Análise de Desempenho de Bancos de Dados em Grafos

Guilherme M. Silva[1]

[1]Departamento de Estatística e Informática (UFRPE)
Recife - PE - Brasil

guilhermemelo3451@gmail.com

Abstract. *The project proposed in this article aims to demonstrate methods for performance analysis of graph oriented databases. In this way the present work presents the analysis of the performance and the behavior of the database in graphs Neo4j, when submitted to queries in different datasets.*

Resumo. *O projeto proposto neste artigo tem por objetivo demonstrar métodos para análise de desempenho de bancos de dados orientados a grafos. Desta forma o presente trabalho apresenta a análise da performance e o comportamento do banco de dados em grafos Neo4j, quando submetido a cargas de consultas em diferentes conjuntos de dados.*

1. Introdução

Os últimos anos foram marcados pelo uso massivo de Tecnologias de Informação e Comunicação (TICs), cujo principal agente de transformação foi a Internet. Este cenário contribuiu para o avanço na geração de instrumentos de comunicação e compartilhamento de informações. Com a disseminação das TICs, a expansão das redes de informações e o anseio de se obter o melhor desempenho ao menor custo, contribuíram para aumentar o grau de exigência do mercado. [Alvarez and Ceci 2016]

As bases de dados NoSQL (Not Only Structured Query Language) vem ganhando popularidade na era da Web 2.0. Pois, a promessa de alto desempenho quando se envolve dados altamente interconectados, atraiu a atenção dos consumidores de tecnologia; uma tecnologia de banco de dados NoSQL que vem ganhando grande importancia nos últimos anos é o banco de dados em grafos. [Takeshi 2016]

O banco de dados em grafos surgiu como uma alternativa ao banco de dados relacional para dar suporte a sistemas cuja interconectividade de dados é um aspecto importante. Em um banco de dados de grafos relacionamentos são mais naturais e a maior diferença nesse modelo é uma representação explícita no modelo lógico do relacionamento entre os dados, que são implícitos no modelo relacional e somente visíveis através das consultas SQL utilizando custosos JOINs. [Takeshi 2016]

A escolha de um Sistema Gerenciador de Banco de Dados (SGBD), dentre a grande diversidade disponível no mercado é uma tarefa delicada, devido a importância e responsabilidade que essa ferramenta representa por gerenciar uma das maiores riquezas de uma organizacão, que são as informações. Nesta tarefa, vários fatores devem ser considerados, tais como confiabilidade, integridade e segurança dos dados, suporte a linguagem de programação, interoperabilidade, dentre outras. Contudo, com o crescimento constante do volume de dados que os SGBD‘s gerenciam, e que muitas vezes estão armazenados remotamente, um fator que vem se destacando como ponto decisivo para escolha de um SGBD é o **Desempenho**. [Ferreira and Junior 2014].

O projeto proposto neste artigo tem por objetivo demonstrar métodos para análise de desempenho de bancos de dados orientados a grafos. Desta forma o presente trabalho apresenta a análise da performance e o comportamento do banco de dados em grafos Neo4j, quando submetido a cargas de consultas em diferentes conjuntos de dados.

O trabalho está organizado em cinco seções principais. Após a introdução, a segunda apresenta os trabalhos relacionados ao projeto. A terceira seção descreve a metodologia utilizada. Na quarta, é apresentado o experimento e avaliação de resultados da solução proposta. Por fim, na última seção, é apresentado a conclusão deste artigo .

2. Trabalhos Relacionados

Para a evolução deste projeto alguns artigos foram estudados e analisados, com destaque para alguns. Em [Alvarez and Ceci 2016] os autores realizaram uma análise comparando o desempenho entre dois tipos de bancos de dados, O OrientDB que é um banco de dados multi-modelo e o Neo4J que é um banco de dados orientado a grafos. A solução proposta foi a utilização de informações fictícias simulando uma rede social sobre os dois tipos de bancos de dados. Verificou-se que o uso do banco OrientDB apresenta uma solução mais amigável, mas para consultas recursivas o Neo4J apresenta um melhor resultado.

O trabalho realizado em [Penteado and Schroeder 2010] apresenta uma análise comparativa entre alguns sistemas de bancos de dados em grafos classificados como nativos, atuais, neste artigo o autor descreve processos de construção e manipulação de uma base de dados em grafos, realizando um estudo comparativo entre cinco sistemas de bancos de dados em grafos nativos. Os sistemas utilizados foram : InfiniteGraph, Neo4j, OrientDB, Titan e o Trinity.

