Preface

TODO: completar ¿Qué ponemos acá?

April 1995

Walter Olthoff
Program Chair
ECOOP'95

Table of Contents

Bases de datos de grafos con manejo de datos espaciales. Un análisis comparativo

Federico Martinez[1] and Ariel Aizemberg[2]

[1] Universidad Nacional de Quilmes, Roque Sáenz Peña 352, Bernal, Buenos Aires, Argentina,
federico.martinez@unq.edu.ar
[2] Instituto Tecnologico Buenos Aires, Av. Madero 399, Buenos Aires, Argentina,
aaizemberg@itba.edu.ar

Abstract. The abstract should summarize the contents of the paper using at least 70 and at most 150 words. It will be set in 9-point font size and be inset 1.0 cm from the right and left margins. There will be two blank lines before and after the Abstract. ...

Keywords: computational geometry, graph theory, Hamilton cycles

1 Bases de datos de grafos con manejo de datos espaciales

1.1 Introducción

Las bases de datos NoSQL han adquirido una gran popularidad en el último tiempo. Entre los distintos tipos que existen, las bases de datos de grafos se destacan por su capacidad de almacenar elementos de características diferentes, no estructuradas y fundamentalmente, las relaciones que existen entre ellos.

En la actualidad existen varias bases de datos de grafos, tanto open source, como de código propietario. Son varias las que introducen la posibilidad de manejar además datos espaciales o georreferenciados. Entre ellas podemos citar: Neo4j[3], ArangoDB[4] , OrientDB[5], Titan[6]; las cuales permiten en distinta medida almacenar y consultar datos espaciales. Las bases de datos de grafos son muy útiles cuando el problema que se quiere resolver tiene distintos tipos de entidades relacionadas entre sí. Son eficientes para resolver problemas de búsquedas en grafos, pero sabemos que los datos almacenados suelen contener información espacial o geográfica. Entonces, nos preguntamos ¿que pasa cuando incorporamos la componente espacial en nuestras consultas?

Elegimos a Neo4j y ArangoDB ya que estas poseen manejo de datos geográficos sin necesidad de motores de indexado externos para realizar una comparación. Un ejemplo típico es una red social. En ella los usuarios son nodos en

[3] http://neo4j.com/
[4] http://www.arangodb.com/
[5] http://www.orientechnologies.com/orientdb/
[6] http://thinkaurelius.github.io/titan/

un grafo y la "amistad" entre dos usuarios es un eje que los relaciona. Estos nodos pueden tener propiedades como el nombre, edad, género, etc. Las fotos subidas por los usuarios pueden ser también un nodo, con otro conjunto de propiedades: url, fecha de publicación, etc. En este caso, pueden existir relaciones entre usuarios y fotos con distinta semántica: le gusta la imagen, comparte la imagen, está etiquetado en la imagen, etc. Cada una de estas relaciones puede modelarse con un eje. En este escenario, podemos pensar que una consulta típica es: ¿Cuáles son los amigos de mis amigos?

Existen problemas donde estos nodos se encuentran distribuidos en el espacio, por ejemplo los nodos pueden representar ciudades y los ejes autopistas, rutas o calles entre ellos. Para estos casos es interesante contar con la capacidad de realizar consultas sobre la georeferenciación de los datos y no solo sobre sus relaciones, a fin de poder responder, por ejemplo, cual es el camino más corto entre dos ciudades.

1.2 Metodología

Para analizar las capacidades de manejo de datos geográficos que presentan ambas bases de datos decidimos utilizar un conjunto de datos proveniente del sitio openflights[7]. El mismo presenta información de 6977 aeropuertos, los cuales tienen, entre otros atributos, latitud y longitud. Además hay 5888 aerolíneas y 59036 rutas. Las rutas van de un aeropuerto a otro y son gestionadas por una aerolínea.

