IIC3432 — Tópicos Avanzados en Bases de Datos

Tarea 1: Enunciado y Proyectos

1. Administrativos

Tu grupo debe investigar sobre uno de los temas al reverso de la página, y presentar los resultados obtenidos en una presentación de 15 minutos en la clase del 8 de Septiembre. Para los grupos armados (máximo 3 personas): Un responsable del grupo debe enviar un correo indicando el grupo y las primeras tres preferencias de proyectos (en orden) a Juan. Para las personas que no tienen grupo, pero que les gustaría uno: deben enviar un correo con las primeras tres preferencias de proyectos (en orden) a Juan. Se recomienda hacerlo a más tardar el viernes 21 de Agosto a las 8am, de lo contrario tu grupo perderá prioridad.

2. FAQ (o PF)

¿Cómo se entrega el trabajo?
No se entrega, sólo debes hacer la presentación.

¿Pero y si tengo otras cosas? ¿Código? ¿Texto? ¿Números?
Mejor aún!! todo tipo de material que puedas generar nos va a ayudar a comprender mejor tu video (y rescatar lo que hiciste). Puedes subir eso al github.

¿Cuál es la estructura de la presentación?
Son 15 minutos, así que deberían ir al grano rápidamente: primero explicar lo que quieren hacer, luego detallar el problema, luego la solución. Piensen que la idea es que todos los alumnos del curso la puedan seguir.

Nuestra presentación solo habla de lo que había en un link en Google. ¿Estamos bien?
¡No! Todos estos proyectos están diseñados para que ustedes metan sus manos, y no se queden con lo de internet solamente: piden o un análisis empírico o el diseño de modelos formales. Aprovechen si la información que encuentren para hacer una mejor presentación.

Queremos hacer una presentación con letra tamaño 9, para que se alcance a ver todo lo mucho que trabajamos en esta tarea. ¿Estamos bien?
¡No! Una parte fuerte del curso es que logren aprender a resumir lo que investigaron en una presentación corta. ¡Esto no es fácil! En general lograr buenas presentaciones cortas requiere un mejor conocimiento del tema que una mala presentación larga; en este curso se espera que ustedes desarrollen lo suficiente como para ser capaces de resumir todo en conclusiones generales, y la presentación debe estar orientada en base a esas conclusiones.

No me tinca ningún tema
¡Puedes proponer uno tú también! Para eso mándale un correo a Juan.

Estamos atascados, nos cuesta decidir que hacer, o no sabemos como avanzar
Para eso estamos. El próximo martes voy a estar conectado recibiendo consultas, la idea es que aprovechen para trabajar en esa hora. Pueden también fijar una reunión mandándole un correo a Juan. Opcionalmente, pueden organizar un discord si quieren ir compartiendo cosas entre ustedes y/o entre varios grupos.

3. Proyectos

Nota: hay varios de estos proyectos que soportan dos grupos, pero ¡siempre y cuando no hagan exactamente lo mismo!

3.1. Proyectos que investigan sobre rendimiento de sistemas (benchmarking)

Para estos proyectos puedes apoyarte en `https://graphbenchmark.com/` o `https://github.com/graphMark/gmark`. Estos son benchmarks propuestos donde se pueden generar grafos y consultas. Pero también podrías diseñar tus propias consultas sobre alguna base de datos bien usada (por ejemplo, wikidata, o `https://github.com/snapframework/snap-benchmarks`). Si buscas patrones complejos, acá se propone un benchmark sobre wikidata: `https://users.dcc.uchile.cl/~ahogan/docs/SPARQL_worst_case_optimal.pdf`.

B1. Patrones en bases de datos de grafos . Comparar neo4j contra al menos dos bases de datos adicionales, ya sea de grafos, RDF, o incluso postgres (obviamente debes ir más allá de lo que está en los benchmarks de arriba). La comparación en este caso tiene que ver con rapidez de ejecución de patrones, y tal vez cosas por encima: agregación, union, diferencia, etc.

B2. Patrones en bases de datos de grafos . Comparar neo4j contra al menos dos bases de datos adicionales, ya sea de grafos, RDF, o incluso postgres (obviamente debes ir más allá de lo que está en los benchmarks de arriba). La comparación en este caso tiene que ver con rapidez de en la parte de consultas de caminos de largo variable, y tal vez en la parte de entregar caminos.

B3. Caminos versus BFS . Como vimos en clases, una forma que parece ser eficiente a la hora de consultar caminos es hacer alguna forma de BFS. En este proyecto la idea es comparar la ejecución de Neo4j contra un algoritmo que use el mismo esqueleto que neo4j pero haga BFS (una forma fácil de lograr esto es montar el algoritmo en python y recurrir al driver de Neo4j para pedir vecinos, nodos y demases). Acá hay algo más de info `http://ceur-ws.org/Vol-1666/paper-04.pdf`.

