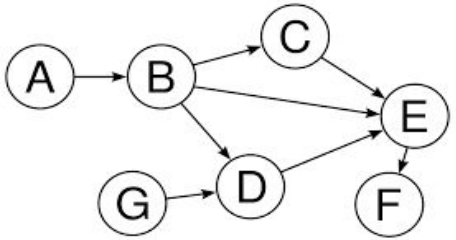
Comparación de bases de datos relacionales (MySQL) con bases de grafos (Neo4j)

Marcos Sander

Descripción del trabajo

- Basado en el artículo “A comparison of a graph database and a relational database: A data provenance perspective” de Vicknair y colaboradores (año 2010).
- Bases de datos de procedencia de datos.
- Grafos dirigidos acíclicos (DAG).
- Representación relacional sencilla:
  - Nodo(id_nodo, info_nodo)
  - Arista(id_nodo_a, id_nodo_b)
- Generación de las bases de datos.
- Comparación de performance ejecutando diferentes consultas.



Generación de Bases

- Se crearon 6 bases MySQL y 6 bases Neo4j.
- Tamaños 1000, 5000 y 10000 nodos con aprox. 2000, 10000 y 20000 aristas.
- Datos numéricos (entero) y datos texto (hasta 4KB).
- Aplicación Java que genera las bases aleatoriamente en base a parámetros (cantidad de nodos, cantidad máxima de nodos por capa, conexiones, etc.).

Generación de bases (2)

- Generación en capas
  - Cantidad aleatoria de nodos en cada capa.
  - Se generan aristas de conexión entre la capa actual y la anterior.
- Se crean ambas bases en paralelo.
- Generación de enteros aleatorios sobre datos numéricos.
- Generación de secuencia de palabras sobre datos texto.

Consultas

- Las consultas que se realizaron son de 2 tipos:
  - Que recorren la estructura del grafo.
  - Sobre la información de cada nodo.
- Se ejecutó 10 veces cada una, descartando máximo y mínimo y se devuelve tiempo promedio en milisegundos.
- Tanto en MySQL como en Neo4j se ejecutaron desde una aplicación Java.
- Hardware y S.O.: Notebook Core i7, 16 GB RAM, Disco duro mecánico SATA, Windows 10.

Consultas estructurales

- Recorrido sobre la estructura del grafo
- 3 consultas
  - E1 - Nodos aislados.
  - E2 - Buscar hijos hasta profundidad 5.
  - E3 - Buscar hijos hasta profundidad 40.

Consultas estructurales (2)

Ejemplo nodos aislados:

```
/* SQL */
select count(n.id)
from nodo n
where n.id not in (select id_nodo_a from arista) and
      n.id not in (select id_nodo_b from arista);

/* NEO4J */
match (n:Nodo)
where not (n)-[:Arista]-() and not ()-[:Arista]-(n)
return count(n);
```

Consultas sobre datos

- Búsqueda de condiciones sobre los datos de los nodos.
- 4 consultas (2 sobre datos numéricos y 2 sobre texto)
  - N1 - Buscar nodos mayores a cierto valor.
  - N2 - Buscar nodos igual a cierto valor.
  - T1 - Buscar nodos que contengan una palabra.
  - T2 - Buscar nodos que contengan una secuencia de 3 palabras.

Consultas sobre datos (2)

