Método para la indexación de grafos RDF desde un SPARQL Endpoint

Alejandro Jesús Mariño-Molerio[1], Juan Carlos Moreira de Lara[1], Leduan Flores-Riera[1], Yusniel Hidalgo-Delgado[1]

Grupo de Web Semántica, Universidad de las Ciencias Informáticas, Cuba
{ajmarino, jcmoreira, lflores, yhdelgado}@uci.cu

Abstract. Linked Data refers to a set of principles and best practices for publishing and linking structured data on the web. The linked data research community has been publishing library data using the standard RDF, a graph-based data model. These RDF graphs are stored in a specific type of database called triplestore and provides a SPARQL endpoint for querying them. However, these triplestores are not optimized for executing real-time queries submitted by the users, affecting the response time and usability in real environments. In this paper, we propose an index-based method for optimizing the query response time and solve the issues related to search and retrieval in large RDF graphs stored in triplestores. To prove the feasibility of the method proposed, it was applied to a semantic digital library, improving the query response time.

Keywords: Inverted Index, Linked Data, RDF Graph, Triplestore, Semantic Web

Resumen Los datos enlazados se refieren a un conjunto de principios y buenas prácticas para la publicación y enlazado de datos estructurados en la web. En los últimos años se han publicado datos bibliográficos utilizando el modelo de datos basado en grafos RDF. Los grafos RDF son almacenados en un tipo específico de base de datos conocidos como almacén de tripletas. Los almacenes de tripletas proporcionan un punto de acceso para la realización de consultas en el lenguaje SPARQL. Sin embargo, estos sistemas de almacenamiento no están optimizados para la ejecución de consultas realizadas por los usuarios en tiempo real, afectando el tiempo de respuesta y por tanto la usabilidad de los datos en entornos reales. En este artículo se propone un método basado en índices para la optimización del tiempo de respuesta a consultas realizadas en un punto de acceso SPARQL, mejorando los procesos de búsqueda y recuperación de datos en grandes grafos RDF. Para comprobar la aplicabilidad del método propuesto, se aplicó el mismo en una biblioteca digital basada en datos enlazados, obteniéndose una mejora en los tiempos de respuestas obtenidos en la experimentación.

Palabras claves: Índice Invertido, Datos Enlazados, Grafos RDF, Almacén de Tripletas, Web Semántica

1 Introducción

El tránsito hacia la Web Semántica requiere de una adecuada estructuración e integración de datos; esto propició que [25] enunciara el concepto de Datos Enlazados: "Los datos enlazados se refieren a un conjunto de principios y buenas prácticas para la publicación y enlazado de datos estructurados en la Web". La idea que persiguen los datos enlazados es utilizar la arquitectura general de la web para la compartición de datos estructurados a escala global [12].

La Web Semántica se basa en dos conceptos fundamentales: (1) la descripción del significado que tiene los contenidos en la web y (2) la manipulación automática de estos significados [5], [23]. En el caso del primero, intervienen conceptos como la semántica, que es el estudio y significado de los términos lingüísticos procesables por las máquinas; los metadatos como contenedores de información semántica sobre los datos; y las Ontologías para definir conceptos y relaciones de un dominio específico. Los metadatos y las Ontologías forman parte del campo de la representación del conocimiento. Para describir la semántica se requiere de un lenguaje apropiado llamado lenguaje de representación. Los lenguajes de representación como RDF(S)(Resource Description Framework Schema) y OWL(Web Ontology Language) proporcionan un estándar adecuado para la web [11], [2].

En el caso de RDF se define como un lenguaje para representar la información acerca de los recursos en la web. Está destinado especialmente para la representación de metadatos sobre recursos web, como el título, autor y fecha de modificación, entre otros, así como la disponibilidad para algunos recursos compartidos. RDF se encuentra recogido en 6 recomendaciones del W3C: Primer, Concepts, Syntax, Semantics, Test Cases y Vocabulary (Schema), usado para describir las propiedades y las clases de los recursos RDF con una semántica para establecer jerarquías de generalización entre dichas propiedades y clases [9].