Em [Takeshi 2016] o autor oferece 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. O modelo de dados construído para os dois tipos de bancos foi um grafo composto pelos artigos da Wikipedia e as hiperligações que conectam uma página à outra. 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 definidas. No final o autor demonstra como algoritmos em grafos resolvem problemas de forma eficiente no Neo4j.

3. Metodologia

Nesta seção serão abordados todas as etapas propostas para a realização deste projeto, está seção seguirá a ordem dos seguintes subtópicos: (1) Ferramentas Utilizadas no projeto, (2) Banco de Dados em Grafos, (3) Construção da Base de Dados e (4) Experimentos. A Figura 1 detalha todas as etapas para a construção do projeto proposto neste artigo.

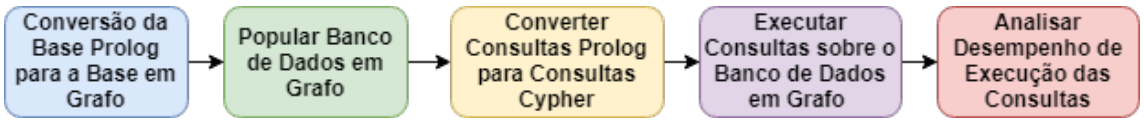

Figura 1. Etapas do Projeto

3.1. Ferramentas Utilizadas

Neste projeto foram utilizadas duas ferramentas para a construção e manipulação do banco de dados em grafos, o **Neo4J** e o **JCypher - Generic Graph Model**.

O **Neo4j** é o principal banco de dasos em grafos do mercado, foi criado em 2010 pela Neo Technology e possui código aberto escrito sobre a JVM (com Scala e Java). O Neo4j é escalável, possui suporte a alta disponibilidade, pode ser utilizado embarcado ou como servidor, fornece uma API REST para operações e possui uma linguagem própria de consulta chamada **Cypher**. O Neo4j é transacional e oferece dois tipos de arquitetura, centralizada e a distribuída com suporte a replicação. Possui o modelo de grafos de propriedade e quanto ao armazenamento físico é baseado em repositórios chave-valor. [Neo4j 2018]

O **JCypher** fornece um modelo para a execução de operações sobre o banco de dados **Neo4J**, permitindo integração com o Java. O modelo consiste em nós, relações e caminhos, juntamente com propriedades, rótulos e tipos. Embora simples, o modelo permite navegar e manipular facilmente gráficos, tomando como base a linguagem **Cypher** integrada a linguagem Java para construção de operações para manipulação do banco de dados Neo4J. [JCypher 2018]

3.2. Banco de Dados em Grafos

Os bancos de dados do orientados a grafos podem ser caracterizados como aqueles nos quais as estruturas de dados são modeladas na forma de grafos, e a manipulação de dados é expressa através de operações orientadas a grafos. Nesse tipo de estrutura, os dados são representados por nós, arestas e propriedades. Os nós representam as entidades, as arestas expressam as relações entre os nós e as propriedades apresentam características das entidades e relacionamentos.

A velocidade de se comparar um banco de dados relacional com um banco de dados de grafos vem na hora de se comparar uma busca cheia de “JOINS” em um banco SQL e a simplicidade de se buscar relacionamentos em grafos. A Figura 02 detalha um banco de dados orientado a grafos.

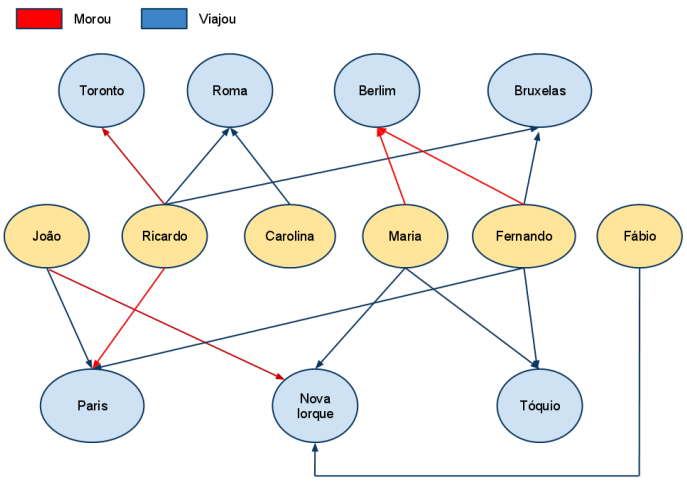

Figura 2. Banco de dados em Grafo

3.3. Construção da Base de Dados

Nesta sessão serão abordados as caracteristicas, os tipos e a estrutura dos dados que foram utilizados neste projeto e também a forma como esses dados foram carregados e estruturados no banco de dados em grafo Neo4J.

