Análise de Desempenho de Bancos de Dados em Grafos

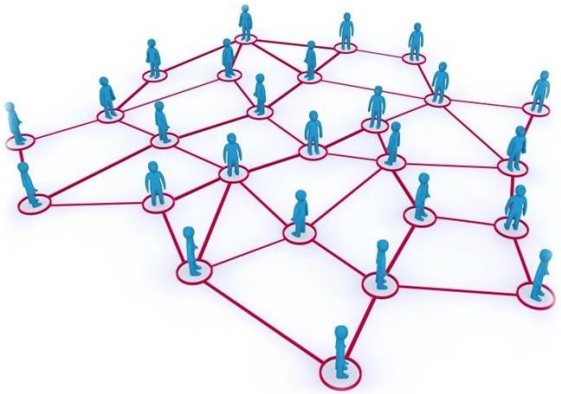
Aluno: Guilherme Melo
Professor: Rinaldo Lima

Introdução

- 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.

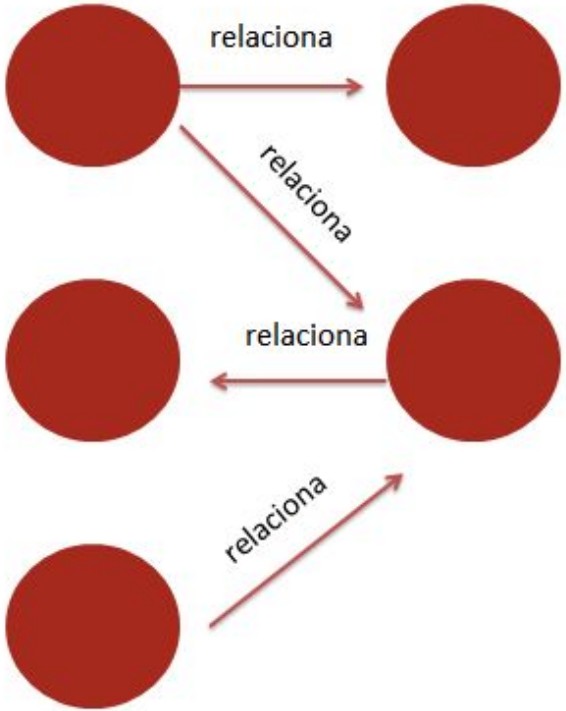
Introdução

- Banco de Dados em Grafos
    - 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.



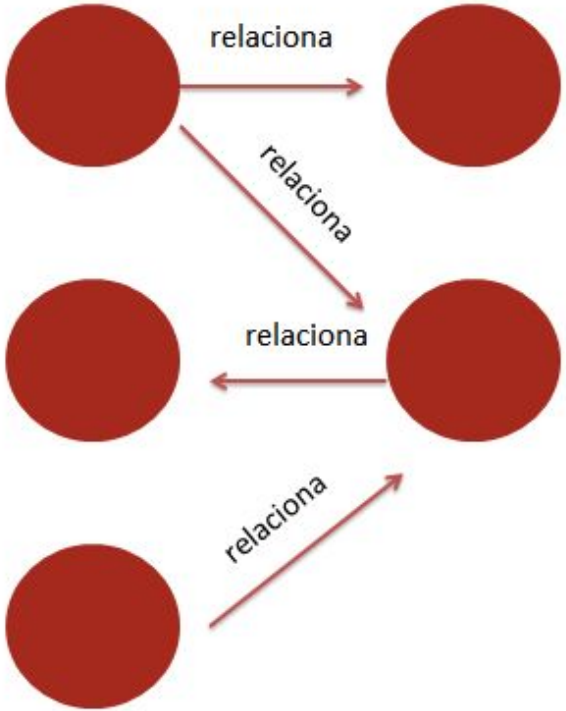
Introdução

- Porque usar BD em Grafos ?
  - Em um banco de dados de grafos, relacionamentos são mais naturais.
  - Temos as entidades chamadas de vértices (ou node) que são ligadas entre elas pelas arestas (ou relationships) cada um podendo guardar dados entre os relacionamentos e cada relacionamento pode ter uma direção.



