Unioeste - Universidade Estadual do Oeste do Paraná
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
Colegiado de Ciência da Computação
Curso de Bacharelado em Ciência da Computação

SGBD orientado a grafos: um estudo comparativo entre Neo4j, OrientDB, ArangoDB e AgensGraph

Adriano Montezano Grams

CASCAVEL
2020

Adriano Montezano Grams

SGBD orientado a grafos: um estudo comparativo entre Neo4j, OrientDB, ArangoDB e AgensGraph

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação, do Centro de Ciências Exatas e Tecnológicas da Universidade Estadual do Oeste do Paraná - Campus de Cascavel

Orientador: Prof. Dr. Guilherme Galante

CASCAVEL
2020

Adriano Montezano Grams

SGBD orientado a grafos: um estudo comparativo entre Neo4j, OrientDB, ArangoDB e AgensGraph

Monografia apresentada como requisito parcial para obtenção do Título de Bacharel em Ciência da Computação, pela Universidade Estadual do Oeste do Paraná, Campus de Cascavel, aprovada pela Comissão formada pelos professores:

Prof. Dr. Guilherme Galante (Orientador)
Colegiado de Ciência da Computação,
UNIOESTE

Prof. Dr. Ivonei Freitas da Silva
Colegiado de Ciência da Computação,
UNIOESTE

Prof. Dr. Clodis Boscarioli
Colegiado de Ciência da Computação,
UNIOESTE

Cascavel, 26 de julho de 2021

AGRADECIMENTOS

Gostaria de agradecer a todos aqueles que estiveram comigo durante a jornada pela universidade, meus colegas de classe, os professores, minha família e amigos, que me ajudaram a passar por essa importante etapa. Também gostaria de agradecer a Unioeste pelas oportunidades disponibilizadas para o enriquecimento da minha formação. Por último gostaria de agradecer a meu mentor Guilherme Galante, pela ajuda no desenvolvimento do TCC e dos projetos aos quais participei com sua orientação. Serei eternamente grato a todos que me ajudaram a chegar aonde cheguei e me tornar a pessoa que sou hoje.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

ACID	Atomicidade, Consistência, Isolamento, Durabilidade
BASE	*Basically Available, Soft state, Eventual consistency*
CRUD	*Create*, *Read*, *Update and Delete*
HDD	*Hard Drive Disk*
JSON	*JavaScript Object Notation*
LOPS	*Loading Operations per Second*
NoSQL	*Not Only SQL*
OLTP	*Online Transaction Processing*
RAM	*Random-access memory*
RDF	*Resource Description Framework*
RPM	Rotações por minuto
SDD	*Solid-State Drive Disk*
SGBD	Sistema de Gerenciamento de Banco de Dados
SGBDR	Sistema de Gerenciamento de Banco de Dados Relacional
SQL	*Structured Query Language*
TEPS	*Traversed Edges per Second*
WAL	*Write-Ahead Logging*
XML	*Extensible Markup Language*

Sumário

Resumo

Nos últimos anos, banco de dados orientado a grafos tem ganho uma grande popularidade por parte da comunidade, devido ao seu enfoque em tratar grandes volumes de dados altamente conectados. Com esse aumento de popularidade, várias soluções de Sistemas de Gerenciamento de Banco de Dados (SGBD) que implementam esse modelo surgiram. Esses sistemas se diferenciam pelos recursos disponibilizados, performance para diferentes tipos de operações, suporte da comunidade e dos desenvolvedores, entre outros fatores que fazem da tarefa de escolher o melhor SGBD para um projeto, situação ou trabalho uma difícil missão. Com isso, este trabalho tem com objetivo auxiliar desenvolvedores e projetistas de banco de dados a escolherem a melhor opção de SGBD orientado a grafos para seus sistemas ou objetivos. Para isso, foram feitas uma análise documental e uma comparação de desempenho entre os SGBD Neo4j, ArangoDB, OrietnDB e AgensGraph. Nossos resultados mostraram que as melhores opções dos SGBD selecionados foram o Neo4j e ArangoDB, tanto em desempenho quanto aspectos técnicos, extraídos da análise documental, com uma vantagem maior ao Neo4j em razão de sua maior escalabilidade. O OrientDB se mostrou inferior em ambos os aspectos comparados, especialmente em questões de desempenho, em relação as duas soluções anteriores. Por último, o AgensGraph se mostra como um projeto descontinuado, sem perspectivas para uma retomada no desenvolvimento.

Palavras-chave: banco de dados orientado a grafos, SGBD, comparação, análise documental, análise de desempenho

Capítulo 1

Introdução

Na atualidade, a área de banco de dados tem sido uma das mais importantes na computação, isso pois, as soluções de banco de dados atuais permitem o armazenamento e manipulação de dados de forma rápida, segura e confiável. Esses são normalmente classificados em grupos, dependendo do modelo que suportam, sendo o modelo relacional o mais conhecido e utilizado. Porém, devido às limitações desse modelo em certos cenários, como tratamento de grandes volumes de dados, fragmentação da base de dados, velocidade de reposta para operações de leitura, entre outros, os bancos de dados não relacionais têm ganhado espaço no cenário. Os bancos de dados não relacionais são divididos em outros modelos mais específicos, como por exemplo, banco de dados orientado a documentos, orientado a objetos, chave e valor, entre outros (SILBERSCHATZ; KORTH; SUDARSHAN, 2020).

Dentre esses modelos, os bancos de dados orientados a grafos tem apresentado uma crescente popularidade, como apresentado na Figura 1.1. Esses bancos de dados, caracterizam-se por representar os dados na forma de grafos, nos quais, os nós representam os objetos e as arestas as relações entre eles (ANGLES; GUTIERREZ, 2008). Seu principal objetivo é facilitar o desenvolvimento de banco de dados altamente conectados e com grandes volumes de informações, o que nos dias de hoje, vem de encontro com as necessidades das aplicações atuais, como, redes sociais, sistemas voltadas para áreas química e da biologia, onde o volume de dados é grande, e esses dados, são altamente conectados um com os outros (MILLER, 2013).

Atualmente, existem diversos Sistemas de Gerenciamento de Banco de Dados (SGBD) que implementam o modelo orientado a grafos, apresentando diferentes níveis de desempenho em relação ao tempo de execução, consumo de memória RAM, espaço ocupado em disco, recursos disponíveis aos usuários, custo, suporte dos desenvolvedores e da comunidade, entre outras

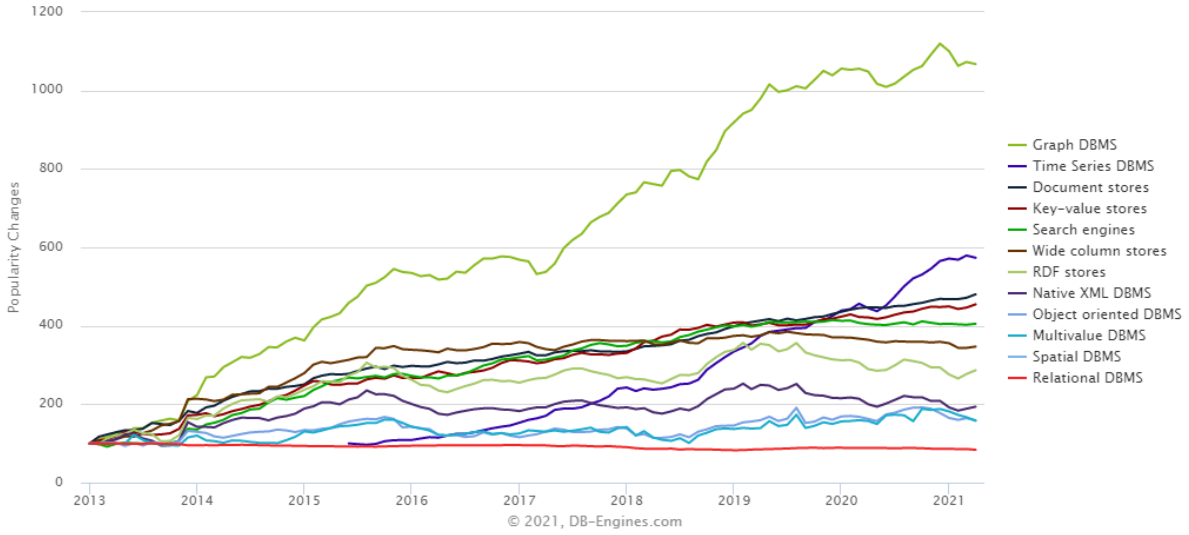

Figura 1.1: Gráfico de popularidade dos modelos nos últimos anos (DB-ENGINE, 2020b)

características. Analisando as diversas opções disponíveis de SGBD orientados a grafos, a tarefa de escolher qual é o mais apropriado para um determinado cenário não é trivial.

Nesse contexto, este trabalho tem como objetivo auxiliar desenvolvedores e projetistas de banco de dados a escolher o SGBD orientado a grafos mais apropriado para sua situação, por meio de uma análise documental e análise de desempenho de 4 SGBD: Neo4j, ArangoDB, OrientDB e AgensGraph. Na análise documental, se avaliou as características de implementação, funcionalidades e recursos desses SGBD, baseados em suas documentações, além de fatores externos como a comunidade de desenvolvedores que utiliza a solução e a maturidade da ferramenta. Já na avaliação de desempenho, diversos experimentos foram realizados, usando diferentes critérios, visando verificar qual é o melhor SGBD para cada critério.

Esse documento segue assim dividido: O Capítulo 2 apresenta a fundamentação teórica para o entendimento dos conceitos e procedimentos usados neste trabalho; O Capítulo 3 tem como objetivo apresentar brevemente os SGBD utilizados na comparação e a metodologia usada nas comparações de desempenho e análise documental; O Capítulo 4 apresenta e discute os resultados obtidos na análise documental dos SGBD e comparações de desempenho; O Capítulo 5 conclui o trabalho, apresentando as considerações finais e propostas para trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Um banco de dados consiste em uma coleção de dados armazenados em algum tipo de dispositivo de hardware como *Hard Drive Disk* (HDD), *Solid-state Drive Disk* (SSD), entre outros. Um sistema de gerenciamento de banco de dados (SGBD) é um sistema que gerência as manipulações nos dados, em um banco de dados, e serve como interface entre o usuário/desenvolvedor e os dados armazenados em hardware. As características dos SGBD podem variar bastante entre eles, mas as funções básicas que todos os SGBD possuem consistem em, criar, recuperar, atualizar e deletar, ou também conhecido como CRUD (*create, read, update* e *delete*).

Algumas outras funções comuns dos SGBD são prover um conjunto de ferramentas para o gerenciamento efetivo do banco de dados, incluindo, importação e exportação de dados, monitoramento do consumo de recursos por parte do SGBD em relação ao sistema, desfragmentação de disco, entre outras análises para que o desenvolvedor possa utilizar o SGBD da melhor forma possível (CONNOLLY, 2014).

Os SGBD podem ser categorizados baseados no modelo de dados os quais eles implementam, sendo esse modelo uma coleção de conceitos que podem ser usados para descrever a estrutura de um banco de dados. Esses modelos podem ser de alto nível, ou conceituais, que apresentam conceitos mais próximos a como os usuários entendem os dados, e de baixo nível ou físico, que se refere a conceitos que descrevem detalhes de como os dados são armazenados em dispositivos físicos. Entre os modelos de alto e baixo nível, existem os representacionais (ou implementações) que apresentam conceitos que podem ser entendidos pelos usuários mas também envolvem conceitos de como os dados são organizados em *hardware*.

Dentro dos modelos de alto nível, temos o modelo entidade relacionamento, que divide os dados em entidades e relacionamentos. As entidades representam objetos ou conceitos do

mundo real, e essas entidades possuem atributos que servem para descrever esses objetos. Os relacionamentos por sua vez se referem a associação entre entidades.

Os modelos representacionais são os mais usados em soluções de SGBD, e alguns desses são o modelo relacional, e outros modelos legado como *network* e hierárquico. Os representacionais tratam os dados como uma estrutura de registros, por isso podem ser também chamados de modelo orientado a registros.

Para os modelos de baixo nível temos o modelo autodescritivo, onde a descrição dos dados são combinadas com os valores dos mesmos. Em comparação com o modelo relacional, a descrição dos dados é separada dos valores através dos esquemas, que restringem as instâncias ao esquema preestabelecido para os dados. No caso dos autodescritivos essa restrição não existe. Nesse modelo temos o XML, JSON e os NoSQL, que usam dessa estrutura para o armazenamento de dados com mais flexibilidade e permitem o armazenamentos de dados com diferentes estruturas e atributos (ELMASRI; NAVATHE, 2015).

Também temos os NewSQL, que são uma classe do modelo relacional, que procura entregar os mesmos benefícios de desempenho em escalabilidade que os NoSQL, mas garantindo as propriedades ACID para as transações (PAVLO; ASLETT, 2016). Neste trabalho será focado nos bancos de dados orientado a grafos, os quais são uma sub-classe dos NoSQL.

2.1 Banco de dados Relacionais

O modelo de banco de dados relacional modela os dados em formato de linhas e colunas dentro de tabelas, com cada linha tendo uma chave única para identificação, e para seu uso, vários dos SGBD relacionais utilizam a linguagem SQL (*Structured Query Language*), para escrever as consultas aos dados (CONNOLLY, 2014).

Dentro de um banco de dados relacional as tabelas representam entidades do mundo real, nas quais as linhas são as instâncias dessas entidades e as colunas são os atributos dessas instâncias. Cada uma dessas instâncias possui uma chave primaria (*Primary Key*), que é utilizada para identificar aquela instância como única. Com isso, essa chave deve ser única para aquela instância dentro daquela tabela. Para representar relacionamentos dentro de um SGBD relacional (SGBDR), são utilizadas chaves estrangeiras (*Foreign Key*) que representam uma outra instância da mesma, ou de outra tabela, sendo que, para relacionamentos mais complexos, pode

ser necessário a criação de uma ou mais tabelas intermediárias (SILBERSCHATZ; KORTH; SUDARSHAN, 2020).

Os SGBDR seguem o conceito de ACID (Atomicidade, Consistência, Isolamento, Durabilidade) (HAERDER; REUTER, 1983) onde:

- **Atomicidade**: Em uma transação, ou a transação será totalmente executada ou ela não será executada, tornando as transações binárias.
- **Consistência**: Ou a transação cria um novo estado válido aos dados, ou gera uma falha e retorna o banco ao seu último estado antes da transação.
- **Isolamento**: Uma transação ainda em andamento deve permanecer isolada de outras operações, não tendo interferência de outras transações.
- **Durabilidade**: Os dados já persistidos no sistema sempre estão disponíveis de forma correta, indiferente de falhas ou outras situações referentes ao sistema onde os dados estão armazenados.