Para [23] RDF es un modelo de datos en forma de grafo dirigido y etiquetado que permite definir relaciones semánticas entre distintas URIs como recursos, asociándoles un conjunto de propiedades y valores con el fin de representar información sobre recursos en la web. Por su parte [22] lo define como un marco para expresar la información acerca de los recursos. Los recursos pueden ser documentos, personas, objetos físicos, y los conceptos abstractos.

RDF está basado en la idea de que los recursos (sujeto) a describir, poseen propiedades (predicado) que a su vez tienen valores (objeto). Estos recursos pueden ser descritos formulando "declaraciones" que especifican estas propiedades y valores, en forma de grafo de nodos y arcos que representan los recursos, y sus propiedades y valores [17]. RDF y el lenguaje RDF Schema se fundamentaron en investigaciones sobre metadatos realizadas por comunidades de Bibliotecas Digitales, pudiendo considerarse RDF como una implementación del Warwick Framework donde RDF es una evolución de este último, que permite que cada vocabulario de metadatos posea una sintaxis distinta. Para [17] en RDF es fundamental utilizar palabras que transmitan un significado inequívoco con el fin de que las aplicaciones entiendan el enunciado para un procesamiento correcto.

Asociado a RDF se define la estandarización del lenguaje de consultas SPARQL. Este lenguaje se puede utilizar para expresar consultas a través de diversas fuentes de datos, ya sea que los datos se almacenen de forma nativa como RDF o se visualicen como RDF a través de un middleware. SPARQL contiene capacidades para consultar patrones sobre grafos obligatorios y opcionales junto con sus conjunciones y disyunciones. SPARQL también admite agregación, subconsultas, negación, creación de valores por expresiones y consultas restrictivas por grafo RDF. Los resultados de las consultas SPARQL pueden ser conjuntos de resultados o grafos RDF [10].

La evolución de RDF y SPARQL, en el contexto de la web semántica, ha propiciado el surgimiento de muchos sistemas capaces de almacenar, consultar y actualizar RDF, tales como Ontotext GraphDB [1] , Virtuoso [2], Jena [3] ,etc. Cada uno de estos sistemas provee de un SPARQL Endpoint. Un SPARQL Endpoint es el principal modo para acceder a los datos porque es una forma flexible de interactuar con la Web de los datos. Además, devuelve respuestas a consultas en varios formatos, como XML y JSON, que son ampliamente utilizados como estándares de intercambio de datos en diversas aplicaciones [20]. Con el crecimiento de la Web de los datos el número de SPARQL Endpoints que construyen consultas SPARQL sobre ella usando HTTP también crece rápidamente. Esto ha permitido que las instituciones agreguen datos de múltiples SPARQL Endpoints similares a las bases de datos distribuidas convencionales. Sin embargo, la escalabilidad de estos sistemas se ve afectada por el aumento del tamaño de los grafos RDF. Esto ha supuesto un desafío para la comunidad científica dado que la nube de datos enlazados crece constantemente.

En [3] se realizan varios experimentos para analizar el tiempo de respuestas a diferentes tipos de consultas realizadas sobre diferentes SPARQL Endpoints para obtener grandes conjuntos de respuestas. Los resultados obtenidos en tiempo de respuesta varían desde los 5.5 hasta los 723.8 segundos con una media de 50.7 segundos, como se muestra en la figura 1. Las consultas realizadas, utilizan diferentes límites para la obtención de los resultados con la ejecución de la consulta. En cuanto a los tiempos de respuesta durante la ejecución de consultas por los usuarios al SPARQL Endpoint desde el punto de vista de la interacción humano-computadora, se definen tres límites de tiempos para las aplicaciones web [18]:

1. 0.1 segundos límite para los usuarios que sienten que están manipulando objetos directamente en la interfaz de usuario.
2. 1 segundo límite para los usuarios que sienten que están navegando libremente por el espacio de comandos sin tener que esperar indebidamente la computadora. Un retraso de 0.2-1.0 segundos significa que los usuarios notan el retraso y sienten que la computadora está "trabajando"en el comando, en lugar de tener el comando como un efecto directo de las acciones de los usuarios.