Introdução

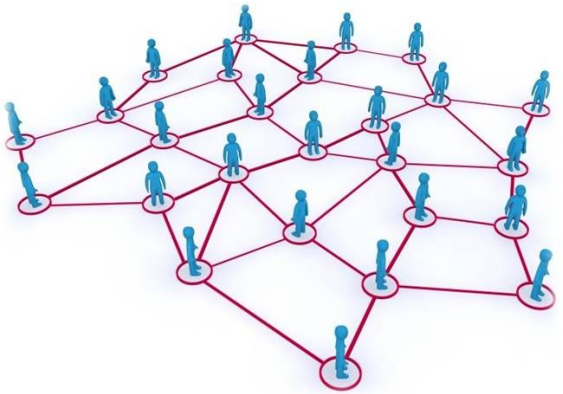
- Porque usar BD em Grafos ?
    - A velocidade de se comparar um banco de dados relacional com um banco de dados de grafos vem na hora de se comparar uma busca cheia de “JOINS” em um banco SQL e a simplicidade de se buscar relacionamentos em grafos.



Problemática

- 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.

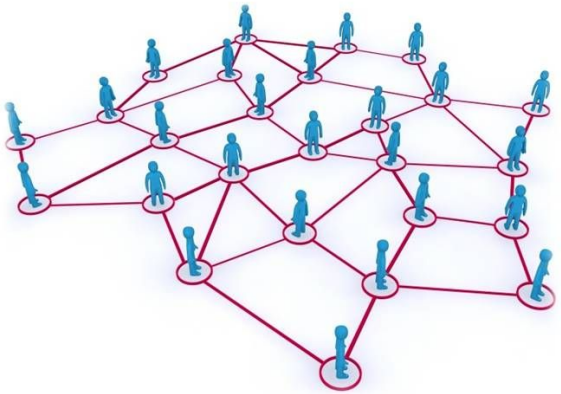
- Assim é de grande importância a análise de desempenho prévia antes da escolha e utilização de algum banco de dados.



Proposta

- O projeto proposto tem por objetivo demonstrar métodos para análise de desempenho de bancos de dados orientados a grafos.

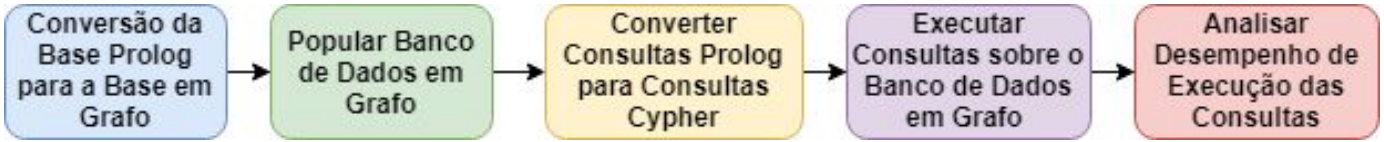
- Desta forma o presente trabalho apresenta a análise da performance e o comportamento do banco de dados em grafos Neo4j, quando submetido a cargas de consultas em diferentes conjuntos de dados



Ferramentas Utilizadas

- Neo4J
- JCypher

Metodologia



Etapas do Projeto

Data Set

- Os dados utilizados e manipulados neste projeto são Sentenças ou frases estruturadas em arquivos Prolog.
- São formados por:
  - Tokens - Palavras
  - Chunks - Conjunto de Tokens
  - Relações - Entre Tokens e Chunks
- Tokens e Chunks possuem Atributos

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)