As quatro propriedades do ACID são garantias pelo SGBDR para as manipulações realizadas nos dados por parte dos usuários. Essas garantias, permitem um armazenamento seguro dos dados de forma a criar transparência por parte do SGBD com o usuário, que não necessita considerar essas questões na hora de utilizar o SGBD para manipular seus dados.

2.2 Banco de dados Não Relacional

SGBD NoSQL ou não-relacionais, são SGBD que armazenam e recuperam dados que são modelados de uma forma diferente de um SGBD relacional. A motivação para a existência desses modelos é a resolução de problemas específicos, não resolvidos de forma satisfatória pelos banco de dados relacionais, como, modelos mais apropriados para tipos de dados específicos ou o escalonamento do tamanho do banco, com distribuição para múltiplas máquinas (LEAVITT, 2010). Os bancos de dados NoSQL podem ser divididos em modelos mais específicos, podendo citar como exemplo os bancos de dados orientado a objetos, orientado a grafos, orientado a documentos, entre outros.

Muitos bancos NoSQL não garantem as propriedades ACID, comprometendo especialmente a consistência em favor de outras vantagens como velocidade, disponibilidade, escalabilidade, entre outros. Em vez disso, muitos NoSQL utilizam um conceito de *eventually consistent*, que foca em conseguir maior disponibilidade dos dados em detrimento da consistência (VOGELS, 2009).

Em contraste com o modelo relacional o *eventually consistent* possui as propriedades BASE (*Basically Available*, *Soft state*, *Eventual consistency*), que consistem no seguinte (PRITCHETT, 2008):

- ***Basically Available***: Operações de leitura e escrita estão sempre que possível disponíveis, mas com nenhuma garantia de consistência, ou seja, os dados lidos podem não ser os mais atualizados, e a escrita pode não ser persistida em caso de conflitos.
- ***Soft State***: Sem a garantia da consistência, não é possível saber o estado do banco de dados de forma garantida.
- ***Eventually consistent***: Se esperarmos o suficiente após uma consulta, a base de dados estará consistente, e quando isso acontece, é possível saber o estado do banco de dados.

Outro conceito importante para os bancos de dados não relacionais, é o da escalabilidade horizontal e vertical. A escalabilidade horizontal consiste na distribuição de carga de trabalho para vários servidores, onde o uso dessas várias máquinas aumenta o poder de processamento dos dados, permitindo consultas mais rápidas (ALI; ABDULLAH, 2019). E a escalabilidade vertical corresponde ao aprimoramento da máquina, como mais memória, mais núcleos no processador, maior clock, entre outras características relacionados ao aprimoramento da máquina onde o banco de dados está armazenado (EL-REWINI; ABD-EL-BARR, 2005).

2.2.1 Banco de dados orientado a grafos

O modelo de banco de dados orientado a grafos é um tipo de banco de dados não relacional, que consiste em uma estrutura de dados onde os esquemas e instâncias são modelados como um grafo direcionado, que pode ser rotulável, ou para uma generalização da estrutura de um grafo, onde a manipulação dos dados é expressa por operações orientadas a grafos e tipos construtores (ANGLES; GUTIERREZ, 2008).

Um grafo é um conjunto de objetos chamados nós/vértices, que possuem relacionamentos entre si, chamados de arestas, essas duas estruturas básicas formam um grafo. A Figura 2.1 mostra um exemplo de grafo de relacionamentos entre pessoas.

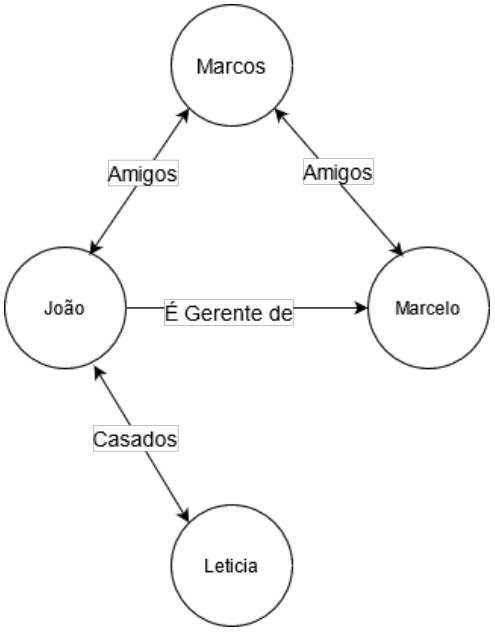

Figura 2.1: Exemplo de grafo

Esse tipo de modelo de banco de dados pode ser aplicado em diversas áreas onde existe a interconexão dos dados, ou onde a topologia é tão importante quanto os dados em si, como por exemplo redes sociais e as áreas da biologia e química. Nesse tipo de situação, os dados e relacionamentos entre os dados estão no mesmo nível, onde os dados são as entidades que representam objetos que existem como uma unidade, e as relações são propriedades que estabelecem a conexão entre uma ou mais entidades (ANGLES; GUTIERREZ, 2008).

Uma entidade, pode ser representada tanto no nível de esquema, quanto de instância, onde o esquema representa o tipo da entidade, semelhante a estrutura das tabelas no modelo relacional, onde se tem um template, normalmente representado por um rótulo, que representa uma generalização de um objeto do mundo real. Já a instância é uma entidade concreta, ou seja, um objeto único com suas propriedades. No caso de relacionamentos/arestas, eles podem ser vistos tanto como atributos, ou entidades, dependendo da implementação, sendo o atributo mono-valorado ou multi-valorado, e no caso de entidade, seguindo uma lógica semelhante aos objetos (ANGLES; GUTIERREZ, 2008).

Os banco de dados orientado a grafos podem ser sub-divididos em 3 grupos, com diferenças em seus modelos, sendo esses, baseados em grafos de propriedades rotuladas, *Resource Description Framework* (RDF), e os hipergrafos (ROBINSON; WEBBER; EIFREM, 2015). O grafo de propriedades rotuladas é representado por um grupo de nós, relacionamentos, propriedades e rótulos, onde tanto os nós quanto as relações são rotulados e podem armazenar propriedades em um esquema de pares chave/valor. Os rótulos servem para agrupar os nós e relacionamentos em grupos. As arestas nesse modelo sempre devem começar em um nó e terminar em outro, e devem possuir uma direção, da onde vem e para onde vão, sendo assim, um grafo direcionado (FRISENDAL, 2017).

O RDF representa dados no formato sujeito-predicado-objeto, conhecido como triplas, onde o sujeito representa o recurso, o predicado as características do recurso, além de representar a relação entre o sujeito e o objeto. Para melhor entendimento, pode se fazer um paralelo com o modelo entidade-atributo-valor, onde o sujeito é a entidade, o atributo o predicado e o valor o objeto (W3C, 2004).

Por último, temos os hipergrafos, os quais tem como diferencial o fato de que uma aresta pode conectar qualquer número de nós, diferente dos grafos de propriedades e RDF que são limitados a apenas dois nós conectados por uma aresta (IORDANOV, 2010).

Para a realização das consultas específicas, os bancos de dados orientado a grafos usam diversas linguagens, que se diferenciam na forma de escrever consultas e nos recursos disponíveis para serem usados. Alguns exemplos de linguagens de consulta são SPARQL, Gremlin, Cypher e SQL (BESTA et al., 2020).

As vantagens que os banco de dados orientado a grafos oferecem, partem do fato de que boa parte dos SGBD que empregam esse modelo implementam esquemas flexíveis, permitindo mudanças e alterações na base de dados com facilidade. Além disso, a forma a qual os bancos de dados orientado a grafos modelam os dados se assemelha a forma como os objetos do mundo real interagem, onde se tem os objetos, representado no grafo como os nós e os relacionamento ou interações entre esse objetos, representado pelas arestas. Alguns exemplos de cenários onde os bancos de dados orientado a grafos se destacam são: software de recomendações, localização geoespacial, redes de computadores, redes sociais, entre outros (ROBINSON JIM WEBBER, 2015).

Capítulo 3

Estudo Comparativo entre Banco de Dados Orientado a Grafos

Neste capítulo são apresentadas as metodologias usadas para comparar os SGBD escolhidos, a motivação por trás da escolha desses SGBD e os trabalhos relacionados usados na construção dessa monografia.

Os SGBD escolhidos foram selecionados com base nos seguintes critérios: modelo suportado, recursos disponíveis, posição no ranking no DB-Engine e licença de código aberto. Além disso, outro fator importante é suporte a drivers para a linguagem Java, na qual são programados os testes de desempenho, e suporte ao sistema operacional Windows 10, onde os testes serão executados. Baseado nesses critérios, os SGBD escolhidos, foram: Neo4j, ArangoDB, OrientDB e AgensGraph.

O Neo4j é um SGBD com modelo de banco de dados orientado a grafos (NEO4J, 2021b), sua escolha foi devido a sua grande adesão por parte dos desenvolvedores, no ranking do DB-Engine o Neo4j ocupa a 20° posição no ranking geral e 1° posição no ranking de banco de dados orientado a grafos (DB-ENGINE, 2020a). Além disso, o Neo4j possui uma versão de código aberto com diversos recursos disponíveis, o que permite o melhor uso do SGBD para o comparativo. O Neo4j utiliza rótulos para agrupar os dados, tanto nós quanto arestas (NEO4J, 2021b).

O ArangoDB é um banco de dados multi-modelo, com os seguintes modelos suportados: modelo orientado a grafos, orientado a documentos e orientado a chave-valor. Ele é código aberto e ocupa a 68° posição no ranking geral e 3° posição no ranking de banco de dados orientado a grafos (DB-ENGINE, 2020a). O ArangoDB utiliza coleções para agrupar os dados,

tanto nós quanto arestas (ARANGODB, 2021).

O OrientDB é um banco de dados multi-modelo, com os mesmos modelos suportados pelo ArangoDB, sua escolha foi pelo suporte ao modelo de grafos, ser de código aberto e ocupar a 76° posição do ranking geral e 4° posição no ranking de banco de dados orientado a grafos (DB-ENGINE, 2020a). O OrientDB agrupa os dados por classes, tanto nós quanto arestas (ORIENTDB, 2021a).

O AgensGraph e um banco de dados multi-modelo que suporta o tipo orientado a grafos e o relacional. Sua escolha foi por seu modelo orientado a grafos e ser de código aberto. Ele ocupa a 321° posição no ranking geral e 31° posição no ranking de banco de dados orientado a grafos (DB-ENGINE, 2020b). Mesmo com sua baixa posição no ranking se comparado com os SGBD anteriores, as outras opções no ranking são exclusivas para sistemas Linux ou possuem apenas versões pagas. O AgensGraph utiliza rótulos para agrupar os dados, assim como o Neo4j (AGENSGRAPH, 2021).

Mais detalhes sobre cada um desses SGBD, como seus recursos e desempenho serão apresentados no Capítulo 4, onde é feita a análise documental e de desempenho.

3.1 Metodologia

Nessa seção serão apresentadas as metodologias utilizadas na comparação dos SGBD. A metodologia está dividida em 2 partes, na primeira parte é realizada uma análise documental, cuja metodologia é apresentada na Seção 3.2.1, na segunda parte, apresenta-se uma análise de desempenho, a metodologia empregada nesses experimentos é descrita na Seção 3.2.2. Na Seção 3.2 os trabalhos relacionados são apresentados.

3.1.1 Análise Documental

A análise documental tem como objetivo comparar as características técnicas, tais como maturidade dos SGBD, comunidade, material para o aprendizado, entre outras características que afetam os desenvolvedores das mais diversas formas, com objetivo de auxiliar o usuário a escolher a melhor opção para o seu projeto não baseado unicamente em aspectos de desempenho.

A análise documental foi dividida em 5 critérios, inspirado no trabalho de Chen (CHEN,

2016), porém com diferenças relacionadas a interpretação de cada critério. Os elementos avaliados são: Maturidade/Nível de Suporte, Comunidade, Recursos para o Aprendizado, Segurança, Interface de Usuário.

Maturidade/Nível de suporte se refere a questões de maturidade do SGBD, como, tempo no mercado, número de versões, frequência das atualizações, investimentos, equipe de desenvolvimento, parceiros, entre outras questões que se referem ao estado atual do SGBD e as previsões desse SGBD para o futuro baseado em seu histórico e contexto. Essa análise tem como objetivo extrair informações sobre o SGBD de forma que seja possível inferir o estado atual do software em termos de maturidade dos recursos e funcionalidades implementados até o momento, e de previsões para o contínuo suporte do mesmo.

Comunidade aborda questões de engajamento da comunidade com o SGBD, por exemplo fóruns exclusivos, número de membros desses fóruns ou comunidades, número de perguntas feitas e a porcentagem de perguntas respondidas, entre outros fatores que podem ser extraídos a partir dessas comunidades para se ter uma melhor caracterização do suporte existente de outros desenvolvedores para o SGBD. Para essa análise, as informações foram extraídas do Stack Overflow (OVERFLOW, 2021) e de fóruns exclusivos, caso existam. Caso não existam, foi buscado em outras comunidades que os desenvolvedores do SGBD promovem no site oficial da ferramenta. O objetivo dessa análise é ter uma visão geral da comunidade que envolve esse SGBD, de forma que seja possível comparar objetivamente as comunidades dos SGBD analisados.

Recursos para o Aprendizado envolve os recursos que os desenvolvedores disponibilizam para o aprendizado desse SGBD como documentação, tutoriais, cursos, livros, entre outros materiais gratuitos, que oferecem possibilidades para o usuário aprender a utilizar o SGBD. Essa análise inclui apenas material oficial fornecido pelos desenvolvedores e não consideram tutoriais ou materiais externos. O objetivo dessa análise é comparar a variedade de opções e quantidade de material para cada uma dessas opções entre os SGBD, de forma que seja possível verificar qual SGBD possui a maior diversidade e volume de material para o aprendizado da ferramenta.

Segurança envolve recursos do SGBD disponibilizados para atender as questões de segurança dos dados. Como exemplo temos, uso de logs, sistema de usuários e privilégios, ferra-

mentas de backup, entre outros sistemas que asseguram o acesso controlado as bases de dados e a confiabilidade dos dados ali armazenados. Essa análise tem como objetivo extrair os recursos mais relevantes (recursos como protocolos de segurança, sistemas de acesso e restrição de usuário e criptografia dos dados) de segurança dos SGBD analisados.