[1] http://graphdb.ontotext.com/
[2] https://virtuoso.openlinksw.com/
[3] https://jena.apache.org/

3. 10 segundos límite para los usuarios que mantienen su atención en la tarea. Todo lo que sea inferior a 10 segundos necesita un indicador de porcentaje de realización y una forma claramente señalizada para que el usuario interrumpa la operación.

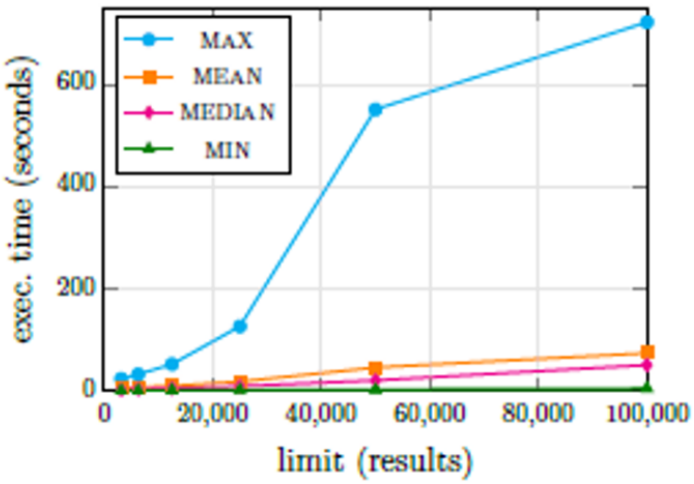

Figura 1. Comparación del tiempo de respuesta con diferentes límites en una consulta SPARQL. Fuente: [4].

Cuando se realiza la comparación entre los tiempos obtenido por [3] para la obtención de los resultados de las consultas realizadas a un SPARQL Endpoint y los límites que se establece en [18] para los tiempos de respuestas a las interacciones (consultas, en este caso) entre el usuario y el sistema (SPARQL Endpoint) se puede comprobar que en muchos casos los usuarios abandonarían la espera de los resultados de dichas consultas. Esto afecta notablemente el uso extendido de estos sistemas, al no proveer de un tiempo de respuesta razonable a consultas realizadas por los usuarios. En este sentido el uso de estructuras de datos optimizadas ha supuesto una alternativa eficiente para crear aplicaciones utilizando grafos RDF en la que las consultas realizadas por los usuarios sin perder la interoperabilidad que este modelo brinda.

Los principales enfoques [14] se centran en usos de esquemas NoSQL tipo clave-valor o con el uso de MapReduce framework [8] y su implementación open source Hadoop[4]. En el caso de las implementaciones NoSQL aunque comparten los elementos básicos de sus interfaces, estos sistemas difieren con respecto a su arquitectura interna (cliente-servidor vs. basado en P2P), políticas de control de acceso, autenticación y consistencia. Una diferencia que afecta el diseño de un almacén de tripletas que depende en tales plataformas es si el índice ofrecido

[4] http://hadoop.apache.org/

en la clave es basado en hash (permitiendo solo búsquedas directas) u ordenado (lo que además permite búsquedas de prefijos) [14]. Por su parte, MapReduce se enfoca en tareas para el análisis de los datos y no para su explotación en entornos donde es necesario consultas complejas sobres los datos, ya que no admite directamente operaciones más complejas como las uniones.

En este artículo se propone un método para la indización de grafos RDF desde un SPARQL Endpoint. El método propuesto soporta la indización de las tripletas almacenadas en un almacén de tripletas hacia un servidor de indización. La indización de estas tripletas permite la reducción del tiempo de respuesta de consultas realizadas por los usuarios sobre el conjunto de datos RDF almacenados en el almacén de tripletas.