3.2. Proyectos que investigan sobre Neo4j y Cypher

NC1. Modelo de datos . Este proyecto tiene que ver con el modelo de datos en Neo4j. En qué consiste, cómo se cargan y almacenan los datos, que tan rápido es al cargar, cuán fácil es cargar datos desde otras fuentes como csv, etc.

NC2. Índices en Neo4j Explicar y analizar qué tipo de índices soporta Neo4j. ¿Cuándo son estos índices útiles? Demostrar su utilidad de forma empírica.

NC3. Planes de consulta y optimización en Neo4j Explicar y analizar cómo funciona la planificación de consultas en Neo4j. El análisis puede ser teórico, explicando un modelo formal de cómo se generan sus planes, o práctico, analizando el impacto de las operaciones disponibles para el motor de consultas.

NC4. Computo de variable-length patterns Explicar en profundidad el algoritmo que usa Neo4j para computar consultas de caminos (variable-length patterns). Identíficar cuales son los puntos críticos que hace que no funcione tan bien, proponer mejoras.

3.3. Proyectos que investigan sobre el poder expresivo de cypher

Para estos proyectos puede que te sea útil `http://homepages.inf.ed.ac.uk/libkin/papers/sigmod18.pdf` o un poco más simple `http://ceur-ws.org/Vol-1912/paper27.pdf`.

EC1. Path unwinding Cypher tiene herramientas para transformar caminos de vuelta en una lista. Esto puede ser de forma implícita, como en la siguiente consulta, que pregunta por dos caminos disjuntos entre c1 y c5 (un problema que es np-completo):

```
MATCH p1 = (a:City {name:"c1"}) -[*]- (b:City {name:"c5"})
MATCH p2 = (a:City {name:"c1"}) -[*]- (b:City {name:"c5"})
WHERE none(x IN nodes(p2) WHERE (x IN nodes(p1) AND x<>a AND x<>b))
RETURN p1, p2
```

O de forma explícita, con el operador UNWIND. Este proyecto debe explicar hasta donde llega esta idea de comparar y/o desgranar caminos, ver cómo se evalúa y para qué se podría usar.

EC2. Variable-length patterns Proveer una comparación formal entre las consultas de caminos expresables en Neo4j y aquellas expresables como path queries con expresiones regulares (ojo que uno puede hacer cosas más allá de los variable-length patterns que podrían añadir al poder expresivo en términos de consultas, como diferencia o union de consultas.

EC3. Cypher vs otros lenguajes . Hay varios otros lenguajes de grafos: SPARQL, gremlin, y algunos otros. Acá se trata de dar una comparación formal de la expresividad de un lenguaje, contra otro. Por ejemplo, podemos expresar todos los patrones de cypher (así como su semántica) usando SPARQL o gremlin?

EC4. Nulos en cypher En comparación con SQL, ¿Cómo funcionan los nulos en cypher? ¿Pueden insertarse valores nulos en los grafos? ¿Atributos nulos? ¿Etiquetas? ¿Son los nulos igual de contradictorios que en SQL?

3.4. SCH1. Esquemas para grafos

No hemos discutido sobre esquemas, pero también es una parte importante en sistemas de bases de datos. En grafos, la noción de esquema tiene que ser un poco más flexible que una idea de "esquema relacional", pero tiene que indicarnos más o menos qué tipo de cosas se almacenan en grafos.
Puedes ver lo que hace Neo4j acá: `https://neo4j.com/docs/getting-started/current/cypher-intro/schema/`, y algunas otras opciones en `http://olafhartig.de/files/HartigHidders_GRADES2019_Preprint.pdf`, `http://jreutter.sitios.ing.uc.cl/SHACL_18.pdf` (para RDF).
¿Cómo se comparan estas opciones? ¿Cuál es más expresiva? ¿Hay alguna forma de unificarlas?

3.5. SA1. Estado del arte: Bases de datos de grafos

¿Qué otras bases de datos de grafos existen? ¿Usan también algo parecido a Property Graph? ¿Cómo se comparan con Neo4j? Mostrar ejemplos, instalando los sistemas y experimentando de la misma forma que lo hecho en clases.

3.6. GDB1. Trabajo con graphDB

GraphDB es el nombre postizo que tiene la base de datos de grafos in-house que estamos desarrollando en el Instituto Fundamentos de los Datos (en C++). En la primera etapa del proyecto, pueden analizar y estudiar lo que lleva graphDB hasta el momento, y proponer ideas de "pull-request"para añadir funcionalidades puntuales. Por ejemplo, un buen proyecto es proponer alguna forma de poder procesar uniones de patrones de grafos, lo que aún no está implementado. ¡Pero hay mucho que hacer en graphDB!

3.7. Propone tu propia tarea

Simplemente manden un mail a Juan para que quede confirmado el nuevo tema.