Bancos de dados orientados a grafos

Uma comparação conceitual e de performance com as ferramentas Neo4j e PostgreSQL

Gabriel Takeshi Andrade Higa

Orientação: Prof. Dr. João Eduardo Ferreira

Instituto de Matemática e Estatística – Universidade de São Paulo

Resumo

Neste trabalho, oferecemos uma comparação justa entre bancos de dados orientados a grafos e bancos de dados relacionais. As duas ferramentas escolhidas para os testes práticos foram o Neo4j e o PostgreSQL. Começamos explicando a diferença entre os modelos lógicos e como eles afetam o processo de modelagem. Além disso, comparamos a facilidade de expressar nossas consultas nas duas linguages estudadas, SQL e Cypher. Em seguida, apresentamos uma análise detalhada explicando como as consultas são executadas e as diferenças de performance para os dois casos. Para complementar, demonstramos como algoritmos em grafos resolvem problemas de forma eficiente no Neo4j.

Palavras-chave: banco de dados, modelo relacional, grafos, comparação de performance.

Introdução

Bancos de dados orientados a grafos são uma de muitas categorias de bancos de dados (BDs) NoSQL propostas nos últimos anos. A grande diferença para o modelo clássico está na representação explícita de relacionamentos entre os dados, através de um modelo com vértices, chamados de *nós*, e arcos, chamados de *relações*. Esperamos que esse modelo, menos genérico que o modelo relacional, nos forneça uma modelagem mais simples, uma linguagem mais natural para descrever nossas consultas e melhor performance, tanto pela implementação livre de operações custosas como `JOINs`, quanto pelo uso de algoritmos de grafos.

Para verificar as vantagens e desvantagens do modelo de grafos, estudamos a ferramenta Neo4j em comparação o PostgreSQL, um sistema de gerenciamento de BDs relacional, ambas de código aberto. O modelo de dados construído foi um grafo composto pelos artigos da Wikipedia e as hiperligações que conectam uma página à outra. A inspiração foi a hipótese popular ”Seis graus de separação”, adaptada para vários contextos diferentes, incluindo os artigos da Wikipedia. Um artigo descrevendo a ideia pode ser encontrado em https://en.wikipedia.org/wiki/Wikipedia:Six_degrees_of_Wikipedia.

Objetivos

Enquanto este trabalho foi desenvolvido, notamos uma falta de testes de performance bem documentados e imparciais. Esperamos suprir essa necessidade, além de oferecer uma visão geral do porquê o uso de BDs orientados a grafos traria vantagens para algumas aplicações. Dessa forma, nossos objetivos principais foram:

1. Comparar os dois modelos lógicos e demonstrar como um domínio próprio para grafos é facilmente modelado para o Neo4j;
2. Escrever consultas nas linguagens SQL e Cypher (a linguagem utilizada pelo Neo4j), comparando a facilidade e flexibilidade com que podemos expressar nossas perguntas;
3. Analisar cuidadosamente as performances na execução das consultas descritas anteriormente, explicando como as ferramentas obtém o desempenho exibido nos testes e o que isso significa para uma aplicação real;

Resultados de Performance

Para comparar a performance, foram escritas consultas em ambas as linguagens para encontrar todos os artigos que podem ser alcançados a partir de alguma página inicial fixa qualquer em menos de n hiperligações. Nos gráficos a seguir, podemos ver quanto tempo cada ferramenta levou para concluir as consulta para diferentes valores de n. Os testes foram executados no mesmo computador, com configurações similares para as duas ferramentas.

Foram feitos dois experimentos, um evitando expansões sobre nós repetidos e outro não. A razão para isso é que a consulta padrão do Neo4j busca todos os caminhos possíveis entre dois nós, levando a um trabalho extra desnecessário para as consultas do experimento. Como no PostgreSQL era muito simples evitar isso, e como o Neo4j permite expressar a mesma coisa, ainda que de forma não natural, testamos e exibimos aqui o resultado de ambas as formas de consulta, chamadas de consulta com filtro e sem filtro, para tornar o experimento mais justo.

