Escuela de Ingeniería en Computación

IC-4302 Bases de datos II

Investigación 1

Neo4j

Estudiante

2022091724 - Laura Vanessa Amador Salas

2021049519 - Wesley Yorjani Esquivel Mena

2023207915 - Josué Daniel Soto González

2023086373 - Isaac Picado Ortega

Profesor

Erick Hernández Bonilla

Agosto 2024

Índice:

1. Objetivo de la Investigación

Objetivo general:

Describir los principales elementos del motor de base de datos Neo4j, incluyendo su historia, arquitectura, construcción, consistencia, usos y funcionalidad, con el fin de comprender sus ventajas y aplicaciones.

Objetivos específicos:

Examinar los antecedentes, el funcionamiento, la arquitectura, construcción y consistencia del motor de base de datos Neo4j.

Analizar los usos y las experiencias de desarrollo actuales de Neo4j en diferentes sectores.

Desarrollar un prototipo funcional usando Neo4j que demuestre las ventajas del motor de base de datos.

2. Antecedentes

En el año 2000 los fundadores de Neo4j Emil Eifrem, Johan Svensson y Peter Neubauer comenzaron a construir el primer prototipo de la base de datos debido a que encontraron problemas para manejar grandes cantidades de datos interconectados en las bases de datos relacionales tradicionales. En 2007 fundan la compañía Neo4j. En el año 2009 realizaron el primer lanzamiento comercial de la base de datos para uso de empresas. En 2012 se realizó el lanzamiento de la versión para uso general (Canvas Business Model, 2024).

En 2015, Neo4j lanzó la versión 3.0, que introdujo características significativas como soporte nativo para datos masivamente distribuidos y la creación de un protocolo de comunicación completamente nuevo llamado Bolt. Este protocolo mejoró la eficiencia en la transmisión de datos entre el servidor y el cliente. Durante los últimos años Neo4j impulsó el uso de Cypher, un lenguaje de consulta gráfica intuitivo que simplifica la manera en que los desarrolladores consultan y manejan los datos dentro del grafo (Canvas Business Model, 2024).

3. Descripción Detallada

Según Anglès (2018) Los sistemas de bases de datos de grafos se diferencian de otros modelos en que se administra a un grafo para empeñar las funciones de una base de datos, son una de las categorías de bases NoSQL, donde cada unidad de información es un nodo y pueden conectarse entre sí. Un grafo está compuesto por sus nodos y las relaciones que los interconectan, aunque las bases de grafos utilizan un tipo de grafo llamados grafos de propiedad. Se caracteriza por ser capaces de etiquetar las relaciones con un nombre como para almacenar en la relación las propiedades e información que une a ambos nodos, este tipo de grafo se utiliza tanto en Neo4j como en otras bases de datos de grafos aunque difieren en su esquema y el modelo de diseño. Es un proyecto open source y también sigue a los principios de ACID para operar a pesar de también ser parte de la familia de bases NoSQL, cada nodo representa a una entidad discreta las cuales pueden relacionarse entre sí.

En la investigación hecha por Fernandes (2018) se afirma que en comparación con los otros motores de grafos Neo4J tiene una mínima rigidez con su Schema, solo teniendo en consideración la propia estructura del grafo y permitiendo el uso de índices y constraints. Los índices se utilizan como puntos de partida en el grafo para realizar operaciones y navegar la red de nodos, estos pueden asignarse a un nodo según una llave única para el nodo para identificarlo, también se puede asegurar que un valor sea único con un constraint de unique.

Los constraints son reglas que deben cumplirse sobre el dominio de la base de datos, en Neo4J también cumplen el propósito de filtrar las operaciones que sean realizadas para mantener las condiciones del Schema y las propiedades de los nodos. Para realizar sus consultas se utiliza un lenguaje llamado Cypher, que tiene similitudes con SQL, pero su sintaxis difiere y en lugar de “select” se utiliza un operador “match” para encontrar los datos que siguen al patrón de las condiciones del query. Una situación importante según la propia documentación de Neo4J es como diferenciar cuándo colocar una propiedad de información dentro de un nodo o en una relación hacía otro nodo para simplificar el patrón para el query de búsqueda.