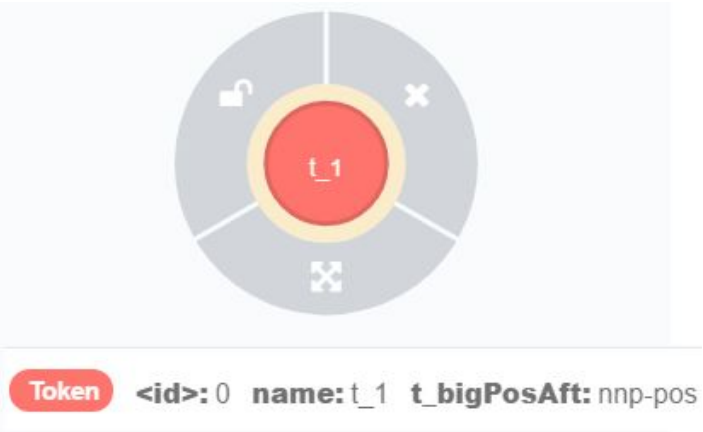
Relações entre Componentes de uma Frase

Base de Dados em Grafo

- Necessária a conversão das sentenças em prolog para grafos.
  - Conversão Tokens

```
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).
```

Tokens em Prolog



Tokens - Nó Grafo

Base de Dados em Grafo

- Necessária a conversão das sentenças em prolog para grafos.
  - Conversão Chunks

```
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).
```

Chunks em Prolog

Chunk <id>: 40 ck_hasType: pp name: ck_1

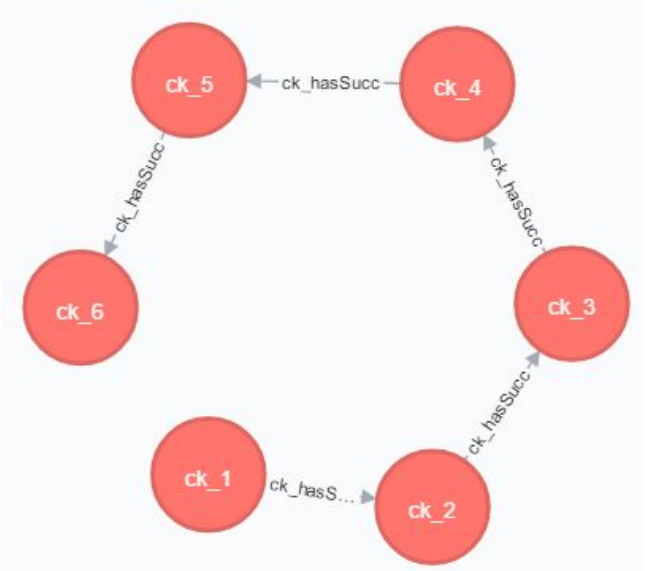
Chunks - Nó Grafo

Base de Dados em Grafo

- Necessária a conversão das sentenças em prolog para grafos.
  - Conversão Relações Chunks
    - **hasSucc** .

```
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).
```

Relação hasSucc Prolog



Relação hasSucc Aresta Grafo

Base de Dados em Grafo

- Necessária a conversão das sentenças em prolog para grafos.
  - Conversão Relações Tokens
    - **next**.
    - **dependências**

```
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).
```