2 Trabajos relacionados

Los índices son estructuras de datos optimizadas que permiten transformar el texto en un formato donde la búsqueda sea más rápida, eliminando el proceso de exploración lento a consultas formuladas por los usuarios. Este proceso de conversión es llamado indización mientras al archivo resultante se le llama índice [13]. La indización es un requisito necesario para un adecuado almacenamiento y recuperación de la información contenida en un fondo documental. En [6] se asume la indización como una lista de información bibliográfica o citas hacia un cuerpo literario, usualmente arreglados en orden alfabético y basado en algunos datos específicos, tales como autor, tema o palabras claves. Si bien ambas ideas poseen similar línea de pensamiento, en la investigación se asume la definición dada en [13] por ser la que más se ajusta en el marco de la investigación.

La indización, según [21], es una de las etapas del procesamiento analítico sintético de la información. Se define como la enumeración sucesiva de los diferentes encabezamientos (términos) que expresan el(los) tema(s) contenido(s) en un documento, y que requiere de la aplicación de criterios uniformes; así como del establecimiento previo de una lista de términos en la cual se basa dicha indización. Su importancia radica esencialmente en la necesidad de habilitar un sistema de búsqueda y recuperación de la literatura científica existente en los fondos documentales de las entidades informativas. El producto final de este proceso es generalmente un índice bibliográfico, una base de datos automatizada o simplemente un catálogo alfabético de materias, indispensables para asegurar el acceso y consulta de la información a los usuarios.

Para lograr la indización correcta de un documento o solicitud de búsqueda es necesario utilizar los lenguajes de indización existentes. Estos lenguajes artificiales, llamados lenguajes de búsqueda informativa, lenguajes de indización, lenguajes documentales, lenguajes de almacenamiento y recuperación, entre otras denominaciones, son herramientas auxiliares, creadas por el hombre con el propósito de expresar el contenido semántico fundamental de los documentos o solicitudes de información y localizar la información que responda a las necesidades de los usuarios.

Para considerar que un documento se indizó correctamente es necesario considerar dos aspectos fundamentales en la indización: (1) la exhaustividad y (2) la especificidad. La exhaustividad se define como la cantidad de conceptos considerados que son representativos del contenido íntegro de un documento. La especificidad es el nivel de detalle y exactitud de la representación de un concepto particular. Indica [19] que en la indización automática, la máquina separa cadenas de caracteres ya sea en el título, en el resumen, descartando únicamente las llamadas palabras vacías o reconociendo en el texto completo los sintagmas nominales.

Como parte de las herramientas de indización examinadas destacan:

Solr: Un índice en Solr posibilita llevar a cabo de manera óptima: la búsqueda de texto completo, agregados y filtrado. Solr acepta documentos JSON, pudiendo transformar su documento RDF en un documento JSON-LD (que es un formato de serialización RDF). Por otra parte SolrRDF (entiéndase Solr + RDF) es un conjunto de extensiones para la gestión de Solr (índice y búsqueda) de datos RDF siendo posible indizar triples al clúster y realizar consultas SPARQL (ASK, CONSTRUCT, SELECT y DESCRIBE) y actualizaciones (como INSERT y DELETE) a cualquier nodo del clúster obteniendo como respuesta un XML, siendo compatible con SPARQL Endpoint 1.1.

Elasticsearch[5]**:** Basado en Lucene proporciona capacidades distribuidas. Distribuido, con búsqueda RESTful y análisis capaz de resolver un número creciente de casos de uso.

Jena ARQ: es un motor de búsqueda de Jena que soporta el lenguaje de consultas SPARQL. En Jena, toda la información de estado proporcionada por un conjunto de tripletas RDF está contenida en una estructura de datos llamada modelo. El modelo representa un grafo RDF, llamado así porque contiene una colección de nodos RDF, unidos entre sí por relaciones marcadas. Entre las capacidades de Jena ARQ destacan: búsqueda de texto libre a través de Lucene; actualización, acceso y la extensión del álgebra de SPARQL; apoyo a las funciones de filtro personalizados; funciones de propiedad para un tratamiento personalizado de relaciones semánticas; y apoyo al cliente para el acceso remoto a cualquier SPARQL Endpoint.