4. Funcionamiento, Arquitectura y Construcción Interna

4.1 Álgebra de Grafos

Neo4j es una base de datos no relacional diseñada para almacenar nodos y las relaciones que hay entre ellos. A diferencia de la mayoría de bases de datos que están basadas en el álgebra relacional para su funcionamiento, Neo4j está basada en la teoría de grafos e intenta maximizar las propiedades de los grafos. Esto significa que la base de datos internamente hace uso de distintos algoritmos de recorrido de grafos para realizar sus consultas como el “a estrella”.

Los vértices en esta álgebra pueden tener aristas para conectarse entre ellos, en Neo4j a estos se les llama nodos y las conexiones entre ellos son denominadas relaciones. Los nodos pueden tener propiedades que se caracterizan por ser pares clave-valor y son accedidos mediante el nodo como clave. Las relaciones entre los nodos son dirigidas de un nodo inicial a un nodo final y pueden tener propiedades que permitan expresar con mayor detalle la naturaleza de la conexión entre los nodos. Un nodo de Neo4j puede tener varias relaciones con otros nodos y viceversa.

4.2 Cypher

Para implementar las consultas en la base de datos de grafos se creó el lenguaje de consultas Cypher. Cypher es un lenguaje declarativo de consultas que se utiliza en Neo4j y algunas otras bases de datos de grafos para hacer consultas. Cypher fue creado principalmente por Andrés Taylor mientras trabajaba para Neo y fue diseñado para permitir el acceso fácil y rápido a los nodos en grafos.

Cypher fue fuertemente influenciado por SQL durante su diseño. Una ventaja de esto es que al SQL ser uno de los lenguajes de consultas más populares del mundo, la introducción de Cypher a una persona que ya conoce SQL es más directa y sencilla permitiendo que la familiaridad del código ayude a los usuarios a aprender el lenguaje.

Las consultas en Cypher están diseñadas de forma similar a SQL como se puede ver en la Figura. Primero se selecciona el grafo donde se va a consultar (en caso de existir múltiples

grafos) utilizando la cláusula USE. Luego en la cláusula de MATCH se emparejan los nodos y relaciones deseados mediante flechas dibujadas en ascii (-> y <-), dependiendo de la relación buscada se pueden realizar múltiples MATCH en una misma consulta. Se pueden obtener resultados parciales de una consulta a otra mediante la cláusula WITH lo que permite anidar consultas de manera similar a como se realiza en SQL. Finalmente con WHERE se filtran los resultados de acuerdo con condiciones específicas. Para retornar los datos se especifican en la cláusula RETURN y la consulta otorga los resultados.

```
[USE]
[MATCH [WHERE]]
[OPTIONAL MATCH [WHERE]]
[WITH [ORDER BY] [SKIP] [LIMIT]
[WHERE]]
RETURN [ORDER BY] [SKIP] [LIMIT]
```

Figura: Comandos usados en el lenguaje Cypher.

Cypher fue diseñado para buscar relaciones entre nodos en grafos con grandes cantidades de conexiones de la forma más óptima posible pero sin complicar la implementación del usuario. A diferencia de varios lenguajes de consultas imperativos, Cypher es declarativo y se enfoca en los datos que se desean obtener y no en la forma de obtenerlos. Esto implica que se puede dar mayor optimización por parte del motor de bases de datos a las consultas (Lindaaker, 2012).

4.3 Recorridos (Traversals)

Para recorrer los grafos Neo4j diseñó los recorridos (traversals en inglés) que son una parte esencial de los algoritmos de consulta y lo que Neo4j utiliza para realizar sus consultas. En la cláusula MATCH de las consultas se definen las relaciones y nodos deseados, esto se puede representar con un grafo de los datos deseados. Un ejemplo serían todas las personas que son seguidas por una persona A y una persona B a la vez que A sigue a B como se muestra en la Figura. Esta consulta se representa de la forma: a->b, b->c, a->c.