Interface de Usuário, se refere a recursos de interface disponibilizados para a visualização, manipulação e gerenciamento referentes ao banco de dados. Esses recursos podem ser como, visualização do consumo do sistema pelo SGBD, visualização dos dados de forma gráfica, meios para a simplificação do gerenciamento dos bancos de dados, entre outras soluções que permitem/facilitam a utilização do SGBD por parte dos administradores de banco de dados. O objetivo dessa análise é extrair características que facilitam ou fornecem recursos exclusivos para a manipulação e gerenciamento do banco de dados.

3.1.2 Comparação de Desempenho

A comparação de desempenho será feita a partir da comparação do tempo de execução das operações escolhidas aos SGBD. Essas execuções consistem em operações básicas como inserção, remoção, seleção e atualização dos dados, sendo essas as operações básicas CRUD. Além dessas, serão realizadas operações mais específicas a grafos, como por exemplo, menor caminho entre dois nós e k-vizinhos (todos os nós conectados a um nó específico). Esses testes serão realizados para verificar o desempenho dos SGBD para diferentes operações e em diferentes tamanhos de uma mesma base de dados, para testar como o tempo de execução dessas operações varia com o crescimento da base. A escolha das operações e bases de dados geradas foram pautadas em outros artigos e publicações, porém de forma personalizada, para assim gerar resultados diferentes e complementares a esses outros trabalhos. Mais detalhes desses trabalhos na Seção 3.2. O código usados nestes testes pode ser encontrado no github (GRAMS, 2021).

Os testes de desempenho foram divididos em 6 partes: testes básicos, *workload*, operações paralelas, múltiplos núcleos, uso de memória RAM e uso de armazenamento de disco.

Testes Básicos

Os testes básicos são compostos por operações simples, como criar, ler, deletar vértices e arestas. Além dessas operações, também entram nos testes básicos, operações de k-vizinhos (todos os elementos que saem de um vértice), menor caminho entre dois elementos (menor

Tabela 3.1: Categorias das operações básicas

Categoria	Descrição	Operações pertencentes
Criação	Cria tanto vértices quanto arestas no banco de dados	Add Vértice e Add Aresta
Remoção	Remove tanto vértices quanto aresta do banco de dados	Del Vértice e Del Aresta
Busca	Busca um nó ou aresta baseado em algum critério	Busca Vértice, Busca Aresta
Caminhamento	Caminha pelo grafo em busca de nós e arestas baseado em algum critério	K-Vizinhos e Menor Caminho
Agrupamento	Agrupa os dados baseado em algum critério	Função media

caminho não direcionado) e buscas por vértices com o uso de filtros. Essas operações podem ser divididas em categorias, dependendo de sua funcionalidade, essas categorias podem ser verificadas em mais detalhes na Tabela 3.1. Essas categorias focam em diferentes aspectos da manipulação de um banco de dados, de forma que cada categoria explora e testa um escopo diferente de execuções no banco de dados.

Esse teste foi construído baseado principalmente no trabalho de Lissandrini *et al.* (LISSANDRINI; BRUGNARA; VELEGRAKIS, 2018), porém outros trabalhos também fizeram testes semelhantes, onde são executadas operações básicas para diferentes base de dados e analisado os resultados, e tem como intuito verificar o desempenho de cada SGBD para operações básicas e como essas operações escalam conforme a base dados aumenta em número de elementos (nós e arestas). As informações com todas as operações individuais e suas descrições podem ser vistas na Tabela 3.2 (O apelido serve como um nome substituto que será usado para se referir a essas operações de forma mais curta).

Para a extração dos tempos, foi executado o conjunto de operações num total de 6 execuções, onde foi excluído o tempo da primeira execução, e depois pego a mediana dos 5 tempos restantes. Os valores usados como parâmetros para as operações foram gerados de forma aleatória e variando para cada execução. Para a operação de menor caminho foi calculado a média entre esses valores em vez de se pegar a mediana. Além disso, só foram considerados execuções as quais encontravam um caminho entre os nós, em casos em que o caminho não era encontrado, os tempos eram descartados. O método de melhor caminho utilizado foi bidirecional, ou seja, o caminho pode ser nó 1→nó 2 ou nó 2→nó 1, o importante é o caminho ser o menor. Devido a uma característica dos drivers Java dos SGBD, o valor da primeira operação executada pelo

Tabela 3.2: Operações básicas

Operação	Parametros	Descrição	Apelido
Criar Vértice	Nó n	Adiciona um novo vértice n	Add Vértice
Criar Aresta	int origem, int destino, int id	Adiciona uma nova aresta com o ide da origem e destino, e um identificador único (id) para a aresta	Add Aresta
Remover Vértice	int id	Remove um vértice baseado no id do nó	Del Vértice
Remover Aresta	int origem, int destino	Remove uma aresta baseado no id da origem e destino	Del Aresta
Alterar Vértice	Nó n	Altera um nó n	Alt Vértice
Busca por Vértice	int id	Busca por um vértice, baseado no id desse vértice	Busca Vértice
Busca por Aresta	int origem, int destino	Busca por uma aresta, baseado no id de origem e destino da aresta	Busca Aresta
Busca com Filtro	String nome	Busca um vértice baseado no atributo "nome"do nó	-
K-Vizinhos	int id	Busca os k vizinhos de um nó, baseado no id desse nó	-
Menor Caminho	int id1, int id2	Busca o menor caminho entre dois nós baseados nos id's desses nós	-
Função Media	Nenhuma	Calcula a media dos nós para o atributo "número"	-

programa sempre produzia um valor de tempo maior que a média, para evitar que esse valor afetasse os resultados obtidos, antes da execução das operações, sempre é executado uma operação de busca por vértice, com identificador único (id) 1, para que esse tempo não afete os valores obtidos por cada operação.

Operações Paralelas

O teste de operações paralelas consiste na simulação de múltiplas requisições sendo enviadas ao banco de dados de forma paralela para avaliar como os SGBD lidam com uma situação como essa. São usadas 3 métricas nesse teste, a primeira é o tempo da operação, que consiste no tempo individual que cada operação leva para executar. A segunda é o tempo total, que se refere ao tempo total que leva para todas as requisições em paralelo serem executas e finalizadas. E por último temos o tempo sequencial, que indica o tempo que leva para executar todas as operações de forma sequencial, e não paralela.

O número de requisições usados foi de 1, 2, 4, 6 e 12, sendo esse o número de núcleos do processador. Esse experimento tem como objetivo testar como os SGBD se comportam em questão de tempo de execução quando requisitados por múltiplas operações de forma simultânea, com cada requisição com suas conexões e sessões próprias ao banco de dados. A motivação

por trás desse teste vem devido ao aumento no uso de funções assíncronas nos servidores que se comunicam com o banco de dados. Essas funções permitem que múltiplas requisições sejam enviadas ao banco de dados de forma quase paralela, fazendo com que o SGBD tenha que lidar de alguma forma com essas requisições. Essa ideia pode ser visualizada na Figura 3.1.

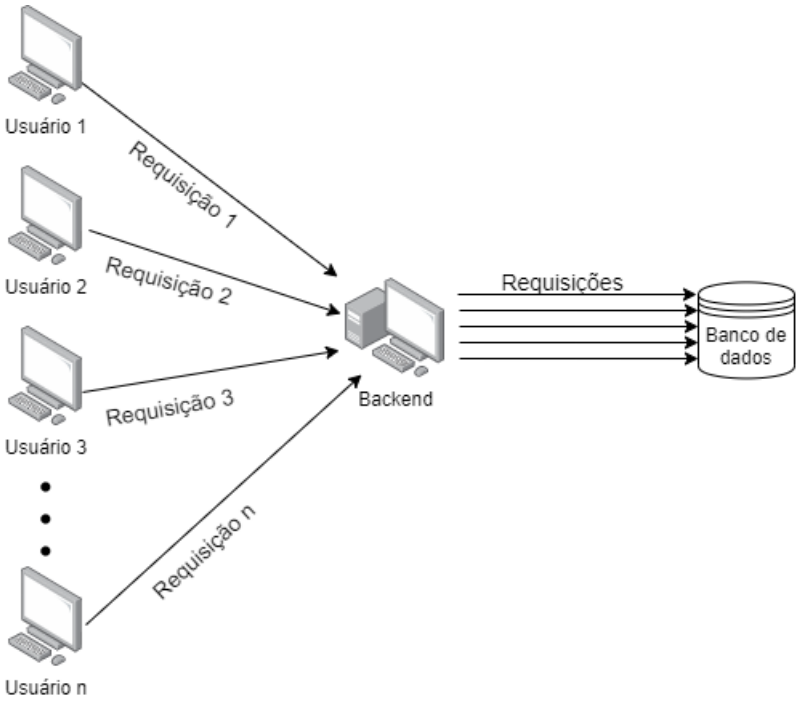

Figura 3.1: Múltiplas requisições em paralelo

Múltiplos Núcleos

Para esse teste foi executado os testes básicos nos SGBD variando o número de núcleos disponíveis em 1, 2, 4, 6 e 12, para o SGBD. Para desativar os núcleos disponíveis para os SGBD, foi utilizado o sistema de afinidade do gerenciador de tarefas do Windows. O banco de dados possuía 2 milhões de elementos no momento da execução. Esse teste tem como intuito verificar se existe algum ganho de desempenho para operações individuais com o uso de múltiplos núcleos de um processador.

Workload

Essa operação consiste em uma inserção massiva de dados no banco de dados de uma vez, também conhecido como *Massive Insertion Workload*, porém chamaremos apenas de *workload*,

e serve para popular um banco de dados com um conjunto grande de registros de forma mais rápida e eficiente comparada a adicionar um registro por vez. Esse teste foi executado tanto para vértices quanto arestas, com valores do conjunto de *workload* variando entre 10 mil e 100 mil, indo de 10 mil para 20 mil, 40 mil, 60 mil e 100 mil. O banco de dados possuía 100 mil elementos no momento da execução. Esse teste tem como objetivo verificar como os SGBD lidam com esse tipo de operação, tanto para vértices quanto arestas e para diferentes tamanhos de conjuntos.

Uso de Memória RAM

Esse teste tem como objetivo verificar quanto de memória RAM os SGBD consomem quando ativos. Foram testados o valor de memória RAM em 3 cenários, quando iniciado o banco de dados, mas nenhuma operação foi executada nele, quando executado um conjunto de testes básicos apresentados anteriormente na Tabela 3.2, e para a operação de *workload* (de forma separada, ou seja, o banco de dados foi reiniciado e executado apenas essa operação para esse cenário). Isso foi feito quando o banco de dados estava com 2 milhões de elementos (1 milhão de arestas e 1 milhão de nós), e tem como objetivo verificar o consumo de memória RAM em diferentes cenários pelos SGBD. As informações foram coletadas pelo gerenciador de tarefas do Windows.

Uso de armazenamento em disco

Esse teste verificou quanto cada banco de dados ocupa em disco, tanto quando vazio, quando com dados (cheio). Essa informação foi extraída da pasta que armazena os conjuntos de dados, utilizando a informação fornecida pelo sistema de arquivos do Windows. A base possuía 2 milhões de elementos para o teste cheio. O objetivo desse teste é avaliar o consumo de memória de disco pelos bancos de dados dos respectivos SGBD.

Conjunto de dados

A base de dados utilizada para os testes de desempenho foi construída exclusivamente para esse trabalho, e foi inspirada em bases como as geradas pelo gerador *LFR-Genrator* e base de dados disponibilizadas pela *Stanford Large Dataset* (LESKOVEC; KREVL, 2014). A estrutura dessa base pode ser observado na Figura 3.2, onde se tem um tipo de nó, e um tipo de aresta,

Tabela 3.3: Detalhes do nó

Propriedade	Descrição
Id	Identifica um nó de forma única, consiste em um inteiro
Nome	String aleatória de tamanho entre 30 e 50, com letras maiúsculas, minúsculas e números
Número	Número inteiro com valor de 0 a 99999

com as arestas direcionadas conectando aleatoriamente os nós. Os nós possuem 3 atributos, como pode ser visto na Tabela 3.3, que servem para caracterizar um objeto simples para nossos testes. As arestas apenas possuem um atributo, tirando o nó destino e origem que ela conecta, sendo esse o id, com o mesmo significado que no nó, identificar de forma única a aresta. Esses atributos foram colocados para simular uma base de dados simples, de forma que cada nó tenha mais que apenas seu id, e as arestas sirvam apenas para ligar esses nós, sem que as ligações tenham atributos próprios. A base varia de 200 mil até 2 milhões de elementos (com metade sendo arestas e a outra metade nós), passando por 200 mil, 400mil, 600mil, 800mil, 1,2 milhões, 1,6 milhões e 2 milhões. Os valores para os atributos "nome"e "número"dos vértices foi gerado aleatório, e o id foi gerado de forma incremental.

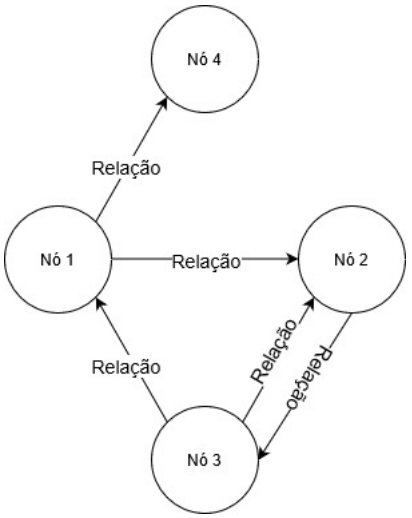

Figura 3.2: Exemplo de grafo da base de dados gerada

O motivo do uso dessa base foi porque mesmo que base de dados reais possuam dados com características mais realistas, como por exemplo, as bases do *Stanford Large Dataset* (LESKOVEC; KREVL, 2014), essas bases são fornecidas com tamanhos fixos, e como esse trabalho tem como objetivo analisar de forma controlada o impacto do tamanho da base de dados no tempo de

execução das operações, foi preferível utilizar uma base de dados que pudesse ser construída de forma iterativa, onde novos dados são incrementados em grupos de tamanho pré-estabelecidos, e os anteriormente adicionados não são afetados. Foi considerado a utilização do gerador *LFR-Generator* para a geração da base de dados, porém, devido à natureza imprevisível na geração das arestas por parte do gerador, onde o número de arestas é aleatório, e a impossibilidade de gerar a base de dados de forma iterativa, fez com que se fosse descartada essa opção.

Máquina

Para executar os testes de desempenho foi utilizado um computador desktop, com Windows 10, i5-10400, 16GB RAM, Placa de Vídeo GTX 1060 e um SSD 240GB, com 400MB/s de leitura e 450MB/s de gravação.

