Resumen #5 Graph Databases For Beginners

Dillan Almendares Barrantes - 2020033336

Capítulo 1

A continuación se explican algunas ventajas de las bases de datos orientadas a grafos frente a las bases de datos tradicionales.

Rendimiento

Un problema en las bases de datos tradicionales es el acelerado incremento en la cantidad de relaciones o joins que se deben hacer para consultar los datos, lo que provoca que se ralenticen por la cantidad y profundidad de las relaciones, mientras que las orientadas a grafos mantienen un rendimiento constante con el aumento de datos.

Flexibilidad

No requieren de gran planificación para el modelo y se pueden hacer agregaciones facilmente sin comprometer el modelo actual.

Agilidad

Funcionan bien con las metodologías de desarrollo agíl para que la aplicación pueda evolucionar con los nuevos requerimientos.

¿Qué son las bases de datos orientadas a grafos?

Un grafo está compuesto de dos elementos: nodos y relaciones. Cada nodo representa una entidad y una relación representa cómo están asociados 2 nodos.

Usando a Twitter como ejemplo, cada usuario se puede representar con un nodo y los follows serían relaciones entre estos nodos.

En las bases de datos orientadas a grafos la prioridad son las relaciones, para las aplicaciones esto evita tener que usas foreign keys o out-of-band processing como MapReduce para conectar los datos requeridos.

Dos propiedades importantes a tener en cuenta:

1. **Graph Storage:** Algunas bases de datos usan uno nativo que está diseñado para grafos, mientras que otras usan una modelos relacionales u orientados a objetos. Los graph storage nativos suelen ser más rápidos.
2. **Graph Processing Engine:** De igual manera existe uno nativo conocido como **"index-free adjacency"** que es la manera más eficiente porque los nodos se apuntan físicamente en la base de datos, por el otro lado, los no nativos usan otras maneras para procesar las operaciones de CRUD.

Las bases de datos orientadas a grafos están creciendo en su uso por su funcionalidad para grandes datasets y porque permite representar elementos de la vida real por su representación de las relaciones.

Capítulo 2

Un gran problema de las bases de datos relacionales es la poca flexibilidad que tienen sus esquemas, perjudicando cuando hay cambios en los requerimientos y toma mucho trabajo implementar esas necesidades al esquema.

En el caso de las bases de datos NoSQL, pueden tener mejor rendimiento pero por su estructura inconexa dificulta aprovechar las relaciones de los datos.

Ese es el problema que solucionan las bases de datos orientadas a grafos, ya que en los grafos las relaciones entre elementos están explícitamente representadas, así se pueden agregar nuevos nodos y relaciones sin afectar la red existente o se evita tener que migrar los datos a un esquema nuevo que soporte los cambios. Además de la flexibilidad, para las consultas donde interesan las relaciones entre los elementos, las bases de datos orientados a grafos son considerablemente más rápidas.

Capítulo 3

Modelado de datos

La etapa que consiste en la abstracción de los datos que se van a almacenar y la manera en que se estructuran, es otra etapa que se simplifica en las bases de datos orientadas a grafos, si inicialmente se prepara un modelo dibujado con los elementos necesarios, en las bases de datos relacionales se debe pasar forzosamente a un modelo físico tabular, mientras que con los grafos el modelo puede ser una réplica del modelo dibujado ya que posiblemente ya sea un grafo.

Hay que resaltar que el modelado no es solamente una etapa inicial que solo se realiza una vez para cada aplicación, pueden surgir nuevas funcionalidades que cumplir que requieran de cambios en el modelo y las bases de datos orientadas a grafos facilitan estos cambios.

Capítulo 4

Errores que evitar en el modelado de datos

Se usará como ejemplo una aplicación para detectar fraudes por email.

Establecer como una forma de relación algo que sería conveniente como otro tipo de nodo, hay que tener claro el objetivo de la aplicación y que el modelo facilite ver las relaciones del elemento principal y no usarlo como una posible relación entre nodos que no son prioritarios para el objetivo. En el caso de la aplicación, si el objetivo es rastrear todas las relaciones del email, antes de conectar nodos de **usuarios** con **EMAILED** para representar que uno le envió un correo a otro, sería mejor tener un nodo de **Email** con relaciones como: **SENT, TO, CC y BCC** con nodos de usuarios.

Otro error sería si para representar los usuarios que respondieron a un email se hace mediante una relación como **REPLIED_TO** de **Usuario** a **Email**, cuando se podría agregar otra etiqueta a un nodo **Email** para representar que es una respuesta.