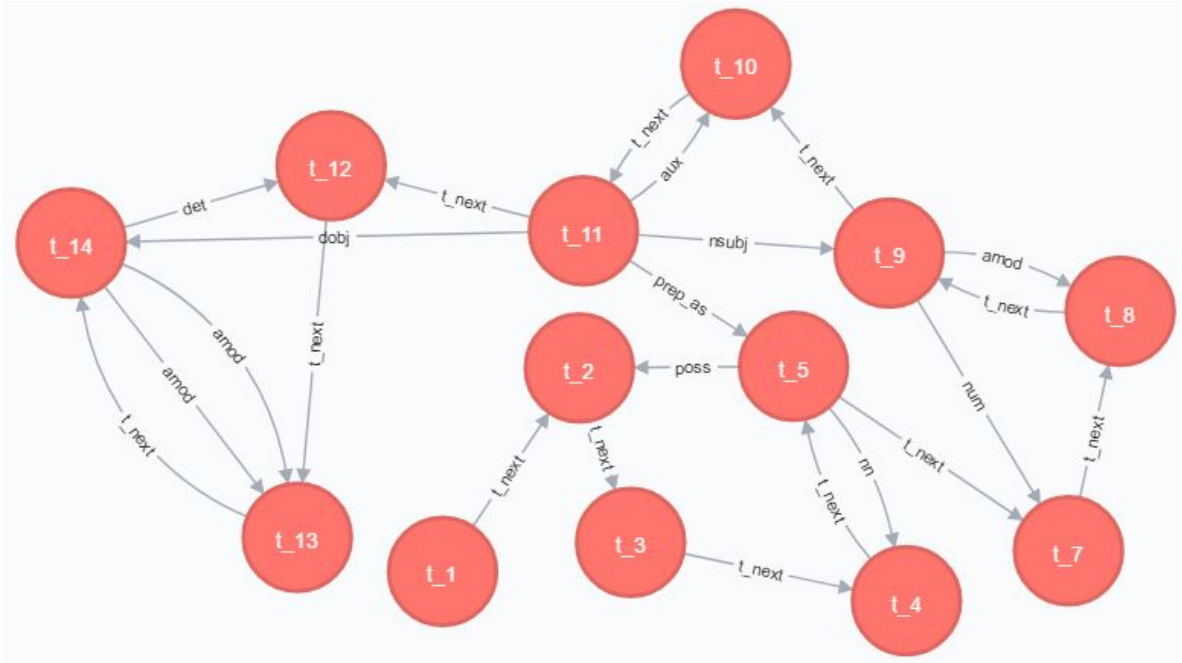
Next Tokens Prolog

```
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).
```

Dependências entre Tokens Prolog

Base de Dados em Grafo

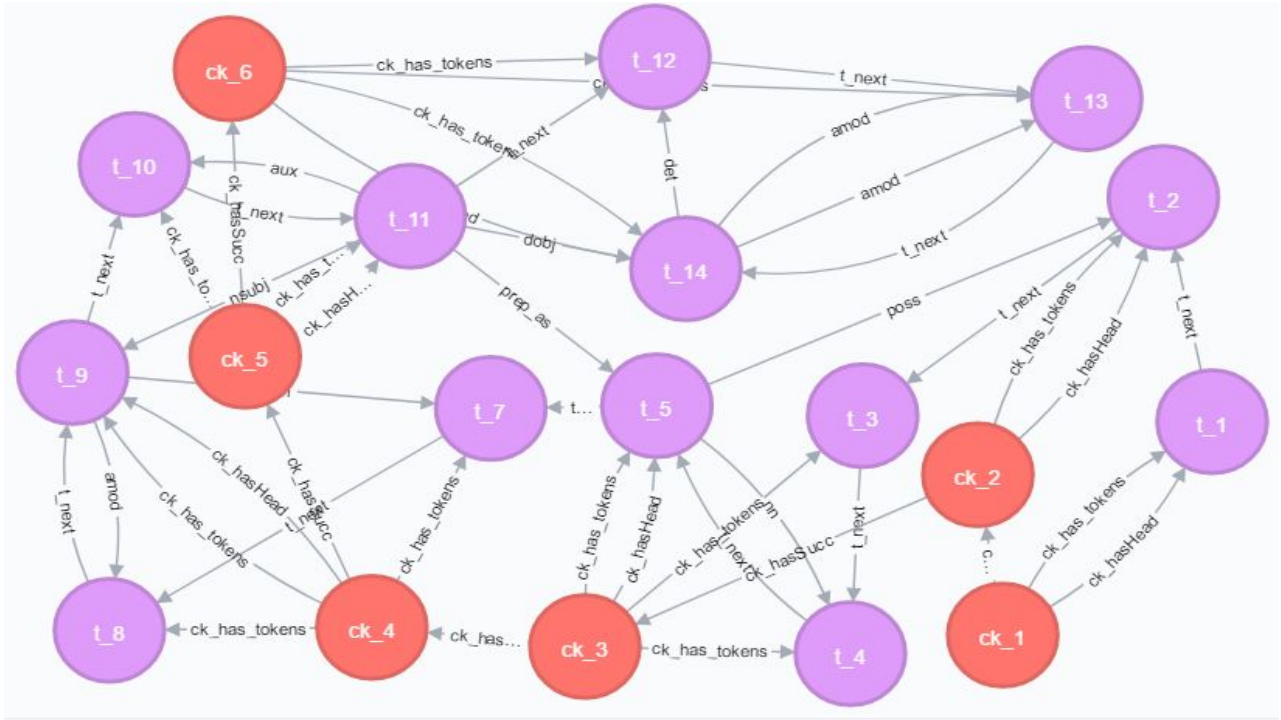
- Necessária a conversão das sentenças em prolog para grafos.
  - Conversão Relações Tokens
    - **next**.
    - **dependências**