3.2 Trabalhos Relacionados

Em relação aos trabalhos relacionados a comparação de banco de dados orientado a grafos, no aspecto de desempenho, temos o trabalho de Lissandrini *et al.* que compara os SGBD ArangoDB, BlazeGraph, Neo4J, OrientDB, Sparksee, SQLG, Titan, em uma série de testes de operações básicas CRUD em bases de dados reais e artificiais, com milhares e milhões de nós e arestas. Os parâmetros usados na comparação foram tempo de execução, espaço ocupado em disco pelas bases de dados e o *workload* de dados.

O trabalho de Kolomičenko (KOLOMIČENKO; SVOBODA; MLÝNKOVÁ, 2013) desenvolveu um novo benchmark chamado de Bluebench, usando como parâmetros de comparação o tempo de execução, os objetos carregados por segundo (LOPS) e o número de arestas atravessadas por segundo (TEPS) para comparar bancos de dados orientado a grafos. Para testar seu benchmark foi utilizado os SGBD DEX, InfiniteGraph, Neo4j, OrientDB e Titan.

Semelhante ao trabalho Kolomičenko, o trabalho de Matyjaszczyk *et al.* (MATYJASZCZYK; ROSOWSKI; WREMBEL, 2020) desenvolve um benchmark chamado GooDBye, usando como parâmetro de comparação o tempo de execução em uma base de dados artificialmente gerada. Para testar seu benchmark foi usado os SGBD ArangoDB, OrientDB, TitanDB, JanusGraph, GraphX.

Beis (BEIS; PAPADOPOULOS; KOMPATSIARIS, 2015) desenvolveu um trabalho com-

parando 3 SGBD, sendo esses, Neo4j, Titan e OrientDB, onde são comparados os tempos de execuções para operações de inserção em massa, inserção única e operações de caminhamento no grafo, sendo essas, busca pelos vizinhos (k-vizinhos), menor caminho e e busca pelos nós adjacentes.

O artigo de Jouli e Vansteenberghe (JOUILI; VANSTEENBERGHE, 2013) compara os SGBD Neo4j, DEX, Titan-BDB, Titan-Cassandra e OrientDB, em testes envolvendo operações de carregamento de dados, operações de caminhamento e operações de busca e adição de elementos no grafo usando múltiplos clientes/usuários (semelhante as operações paralelas deste trabalho).

Em relação a trabalhos de comparação teórica, temos o trabalho de Angles (ANGLES, 2012) que compara os SGBD AllegroGraph, DEX, Filament, G-Store, HyperGraphDB, InfiniteGraph, Neo4j, Sones e VertexDB em diferentes tipos de características, analisando a existência ou não do suporte a tais características como formas de armazenamento, recursos para manipulação dos dados, forma de estruturar os dados, entre outros.

Por último, temos o trabalho de Chen (CHEN, 2016), o qual compara os SGBD Neo4j e MySQL de forma quantitativa com o custo de armazenamento da base de dados e o tempo de execução e uma comparação qualitativa nos seguintes pontos: Maturidade/nível de suporte, facilidade de programar, flexibilidade, segurança e visualização de dados. As comparação qualitativa foi feita utilizando uma base de dados gerada artificialmente simulando uma grande base de dados social.

Os principais diferencias desse trabalho são: A utilização versões mais recentes desses SGBD para comparação, o que inclui novos recursos e melhoras em aspectos de desempenho; O uso de multi-thread na execução de algumas consultas CRUD, para verificar suas capacidades em lidar com múltiplas consultas paralelas; A comparação documental customizada do trabalho do Chen (CHEN, 2016) com os SGBD escolhidos; Testes de desempenho com diferentes critérios e situações.

Capítulo 4

Análise Comparativa

Neste Capítulo serão apresentados e discutidos os resultados obtidos na análise documental e comparação de desempenho, e também será avaliado esses resultados em comparação com outros trabalhos similares.

4.1 Análise Documental

Esta seção apresenta uma comparação documental, que tem como foco as características de cada um dos SGBDs analisados, para uma melhor noção de suas capacidades e recursos disponíveis para desenvolvedores. A análise documental consiste na avaliação e comparação dos SGBDs escolhidos nos critérios de maturidade/nível de suporte, comunidade, recursos para o aprendizado, segurança e interface de usuário. A maior parte das informações aqui apresentadas, advém diretamente da documentação de cada respectivo SGBD.

4.1.1 Neo4j

Em questão de maturidade e nível de suporte, o Neo4j é um SGBD desenvolvido desde os anos 2000, com sua primeira versão em 2002. Seu lançamento oficial da versão 1.0 foi em 2010 e a empresa por trás do software é a própria Neo4j. Durante todos esses anos, o Neo4j recebeu 6 investimentos maiores que 1 milhão de dólares, por parte de diversas empresas e organizações, sendo o maior deles no valor de 330 milhões em Junho de 2021 (NEO4J, 2021b).

Além disso, o Neo4j possui 119 parceiros, com alguns deles sendo, Volvo, Adobe, Walmart, Microsoft, Ebay, Linkedin, entre outras. Em questão de pessoal, a Neo4j possui mais de 360 funcionários trabalhando na empresa. A versão atual do Neo4j enterprise para desenvolvedores

é a 4.2.6, e a última atualização foi em Maio de 2021. As linguagens que possuem drivers oficiais do Neo4j, são: .Net, Java, Spring, JavaScript, Go e Python (NEO4J, 2021b).

Com esses dados podemos observar que o Neo4j possui um alto nível de maturidade, pois seu desenvolvimento se estende por um longo período, e a versão 4.2.6 mostra que o sistema está em uma versão avançada, sendo assim, muitos recursos já implementados até o momento. Os investimentos milionários durante os anos mostram confiança dos investidores no Neo4j no longo termo. As grandes empresas parceiras, também aumentam sua confiança, pois significa que essas empresas estão dispostas a apostar seus sistemas no Neo4j. Sua última atualização sendo recente, mostra que o suporte ao software é frequente.

Em questão de comunidade, o Neo4j possui mais de 20 mil, com 28,7% das questões não respondidas até hoje no Stack Overflow (OVERFLOW, 2021). O Neo4j possui um fórum oficial, com 12,7 mil usuários, sendo 808 ativos nos últimos 30 dias, e com 71 mil posts feitos no total. Isso mostra que o Neo4j possui uma comunidade ativa, isso garante que dificuldades encontradas durante o uso do SGBD possam ser solucionadas por perguntas já feitas, ou pela criação de novas perguntas que podem ser respondidas pela grande comunidade do Neo4j (NEO4J, 2021a).

Na questão de recursos para o aprendizado, o Neo4j fornece sua documentação, cobrindo todas as ferramentas e recursos disponibilizados pelo software, além de guias para inciantes, com os primeiros passos na utilização do Neo4j. O Neo4j também possui cursos divididos em temas específicos, além um canal do Youtube com centenas de vídeos de palestras e vídeos sobre funções do Neo4j, soluções usando o Neo4j, guias de desenvolvimento, entre outros temas. Também são disponibilizados livros com tópicos relacionados ao uso de banco de dados orientado a grafos em diversos cenários, como ciência de dados e aprendizado de máquina, usando o Neo4j para as explicações e exemplificações. Por último, o Neo4j possui um guia/tutorial interativo básico junto a interface desktop e uma base de conhecimento, com uma lista de soluções de problemas comuns que os desenvolvedores encontram ao longo do desenvolvimento e são listados para fácil e rápido acesso (NEO4J, 2021b).

É possível ver que o Neo4j oferece diferentes possibilidades para o aprendizado, com soluções em vídeo, tutoriais interativos, livros, documentação e soluções rápidas, dando opções e volume dessas opções, para os desenvolvedores aprenderem a utilizar o Neo4j.

Em questão de segurança, o Neo4j possui os seguintes recursos: Sistema de logs para as operações realizadas no banco de dados para monitoramento; Sistema de autenticação de usuários para confirmar a autenticidade do usuário e autorização por *role-based access control* (RBAC); Segurança da comunicação entre canais usando SSL/TLS (esses são protocolos de comunicação feitos para providenciar uma comunicação segura em uma rede) (NEO4J, 2021b); Backup dos banco de dados. A segurança do Neo4j é padrão dos SGBDs, sem recursos extras disponíveis oficialmente.

No quesito interface de usuário, o Neo4j fornece uma interface desktop, com informações referentes ao banco de dados, e algumas manipulações possíveis, como criação de usuários e a designação de papéis a esses usuários, criação de scripts em Cypher, para a execução automática de consultas mais complexas, além de uma interface gráfica para a visualização e manipulação dos dados resultantes de uma consulta (NEO4J, 2021b). A Figura 4.1 mostra a interface do Neo4j.

A interface desktop do Neo4j fornece alguns recursos para facilitação da visualização de certas informações sobre a base de dados, e algumas funções de manipulação do banco, mas sem muitos recursos relevantes para administradores de banco de dados.

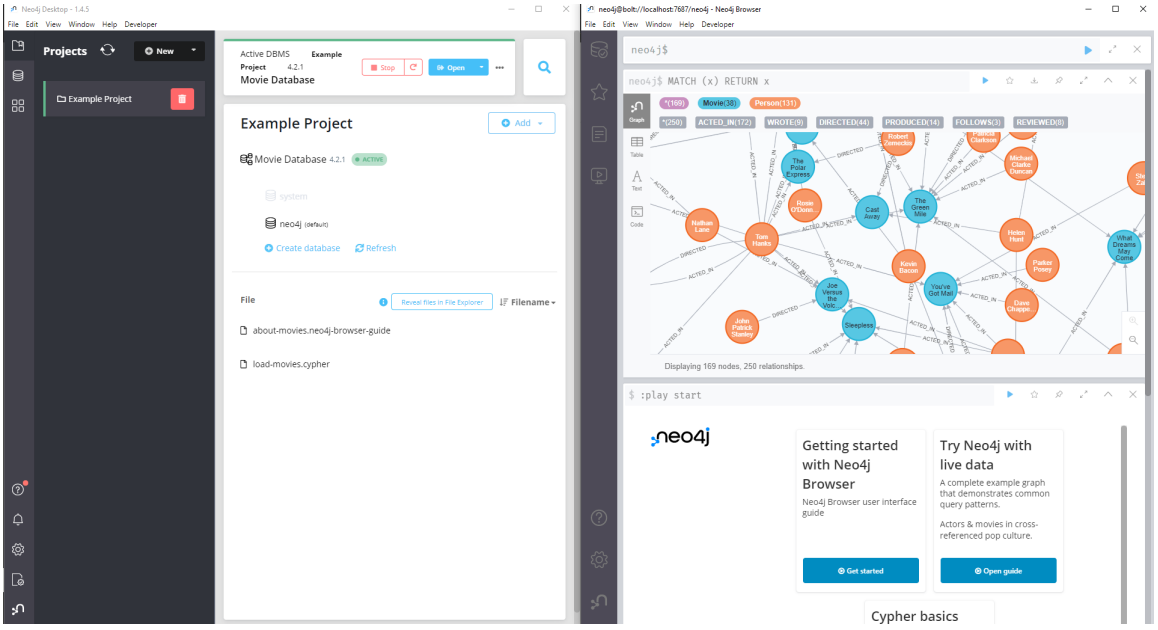

Figura 4.1: Interface Neo4j

4.1.2 OrientDB

Em questão de maturidade e nível de suporte, o OrientDB teve seu desenvolvimento iniciado em 2009 por Luca Garulli, e foi adquirido pela CallidusCloud em 2017 (PICART, 2017), que logo em seguida foi adquirida pela SAP em 2018 (MILLER, 2018). Está atualmente na sua versão 3.0.32, e sua última atualização foi em Abril de 2021. As linguagens que possuem drivers oficiais do OrientDB são: Java, PHP e .Net (ORIENTDB, 2021a).

O OrientDB está em sua versão 3.x de desenvolvimento, com atualizações recentes, o que representa um bom sinal para o estado atual da ferramenta. Além disso, o fato de que o OrientDB pertence a uma grande empresa como a SAP dá esperança para o suporte futuro do SGBD, devido ao seu grande suporte financeiro possível por sua empresa detentora. Por outro lado, devido a falta transparência em relação a equipe que trabalha no projeto, os parceiros que utilizam o OrientDB, e o fato que a SAP adquiriu a CallidusCloud, que possuía outros softwares em seu nome, não apenas o OrientDB, diminuem um pouco a confiança desse suporte no longo prazo do OrientDB.

Na questão de comunidade o OrientDB possui no Stack Overflow 2672 questões no total, com 39,7% dessas questões não respondidas (OVERFLOW, 2021). O OrientDB também possui um fórum próprio com 480 usuários, com 12 usuários ativos nos últimos 30 dias e com 2,8 mil posts feitos até hoje (ORIENTDB, 2021b). Isso mostra que a comunidade do OrientDB está consideravelmente abaixo se comparada com o Neo4j, porém já possui seu espaço.

Na questão de recursos para o aprendizado do OrientDB, ele fornece sua documentação referente ao SGBD, semelhante ao Neo4j, e também disponibiliza um curso grátis para iniciantes, com os conceitos básicos para o desenvolvimento de soluções utilizando o OrientDB (ORIENTDB, 2021a). O OrientDB fornece poucas possibilidades para o aprendizado, tanto em opções, quanto em volume de material, sendo bem limitado nesse aspecto.

Em questão de segurança, o OrientDB é similar ao Neo4j, com sistema de usuários, divididos por papéis que determinam o acesso desses usuários a base de dados, uso de logs, e comunicação usando SSL/TLS e backup (ORIENTDB, 2021a).

No quesito de interface de usuário, o OrientDB fornece uma interface semelhante ao Neo4j, com funções para executar operações, visualizar e manipular os dados resultantes dessas consultas (ORIENTDB, 2021a), porém, em questão de funções de manipulação, o OrientDB possui

uma gama maior de funcionalidades disponíveis através de sua interface, como a criação de papéis para os usuários, visualização das classes de nós e arestas no banco com opções de manipulação, como, deleção, consulta e criação das classes e elementos que compõem essas classes. A Figura 4.2 mostra a interface do OrientDB.

Figura 4.2: Interface OrientDB

4.1.3 ArangoDB

Em questão de maturidade e nível de suporte, o ArangoDB teve seu desenvolvimento iniciado em 2011, e como o Neo4j, recebeu diversos investimentos nos últimos anos, com o último e maior, sendo de 10 milhões. Entre seus clientes temos a Cisco, Dish, VMware, entre outras empresas de grande porte. Na questão de pessoal e atualizações, o ArangoDB possui 55 funcionários, e a versão ArangoDB Community edition é 3.7.11, com sua última atualização em Abril de 2021. As linguagens que possuem drivers oficiais do ArangoDB são: NodeJS, PHP, Java, JavaScript e Go (ARANGODB, 2021). É possível ver que o ArangoDB possui uma boa maturidade e nível de suporte, considerando o Neo4j e OrientDB, devido a sua versão 3.x, investimentos recebidos e parceiros de grande porte.