3.3.1. Data Set

Os dados utilizados e manipulados neste projeto são **Sentenças** ou frases estruturadas em arquivos Prolog. As sentenças são formadas pricipalmente por **Tokens**, que são as palavras que compõe cada sentença, cada Token possui propriedades importantes como: os caracteres da palavra, nome, tipo, tamanho entre outros. Nas sentenças também são definidos os **Chunks** que são formados por um conjunto de Tokens, os Chunks tabém possuem atributos importantes como: tipo , nome e posição na frase.

Nas sentenças também são definidas as relações entre Tokens e Chunks. As relações definidas nas frases estão divididas em relações entre Chunks, entre Chunks e Tokens e entre Tokens. As relações entre Chunks nas sentenças estão definidos em relação à sucessão ou seja as relações indicam quais sucessores os Chunks tem numa frase. As relações entre Chunks e Tokens indicam a composição de Tokens que formam cada Chunk numa sentença. Por último as relações entre Tokens indicam a sucessão entre eles ou seja qual Token é sucessor de outro, como também detalha relacionamentos de dependências gramáticas entre os Tokens de uma frase. A Figura 3 detalha todas as relações possíveis, escritos em prolog, entre os componentes de uma frase (Tokens e Chunks).

Componentes	Relações em Prolog
Chunk e Chunk	ck_hasSucc(ck_1, ck_2)
Chunk e Token	ck_has_tokens(ck_1, t_1) ck_hasHead(ck_1, t_1)
Token e Token	t_next(t_1, t_2) t_hasDep(poss, t_1, t_2)

Figura 3. Relações entre componentes das frases

3.3.2. Base de Dados em Grafos

Para construir e popular a base de dados em grafos no Neo4J foi necessário converter as sentenças descritas em prolog para uma estrutura em grafos. Para conversão da base em prolog para a base em grafos foi utilizado o framework JCypher citado anteriormente.

A construção da base de dados em grafo seguiu o seguinte modelo padrão, os Tokens e os Chunks definidos nas sentenças em prolog foram convertidos em Nós do grafo e todas as propriedades tanto dos Tokens quanto dos Chunks foram convertidos em

propriedades dos Nós do grafo. A Figura 4, 5, 6 e 7 detalha a conversão dos Tokens, Chunks e suas propriedades em Nós no banco de dados em grafos no Neo4J.

```
token(t_1).
t_stem(t_1, 'as').
t_pos(t_1, in).
t_type(t_1, word).
t_ck_ot(t_1, pp).
t_ck_tag_ot(t_1, b-pp).
t_orth(t_1, lowercase).
t_isHeadPP(t_1).
t_length(t_1, 2).
t_bigPosAft(t_1, nnp-pos).
t_trigPosAft(t_1, nnp-pos-nnp).
```

Figura 4. Tokens Prolog

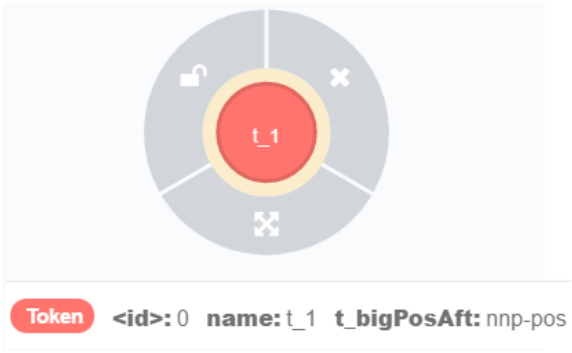

Figura 5. Token - Nó Grafo

```
chunk(ck_1).
%ck_hasTokenList(ck_1, [t_1] ).
ck_hasType(ck_1, pp).
ck_has_tokens(ck_1, t_1).
ck_hasHead(ck_1, t_1).
ck_posRelPred(ck_1, -4).
```

Figura 6. Chunks Prolog

Figura 7. Chunks - Nó Grafo

Após o processo de construção dos Nós do grafo, foram inseridas as arestas do grafo que têm a função de interligar os Nós inseridos. Assim todas as relações entre Tokens e Chunks que compõem as sentenças em prolog, mostradas anteriormente, foram convertidas em arestas de relacionamento entre os Nós do Grafo construído. A Figura 8 e 9 mostram a conversão das relações do tipo **hasSucc** entre os Chunks para o Grafo.

```
ck_hasSucc(ck_1, ck_2).
ck_hasSucc(ck_2, ck_3).
ck_hasSucc(ck_3, ck_4).
ck_hasSucc(ck_4, ck_5).
ck_hasSucc(ck_5, ck_6).
```