Relações entre Tokens de uma Sentença

Base de Dados em Grafo

- Resultado de uma Sentença em grafos no Neo4J.



Resultado de uma Sentença em Grafos

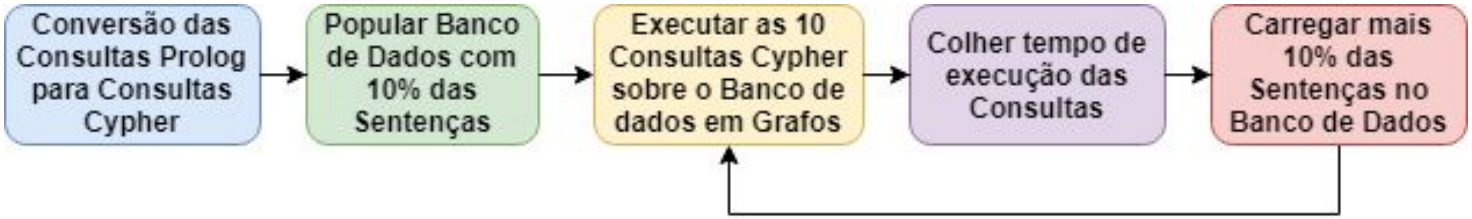
Experimentos

- Base de Dados - 1079 Sentenças.
- Conversão de 10 Consultas de Prolog para Cypher.
- Carregar o Banco em grafo com as sentenças de 10% em 10%.
- Execução das 10 consultas a cada 10% das sentenças carregadas no Banco.
- Colher o tempo de Execução de Cada Consulta.

```
#####    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
```