Na questão de comunidade, o ArangoDB possui uma quantidade pequena de questões no Stack Overflow, com 1739 no total, sendo 25,2% dessas questões não respondidas (OVERFLOW, 2021). O ArangoDB não possui um fórum próprio, mas possui uma comunidade no Slack (SLACK, 2021) com 4292 usuários registrados e um grupo no Google groups (GOO-

GLE, 2021b) com 1492 posts. Com isso é possível ver que a comunidade do ArangoDB é pequena, e a falta de um fórum oficial, apenas reforça essa afirmação.

No quesito recursos para o aprendizado, o ArangoDB fornece diversos recursos para isso, desde artigos, cursos online, um canal no Youtube com tutoriais e seminários ensinando ferramentas e recursos do SGBD. Além disso, o ArangoDB também fornece sua documentação nos mesmos moldes dos SGBDs analisados anteriormente e um guia interativo, semelhante ao Neo4j, porém com conteúdos mais avançados, e totalmente online, através da plataforma Colaboratory (GOOGLE, 2021a), utilizando uma máquina virtual, sem a necessidade de instalar o ArangoDB para o uso (ARANGODB, 2021). O ArangoDB, semelhante ao Neo4j, fornece diferentes recursos para o aprendizado em número opções, porem com menor volume de material total se comparado com o Neo4j.

No quesito segurança, o ArangoDB possui sistemas equivalentes aos anteriores, com controle de usuários, logs, e comunicação com SSL/TLS e backup (ARANGODB, 2021).

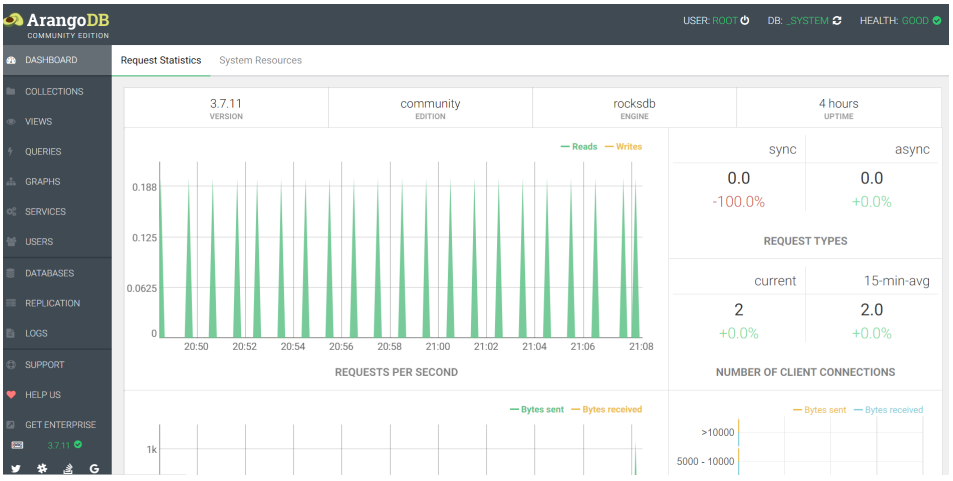

Figura 4.3: Interface ArangoDB

No quesito de interface de usuário, o ArangoDB fornece uma interface para a visualização e manipulação dos dados de forma gráfica, semelhantes ao Neo4j (ARANGODB, 2021). Ele também fornece funcionalidades de gerenciamento de usuários, coleções e grafos. O destaque da interface web do ArangoDB é a visualização completa do estado de funcionamento do banco de dados, com gráficos de operações de leitura e escrita, tempo de requisição, quantidade de dados transferida, além de informações sobre o consumo de memória e processador pelo banco.

Ele também oferece uma visão para sistemas distribuídos, onde é possível visualizar o estado de todas as máquinas e estatísticas referentes a operação do sistema como um todo, contando todas as máquinas. A Figura 4.3 mostra a interface do ArangoDB.

4.1.4 AgensGraph

Em questão de maturidade e nível de suporte, o AgensGraph teve sua primeira versão lançada em 2017. É pertencente a empresa Bitnine. Sua versão atual é 2.1.0 e sua última atualização foi em janeiro de 2019. As linguagens oficiais que possuem driver são: Java e Python (AGENSGRAPH, 2021).

A maturidade do AgensGraph é bem abaixo dos outros SGBD analisados, devido a sua versão 2.x, e seu suporte para o futuro é nulo, com sua última atualização há mais de 2 anos, com nenhuma perspectiva para a continuidade do projeto. Além disso, a empresa Bitnine já está trabalhando em um novo projeto, chamado Apache Age, que ainda está em versão beta, porém recebendo atualizações frequentes.

No quesito comunidade, o AgensGraph possui pouca quantidade de questões no Stack Overflow, com 103 no total, e 62,1% dessas questões não respondidas (OVERFLOW, 2021). O AgensGraph não possui um fórum próprio, porém a Bitnine possui um fórum geral, com apenas 1 post, feito pelo admin há 2 anos (AGENSGRAPH, 2021). Pode se dizer que a comunidade ativa do AgensGraph é inexistente.

No quesito recursos para o aprendizado, o AgensGraph fornece a documentação análoga a dos SGBDs anteriores, alguns tutoriais e artigos, além de um canal no Youtube da Bitnine, que possui alguns vídeos sobre o AgensGraph (AGENSGRAPH, 2021). O AgensGraph possui uma variedade de materiais para o aprendizado, com algum volume para cada opção.

No quesito de segurança, o AgensGraph possui os mesmos recursos dos outros SGBDs analisados.

Na questão de interface de usuário, o AegensGraph traz recursos semelhantes na visualização dos dados ao Neo4j, com funcionalidades focadas na visualização dos dados em forma de grafo, e com poucos recursos utilitários e de manipulação dos dados, sendo algum desses recursos a criação e deleção de rótulos e os dados identificado por esses rótulos. A Figura 4.4 mostra a interface do AgensGraph.

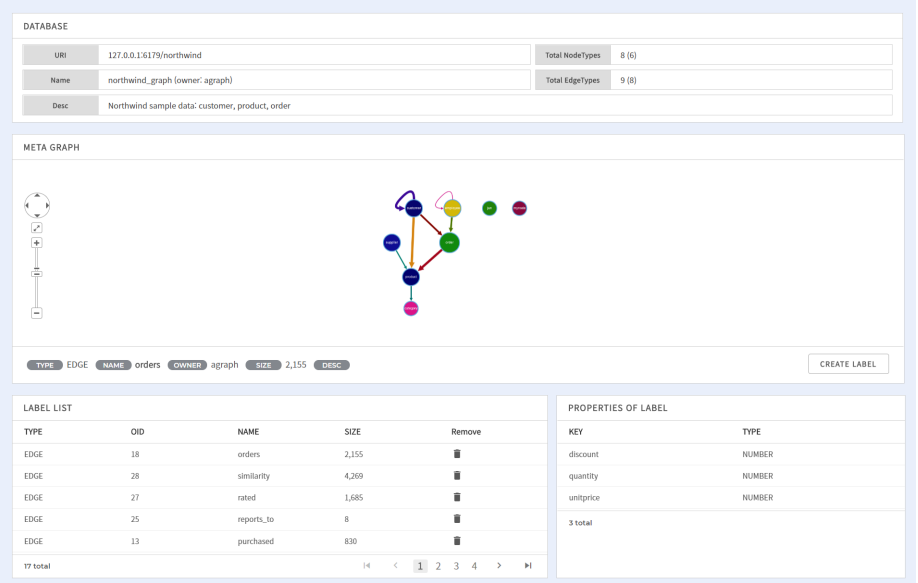

Figura 4.4: Interface AgensGraph (AGENSGRAPH, 2021)

4.1.5 Discussão

A Tabela 4.1 apresenta a comparação direta entre os SGDBS analisados.

Tabela 4.1: Comparação de características entre os SGBDs

SGBDs	Neo4j	OrientDB	ArangoDB	AgensGraph
Maturidade e Nível de Suporte	1°	3°	2°	4°
Comunidade	1°	2°	3°	4°
Recursos para o aprendizado	1°	4°	2°	3°
Segurança	1°	1°	1°	1°
Interface de Usuário	3°	2°	1°	3°

Em questão de maturidade e nível de suporte, o Neo4j ocupa a primeira posição, devido a sua transparência em relação a investimentos, grande número de funcionários, desenvolvimento contínuo, e vários parceiros grandes usando a solução. Em segundo lugar temos o ArangoDB, que possui grandes investimentos, já trabalha com empresas de grande porte e está em uma versão avançada do desenvolvimento, porém ainda fica atrás do Neo4j na maioria dos aspectos considerados. Em terceiro temos o OrientDB, que, devido a falta de transparência da equipe trabalhando nele, investimento, empresas parceiras, e a incerteza na forma como a SAP irá progredir com o projeto, deixam-no com a terceira posição. Por último temos o AgensGraph, com suporte praticamente zero, e pouca maturidade.

No quesito de comunidade, temos o Neo4j em primeiro, pois sua comunidade é a maior entre os 4, com diversos usuários totais e ativos nos fóruns, e um grande número de perguntas feitas, o que demonstra um grande volume de desenvolvedores engajados na comunidade. Em segundo temos o OrientDB, que possui uma comunidade abaixo do Neo4j, porém acima do ArangoDB. Em terceiro o ArangoDB, que possui uma pequena comunidade em relação aos anteriores, com poucos posts no Stack Overflow, e não possuindo um fórum dedicado, apenas comunidades menores, que mesmo sem todas as informações de usuários ativos é possível pressupor que ela é inferior numericamente as anteriores. E por último AgensGraph, com uma comunidade praticamente inexistente.

Em questão de recursos para o aprendizado, o Neo4j fica na frente, devido a sua vasta quantidade de formas de aprender, desde artigos até guias interativos, com um volume grande de material para cada opção. O ArangoDB fica em segundo, pois, mesmo com opções semelhantes e com um guia interativo mais completo e avançado, seu volume moderado de material fica abaixo do Neo4j. Em terceiro AgensGraph, que possui uma boa variedade de opções, porém com pouca quantidade de materiais para cada opção. Por último, temos o OrientDB, com opções da documentação e um curso grátis disponível, sendo poucas opções e material disponível.

No quesito de segurança, os 4 empatam, com nenhuma característica destacável para nenhum dos SGBDs analisados.

Para a interface de usuário, o ArangoDB fica em primeiro, devido ao fato de que possui funcionalidades exclusivas, como a visualização de informações referentes a utilização do banco de dados, além de possuir a maior parte das funcionalidades dos outros SGBDs. O OrientDB fica e segundo, pois sua interface não possui tantos recursos, porém ele se sobressai em relação aos outros dois, devido a algumas funcionalidades a mais disponíveis citadas anteriormente. Em terceiro, temos empatados, o Neo4j e o AgensGraph, isso pois, os dois fornecem poucos recursos de manipulação, e tem maior foco na visualização dos dados. Os recursos de suas interfaces são diferentes, porém, os dois não conseguem se destacar o suficiente para que um fique acima do outro nessa comparação.

Em questão de suporte, a Tabela 4.2 mostra as linguagens suportadas pelos SGBDs, com o Neo4j possuindo a maior quantidade de linguagens suportadas, seguido pelo ArangoDB, OrientDB e por último AgensGraph. Foram apenas consideradas as linguagens suportadas oficial-

mente, porém, o existem drivers alternativos outras linguagens.

Tabela 4.2: Linguagens suportadas oficialmente pelos SGBDs

SGBDs/Linguagens	.Net	Java	Spring	JavaScript	Go	Python	PHP	NodeJS
Neo4j	•	•	•	•	•	•		
OrientDB	•	•					•	
ArangoDB		•		•	•		•	•
AgensGraph		•				•		

4.2 Comparação de desempenho

Nesta seção, serão comparados os SGBDs Neo4j, OrientDB e ArangoDB, em termos de desempenhos nos testes básicos, operações paralelas, múltiplos núcleos, *workload*, uso de memória RAM e uso de armazenamento em disco. O AgensGraph foi excluído dessa análise devido ao seu abandono no desenvolvimento, apurado na análise documental.

4.2.1 Testes Básicos

Começando pela questão de como o tempo das operações básicas é afetado pelo crescimento no número de elementos da base, em geral, a maioria das operações se manteve constante com pouca ou nenhuma variação de tempo conforme novos elementos são adicionados a base, isso pode ser observado na Tabela 4.3. O OrientDB teve os melhores resultados para essas operações, porém, esse tempo está relacionado ao driver usado e pode variar de linguagem para linguagem, pois as operações em si tem custo próximo de nulo, por isso da não alteração no tempo de execução com o aumento da base de dados.

Para a operação de menor caminho, o resultado obtido mostra que a operação em si tem um tempo de execução próximo ao tempo das operações da Tabela 4.3, especialmente para o Neo4j, que manteve um tempo de execução abaixo de 15 ms para todos os testes. Já o OrientDB e ArangoDB tiveram variações maiores, porém, o tempo de execução ainda ficou abaixo de 30ms para a maior parte dos testes. Não foi possível observar o efeito do aumento da base de dados no tempo de execução da operação menor caminho, pois os resultados variam muito, e dependem mais dos vértices de entrada, do que do tamanho da base de dados. Os resultados obtidos com as execuções podem ser vistos na Tabela 4.4.

Tabela 4.3: Tempo de execução das operações básicas (ms)

Banco de dados	**Neo4j**		**ArangoDB**		**OrientDB**	
N° de elementos (mil)	200	2000	200	2000	200	2000
Add Vértice (ms)	8,83	12,66	9,66	9,16	7,2	7,4
Add Aresta (ms)	10,16	12,8	9,6	9,6	5	5,6
Alt Vértice (ms)	9,66	11,6	10,2	10	4,2	4
Del Vértice (ms)	8	11,16	9,15	9,2	3,6	3,6
Del Aresta (ms)	9,33	10	12,33	12,5	4,2	4,8
Busca Vértice (ms)	7,66	7,6	9,36	9,31	2,4	2,4
Busca Aresta (ms)	7,85	8,66	12,4	12,8	2,6	2,6
K-Vizinhos (ms)	9,06	9,2	12,4	12,8	2,6	2,6

Tabela 4.4: Tempo de execução da operação menor caminho (ms)

N° de elementos (mil)	**200**	**400**	**600**	**800**	**1200**	**1600**	**2000**
Neo4j (ms)	13,4	16,8	12,4	12,4	12,4	11,8	11,2
ArangoDB (ms)	19,2	24	26,6	21,6	28,2	32,8	22,4
OrientDB (ms)	22	15,8	15,2	19	33,2	27,2	42,8

Para a função de média, os resultados obtidos e apresentados na Figura 4.5, mostram um alto tempo de execução do OrientDB se comparado com os outros SGBDs, sendo esse tempo 5 vezes maior que o do ArangoDB e 22 vezes maior que o do Neo4j para 2 milhões de elementos. Comparando o Neo4j e ArangoDB separadamente, o Neo4j possui um tempo de execução 2,5 vezes menor que o ArangoDB para a base de 200 mil elementos, sendo que essa variação cresce para uma diferença de 4 vezes para a base de 2 milhões.