Figura 8. Relação hasSucc Prolog

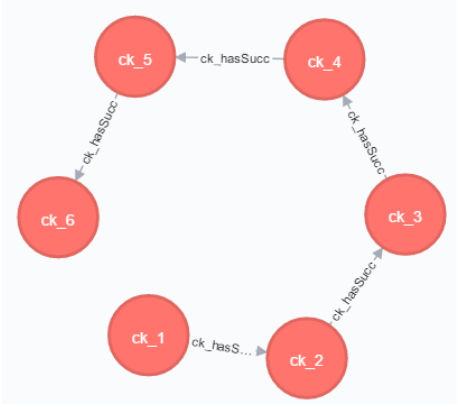

Figura 9. Relação hasSucc Aresta Grafo

As Figuras 10, 11 e 12 detalham a conversão, da base em prolog para a base em grafos Neo4J, dos relacionamentos de sucessão entre os Tokens (**next**) e a conversão dos relacionamentos de dependência existententes entre os Tokens das senteças **(hasDep)**.

```
t_next(t_1, t_2).
t_next(t_2, t_3).
t_next(t_3, t_4).
t_next(t_4, t_5).
t_next(t_5, t_7).
t_next(t_7, t_8).
t_next(t_8, t_9).
t_next(t_9, t_10).
t_next(t_10, t_11).
t_next(t_11, t_12).
t_next(t_12, t_13).
t_next(t_13, t_14).
```

Figura 10. Next Tokens Prolog

```
t_hasDep(poss, t_5, t_2).
t_hasDep(nn, t_5, t_4).
t_hasDep(num, t_9, t_7).
t_hasDep(amod, t_9, t_8).
t_hasDep(prep_as, t_11, t_5).
t_hasDep(nsubj, t_11, t_9).
t_hasDep(aux, t_11, t_10).
t_hasDep(dobj, t_11, t_14).
t_hasDep(det, t_14, t_12).
t_hasDep(amod, t_14, t_13).
```

Figura 11. Dependências entre Tokens Prolog

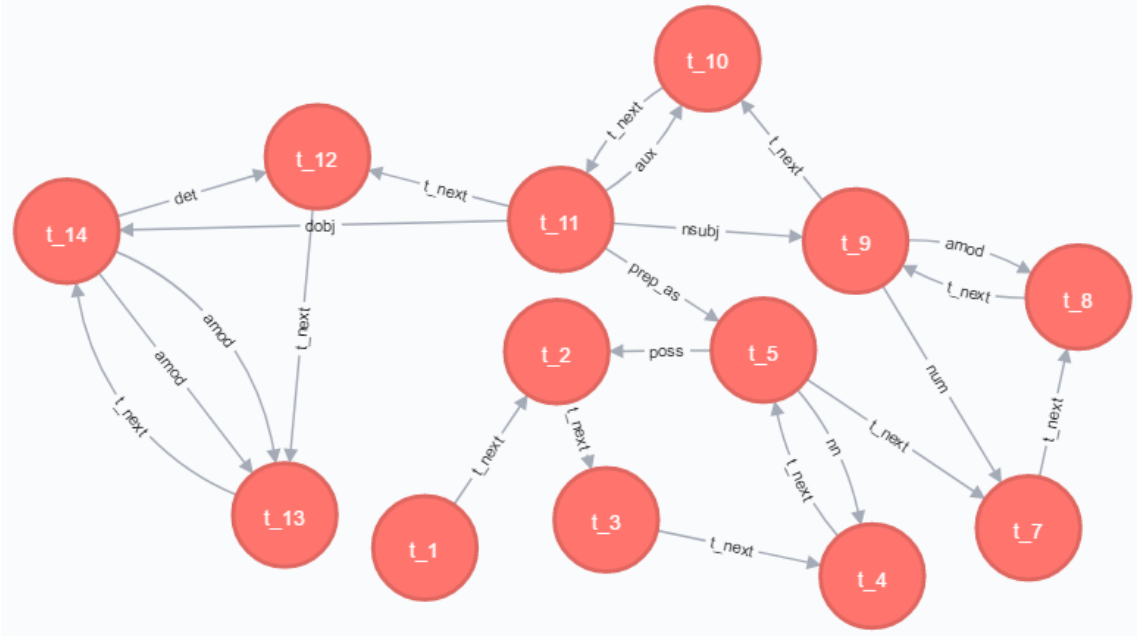

Figura 12. Relações entre Tokens de uma Sentença

A Figura 13 mostra o resultado final da construção de uma sentença no banco de dados em grafos Neo4J, detalhando os relacionamentos entre Chunks e Tokens **(hasTokens e hasHead)**, como também todos os relacionamentos mostrados nos tópicos anteriores deste artigo.