Conversão das Consultas

Experimentos



Etapas para análise de desempenho do Neo4J

Resultados

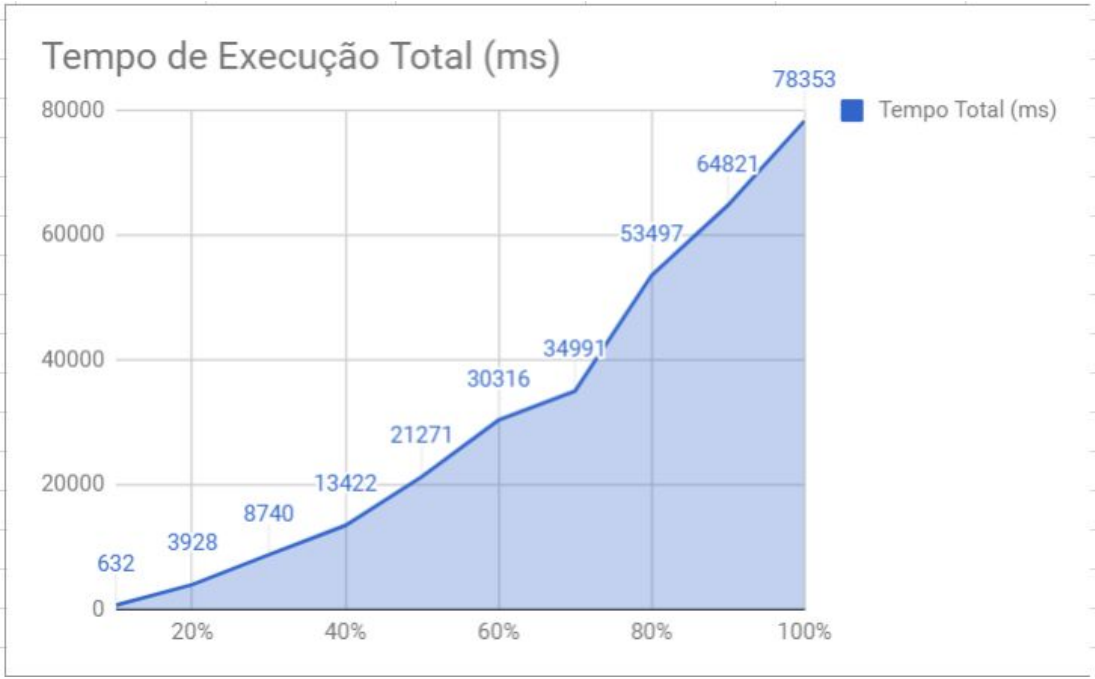
- Resultados Gerais

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

Resultados da Análise de Desempenho

Resultados

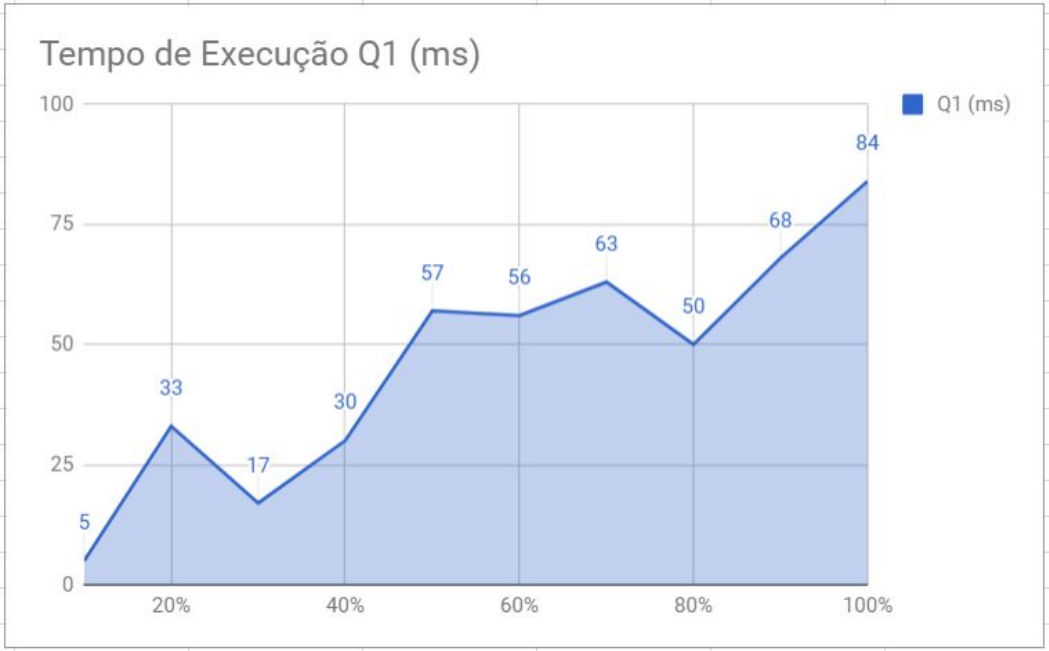
- Tempo de Execução Total