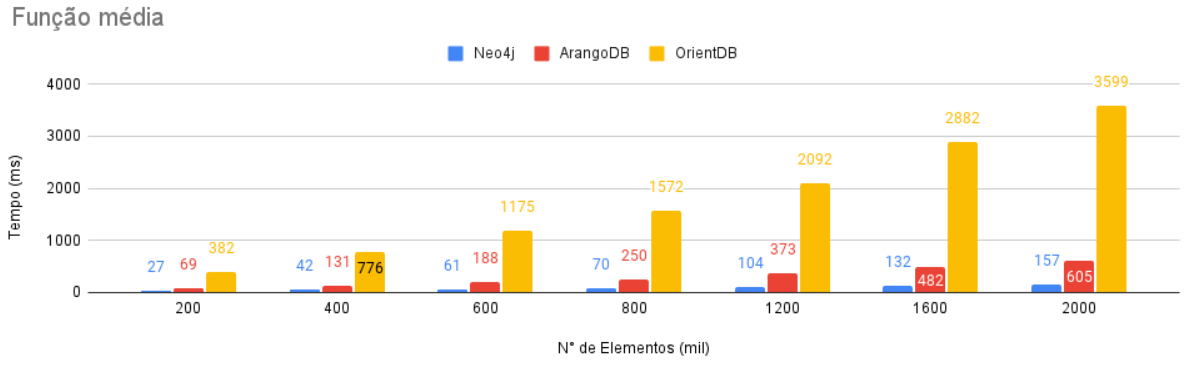

Figura 4.5: Função de média

A operação mais custosa nos testes básicos foi a busca com filtro, onde todos os SGBDs tiveram seu tempo de execução crescente conforme o crescimento da base como um todo. Esse

crescimento aparenta ser constante para todos, porém, o crescimento do OrientDB foi mais acentuado que o Neo4j e ArangoDB, como pode ser visto na Figura 4.6, sendo mais de 11 vezes maior que os dois, para 2 milhões de elementos. Isso demonstra uma grande dificuldade na busca de registros usando propriedades não únicas por parte do OrientDB se comparado com os outros SGBDs analisados.

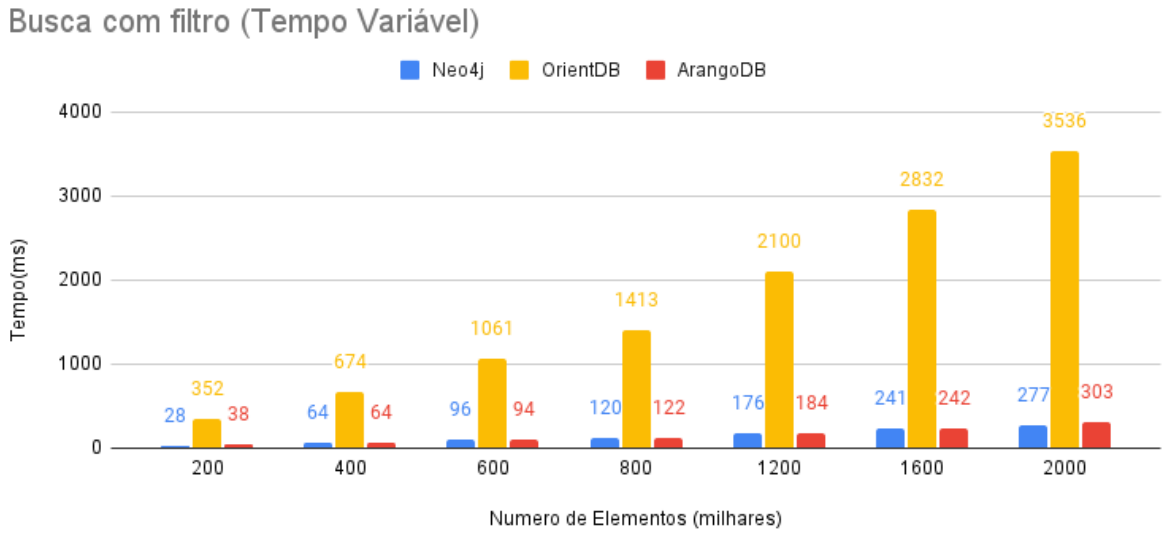

Figura 4.6: Busca com Filtro

Observando o custo da operação de busca com filtro entre o Neo4j e o ArangoDB, é possível visualizar que eles têm tempos de execução similares, com uma variação maior apenas no caso com 2 milhões de elementos.

Uma característica peculiar observada nos testes foram as operações de busca e deleção de arestas do OrientDB. Enquanto os outros SGBDs apresentaram pequenos tempos de execução para essas operação, o OrientDB obteve um tempo extremamente alto se comparado com as outras operações, ainda mais alto que a operação de busca com filtro, com valores chegando na casa de 6 a 7 mil milissegundos de tempo de execução. Porém, esse problema decorre do fato que o OrientDB permite a criação de arestas repetidas, ou seja, que conectam os mesmos vértices na mesma direção, pois ele não trata os atributos de origem e destino como únicos, o que faz com que seu tempo de execução suba acentuadamente. Esse problema pode ser solucionado simplesmente tornando os atributos de origem e destino como um índice único, o que faz com

que o tempo de execução de buscas por arestas diminua para o custo semelhante a busca por vértices como pode ser visto na Tabela 3.2.

4.2.2 *Workload*

Nos experimentos envolvendo o *workload* de vértices, é possível observar na Figura 4.7 que o OrientDB teve o menor tempo de execução, tanto para 10 mil quanto 100 mil, seguido pelo Neo4j, com uma diferença de aproximadamente 1000 ms, e o ArangoDB teve o pior desempenho para o caso de 100 mil, sendo quase o dobro que o OrientDB, porém, para 10 mil seu valor era muito próximo, mas essa diferença foi aumentando conforme o número de elementos foi crescendo. O crescimento do tempo de execução do Neo4j de 10 mil para 100 mil foi pouco mais de 2 vezes, sendo o menor aumento proporcional dos 3, seguido pelo OrientDB, que teve um aumento de pouco mais de 5 vezes, e por último, tivemos o ArangoDB, que teve um crescimento de mais de 7,5 vezes.

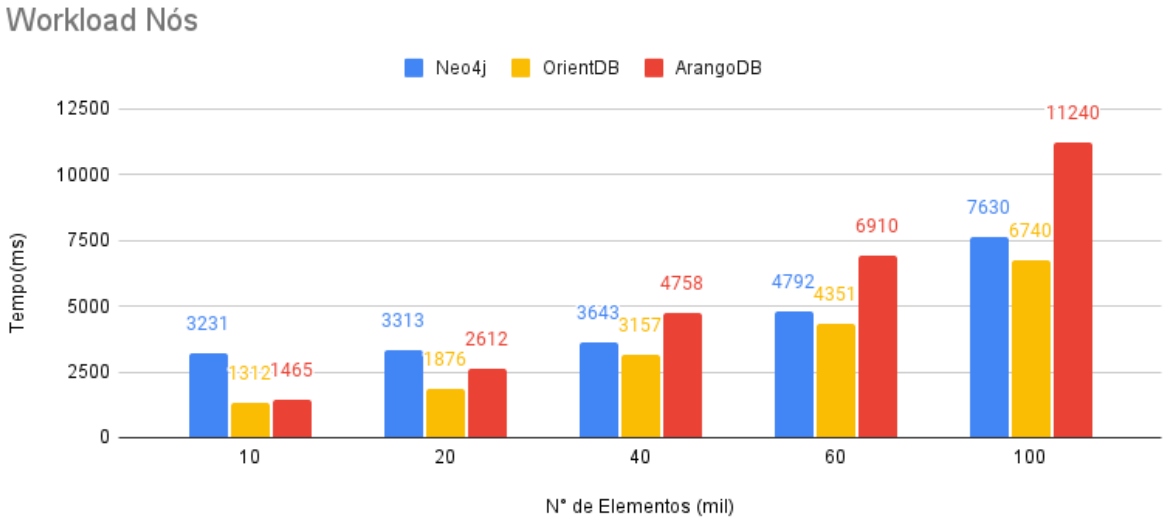

Figura 4.7: *Workload* dos nós

No caso do *workload* para arestas os resultados observados na Figura 4.8 mostram que, diferente do caso dos vértices, o OrientDB apresentou o pior resultado por uma grande margem, começando com uma diferença de aproximadamente 3 vezes ao Neo4j e que sobe para uma diferença de 20 vezes para 100 mil elementos, aparentando inclusive estar crescendo de forma exponencial, pois, proporcionalmente ao valor de tempo inicial, o OrientDB teve um

crescimento de mais de 26,5 vezes de 10 mil elementos para 100 mil, fazendo ser mais vantajoso executar 10 *workloads* de 10 mil do que 1 *workload* de 100 mil. Comparando o Neo4j e o ArangoDB separadamente, é possível observar que, inicialmente o ArangoDB possuía um melhor desempenho que o Neo4j, porém que é revertido para o conjunto de 20 mil em diante. Essa distância aumenta conforme se aumenta o número de elementos, dando uma vantagem ao Neo4j no caso das arestas. Em relação ao crescimento proporcional, o Neo4j teve um crescimento de 4 vezes do conjunto de 10 mil para o de 100 mil, sendo um aumento maior que para o caso de vértices, porém menor que os outros, enquanto o ArangoDB semelhante ao crescimento proporcional que teve com vértices, teve um crescimento de pouco mais de 7,5 vezes.

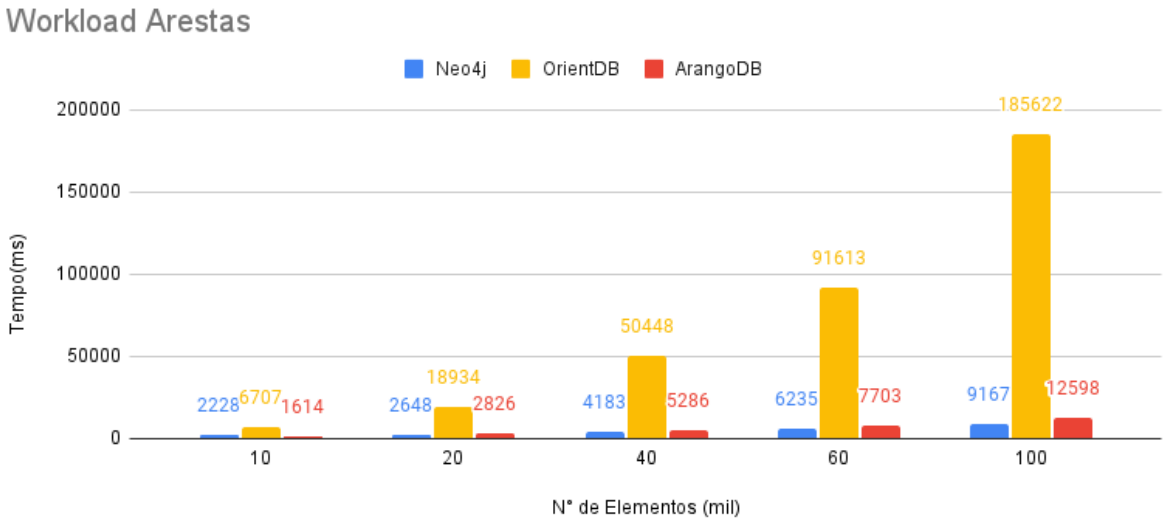

Figura 4.8: *Workload* das arestas

4.2.3 Uso de memória RAM

No quesito de memória RAM utilizado por cada SGBD, podemos observar na Figura 4.9 que o OrientDB utiliza a maior quantidade de memória dos três, e esse consumo aumenta em quase 50% depois de executado os testes, e quando executado o *workload*, seu valor aumentou mais de 3 vezes que o inicial, chegando a 5GB de memória RAM consumidos. O Neo4j vem em segundo, com um consumo de 1,1GB de memória, e que depois dos testes executados sobe um pouco mais de 100MB, sendo um aumento de aproximadamente 10%, e com a execução do *workload*, seu valor subiu pouco mais de 500MB, sendo um aumento de 40% no consumo de

memória RAM em relação ao estado inicial. Por fim, o ArangoDB tem o menor consumo de memória RAM de todos, sendo apenas 170MB quando inicializado e indo para 280MB depois de executados os testes. Esse aumento, mesmo que proporcionalmente grande (65%), ainda é pouco se comparado com o Neo4j e OrientDB em valores discretos. Curiosamente, quando executado o *workload*, o consumo de memória sobe menos que para as operações básicas, indo para apenas 200MB de memória RAM.