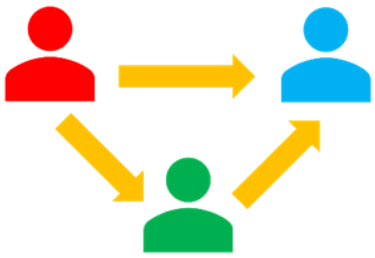

Figura: usuarios conectados

Los recorridos se aseguran de que los datos buscados sigan la estructura definida previamente sin buscar de forma exhaustiva a través de todo el grafo para encontrar las relaciones. Estos primero obtienen todas las posibles relaciones a partir del nodo base (o grafo base) ya establecido y filtra las que ayuden a formar la relación requerida, esto asegura que la exploración obtenga todos los resultados a la vez que la dirige hacia la dirección correcta. Otro aspecto útil de esto son el uso de constraints que permitan buscar sólo resultados únicos, esto

descarta loops de datos que se puedan desencadenar durante el recorrido si el usuario así lo desea (Lindaaker, 2012).

Todas estas optimizaciones hacen que Neo4j sea una de las bases de datos de grafos con mejor velocidad y fácil implementación. Gracias a estos recorridos el tiempo de muchas consultas se reduce considerablemente de una forma que no podría realizar una base de datos relacional. Un ejemplo muy simple de esto es buscar todos los amigos de una persona, en una base de datos relacional esto depende de la cantidad de personas guardadas en la base de datos, ya que la consulta debe verificar todas las relaciones que hay entre personas para determinar los amigos de una persona determinada por lo que escala con la cantidad de personas en la base de datos. En Neo4j este proceso recibe una gran optimización ya que se parte del nodo requerido y se utiliza un recorrido para encontrar todos los nodos de personas que son amigas de la primera, gracias a estos mecanismos esto se puede realizar en tiempo constante sin importar la cantidad de personas almacenadas en la base de datos (Lindaaker, 2012).

4.4 Almacenamiento Interno

Neo4j guarda los nodos, relaciones y propiedades de ambas mediante listas enlazadas de las relaciones creadas. Para un nodo utiliza un espacio de 9 bytes que funciona como puntero al nodo y sus propiedades. El espacio se divide en un byte inicial que indica si el nodo está siendo usado, esto para manejo eficiente de recursos y sobreescrituras. El siguiente espacio está compuesto por 4 bytes que sirven como puntero al nodo inicial en la relación permitiendo el acceso rápido a las relaciones asociadas a un nodo. Finalmente los últimos bytes se utilizan para guardar el puntero a la primera propiedad del nodo, funcionando como un puntero a la lista de propiedades (Lindaaker, 2012).

Las relaciones se manejan de forma similar pero con más propiedades para realizar búsquedas eficientes. Se tienen 33 bytes divididos en 8 punteros de 4 bytes que ayudan a encontrar las distintas referencias requeridas. Se tiene el nodo inicial de la relación y el final que indican cuáles nodos son parte de la relación y la dirección de la misma. Luego se tiene un puntero a la relación en el catálogo de relaciones del grafo. Los siguientes cuatro campos están vinculados con las relaciones hacia y desde estos dos nodos para buscar eficientemente las conexiones en consultas más complejas. Finalmente se asocian los últimos 4 bytes con un puntero a las propiedades propias de la relación (Lindaaker, 2012).

5. Consistencia

En cuanto a consistencia, Neo4j es una base de datos que proporciona cumplimiento ACID. Esto significa que sigue los principios de las características ACID (atomicidad, consistencia, aislamiento y durabilidad). Esto hace que sus transacciones tengan las siguientes características (Robinson, 2020):

a. Las operaciones dentro de sus transacciones se completan todas, o no se completa ninguna.
b. Las transacciones no se afectan entre sí hasta que se completen.
c. Una vez que se confirma una transacción, los cambios persisten incluso en cuando ocurre un error en el sistema.
d. Después de cada transacción, la base de datos permanece en un estado que cumple con todas las restricciones que se asignaron al crear el modelo de grafo.

La arquitectura de clustering de Neo4j permite que las empresas utilicen sus bases de datos en producción. Primeramente provee una solución con alta disponibilidad si una instancia tiene una falla, haciendo que otra instancia la reemplace automáticamente. Segundo, provee alta escalabilidad en donde algunas partes de la aplicación actualizan los datos, pero otras están distribuidas ampliamente y no necesitan acceso inmediato a los datos (Neo4j, 2024).

La consistencia casual en Neo4j significa que una aplicación garantiza constantemente poder leer todos los datos que ha escrito. Este tipo de consistencia resalta en Neo4j, ya que sus bases de datos son implementadas usando arquitectura de clustering con regularidad, para asegurar alta disponibilidad y consistencia (Neo4j, 2024).