Apache Lucene[6]**:** es el principal motor de búsqueda de código abierto y se utiliza en muchas empresas, proyectos y productos, originalmente implementada en Java. Es útil para cualquier aplicación que requiera indexado y búsqueda a texto completo. En esencia, el índice se compone de documentos que se componen de campos. Las consultas de Lucene tienen que pasar a través de los mismos analizadores que se utilizaron durante la indexación, de lo contrario términos idénticos podría no coincidir. Se puede usar Lucene en Grails, como es el caso de los plugins para la integración de Solr y ElasticSearch con Grails.

Muchos sistemas de almacenamiento y consulta de RDF se basan en sistemas de bases de datos relacionales [24]. La mayoría de estos sistemas asocian tablas relacionales con RDF, tripletas, propiedades o instancias de clase. RDFBroker

[5] https://www.elastic.co/
[6] https://lucene.apache.org/

calcula los conjuntos específicos de propiedades utilizadas en cada recurso, y almacena los datos RDF en tablas organizadas [16]. RDFSuite adopta una técnica de almacenamiento "controlada por esquema"; realizando la optimización de consultas mediante el sistema de gestión de base de datos subyacente [1]. El acoplamiento del diseño de la tabla al esquema de los datos RDF permite tener en cuenta las características específicas de las clases y propiedades empleadas y explotar las relaciones de esquema explícitas. Por otro lado, limitar el número de tablas creadas, especialmente cuando se enfrenta a la reestructuración de esquemas dinámicos, se convierte en un problema crucial. En [7] se propone una técnica basada en aprendizaje automático aplicada a datos y consultas para calcular diseños adecuados de tablas relacionales.

3 Método propuesto

El método propuesto consta de tres etapas fundamentales (1) selección del SPARQL Endpoint, (2) generación de consultas en el lenguaje de consultas SPARQL y (3) sincronización de los datos con el índice invertido, ver Figura 2. Este método sigue un enfoque basado en tuberías o filtros donde la salida de una etapa constituye la entrada a la próxima, siguiendo un enfoque iterativo. Su utilización garantiza la sincronización de los datos almacenados en el almacén de tripletas RDF hacia el índice invertido.

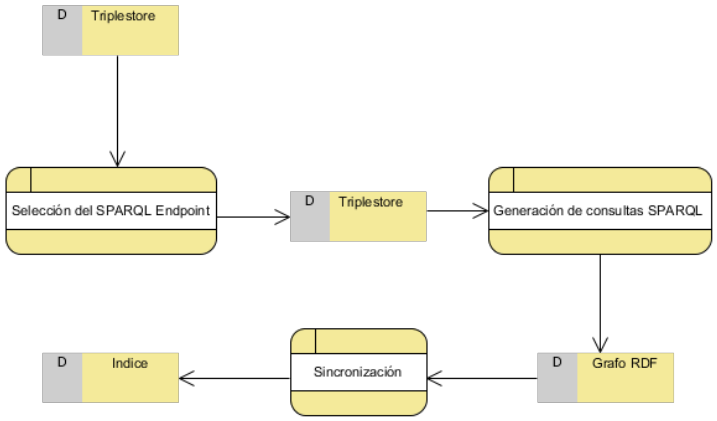

Figura 2. Etapas del método de solución propuesto. Fuente: elaboración propia.

a) Selección del SPARQL Endpoint: el método propuesto utiliza un almacén de triplestas donde son almacenados metadatos bibliográficos que han sido transformados al estándar RDF utilizando el modelo de datos que se define en la Figura 3, a partir de una base de datos relacional (RDB). En este paso se verifica la valides de la URL del SPARQL Endpoint.

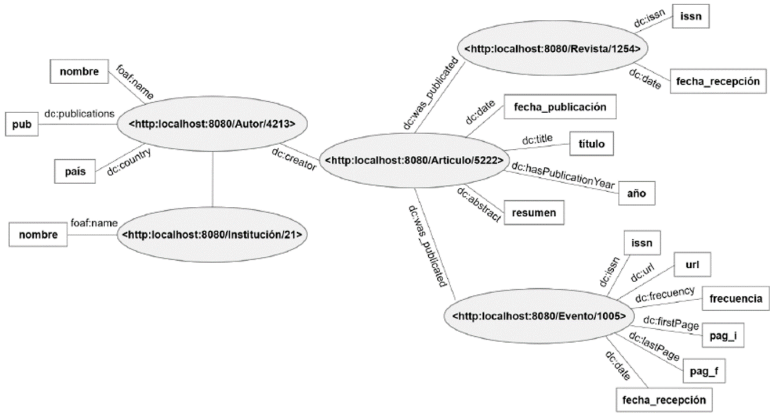

Figura 3. Modelación de los datos con las ontologías de dominio. Fuente: elaboración propia.