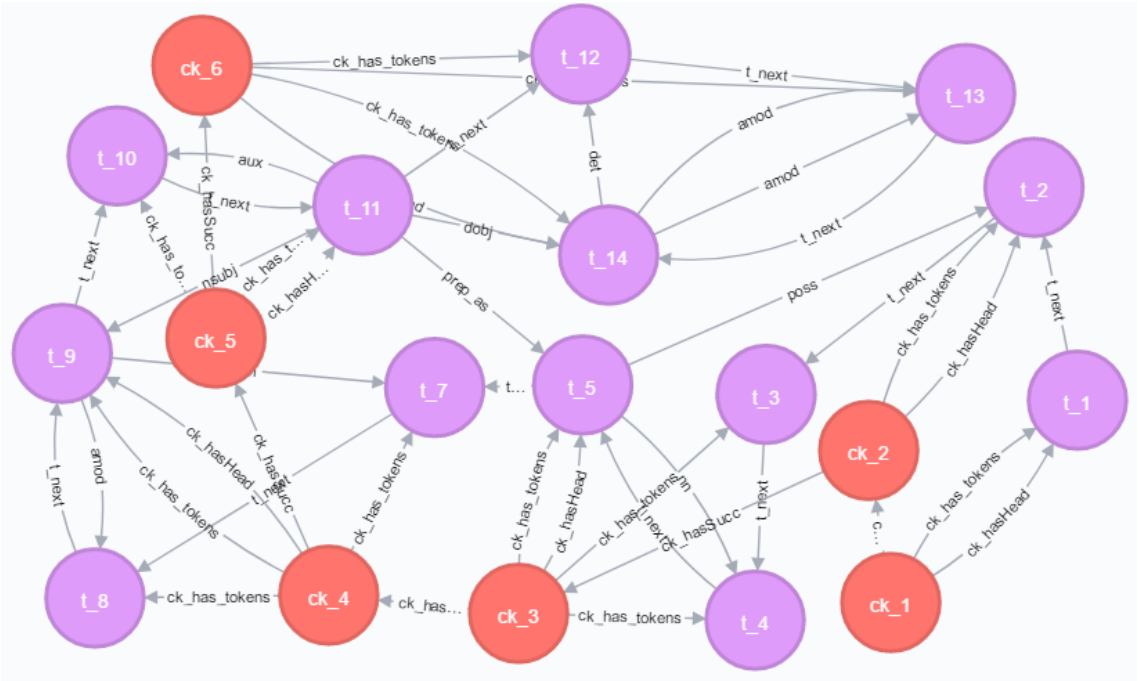

Figura 13. Resultado de uma Sentença em Grafos

3.4. Experimentos

Nesta seção serão detalhados todos os procedimentos e métodos para a execução dos testes de desempenho sobre o banco de dados em grafos criado no Neo4J. A base de dados utilizada neste projeto, completa, é composta por 1079 sentenças; para a execução dos testes de desempenho sobre a base, foram utilizadas consutas prontas, relacionadas as sentenças criadas, convertidas de prolog para a linguagem Cypher. A Figura 14 detalha bem um exemplo de uma conversão das consultas em prolog, utilizadas no projeto, para a linguagem Cypher.

```
#####    Prolog   #######
employ_staff(A,B),t_next(B,C),t_orth(C,lowercase),t_ck_ot(C,np),t_orth(A,mixedcase),t_next(C,A)
#####    Neo4J    #######
MATCH (b:Token)-[:t_next]->(c:Token)-[:t_next]->(a:Token)
WHERE c.t_orth="lowercase" AND c.t_ck_ot= "np" AND a.t_orth="mixedcase"
RETURN a,b
```

Figura 14. Conversão da consulta Prolog para Cypher

Foram utilizados alguns métodos padrões para a realização da análise de desempenho sobre as estruturas, de banco de dados, em grafos geradas no Neo4J. A primeira fase foi converter 10 consultas em prolog, relacionadas as sentenças, para a linguagem Cypher; Após a conversão das consultas, o banco de dados em grafo foi carregado com 10 porcento das sentenças disponíveis no data-set; Após o carregamento das sentenças o banco em grafo foi submetido à carga de execução das 10 consultas utilizadas neste procedimento e após isso foram colhidos os tempos de execução de cada consulta realizada.

