Graph Database: Neo4J da teoria a pratica.

Eduardo Pereira da Silva[1], **Iuri Andreazza**[1], **Paulo Gräbin**[1], **Talita Audibert**[1]

[1]Universidade do Vale dos Sinos (UNISINOS)
São Leopoldo – RS – Brasil

{eduardobursa,iuri.andreazza,plgrabin,tali.audibert}@gmail.com

Abstract. *This article aims to provides an overview about the core concepts of a no-SQL database that uses graphs as representative model and operations using another language rather than SQL to fetch data. It's also reviewed the pratics from the creation of the project to some basic queries with some data to illustrate your queries and functionality. Finally it is possible to view the use of this database and how we can have big gains when use it correctly.*

Resumo. *Este artigo traz uma visão geral dos conceitos de um banco no-SQL que utiliza grafos como modelo representativo dos seus dados e operações e utilizando de uma outra linguagem do que o SQL para consultas aos seus dados. Também é tratado a utilização, desde a criação do projeto até algumas consultas basicas em alguns dados para exemplificar suas consultas. Por fim é possivel visualizar o uso deste tipo de banco de dados e como podemos ter grandes ganhos quando utilizarmos corretamente.*

1. Introdução

1.1. Organização

Este artigo está organizado da seguinte maneira, tendo que, na sessão 1 temos um geral do artigo e a introdução do assunto, já na sessão 2 é tradado dos conceitos básicos do banco de dados como também um geral da teoria dos grafos que está sendo usado. Na sessão 3 é apresentado o uso prático do banco de dados desde a criação do projeto até sua utilização dentro da linguagem *java*, durante a sessão 4 é apresentado os trabalhos relacionados com este artigo e ao final na sessão 5 é mostrado quais beneficios que podemos ter ao utilizar esse tipo de abordagem.

2. Conceito

Um banco de dados orientado a grafo não é um banco de dados para armazenar gráficos ou imagens como o nome pode sugerir. É um banco de dados que usa estruturas de grafos, como nós, propriedades e arestas para representar e armazenar dados. Além disso, ele permite que você represente qualquer tipo de dados, sem a limitação de bases de dados regulares, ou seja os relacionais tradicionais.

Para entender melhor, vamos falar sobre cada uma dessas estruturas individualmente eo que eles representam:

Nó: Um nó pode ser usado para representar qualquer tipo de entidade, seja ele uma empresa, um blog, um local, uma plataforma de petróleo, uma cidade. Um bando de dados orientado a grafos não se importa com o tipo de dados que se está sendo operado.

Propriedades: Propriedades, às vezes chamados de atributos, são os valores nomeados que dizem respeito a nós. Por exemplo, se levarmos em consideração a representação da cidade em um nó, uma das propriedades seria ”nome”, outra seria ”população”, e assim por diante.

Arestas: Arestas, às vezes chamados de Relacionamentos, serve para conectar os nós e organizá-los em estruturas arbitrárias, como um mapa, lista ou uma árvore. É importante notar que quando um nó é o nó de início de uma relação, a relação do ponto de vista de que o nó será uma relação de saída, porém quando um nó está no fim de uma relação, será uma relação de entrada.

2.1. SQL Tradicional

Em uma estrutura SQ: tradicional, cada tupla na tabela representa uma informação na qual pode ser relacionada com outras tuplas em outras tabelas. Se for decidido adicionar propriedades para as tabelas em outro momento será necessário usar instruções DML para alterar a estrutura da tabela e tratar caso a caso em caso de existencia de restrições e outros eventos relacionados com suas propriedades e mesmo que se fosse preciso adicionar propriedades apenas para algumas tuplas da tabela será necessário igualmente alterar a tabela ou mesmo criar uma nova estrutura para comportar esse novo conjunto de informações, este cenário não é bom quando é existem milhões de registros que necessitam de dados especificos.

