perseil, I. (eds.) FHIES. Lecture Notes in Computer Science, vol. 7789, pp. 105–123. Springer (2012)
7. Franconi, E., Guagliardo, P.: The view update problem revisited. CoRR abs/1211.3016 (2012)
8. Guo, Y., Pan, Z., Heflin, J.: An Evaluation of Knowledge Base Systems for Large OWL Datasets. In: Proc. of the International Semantic Web Conference (ISWC). pp. 274–288 (2004)
9. Horridge, M., Bechhofer, S.: The OWL API: A Java API for OWL ontologies. Semantic Web 2(1), 11–21 (2011)
10. Horrocks, I., Li, L., Turi, D., Bechhofer, S.: The Instance Store: DL Reasoning with Large Numbers of Individuals. In: Haarslev, V., Möller, R. (eds.) Description Logics. CEUR Workshop Proceedings, vol. 104. CEUR-WS.org (2004)
11. Kiryakov, A., Ognyanov, D., Manov, D.: OWLIM - A Pragmatic Semantic Repository for OWL. In: Dean, M., Guo, Y., Jun, W., Kaschek, R.H., Krishnaswamy, S., Pan, Z., Sheng, Q.Z. (eds.) WISE Workshops. Lecture Notes in Computer Science, vol. 3807, pp. 182–192. Springer (2005)
12. Laallam, F.Z., Kherfi, M.L., Benslimane, S.M.: A survey on the complementarity between database and ontologies: principles and research areas. IJCAT 49(2), 166–187 (2014)
13. Pan, Z., Heflin, J.: DLDB: Extending Relational Databases to Support Semantic Web Queries. In: In PSSS. pp. 109–113 (2003)
14. Pease, A., Schulz, S.: Knowledge Engineering for Large Ontologies with Sigma KEE 3.0. In: Demri, S., Kapur, D., Weidenbach, C. (eds.) IJCAR. Lecture Notes in Computer Science, vol. 8562, pp. 519–525. Springer (2014)
15. Roldán García, M., Aldana Montes, J.: Evaluating DBOWL: A Non-materializing OWL Reasoner based on Relational Database Technology. In: Horrocks, I., Yatskevich, M., Jiménez-Ruiz, E. (eds.) ORE. CEUR Workshop Proceedings, vol. 858. CEUR-WS.org (2012)
16. Tsarkov, D., Palmisano, I.: Chainsaw: a metareasoner for large ontologies. In: Horrocks, I., Yatskevich, M., Jiménez-Ruiz, E. (eds.) ORE. CEUR Workshop Proceedings, vol. 858. CEUR-WS.org (2012)
17. Zhou, J., Ma, L., Liu, Q., Zhang, L., Yu, Y., Pan, Y.: Minerva: A Scalable OWL Ontology Storage and Inference System. In: ASWC. pp. 429–443 (2006)