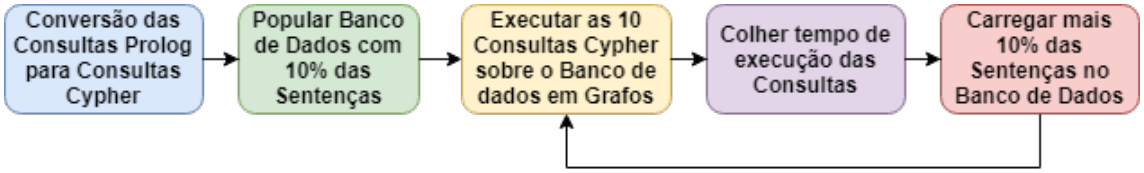

Figura 15. Etapas para análise de desempenho do Neo4J

A Figura 15 detalha todas as etapas e métodos utilizados para a análise de desempenho nas diferentes estruturas de banco de dados em grafo criadas. Como mostrado na Figura 15, todos os procedimento propostos utilizados foram executados repetidamente com o aumento gradativo de 10 em 10 porcento da quantidade de sentenças carregadas no banco de dados em cada ciclo de execução dos testes de análise de desempenho, até que seja carregado 100 porcento do data-set no banco de dados Neo4J. Vale salientar também que para cada execução dos 10 testes de desempenho sobre a base de dados, foram utilizadas as mesmas 10 consultas base, citadas anteriormente neste tópico.

4. Resultados

Nesta seção serão detalhados os resultados obtidos após a realização da análise de desempenho do banco de dados em grafo Neo4J. Após a execução das consultas sobre as diferentes quantidades de sentenças carregadas no banco de dados, foram obtidos alguns resultados importantes em relação ao tempo de execução das consultas como: tempo total da execução das consultas sobre o banco, média do tempo de execução das consultas, desvio padrão do tempo total de consulta e o tempo de cada consulta. A Figura 16 detalha os resultados obtidos após a execução das consultas sobre as diferentes porcentagens de sentenças carregadas no banco de dados; foi obtido o tempo de execução total das consultas e a média e o desvio padão desses tempos de execução.

Resultados	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Tempo Total (ms)	632	3928	8740	13422	21271	30316	34991	53497	64821	78353
Média	63,2	392,8	874	1342,2	2127,1	3031,6	3499,1	5349,7	6482,1	7835,3
Desvio Padrão	124,7	777,5	1731,11	2780,6	4389,12	6394,15	7311,15	11847,1	14648,6	18093,6

Figura 16. Resultados da Análise de Desempenho

As Figuras 17, 18 e 19 detalha em gráficos os resultados dos tempos de execução das consultas, em milissegundos, 1, 5 e 10 nas diferentes porcentagens de sentenças carregadas no banco de dados em grafo. Podemos notar nos gráficos 18 e 19 que o tempo de execução das consutas 5 e 10, que são as mais custosas, crescem proporcionalmente em relação ao aumento da quantidade de setenças carregadas no banco de dados, já no gráfico 17 é mostrado que em alguns casos, como na consulta 1, o tempo de execução varia em relação ao aumento da quantidade de nós e relacionamentos entre os nós (sentenças) no banco de dados em grafo.