b) Generación de consultas SPARQL: a partir del modelo de datos anterior, se definen las consultas para obtener los datos desde el almacén de tripletas que van a ser indizados. En esta etapa se establece la(s) consulta(s) para obtener los datos de acuerdo al modelo RDF. Es necesario conocer *a priori* las relaciones que se van a obtener a partir de las consultas. Las relaciones obtenidas a partir de los resultados de las consultas permiten establecer un esquema de alineación con el cual van a ser guardados los datos en el motor de indización. La siguiente consulta SPARQL muestra la relación entre un recurso artículo dentro del grafo y sus datos correspondientes, tales como autor, resumen, url, año de publicación, entre otros.

```
SELECT DISTINCT ?s ?author ?title ?name
?affiliation ?abstract ?uri ?year
WHERE   {
?s      a      fabio:JournalArticle.
?s      vocab:record_title       ?title.
?s      fabio:abstract       ?abstract.
?s      bibo:uri                 ?uri.
?s      fabio:hasPublicationYear ?year.
?author vocab:author_record      ?s.
?author foaf:name                ?name.
?author swrc:affiliation  ?affiliation.}
```

c) Sincronización de los datos con el índice invertido: en esta etapa se define, a partir de los resultados obtenidos en la(s) consulta(s) en la etapa anterior, utilizando el formato JSON, la estructura que va a tener la información obtenida dentro del índice. Una vez definida, se comprueban los parámetros en el motor de búsqueda y se procede a la carga de los datos hacia el índice. Este proceso de carga se realiza inicialmente a todo el conjunto de datos obtenidos

a través de la(s) consulta(s) realizadas sobre el grafo RDF. Después de la carga inicial de datos hacia el índice se utiliza el enfoque propuesto por [15] para la actualización de los grafos RDF almacenados en el almacén de tripletas y para la actualización del índice en el motor de indización.

Para comprobar la aplicabilidad del método propuesto, se implementó una herramienta informática basada en el mismo. La herramienta implementada utiliza una arquitectura de tubería y filtros basándose en la descripción del método antes detallado, ver figura 4. La herramienta consta de dos componentes fundamentales: el almacén de tripletas Apache Jena Fuseki (como punto de acceso a consultas SPARQL) y el servidor de indización Elasticsearch (como motor de búsqueda).

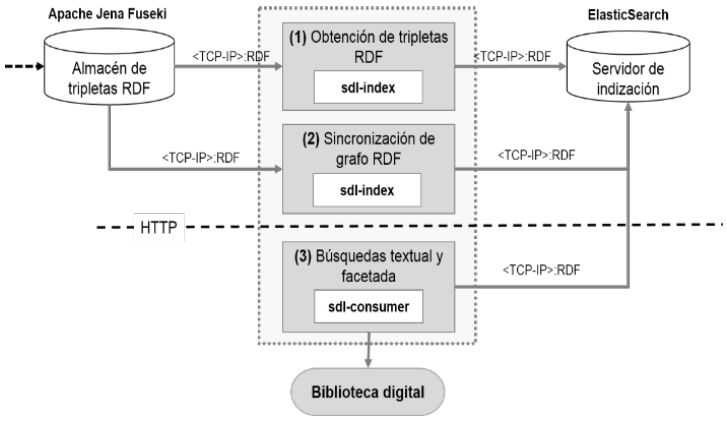

Figura 4. Arquitectura del componente implementado. Fuente: elaboración propia.