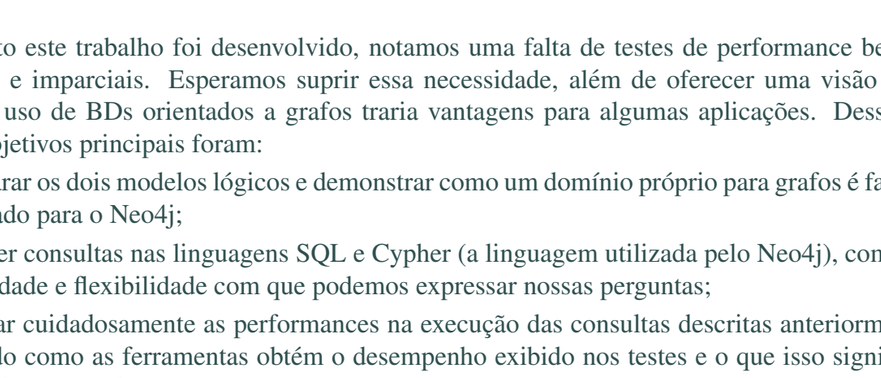

Figura 1: Sem filtro, forma natural da expansão de caminho variável do Neo4j.

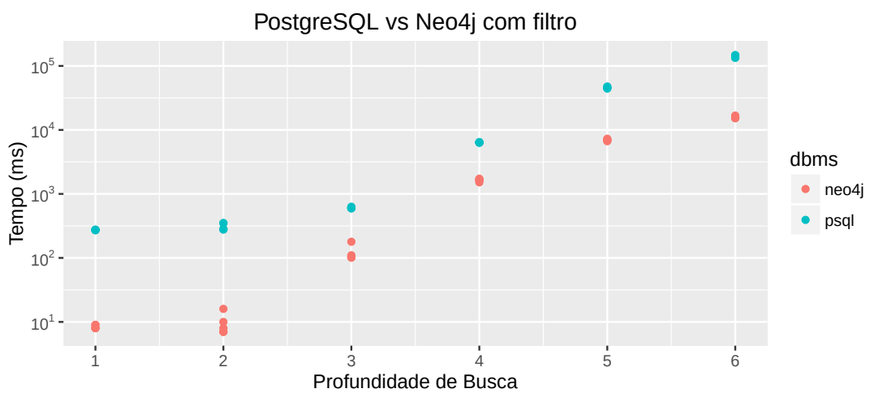

Figura 2: Com filtro, evitando expansão sobre nós repetidos.

Observamos que em ambos os casos o Neo4j ganha por aproximadamente uma ordem de magnitude. Essa performance pode ser atribuida ao conceito de *adjacência livre de índices*, que substitui os custosos `JOINs` do modelo relacional, resultado de uma representação subjacente nativa de grafos, um dos principais pontos de venda do Neo4j.

Conclusões

- O uso de um modelo de grafos compensa a partir do momento que se tem em mente uma aplicação altamente focada no relacionamento dos dados, levando a uma modelagem direta.
- É necessário ter em mente as limitações do Cypher como uma linguagem declarativa, sendo ela menos flexível do que o SQL para expressar detalhes ou consultas que fogem ao modelo de grafos. Apesar disso, é notável sua expressividade e simplicidade para o domínio que foi projetada, além da disponibilidade de algoritmos em grafos apropriados para resolver alguns problemas.
- O Neo4j é mais rápido do que o PostgreSQL para consultas envolvendo os relacionamentos entre os dados e serve muito bem para evitar problemas de performance provenientes de muitos `JOINs`. Entretanto, a diferença não é grande o bastante para viabilizar o uso de uma ferramenta ou outra, com exceção do uso de algoritmos em grafos ou APIs de mais baixo nível no Neo4j, que não foram exploradas neste trabalho.

Ferramentas

PostgreSQL

neo4j

Referências

[1] R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems*. Pearson, 7th edition, 2016.

[2] PostgreSQL documentation. `https://www.postgresql.org/docs/9.6/static/index.html`. Accessed: 2016-11-13.

[3] I. Robison, J. Webber, and E. Elfrem. *Graph Databases*. O'Reilly, 1st edition, 2013.