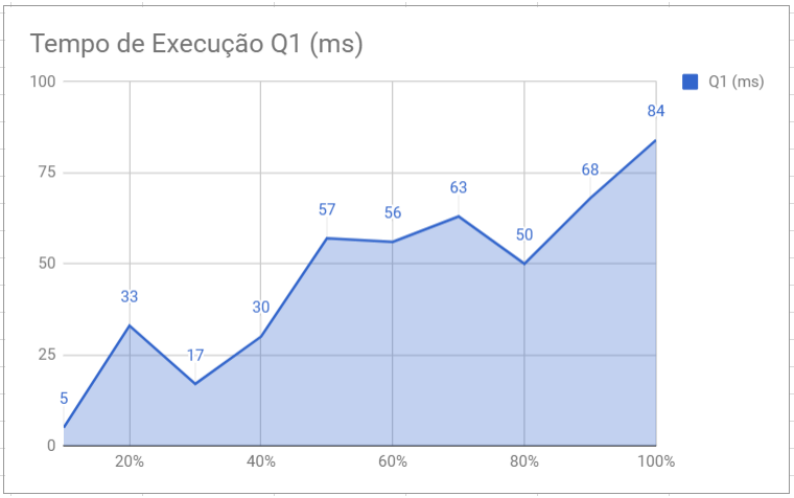

Figura 17. Tempos de Execução da Consulta 1

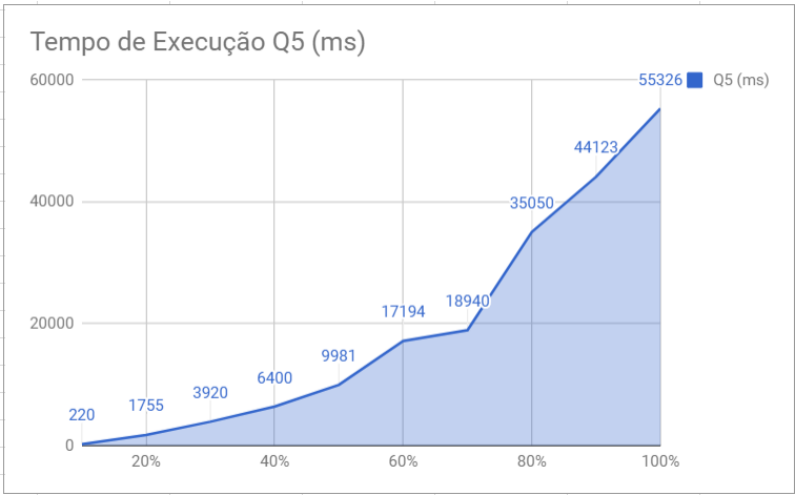

Figura 18. Tempos de Execução da Consulta 5

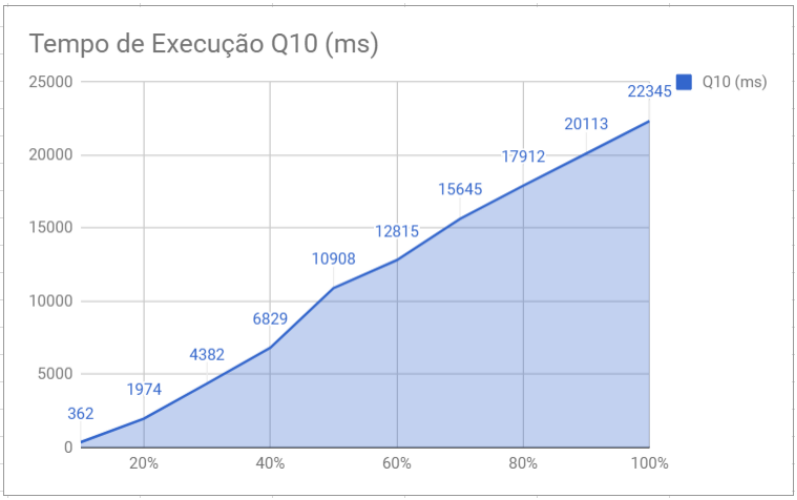

Figura 19. Tempos de Execução da Consulta 10

A gráfico mostrado na Figura 20 detalha os resultados gerais obtidos dos tempos totais de executação das consultas, em milissegundos, sobre o banco de dados Neo4J. No gráfico da figura 20 é possível notar que o tempo total das consultas crescem proporcionalmente ao aumento gradativo da porcentagem de sentenças carregadas no banco de dados ou seja quanto maior for a quantidade de sentenças e maior for a quantidade de nós e relacionamentos entre os nós destas sentenças, maior será o tempo de execução total das consultas sobre o banco em grafos.