Resultados da Análise de Desempenho

Resultados

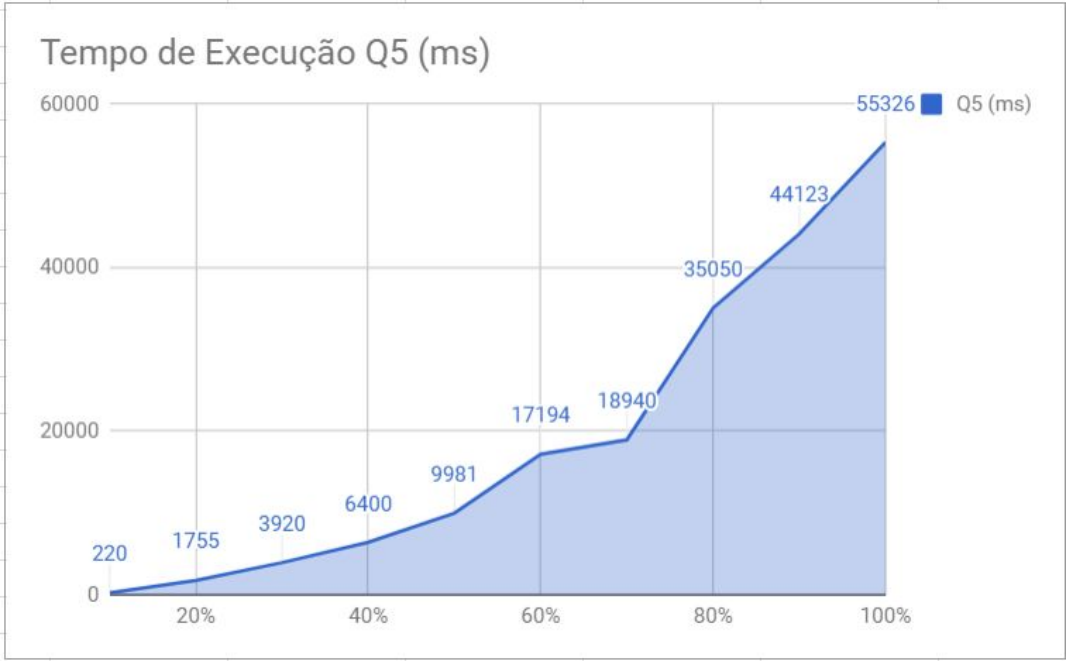
- Tempo de Execução Consulta 1



Resultados da Execução da Consulta 1

Resultados

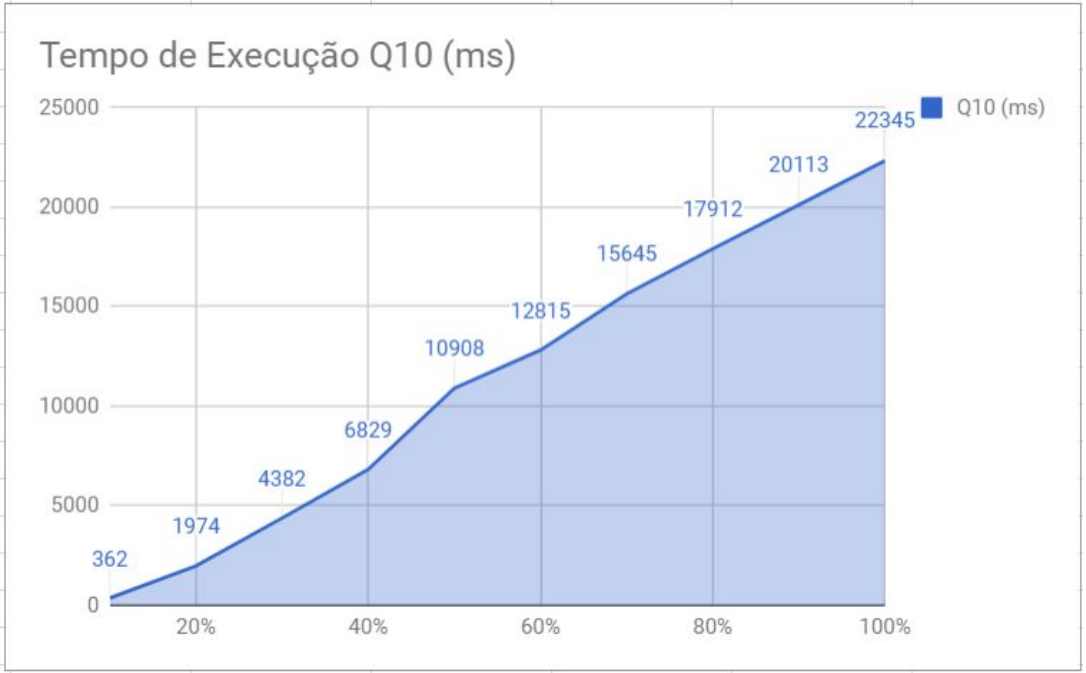
- Tempo de Execução Consulta 5



Resultados da Execução da Consulta 5

Resultados

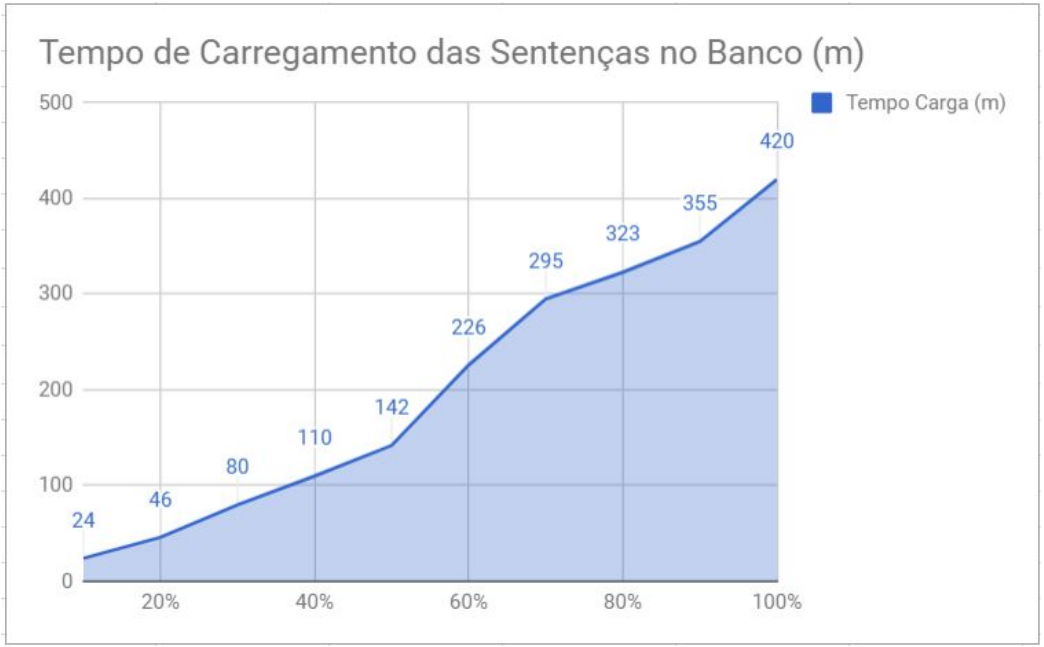
- Tempo de Execução Consulta 10



Resultados da Execução da Consulta 10

Resultados

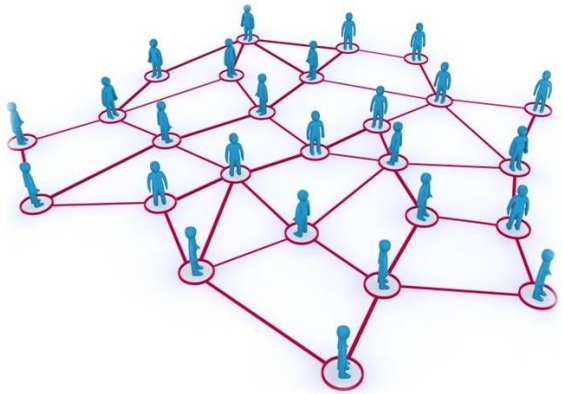
- Tempo de Carregamento das Sentenças



Resultados do Tempo de carregamento das Sentenças

Conclusão

- 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.



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.