6. Distribución

Por mucho tiempo Neo4j ha sido de las bases de datos de grafos más prominentes del mercado, pero siempre había sido limitada en términos de escalabilidad, ya que estaba restringida a correr en un solo servidor. A partir de 2020, con el lanzamiento de Neo4j 4.0, la compañía tomó el primer paso en la creación de una base de datos de grafos distribuída (Leetaru, 2020).

A día de hoy, Neo4j aún no es una base de datos completamente distribuida, pero tiene características distribuidas. Con la funcionalidad de múltiples bases de datos introducida en la 4.0, esencialmente se permite a los usuarios fragmentar una sola base de datos de grafo en múltiples subgrafos que pueden ser administrados de manera distribuida a través de Fabric (Leetaru, 2020).

Fabric es una funcionalidad añadida a Neo4j 4.0, es una manera de almacenar y recuperar datos a través de múltiples bases de datos. Esta funcionalidad hizo más fácil la consulta de información en distintos sistemas de bases de datos utilizando Cypher. Se utiliza como una herramienta de infraestructura para federación de datos (acceder a información disponible en fuentes distribuidas en forma de grafos no conectados) y para fragmentación de datos (acceder a información disponible en fuentes distribuídas en forma de un grafo común particionado en varias bases de datos) (Bondy, 2019).

La distribución de Neo4j a través de múltiples nodos se da por medio de master-slave. Todas las transacciones son procesadas por el nodo maestro y eventualmente aplicadas a los nodos esclavo, la eventualidad se da en base a cierto intervalo de tiempo o cuando una interacción exige el uso de la versión más reciente. Los cambios se aplican a los nodos esclavos en base a el número de id de la transacción local. Los ids de transacción incrementan en orden conforme se hacen las transacciones lo que permite revertir cambios a una versión determinada y actualizar los datos en orden como se recibieron las transacciones (Lindaaker, 2012).

En caso de perder al maestro las instancias se comunican para elegir un nuevo maestro en base al id interno de transacciones y el último tiempo de su escritura. Las reelecciones ocurren cada vez que el maestro es inaccesible y cualquier instancia puede ser candidata para su candidatura. Una vez elegido el nuevo maestro, este es reportado a todas las demás instancias (Lindaaker, 2012).

7. Usos posibles y experiencias de Desarrollo Actuales

Representar las relaciones de las entidades en una base de datos es la mayor ventaja de una base de datos de grafos debido al énfasis en las relaciones de los nodos, esto les permite un

nicho como sistemas secundarios para manejar las relaciones de la base de datos principal. La base con las que más se tiende a utilizar como apoyo son las bases relacionales, en las tablas se pueden representar relaciones entre entidades mediante llaves foráneas. Sin embargo, en su investigación H. Lu et al (2017) indica que estas bases de datos pueden sufrir de tanto redundancia de datos cómo ser más rígidas y difíciles de modificar. Debido a esto también se utiliza en el campo de análisis de datos, Neo4J GDS (Graph data science) es una sección de neo4J dedicada para la ciencia de datos que contiene herramientas y algoritmos especializados para el análisis de los datos en los nodos y relaciones con gran profundidad.

Estas herramientas cumplen un rol importante también para completar un manejo efectivo de grandes volúmenes de datos y para encontrar patrones de información relevantes. Según los casos de uso recopilados por Neo4J, actualmente la NASA utiliza esta base de datos para brindar una mayor facilidad de acceso a su amplio historial de documentos. Desde los años 50's la organización acumuló más de 20 millones de documentos a lo largo de diferentes sitios, lo cual le hizo una tarea muy complicada a sus propios trabajadores analizar sus datos previos para investigaciones y desarrollo de tecnología. Sin embargo esto se pudo enmendar gracias a un subsistema de Neo4J para el manejo de archivos correlacionando temas cercanos. Otro uso similar ocurre dentro del Ejército de los Estados Unidos, donde usan a Neo4J(2021) como un sistema para administrar y seguir el proceso de mantenimiento de las armas y vehículos del arsenal situado alrededor del mundo, facilitando en gran medida la tarea de logística.