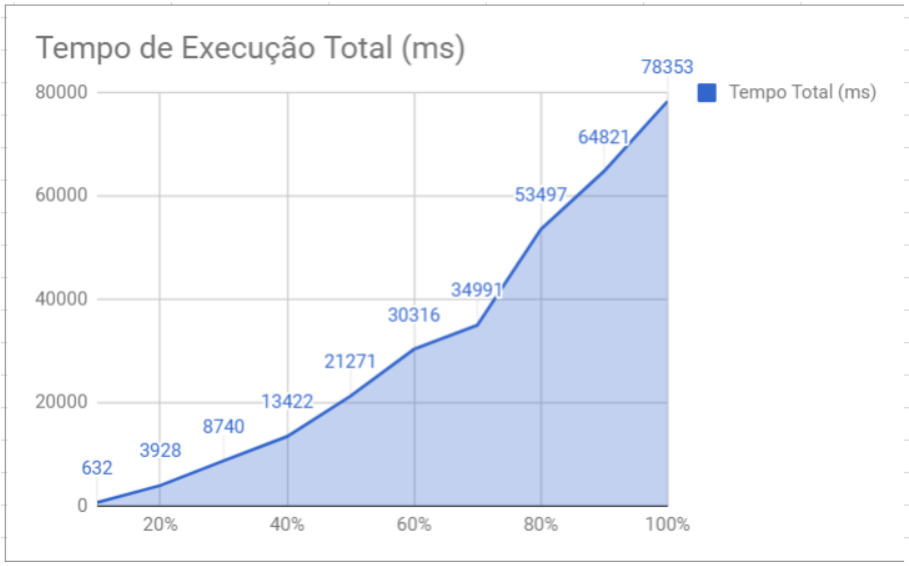

Figura 20. Tempos Totais de Execução das Consultas

É importante detalharmos também o tempo de carregamento de cada porcentagem de sentenças no banco de dados em grafo, assim podemos ter ciência do tempo gasto para popular o banco de dados em relação a este estudo. O gráfico da Figura 21 mostra o tempo para carregamento do banco de dados, em minutos, utilizando o JCypher, para cada porcentagem de sentenças carregadas no banco Neo4J.

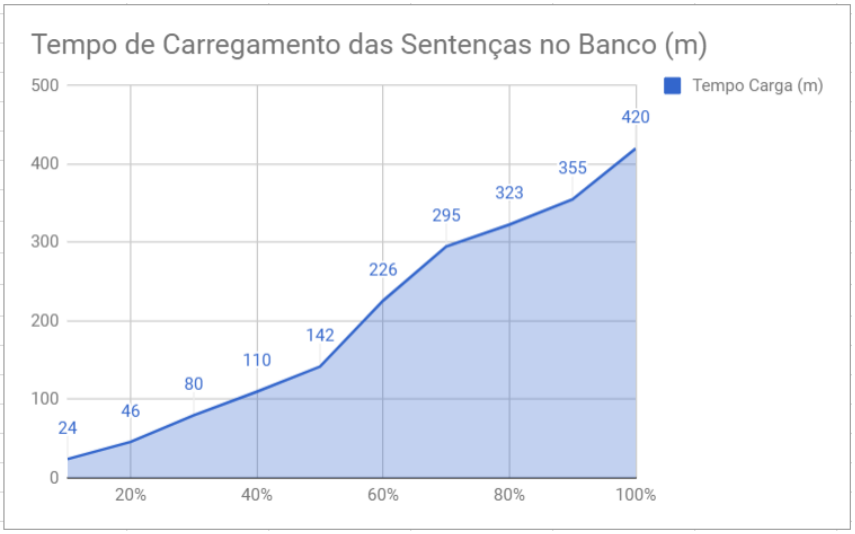

Figura 21. Tempos de Carregamento das Sentenças no Banco

5. Conclusão

Neste artigo foi demonstrado métodos para análise de desempenho e comportamento de bancos de dados orientados a grafos quando submetidos a cargas de consultas em diferentes conjuntos de dados, como base para os estudos e resultados deste trabalho foi utilizado o tempo de execução de consultas sobre o banco de dados orientado a grafos Neo4J .

Com base nos resultados da análise podemos ter informações importantes de desempenho para a tomada de decisão na escolha de um banco de dados em grafo em relação à uma determinada aplicação ou à outros tipos de uso do banco de dados, tornando-se um modelo útil para a comunidade de pesquisa ou para o mercado tecnológico.

Como trabalhos futuros pretende-se utilizar os mesmos métodos de análise de desempenho utilizados no Neo4j em outros bancos de dados em grafos e realizar a comparação de performance entre estes bancos de dados em grafos, assim é possível ter resultados mais plausíveis e importantes para tomadas de decisões na escolha de bancos de dados em grafos.

Referências

Alvarez, G. and Ceci, F. (2016). Análise comparativa dos bancos orientados a grafos de primeira e segunda geração – uma aplicação na análise social. *III Encontro de Inovação em SI, Florianópolis, SC, 17 a 20 de Maio de 2016*, pages 8–11.

Ferreira, E. and Junior, S. (2014). Analise de desempenho de bancos de dados. *(UNIPAC)*, pages 1–10.

JCypher (2018). Documentação oficial generic graph model. *https://github.com/Wolfgang-Schuetzelhofer/jcypher/wiki*.

Neo4j (2018). Documentação oficial neo4j. *https://neo4j.com/graphacademy/*.

Penteado, R. and Schroeder, R. (2010). Um estudo sobre bancos de dados em grafos nativos.

Takeshi, G. (2016). Uma comparacão conceitual e de performance com as ferramentas neo4j e postgresql. *Monografia - Universidade de São Paulo*, pages 1–19.