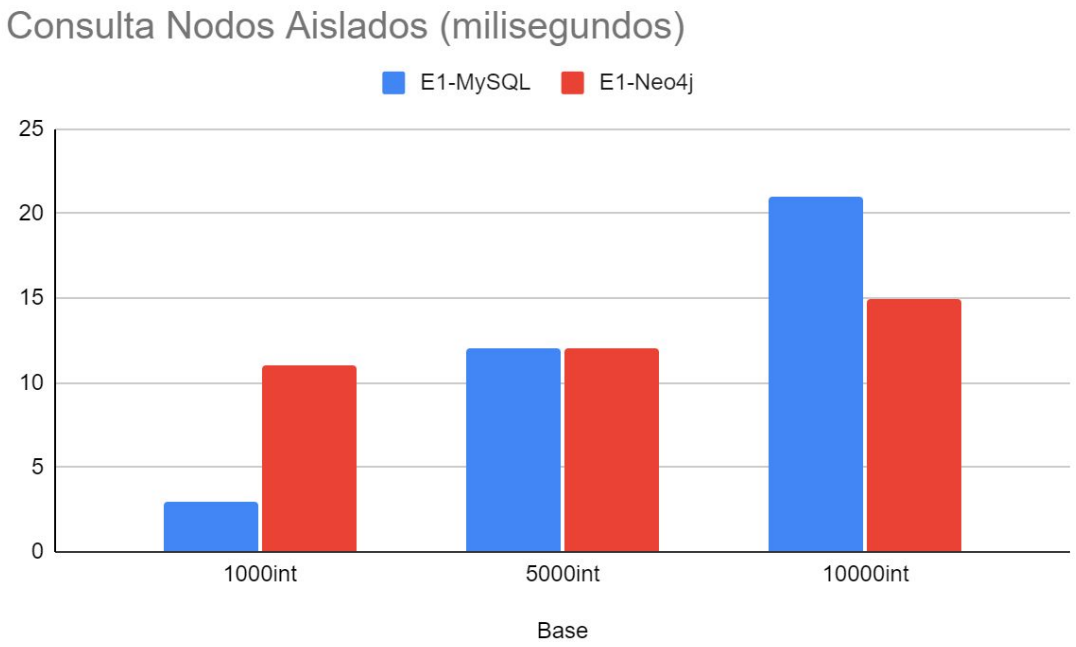
Ejemplo nodos con cierto texto:

```
/* SQL */
select count(n.id)
from nodo n
where n.info like "%:secuencia%";

/* NEO4J */
match (n:Nodo)
where n.info contains ':secuencia'
return count(n);
```