Además, se utiliza con frecuencia como base para motores de recomendaciones en algunas plataformas, la forma de la base de datos facilita el formar correlaciones entre el variado contenido de un sitio web a tiempo real. Tal es el caso con Ebay, la compañía de comercio recientemente implementó dentro del asistente de google una función para realizar búsquedas exactas con poca información, logrando esto mediante el enlazar patrones con Neo4J. Esta manera de utilizar a Neo4J implica que se utilice también a otras bases de datos en un sistema completo Polyglot, sirviendo como un sistema secundario.

8. Conclusiones

- Neo4j surgió en respuesta a los problemas de manejo de datos interconectados en las bases de datos relacionales. Desde su lanzamiento, ha ofrecido innovaciones como el protocolo Bolt y el lenguaje Cypher.
- La base de datos utiliza a un grafo especializado para almacenar sus datos y para cumplir con sus funciones, su lenguaje de consultas Cypher aunque especializado para navegar un grafo mantiene parentesco con SQL en su sintaxis. Debido a esto es considerada de las bases de datos más rápidas de implementar.
- Neo4j permite a sus usuarios obtener una solución eficiente para manejar grandes volúmenes de datos interconectados, algo que normalmente resulta problemático para las bases de datos relacionales tradicionales.
- Utilizando su modelo basado en grafos y su lenguaje de consultas dedicado Cypher, Neo4j proporciona mayores velocidades en cuanto a manejo y ejecución de consultas para relaciones complejas, como las conexiones entre usuarios en redes sociales o recomendaciones personalizadas.

- Las bases de datos de nodos ya tienen su lugar en la industria de las bases de datos, Neo4J actualmente es una herramienta valiosa en múltiples ámbitos como un sistema en las organizaciones que adoptan un estilo polyglot. La base de datos se ha utilizado tanto en empresas pequeñas como grandes organizaciones para tratar con manejo de datos, motores de recomendaciones y análisis.

9. Prototipo

Para demostrar un caso de uso óptimo para una base de nodos como Neo4J se decidió el desarrollo de un sistema de recomendaciones a tiempo real como un ejemplo sencillo, y también para demostrar el funcionamiento de la base de datos desde cerca. El sistema desarrollado es un prototipo de una página web con una sección para conectar a usuarios con hobbies parecidos, el usuario puede darle a like si tiene interés en el hobbie y dislike si por lo contrario no desea interactuar con el hobbie. La base de datos es capaz de identificar patrones en los usuarios para presentar recomendaciones relacionadas con los gustos de usuarios con conexiones similares y evitar recomendar aquellos hobbies con relaciones de dislike.