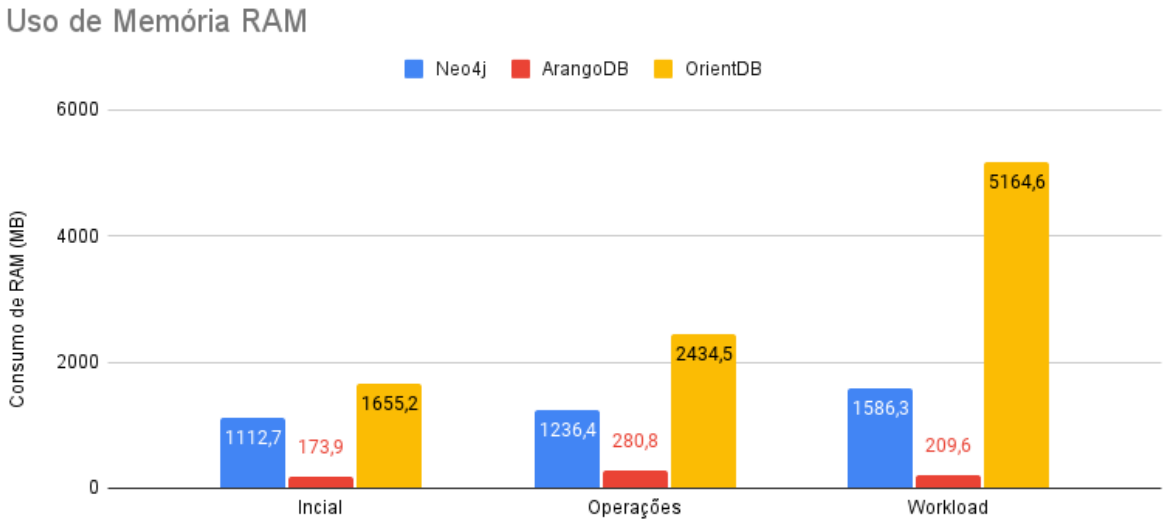

Figura 4.9: Uso de memória RAM

4.2.4 Uso de armazenamento em disco

No consumo de armazenamento de disco, podemos observar na Figura 4.10 que o Neo4j consome a maior quantidade de memória em disco, com 4,6GB para armazenar os dados do banco de dados, mais de 8 vezes maior que os outros SGBDs analisados, e 168MB para uma base de dados vazia. Devido ao ArangoDB não possuir um local específico para cada banco de dados não foi possível saber precisamente seu tamanho, porém, esse tamanho é menor que 512MB pois esse é o total de memória ocupada por todos os bancos de dados do ArangoDB. Não foi possível inferir quanto um banco de dados vazio tem de tamanho para o ArangoDB. Por último temos o OrientDB, que ocupa 570MB em disco e apenas 1,52MB vazio. O OrientDB e ArangoDB possuem um consumo de disco muito semelhante, porém, o ArangoDB possui

vantagem, pois mesmo considerando todos os bancos, ele ainda possui um valor menor que o OrientDB.

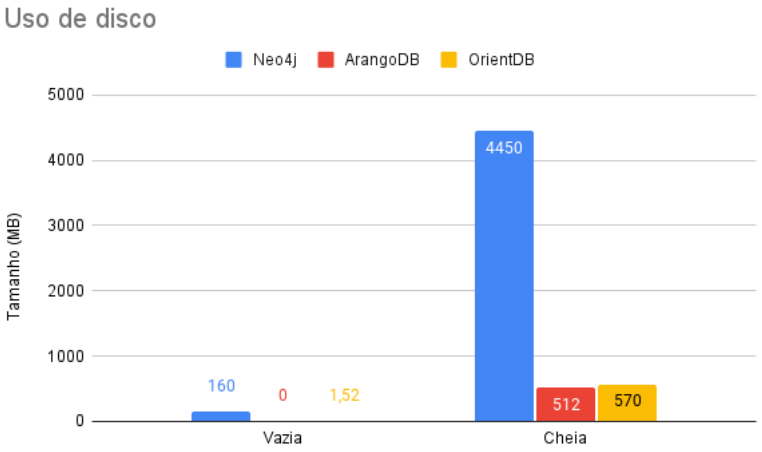

Figura 4.10: Uso de disco

4.2.5 Operações Paralelas

No caso dos testes com operações paralelas, os resultados podem ser observados no Figura 4.11, onde é possível visualizar que o OrientDB teve o pior resultado em relação ao tempo total gasto para responder a todas as requisições, sendo seu tempo 32 vezes maior que o Neo4j e ArangoDB.

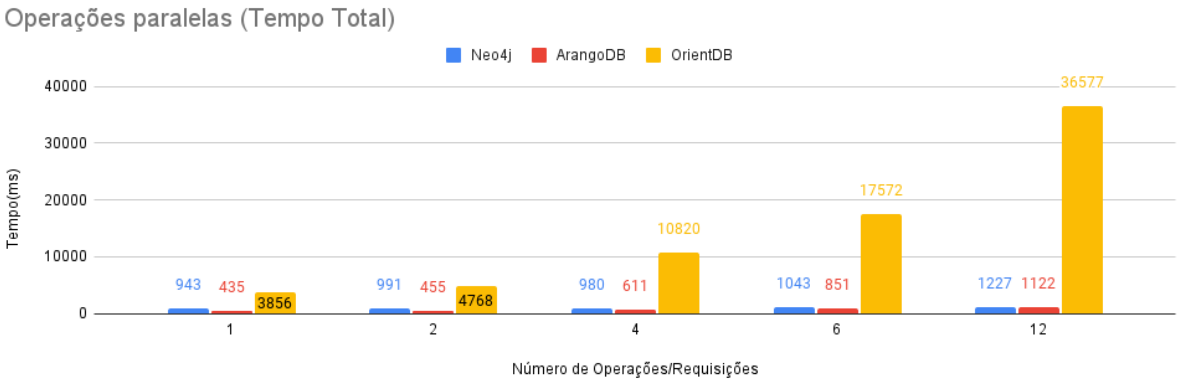

Figura 4.11: Múltiplos Usuários (Tempo Total)

Comparando o Neo4j e ArangoDB separadamente, é possível observar que o Neo4j possui um tempo maior de execução para uma única requisição, porém que diminui sua diferença conforme o número de requisições aumenta, essas diminuições ocorrem pois o tempo do Neo4j cresce pouco proporcionalmente, em comparação com o ArangoDB, o que os deixa bem próximos no teste para 12 operações.

Analisando cada SGBD individualmente nesse teste, começando pelo Neo4j, é possível observar na Figura 4.12, que o tempo de operação e tempo total se mostram praticamente constantes, mesmo aumentando o número de requisições de 1 para 12, com um aumento de 30% de 1 operação para 12. Enquanto o custo sequencial cresce de forma linear, o que era esperado. Isso mostra que o Neo4j lida bem com operações paralelas, aproveitando os múltiplos núcleos da máquina.

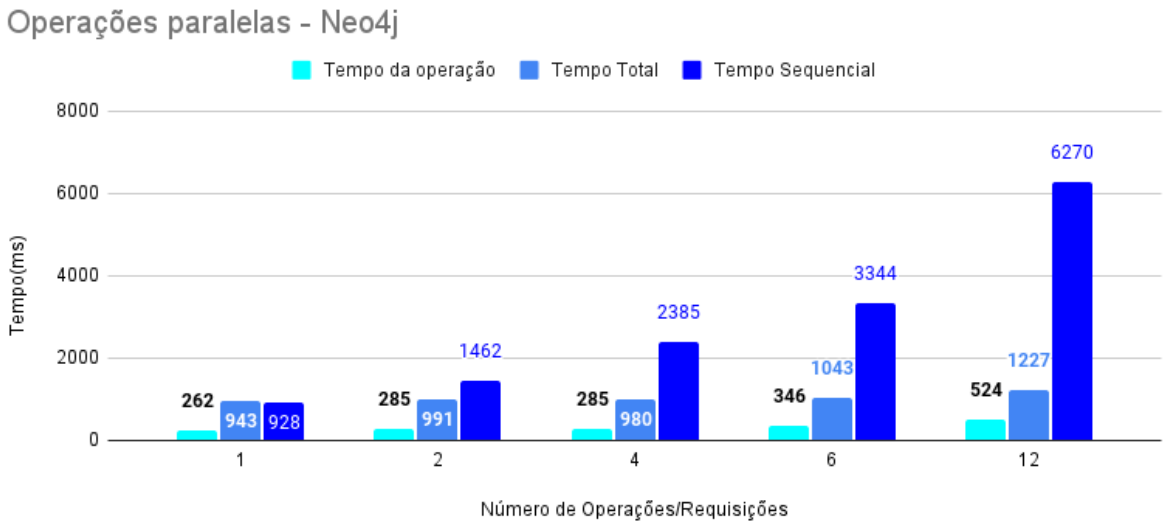

Figura 4.12: Múltiplos Usuários - Neo4j

No caso do ArangoDB, é possível observar um cenário semelhante ao do Neo4j como pode ser visto na Figura 4.13, porém o crescimento proporcional de 1 para 12 operações é maior, sendo aproximadamente 2,5 vezes maior. É possível observar também que o tempo de operação aumenta conforme aumenta o número de requisições, esse crescimento não é brusco, porém é notável, sendo mais de 2 vezes maior que o valor inicial. Isso não impede que o ArangoDB possua um melhor desempenho que o tempo sequencial, com um tempo total 3 vezes menor que o sequencial.

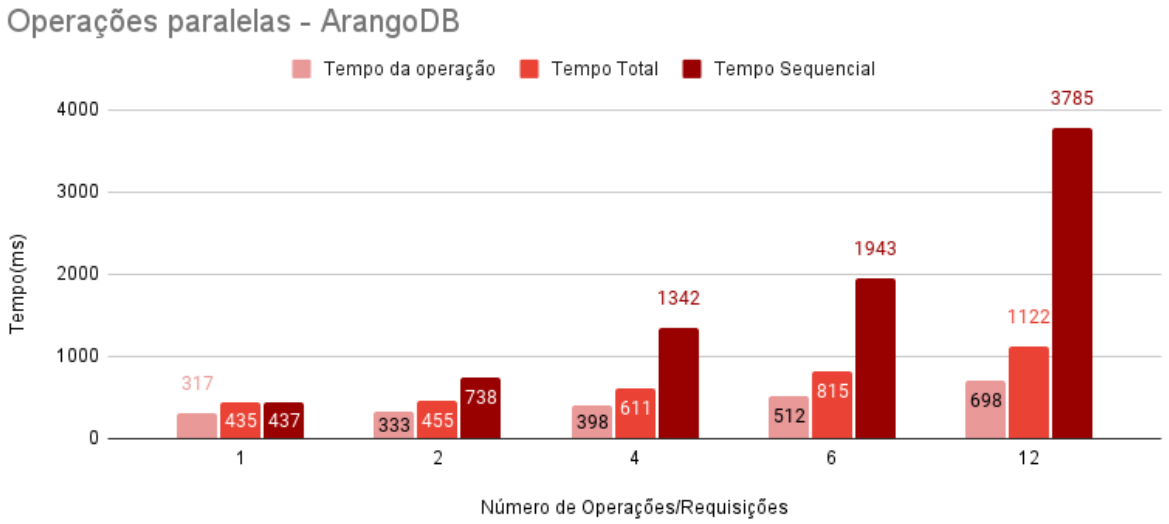

Figura 4.13: Múltiplos Usuários - ArangoDB

Por último temos o OrientDB, onde é possível observar pela Figura 4.14 que o crescimento dos tempos de operação, total e sequencial são praticamente paralelos, o que mostra que o OrientDB possui pouca melhoria de desempenho do caso sequencial e paralelo.

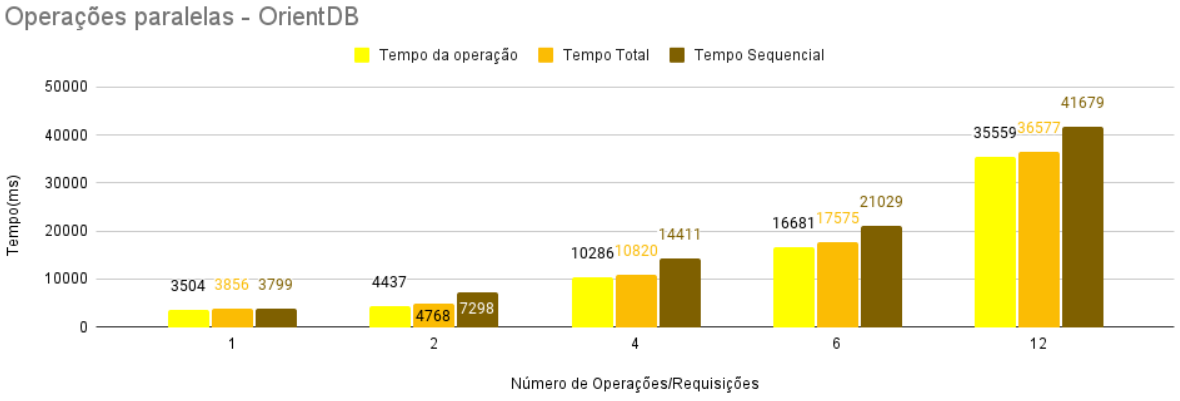

Figura 4.14: Múltiplos Usuários - OrientDB

O maior problema é o tempo de operação aumentando de forma proporcional aos outros dois, isso indica que no caso dos 12 usuários, todos eles deveram esperar 36 segundos para as respostas de suas requisições, isso significa que se fosse utilizado um sistema de fila, onde se

espera finalizar a requisição de uma para ir para a próxima requisição, o tempo final seria muito semelhante, porém o primeiro usuário receberia sua resposta em 3 segundos, o segundo em 6 segundos e assim por diante, e apenas o último teria que esperar 36 segundos. No fim, isso gera pouca vantagem no geral, pois em vez de apenas um usuário ter que esperar mais, todos devem esperar o mesmo tempo, sendo esse tempo o maior possível.

4.2.6 Múltiplos Núcleos

No teste com múltiplos núcleos se observou que não existe diferença significativa entre usar 1 núcleo apenas ou mais para todos os SGBDs analisados e para todas as operações básicas testadas, mesmo as operações de busca pelo menor caminho e busca com filtro. Ou seja, para operações individuais não existe ganho de desempenho pelo maior número de núcleos, fazendo com que para se aproveitar os múltiplos núcleos de uma máquina, seja necessário programar de forma específica para aproveitar os múltiplos núcleos. Ganho de desempenho por múltiplos núcleos, de acordo com nossos testes e resultados, apenas ocorre em situações de operações paralelas sendo processadas pelo banco de dados, e isso apenas serve para o Neo4j e ArangoDB, não se aplicando ao OrientDB.

4.3 Comparação com Trabalhos Relacionados

Comparando esses resultados obtidos com outros trabalhos, verificamos as seguintes situações: Começando pelo trabalho de Lissandrini *et al.*, vemos que os resultado obtidos na comparação de desempenho se assemelham entre os trabalhos, com os melhores resultados para operações básicas e o workload, ao qual eles chamam de loading, sendo na sua maior parte do Neo4j e ArangoDB, se comparado com o OrientDB. Algumas maiores diferenças, como o pior desempenho do ArangoDB em operações de buscas, podem ser atribuídas ao fato que o trabalho é de 2018, foi executado em um sistema operacional linux (Ubuntu) e com a linguagem Python e um HD com 2000 rotações por minuto (RPM), além de algumas pré-configurações que podem ter afetado o desempenho nesses casos.