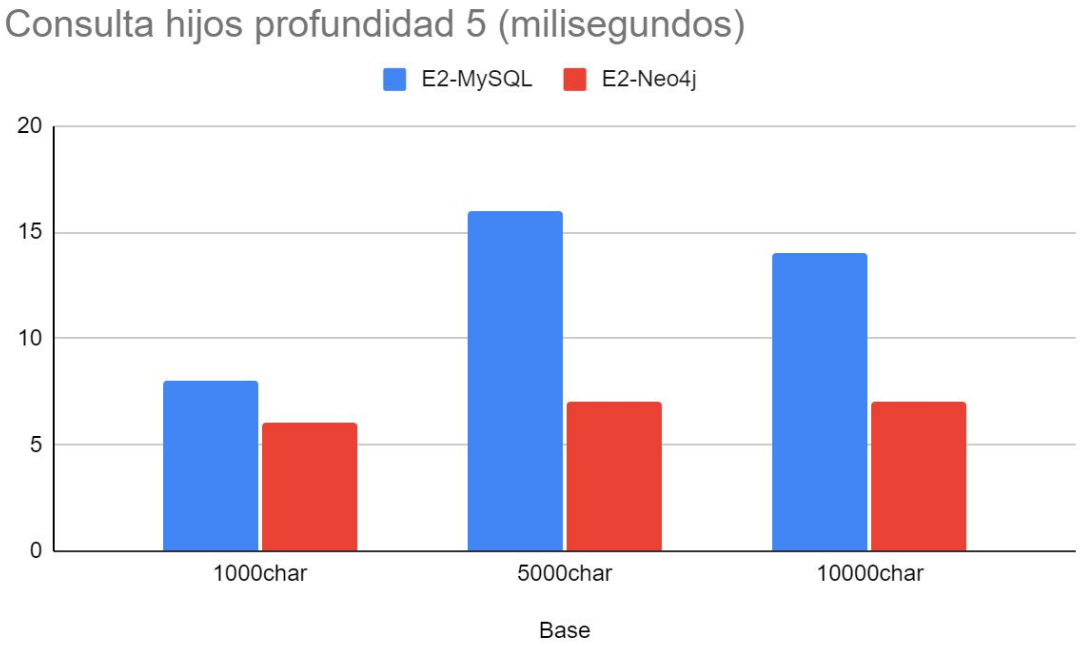
Resultados

Consultas estructurales



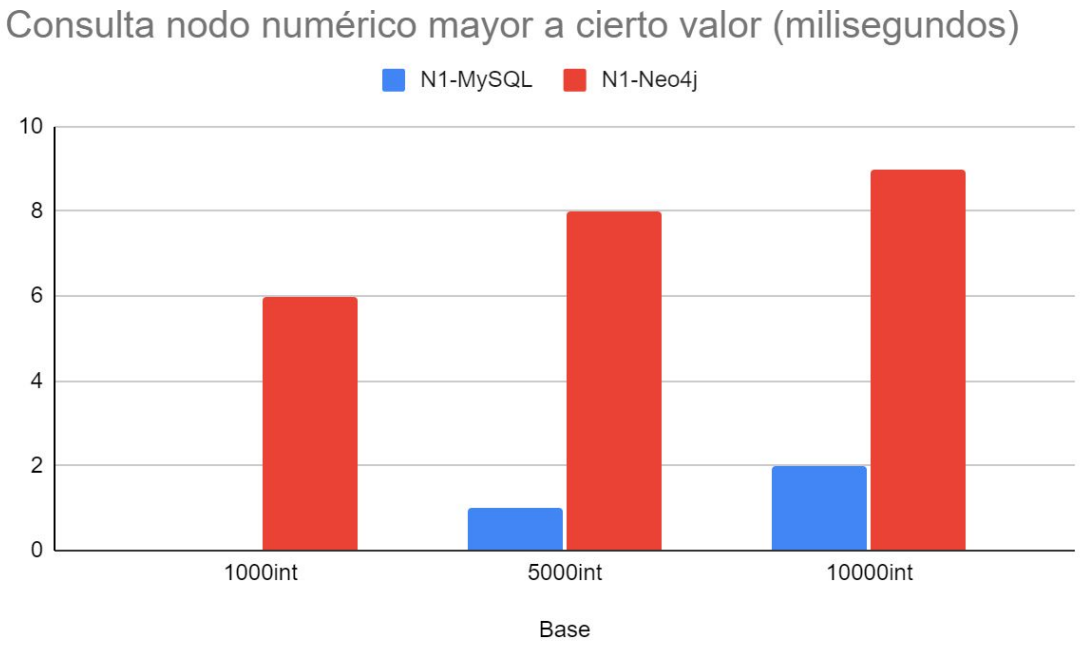
Resultados (2)

Consultas estructurales



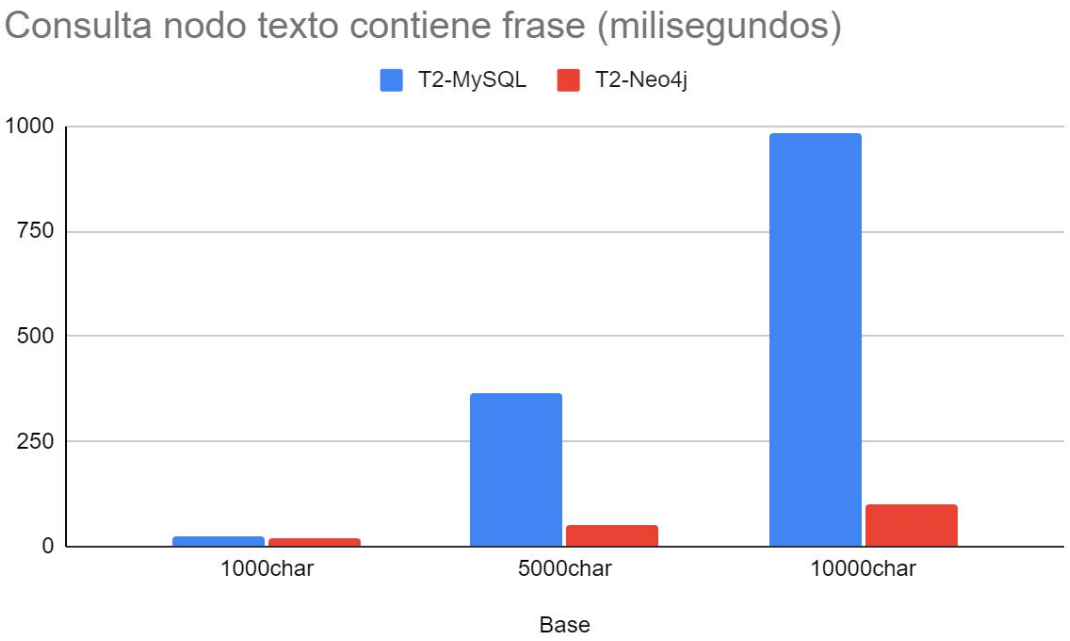
Resultados (3)

Consultas sobre datos del nodo



Resultados (4)

Consultas sobre datos del nodo



Conclusiones

- Ambas bases se comportan relativamente bien para el problema propuesto.
- MySQL mejor performance al consultar datos numéricos (índice en ambas bases sobre ese dato).
- Neo4j mejor performance sobre:
  - Consultas estructurales sobre el grafo (aunque no se ve tan clara la diferencia, faltó una consulta con profundidad mayor).
  - Datos texto (Lucene).

FIN

¿ Preguntas?