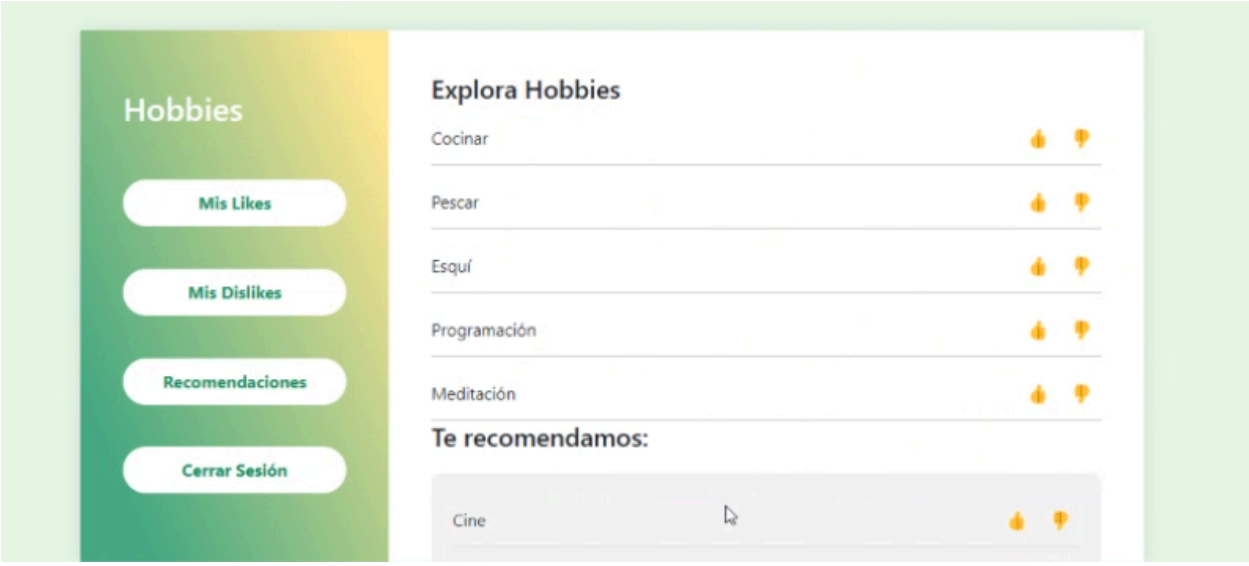

Figura: Interfaz del prototipo

Este prototipo demuestra uno de los usos más ideales para la base de datos, el sistema es capaz de actualizarse inmediatamente contra cada cambio para ser más interactivo con el usuario a tiempo real dando una experiencia más interactiva para el usuario. La base de datos provee en su propia interfaz a herramientas para analizar los datos con los nodos y las relaciones existentes a tiempo real, esto apoya al proceso de desarrollo y es importante para el análisis de datos de manera superficial con la red de nodos. Aunque la interfaz carece de herramientas visuales para modificar la base de datos dependiendo de comandos para todas las acciones, lo cual es un obstáculo para el desarrollo en comparación con otros motores como Oracle SQL que provee herramientas para esto.

10. Bibliografía

Angles, R. (2018). The Property Graph Database model. *AMW*. http://ceur-ws.org/Vol-2100/paper26.pdf

Bondy, C. (2019). *Primeros pasos con Neo4j fabric*. Neo4j. https://neo4j.com/blog/getting-started-with-neo4j-fabric/

Canvas Business Model. (2024). *A brief history of Neo4j*. Canvas Business Model. https://canvasbusinessmodel.com

Chang, V., Songala, Y., Xu, Q., & Liu, B. (2022). Scientific Data Analysis using Neo4j. *SCITEPRESS – Science and Technology Publications*, 75–84. https://doi.org/10.5220/0011036700003206

Chatham, M. (2012). Structured Query Language By Example - Volume I: Data Query Language. *Lulu*. ISBN 1291199519, 9781291199512.

Fernandes, D., & Bernardino, J. (2018). Graph Databases comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB. *7th International Conference on Data Science, Technology and Applications*. https://doi.org/10.5220/0006910203730380

Lindaaker, T. (2012, May 20). *An overview of Neo4j Internals* [PowerPoint slides]. Neo Technology. An overview of Neo4j Internals. https://es.slideshare.net/slideshow/an-overview-of-neo4j-internals/13009803

Lindaaker, T. (2010, April 20). *Graph Databases and Neo4j* [PowerPoint slides]. Neo Technology. NOSQLEU - Graph Databases and Neo4j . https://es.slideshare.net/slideshow/nosqleu-graph-databases-and-neo4j/3792258

Leetaru, K. (2020). *Neo4j se vuelve distribuido con la base de datos de grafos*. Datanami. https://www.datanami.com/2020/02/04/neo4j-going-distributed-with-graph-database/

Lu, H., Hong, Z., & Shi, M. (2017). Analysis of film data based on NEO4J. *16th International Conference on Computer and Information Science (ICIS)*. https://doi.org/10.1109/icis.2017.7960078

Neo4j. (s. f.). *Neo4j administración: Causal clustering en Neo4j*. Neo4j. Recuperado el 4 de octubre de 2024, de https://neo4j.com/graphacademy/training-admin-35/04-neo4jadmin-3-5-causal-clustering-neo4j/#:~:text=Causal%20consistency%20in%20Neo4j%20means,that%20utilize%20multiple%20data%20centers

Neo4j. (2022, July 6). *NASA - Graph Database & Analytics*. Graph Database & Analytics. https://neo4j.com/case-studies/nasa/

Neo4j. Graph technology keeps the army up and running by tracking and analyzing equipment maintenance. (2021). *Neo4J.com*. https://go.neo4j.com/rs/710-RRC-335/images/Neo4j-case-study-US-army-EN-US.pdf

Neo4j. (2024, April 23). *Intelligent commerce for EBay app on Google Assistant*. Graph Database & Analytics. https://neo4j.com/case-studies/ebay/

Robinson, I. (2020). *ACID vs. BASE: El cambio del pH en la gestión de transacciones de bases de datos*. Neo4j. https://neo4j.com/blog/acid-vs-base-consistency-models-explained/