Também temos o trabalho de Beis (BEIS; PAPADOPOULOS; KOMPATSIARIS, 2015) que comparou o OrientDB e Neo4j, apresentou os mesmos resultados aqui vistos, com o Neo4j a frente do OrientDB para operações de workload e caminhamento no grafo (K-Vizinhos e Menor

Caminho). O trabalho de Matyjaszcyk *et al.* (MATYJASZCZYK; ROSOWSKI; WREMBEL, 2020), que compara o ArangoDB e o OrientDB e alguns outros SGBDs, também mostrou melhores resultados para o ArangoDB para a maioria das operações, em relação ao OrientDB. Os trabalhos de Kolomičenko, tanto sua dissertação (KOLOMIČENKO, 2013), quando seu outro artigo (KOLOMIČENKO; SVOBODA; MLÝNKOVÁ, 2013), também apresentaram resultados semelhantes em relação ao Neo4j comparado ao OrientDB, para operações de *workload*, a qual ele também chama de *loading*, e operações de caminhamento pelo grafo.

Salim (JOUILI; VANSTEENBERGHE, 2013) também desenvolveu testes entre diferentes SGBD, incluindo o Neo4j e OrientDB, e também fez testes com múltiplos clientes/usuários (operações paralelas), seus testes tiveram resultados diferentes possivelmente devido ao fato que a máquina a qual foi testada possuía apenas 2 núcleos e devido ao fato que os testes em paralelo foram executados como conjuntos de operações, não de operações individuais, além do fato que o ganho de desempenho pode ter advindo de questões de paralelismo de operações relacionadas ao driver utilizado nos testes.

Capítulo 5

Conclusão

Esse trabalho teve como objetivo auxiliar os desenvolvedores e projetistas de banco de dados, a escolherem a melhor opção de SGBD baseado em grafo para seus projetos, com essa escolha pautada em aspectos técnicos e de desempenho. Para alcançar esse objetivo, realizou-se uma análise documental e uma comparação de desempenho dos SGBDs Neo4j, ArangoDB, OrientDB e AgensGraph.

Primeiramente, é importante ressaltar que os bancos de dados orientado a grafos tem foco principal em base de dados grandes e altamente conectadas, e em funções de caminhamento por grafos, porém, isso não significa que esse modelo não pode ser usado em cenários mais comuns, onde o volume de dados não é alto e não há muitos relacionamentos entre esses dados. Principalmente quando comparados com os banco de dados relacionais, os orientado a grafos podem ser uma interessante alternativa, pois, além de fornecerem um conjunto similar de recursos, operações e características, eles também oferecem propriedades extras que podem ser relevantes para o desenvolvimento futuro do banco de dados. Em caso de bases com características mais especificas, é necessário avaliar outros modelos disponíveis para escolher o que melhor atende as necessidades desse conjunto.

Depois de feita todas as análises e comparações, é possível concluir que o Neo4j e o ArangoDB são as melhores opções para o armazenamento e manipulação de dados em grafos de forma rápida e confiável, dos SGBDs aqui comparados. Com a análise documental foi possível observar que os dois possuem uma boa maturidade, facilidade de programar e visualização, mesmo que o ArangoDB perca um pouco no aspecto de comunidade e o Neo4j em visualização, os dois se mostraram as melhores opções para o uso geral. Já na questão de desempenho, os dois se mostraram muito semelhantes em tempo de execução das operações básicas e *workload*

de nós e arestas. O Neo4j ficou atrás nos quesitos de uso de memória RAM e memória de disco, porém ficou na frente em operações paralelas, aproveitando melhor os recursos do sistema para essa situação. Em geral, os dois tiveram seus prós e contras na questão de desempenho, com uma maior vantagem para o Neo4j, devido a seus tempos de execução menores e sua melhor escalabilidade em relação ao crescimento do conjunto. No fim, os dois se apresentaram como as melhores soluções de SGBDs orientado a grafos.

O OrientDB se mostrou uma opção menos interessante que as duas anteriores. Seus resultados na análise documental mostraram-se pouco animadores, devido a questão de maturidade e, especialmente, a facilidade de programar, pois a falta de opções para o aprendizado dificulta o uso da ferramenta para os desenvolvedores menos experientes. Na questão de desempenho, o OrientDB possui um menor tamanho em disco e mais velocidade de *workload* de vértices, porém seu altíssimo tempo de execução para arestas, maior tempo na execução de consultas de busca com filtro e função média, e desempenho ruim para operações paralelas, além de alto consumo de memória RAM, o tornaram uma opção muito abaixo se comparado com as duas anteriores, mesmo para base de dados pequenas ou projetos menores.

O AgensGraph por seu estado abandonado atual se mostra a opção menos interessante das analisadas, pois sua situação pode gerar diversos problemas futuros com falta de atualizações necessárias para manter o software seguro e otimizado para novos hardwares e recursos. Por isso ele se mostra como o SGBD menos recomendado dos analisados nesse trabalho e não participou da análise de desempenho.

Também se observou que operações individuais não são afetadas por múltiplos núcleos, ou seja, caso o SGBD não esteja aproveitando os múltiplos núcleos de outra forma, como por exemplo pelo processamento de operações em paralelo como visto nos resultados, os núcleos a mais não serão usados pelo SGBD. Isso significa que aqueles que forem utilizar qualquer uma das opções de SGBDs aqui apresentados devem ter isso em mente, que, o SGBD não tirará vantagem automaticamente dos seus múltiplos núcleos, e que é responsabilidade do programador encontrar outros meios de aproveitar os núcleos da máquina.

Um aspecto que todos os SGBDs analisados na análise de desempenho apresentaram, é boa performance com operações de caminhamento em grafos. Essas operações podem ser custosas caso não otimizadas corretamente, porém, nos testes foi possível observar que todos eles tiveram

baixo tempo de execução para a operação de K-Vizinhos e Menor Caminho, mesmo que a última seja uma operação custosa de ser executada, e tudo isso utilizando apenas um núcleo do processador. Isso mostra que esses SGBDs estão otimizados para esse tipo de operação, focada em grafos, como se era esperado de um SGBD orientado a grafos.

5.1 Trabalhos Futuros

Os testes desse trabalho se focaram principalmente em um base de dados simples, com isso, é possível se executar testes semelhantes para base de dados mais complexas, com mais tipos de nós e arestas, e com esses dados possuindo diferentes propriedades, além de testar com maior número de elementos (nós e arestas) no total. É também possível testar com mais SGBDs, e em um sistema operacional Linux, onde existem mais opções de SGBDs disponíveis e os tempos de execução podem variar se comparado com o Windows.

Para futuros trabalhos também podem ser trabalhadas diferentes linguagens de programação, que possuam drivers oficiais para os testes, além de testar mais afundo funções como menor caminho, com a criação de diferentes bases de dados para verificar como os SGBDs se comportam em relação a essa operação para base de dados muito e pouco conectadas e com arestas com pesos, além de variações nos parâmetros das operações.

Uma parte que esse trabalho não abordou são as outras formas de aproveitar os múltiplos núcleos de um processador pelo SGBD, isso inclui funções de paralelismo para operações individuais, inserções em massa e outras possíveis formas que podem diminuir o tempo das operações ao banco de dados, por meio de múltiplos núcleos.

Referências Bibliográficas

AGENSGRAPH. *AgensGraph*. 2021. Disponível em: <https://bitnine.net/agensgraph/>. Acesso em: 29 mar 2021.

ALI, A.; ABDULLAH, M. A survey on vertical and horizontal scaling platforms for big data analytics. *International Journal of Integrated Engineering*, v. 11, p. 138–150, 09 2019.

ANGLES, R. A comparison of current graph database models. In: *2012 IEEE 28th International Conference on Data Engineering Workshops*. [S.l.: s.n.], 2012. p. 171–177.

ANGLES, R.; GUTIERREZ, C. Survey of graph database models. *ACM Comput. Surv.*, New York, NY, USA, v. 40, n. 1, fev. 2008.

ARANGODB. *ArandoDB*. 2021. Disponível em: <https://www.arangodb.com>. Acesso em: 07 mar 2021.

BEIS, S.; PAPADOPOULOS, S.; KOMPATSIARIS, Y. Benchmarking graph databases on the problem of community detection. In: *New Trends in Database and Information Systems II*. Cham: Springer International Publishing, 2015. p. 3–14.

BESTA, M. et al. *Demystifying Graph Databases: Analysis and Taxonomy of Data Organization, System Designs, and Graph Queries*. 2020.

CHEN, Y. *Comparison of Graph Databases and Relational Databases When Handling Large-Scale Social Data*. Dissertação (Dissertação de Mestrado) — University of Saskatchewan, Saskatoon - Saskatchewan, Setembro 2016.

CONNOLLY, C. B. T. *Database Systems: A Practical Approach to Design, Implementation, and Management*. 6. ed. [S.l.]: Pearson, 2014.

DB-ENGINE. *DB-Engines Ranking*. 2020. Disponível em: <https://db-engines.com/en/ranking>. Acesso em: 16 nov 2020.

DB-ENGINE. *DBMS popularity broken down by database model*. 2020. Disponível em: <https://db-engines.com/en/ranking_categories>. Acesso em: 25 nov 2020.

EL-REWINI, H.; ABD-EL-BARR, M. *Advanced Computer Architecture and Parallel Processing*. [S.l.]: John Wiley & Sons, 2005.

ELMASRI, R.; NAVATHE, S. B. *Fundamentals of Database Systems*. 7. ed. [S.l.]: Pearson, 2015.

FRISENDAL, T. *Property Graphs: The Swiss Army Knife of Data Modeling*. 2017. Disponível em: <http://www.dataversity.net/property-graphs-swiss-army-knife-data-modeling/>. Acesso em: 22 mar 2021.

GOOGLE. *Colaboratory*. 2021. Disponível em: <https://colab.research.google.com/notebooks/intro.ipynb>. Acesso em: 27 mar 2021.

GOOGLE. *Google Groups*. 2021. Disponível em: <https://groups.google.com/>. Acesso em: 19 mai 2021.

GRAMS, A. M. *Graph Database Benchmark*. 2021. Disponível em: <https://github.com/adrianograms/Graph_Database_Benchmark>. Acesso em: 20 jun 2021.

HAERDER, T.; REUTER, A. Principles of transaction-oriented database recovery. *ACM Comput. Surv.*, New York, NY, USA, v. 15, n. 4, p. 287–317, dez. 1983.

IORDANOV, B. Hypergraphdb: A generalized graph database. In: *Web-Age Information Managements*. [S.l.: s.n.], 2010. p. 25–36.

JOUILI, S.; VANSTEENBERGHE, V. An empirical comparison of graph databases. In: *2013 International Conference on Social Computing*. [S.l.: s.n.], 2013. p. 708–715.

KOLOMIČENKO, V. *Analysis and Experimental Comparison of Graph Databases*. Dissertação (Dissertação de Mestrado) — Charles University, Praga - República Tcheca, 2013.

KOLOMIČENKO, V.; SVOBODA, M.; MLÝNKOVÁ, I. H. Experimental comparison of graph databases. In: *Proceedings of International Conference on Information Integration and Web-Based Applications & Services*. New York, NY, USA: Association for Computing Machinery, 2013. p. 115–124.

LEAVITT, N. Will nosql databases live up to their promise? *Computer*, v. 43, p. 12 – 14, 03 2010.

LESKOVEC, J.; KREVL, A. *SNAP Datasets: Stanford Large Network Dataset Collection*. 2014. Disponível em: <http://snap.stanford.edu/data>. Acesso em: 12 jun 2021.

LISSANDRINI, M.; BRUGNARA, M.; VELEGRAKIS, Y. Beyond macrobenchmarks: Microbenchmark-based graph database evaluation. *Proc. VLDB Endow.*, VLDB Endowment, v. 12, n. 4, p. 390–403, dez. 2018. ISSN 2150-8097.

MATYJASZCZYK, P.; ROSOWSKI, P.; WREMBEL, R. Goodbye: a good graph database benchmark - an industry experience. In: *EDBT/ICDT Workshops*. [S.l.: s.n.], 2020.

MILLER, J. J. Graph database applications and concepts with neo4j. *SAIS 2013Proceedings*, v. 24, 2013.

MILLER, R. *SAP snags CallidusCloud for $2.4 billion*. 2018. Disponível em: <https://techcrunch.com/2018/01/30/sap-snags-calliduscloud-for-2-4-billion/>. Acesso em: 20 mai 2021.

NEO4J. *Community Neo4j*. 2021. Disponível em: <https://community.neo4j.com/>. Acesso em: 19 mai 2021.

NEO4J. *Neo4j*. 2021. Disponível em: <https://neo4j.com>. Acesso em: 20 jun 2021.

ORIENTDB. *OrientDB*. 2021. Disponível em: <https://www.orientdb.org>. Acesso em: 07 mar 2021.

ORIENTDB. *OrientDB Community*. 2021. Disponível em: <https://discourse.orientdb.org/>. Acesso em: 19 mai 2021.

OVERFLOW, S. *Stack Overflow*. 2021. Disponível em: <https://stackoverflow.com/>. Acesso em: 19 mai 2021.

PAVLO, A.; ASLETT, M. What's really new with newsql? *SIGMOD Rec.*, New York, NY, USA, v. 45, n. 2, p. 45–55, set. 2016.

PICART, V. *CallidusCloud Acquires Leading Multi-Model Database Technology*. 2017. Disponível em: <https://www.globenewswire.com/news-release/2017/09/19/1124824/0/en/CallidusCloud-Acquires-Leading-Multi-Model-Database-Technology.html>. Acesso em: 19 mai 2021.

PRITCHETT, D. Base: An acid alternative: In partitioned databases, trading some consistency for availability can lead to dramatic improvements in scalability. *Queue*, New York, NY, USA, v. 6, n. 3, p. 48–55, maio 2008.

ROBINSON, I.; WEBBER, J.; EIFREM, E. *Graph Databases New Opportunities for Connected Data*. 2. ed. [S.l.]: O'Reilly Media, 2015.

ROBINSON JIM WEBBER, E. E. I. *Graph Databases: New Opportunities for Connected Data*. 2. ed. [S.l.]: O'Reilly Media, 2015.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. *DATABASE SYSTEM CONCEPTS*. 7. ed. [S.l.]: McGraw-Hill Education, 2020.

SLACK. *Slack*. 2021. Disponível em: <https://slack.com/>. Acesso em: 19 mai 2021.

VOGELS, W. Eventually consistent. *Commun. ACM*, Association for Computing Machinery, New York, NY, USA, v. 52, n. 1, p. 40–44, jan. 2009.

W3C. *Resource Description Framework (RDF): Concepts and Abstract Syntax*. 2004. Disponível em: <https://www.w3.org/TR/rdf-concepts/>. Acesso em: 22 mar 2021.