- Cada aeropuerto es un nodo, al igual que las aerolíneas y las rutas.
- Un eje entre una ruta y una aerolínea indica que ésta es responsable de la ruta.
- Las rutas pueden relacionarse con los aeropuertos mediante dos tipos distintos de ejes: Origen y Destino, los cuales indican, respectivamente, si el aeropuerto es donde la ruta comienza o termina.

Para el modelo de datos generado, realizamos las siguientes consultas:

1. Aeropuerto a menos de 100 kilómetros de la Ciudad de Buenos Aires.
2. Aeropuertos a 100 kilómetros de la Ciudad de Buenos Aires que permiten viajar a Barajas.
3. Itinerarios que parten de un aeropuerto a menos de 400 kilómetros de la Ciudad de Buenos Aires, que tengan una escala y terminan en Barajas.
4. Itinerarios que parten de un aeropuerto a menos de 400 km de la Ciudad de Buenos Aires, tienen una escala y terminan a menos de 400 km de Chipre.

1.3 Resultados

Para cada consulta consideramos el tiempo de ejecución hasta obtener los resultados. Analizamos también el tiempo necesario para la carga inicial y el espacio

[7] ttp://openflights.org/data.html

necesario en disco para cada una de las bases. La carga de datos y los experimentos se ejecutaron sobre una máquina virtual, ésta dispone de 3 GB de RAM y 2 cores. La máquina física es un Core i5 4690 de 3.5 GHz con 16 GB de RAM.

Table 1. Resultados de los experimentos

Experimento	Neo4J	ArangoDB
Carga	20 minutos 27 segundos	25 minutos 37 segundos
Espacio en disco	165.9 MB	437.3 MB
Consulta 1	0.063 segundos	0.012 segundos
Consulta 2	0.016 segundos	0.052 segundos
Consulta 3	0.227 segundos	3.697 segundos
Consulta 4	2.319 segundos	4.901 segundos

1.4 Conclusiones

Si bien ambas bases de datos proveen soporte para el manejo de datos geográficos y las dos funcionan muy bien para responder consultas sencillas, Neo4j logra imponerse como una mejor alternativa cuando las consultas requieren recorrer el grafo o funcionalidades más complejas como importar mapas. Si bien Arango logra resolver correctamente las consultas sencillas (por ejemplo la consulta 1, que no requería recorrer las relaciones del grafo, es resuelta más rápidamente por Arango), no está a la altura del soporte que brinda Neo4j. Las principales falencias que encontramos en ArangoDB son: Su lenguaje de consulta no parece estar pensado para recorrer grafos. Parece haber sido diseñado para trabajar con documentos y luego se le agregaron casi de manera ad-hoc nociones de grafos. Esto genera que escribir queries que combinen información espacial y recorrer el grafo sea ineficiente además de complejo. Si bien es muy bueno que la base brinde soporte geográfico nativo, el mismo es muy básico. Por esa razón es imposible escribir consultas que requieren de capacidades más complejas.

El informe completo, las tablas, graficos y el código fuente del presente estudio, se puede consultar en la siguiente URL:

https://github.com/federicoemartinez/itba_graphdb

References

1. Clarke, F., Ekeland, I.: Nonlinear oscillations and boundary-value problems for Hamiltonian systems. Arch. Rat. Mech. Anal. 78, 315–333 (1982)
2. Clarke, F., Ekeland, I.: Solutions périodiques, du période donnée, des équations hamiltoniennes. Note CRAS Paris 287, 1013–1015 (1978)
3. Michalek, R., Tarantello, G.: Subharmonic solutions with prescribed minimal period for nonautonomous Hamiltonian systems. J. Diff. Eq. 72, 28–55 (1988)

4. Tarantello, G.: Subharmonic solutions for Hamiltonian systems via a $\mathbb{Z}_p$ pseudoindex theory. Annali di Matematica Pura (to appear)
5. Rabinowitz, P.: On subharmonic solutions of a Hamiltonian system. Comm. Pure Appl. Math. 33, 609–633 (1980)