Apache Jena Fuseki almacena los grafos RDF generados *a priori*. Como se indica en (b) se realizan consultas utilizando el lenguaje de consultas SPARQL para obtener dichos grafos RDF e indizarlos en el servidor de indización. Estos grafos una vez incorporados a Elasticsearch se encuentran en formato JSON. La tarea de obtención de grafos RDF se acomete en primera instancia, ya que seguidamente basta con sincronizar los grafos RDF del almacén de tripletas con los del índice en el motor de búsqueda. El propósito de esta actividad es consultar el almacén de tripletas para verificar los últimos cambios e indizar sólo esos grafos RDF al índice del motor de búsqueda de Elasticsearch. Este proceso es ventajoso en el sentido de que no se precisa el almacenamiento repetido de los grafos RDF existentes en el almacén de tripletas.

Para validar la propuesta de solución se realiza un experimento. Consta de realizar un conjunto de consultas sobre el grafo RDF. Las consultas ejecutadas se muestran en la tabla 1.

El grafo RDF utilizado en el experimento contiene aproximadamente 52000 tripletas. En la tabla 1 se muestran las diferentes consultas ejecutadas sobre el

ID	Consulta
Q1	Seleccionar el autor X que más ha publicado con el autor Y en el año A y en la afiliación F.
Q2	Encontrar los artículos del autor X que pertenecen a la afiliación F.
Q3	Encontrar la afiliación F que más artículos haya publicado en el año A.
Q4	Encontrar las dos afiliaciones que más artículos hayan publicado.

Cuadro 1. Consultas ejecutadas en el experimento realizado.

grafo RDF. Los tiempos de respuesta obtenidos se muestran en la Figura 5. La comparativa hecha a partir de los resultados obtenido muestran que los tiempos de respuesta disminuyen en un 50 por ciento para cada una de estas consultas. Los tiempos obtenidos son tiempos promedios calculados a partir de la ejecución por diferentes usuarios de la misma consulta en diferentes momentos durante el experimento.

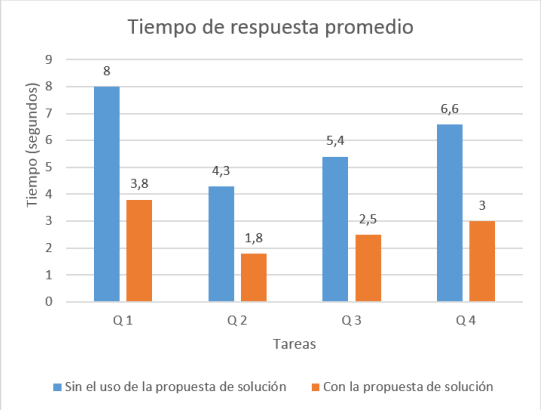

Figura 5. Tiempo de respuesta promedio obtenido. Fuente: elaboración propia.

4 Conclusiones

En este artículo se ha propuesto un método dividido en tres etapas para la indización de grafos RDF desde un SPARQL Endpoint. En la propuesta de solución se hace uso de un motor de indización para almacenar el grafo RDF que se encuentra en un almacén de tripletas. El proceso de indización posee un enfoque incremental para la actualización del índice en el motor de indización si ocurren actualizaciones en el grafo RDF. Con la indización del grafo RDF se logró disminuir el tiempo de respuesta a las consultas formuladas por los usuarios directamente sobre el almacén de tripletas. La disminución de los tiempos

de respuesta fue de un 50 por ciento en relación a los tiempos de respuesta obtenidos en consultas realizadas directamente sobre el almacén de tripletas. Se ha identificado como principal problema la generación de facetas dinámicas a partir de los datos almacenados en el motor de indización. La estructura de los datos del grafo almacenado en el índice dificulta la generación de estas facetas. En trabajos futuros se trabajará para resolver este problema.