2.2. Graph Database

Neo4j é a applicação lider em banco de dados orientados à grafos sendo suportado comercialmente e tendo sua versão *open-source* como fonte de fomento para novos usos assim chamando a comunidade para dar suporte a ferramenta. As principais caracteristicas tem como base a representação intuitiva de dados usando um modelo orientado a grafo. A portabilidade para um número de linguagens de programação, incluindo Ruby, Python e Java. Capacidades de um SGBD, armazenamento nativo completamente otimizado para armazenar estruturas dos grafos para com o máximo de desempenho e escalabilidade. Escalabilidade maciça, podendo lidar com grafos de vários bilhões de nós / relações / propriedades em uma única máquina. Fácil de usar e conveniente possui uma API orientada a objetos. Com grafos grandes não cabem em memória, utiliza um mecanismo de persistencia de alta velocidade que possibilita varreduras em nós em perda de desempenho com o mecanismo transacional funcional. Quadro travessia poderosa para travessias de alta velocidade no espaço de nó. Otimizado para dados altamente conectados. Pegada pequena, somente cerca de 750k arquivo jar. Ele fornece uma interface REST para linguagens não suportadas nativamente. É capaz de atravessar mais de 1000 níveis de profundidade em velocidades de milissegundos e se integra perfeitamente com o Lucene, proporcionando pesquisa de texto completo para nós e relacionamentos, incluindo consultas de frase, consultas curinga, consultas de proximidade, a pesquisa classificatórias e muito mais.

2.2.1. Estrutura da base de dados

On the other hand, a Graph database has no set structure or schema for the data, much like a NoSQL database. Now, let?s understand what the graph above is representing: each

node represents an entity ? a User in our case; each node contains property values, in this case it?s the User?s name; and each line represents a relationship between the nodes. This is as simple as it gets. And to complement the scenario described in the traditional SQL example, if we did decide to add additional properties to a subset of users, we could easily perform this action on a per-node level instead of a table-wide transaction.

grafo

3. Abordagem Prática

Graph Database Example

Let?s examine the following scenario and how we would structure it using a Graph Database: John is friends with Bob, and Bob is friends with Mark. Visually, this is how we could represent the scenario:

You notice that the visual representation is pretty much how we verbally expressed our scenario. This is a very simple example that we can use to compare the implementation of a traditional SQL database structure versus a Graph database structure:

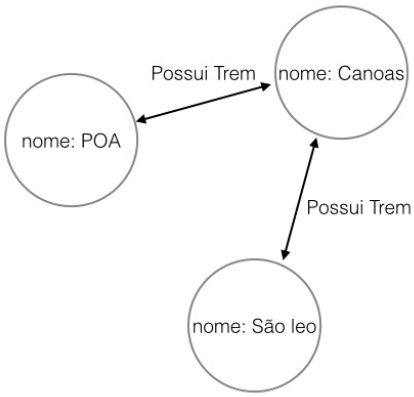

Figure 1. Estrutura base do grafo representando cidades ligadas por uma linha de trem

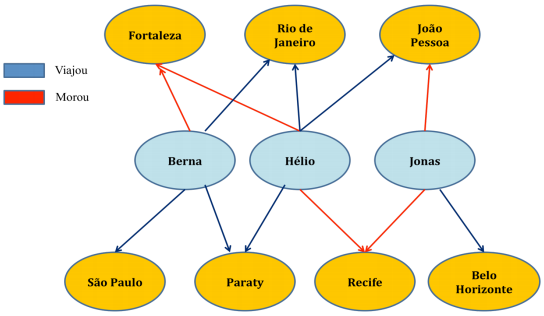

Figure 2. Exemplo mais complexo de um grafo

4. Trabalhos relacionados

5. Conclusão

References

Boulic, R. and Renault, O. (1991). 3d hierarchies for animation. In Magnenat-Thalmann, N. and Thalmann, D., editors, *New Trends in Animation and Visualization*. John Wiley & Sons ltd.

Knuth, D. E. (1984). *The TEX Book*. Addison-Wesley, 15th edition.

Smith, A. and Jones, B. (1999). On the complexity of computing. In Smith-Jones, A. B., editor, *Advances in Computer Science*, pages 555–566. Publishing Press.