Referencias

1. Blin, G., Curé, O., Faye, D.C.: A survey of RDF storage approaches. REVUE AFRICAINE DE LA RECHERCHE EN INFORMATIQUE ET MATHÉMATIQUES APPLIQUÉES 15 (2016)
2. Brickley, D., Guha, R.V.: RDF vocabulary description language 1.0: RDF schema (2004)
3. Buil-Aranda, C., Hogan, A., Umbrich, J., Vandenbussche, P.Y.: Sparql web-querying infrastructure: Ready for action? In: International Semantic Web Conference. pp. 277–293. Springer (2013)
4. Buil-Aranda, C., Hogan, A., Umbrich, J., Vandenbussche, P.Y.: Sparql web-querying infrastructure: Ready for action? In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) The Semantic Web – ISWC 2013. pp. 277–293. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
5. Chávez, M.E., Cárdenas, O., Benito, O.: La web semántica. Revista de investigación de Sistemas e Informática 2(3), 43–54 (2005)
6. Cleveland, A.D., Cleveland, D.B.: Introduction to indexing and abstracting. ABC-CLIO (2013)
7. Curé, O., Blin, G.: RDF database systems: triples storage and SPARQL query processing (2015)
8. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Communications of the ACM 51(1), 107–113 (2008)
9. Frank Manola, Eric Miller: RDF Primer (2004), `https://www.w3.org/TR/2004/REC-rdf-primer-20040210/#intro`
10. Harris, S., Seaborne, A.: SPARQL 1.1 Query Language (2013), `https://www.w3.org/TR/sparql11-query/`
11. Hayes, P., McBride, B.: RDF Semantics. W3C Recommendation, February 2004 (2004)
12. Heath, T., Bizer, C.: Linked Data: Evolving the Web into a Global Data Space. Synthesis Lectures on the Semantic Web: Theory and Technology 1(1), 1–136 (Feb 2011), `http://www.morganclaypool.com/doi/abs/10.2200/S00334ED1V01Y201102WBE001`
13. Hernández, J.P.R., Hernández, G.A.: Indización y Búsqueda a través de Lucene. Veracruz, Sinaloa (2008)
14. Kaoudi, Z., Manolescu, I.: RDF in the clouds: a survey. The VLDB Journal 24(1), 67–91 (2015)
15. Liudmila Reyes-Álvarez, Yusniel Hidalgo-Delgado, Katerin Martinez-Rojas, Maria del Mar Roldan, José F. Aldana-Montes: Actualización incremental de grafos RDF a partir de bases de datos relacionales. In: Proceedings of Jornadas de Ingeniería del Software y Bases de Datos. JISBD 2014. España (2014)

16. Ma, Z., Yan, L.: A Review of RDF Storage in NoSQL Databases. In: Managing Big Data in Cloud Computing Environments, pp. 210–229. IGI Global (2016)
17. Moreno Agudelo, C.A., Sánchez Reyes, Y.: Prototipo de buscador semántico aplicado a la búsqueda de libros de ingeniería de sistemas y computación en la biblioteca Jorge Roa Martínez de la Universidad Tecnológica de Pereira. Ph.D. thesis, Universidad Tecnológica de Pereira (2012)
18. Nielsen, J.: Web-based application response time (2014), https://www.nngroup.com/articles/response-times-3-important-limits/
19. Peña, C.N.: Indización y clasificación: Un problema conceptual y terminológico Indexation and classification: A conceptual and terminologic problem. Documentación de las Ciencias de la Información 26, 23–40 (2003)
20. Rakhmawati, N.A., Umbrich, J., Karnstedt, M., Hasnain, A., Hausenblas, M.: Querying over Federated SPARQL Endpoints—A State of the Art Survey. arXiv preprint arXiv:1306.1723 (2013)
21. Rodríguez Suárez, A., Bermello Navarrete, R., Pinillo León, A.L.: Indización en línea:¿ capricho o necesidad? Acimed 15(1), 0–0 (2007)
22. Schreiber, G., Raimond, Y.: RDF 1.1 Primer. W3C Working Group Note 25 (2014)
23. Tello, J.C.: La Web Semántica y el lenguaje RDF (2006)
24. Theoharis, Y., Christophides, V., Karvounarakis, G.: Benchmarking database representations of RDF/S stores. In: International Semantic Web Conference. vol. 3729, pp. 685–701. Springer (2005)
25. Tim Berners-Lee: Linked Data - Design Issues (2006), https://www.w3.org/DesignIssues/LinkedData.html