Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnologia
Curso de Ciência da Computação

FLAVIANE DOS SANTOS OLIVEIRA

ANÁLISE COMPARATIVA DE UM BANCO DE DADOS RELACIONAL E UM BANCO DE DADOS EM GRAFOS

São Luís
2019

FLAVIANE DOS SANTOS OLIVEIRA

ANÁLISE COMPARATIVA DE UM BANCO DE DADOS RELACIONAL E UM BANCO DE DADOS EM GRAFOS

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Profº Me. Carlos Eduardo Portela Serra de Castro

São Luís

2019

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Núcleo Integrado de Bibliotecas/UFMA

Oliveira, Flaviane dos Santos.
Análise Comparativa de um Banco de Dados Relacional e um Banco de Dados em Grafos / Flaviane dos Santos Oliveira. – 2019.
47 f.

Orientador(a): Carlos Eduardo Portela Serra de Castro.
Monografia (Graduação) – Curso de Ciência da Computação, Universidade Federal do Maranhão, Auditório DEINF CCET UFMA, 2019.

1. Banco de Dados Grafos. 2. Cypher. 3. Linguagem de Consulta. 4. Neo4j. I. Serra de Castro, Carlos Eduardo Portela. II. Título.

FLAVIANE DOS SANTOS OLIVEIRA

ANÁLISE COMPARATIVA DE UM BANCO DE DADOS RELACIONAL E UM BANCO DE DADOS EM GRAFOS

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 11 de Abril de 2019

BANCA EXAMINADORA

Profº Me. Carlos Eduardo Portela Serra de Castro (Orientador)
Universidade Federal do Maranhão

Profº Dr. Carlos de Salles Soares Neto
Universidade Federal do Maranhão

Profº Me. Caio Eduardo Falcão Matos
Universidade Federal do Maranhão

São Luís
2019

A minha mãe Osmarina, com toda minha gratidão, por tudo que fez por mim ao longo da minha vida.

Ao meu eterno companheiro, Aitan, pela dedicação, paciência e carinho para comigo.

Agradecimentos

Agradeço primeiramente a Deus por ter me dado o sustento e minha base de apoio;

As meus pais, Osmarina Mendes dos Santos e João Batista do Santos Oliveira, por sempre me mostrarem que a educação é o caminho para realização de sonhos maiores. Em especial a minha mãe que em momento algum deixou que desistisse dessa caminhada e me apoiou incondicinalmente em todas as decisões por mim tomadas;

Ao meu companheiro, Aitan, por segurar a minha mão e não ter soltado nunca mais, ajudando assim, a levantar em cada tombo da vida;

A minha filha, Alice, por me alegrar todos os dias e fazer enxergar a vida de uma maneira mais leve.

Ao meu irmão, Flávio, pelos conselhos paternos e por ser meu apoio em todos os momentos da minha vida.

Ao meu orientador, Carlos Eduardo Portela, pela oportunidade concedida, por não ter desistido de mim em nenhum momento, pela paciência em corrigir meus textos, por responder sempre com celeridade a todos meus questionamentos e por todo o apoio dado;

À todos os meus amigos que estiveram ao meu lado durante esta jornada, me apoiando e me proporcionando momentos de descontração e diversão. Um agradecimento especial aos amigos Igor Leonardo, Márcio Fernando, Benedito Mendes, Carlos Augusto e Samir Fahd, por me ensinarem que podemos ser o que quiser e mesmo assim estarmos juntos;

A todos os professores que fizeram parte de toda essa trajetória e me deram a chance de poder concluir meu sonho.

RESUMO

Palavras-chave: Banco de Dados Grafos, Linguagem de Consulta, Neo4J, Cypher.

Os bancos de dados são coleções de informações, que são especialmente organizadas para busca e recuperação rápida. Para isso, são estruturados de forma a facilitar o armazenamento, recuperação, alteração e exclusão de dados em um conjunto, por meio de operações de processamento. Durante muito tempo a modelagem relacional prevaleceu para a organização dessas bases de dados. Contudo, com as evoluções tecnológicas, surgimento de novas aplicações e o aumento considerável dos volumes de dados, foram surgindo outras metodologias de modelar os bancos, chamadas Not only SQL. Dentre as modelagens não SQL, os bancos de dados grafos tem ganhado destaque por apresentar bons desempenho no processamento de grande volumes de dados conectados. Desta forma, neste trabalho é realizada uma análise comparativa entre os bancos de dados grafos e relacionais, apresentando as principais semelhanças e diferenças entre eles, além de evidenciar os pontos positivos e negativos de cada um, através de um estudo de caso.

ABSTRACT

Keywords: Graph Database, Query Language, Neo4j, Cypher.

Databases are collections of information, which are specially organized for search and fast retrieval. For this, they are structured in order to facilitate the storage, recovery, alteration and exclusion of data in a set, through processing operations. For a long time relational modeling prevailed for the organization of these databases. However, with the technological developments, the emergence of new applications and the considerable increase of the data volumes, other methodologies of modeling the databases, called NoSQL, appeared. Among NoSQL modeling, graph databases have gained prominence because they perform well in processing large volumes of connected data. Thus, in this work a comparative analysis between the graph databases and relational databases is presented, presenting the main similarities and differences between them, besides evidencing the positive and negative points of each one, through a case study.

Lista de ilustrações

Lista de listagens

Lista de tabelas

Lista de abreviaturas e siglas

ACID	Atomicidade, Consistência, Isolamento e Durabilidade
BD	Banco de Dados
BDG	Banco de Dados Grafos
BDR	Banco de Dados Relacional
CMS	Content Management System
FBC	Filtragem Baseada em Conteúdo
FC	Filtragem Colaborativa
IP	Internet Protocol
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
MAC	Media Access Control
NoSQL	Not Only SQL
SGBD	Sistemas Gerenciadores de Banco de Dados
SGBDR	Sistemas Gerenciadores de Banco de Dados Relacionais
SQL	Structured Query Language
SR	Sistema de Recomendação
UML	Unified Modeling Language

Sumário

1 Introdução

Com o surgimento das redes sociais e a crescente comercialização por meio da Internet (e-commerce) os Bancos de Dados tomaram volumes notáveis (PENTEADO et al., 2014). Como exemplo de redes sociais pode-se citar o Twitter e o Facebook, as duas oferecem os serviços de recomendação de amigos próximos e inclusão de hashtags, operações muito complexas e demoradas quando realizadas em uma base de dados relacional (ROBINSON; WEBBER; EIFREM, 2015). Onde uma base de dados relacional nada mais é do que uma coleção de dados que possuem relacionamentos dentro de um sistema de informação.

Banco de Dados Relacionais estão enfrentando mais dificuldades para acomodar tendências, como a inclusão de hashtags. Por outro lado, Bancos de Dados Orientados a Objeto, Banco de Dados Hierárquicos e muitos outros não inserem nada de novo no contexto atual. Porém, nos últimos anos houve o surgimento da Base de Dados NoSQL, a qual se utiliza do grafo para construir modelagem de estruturas complexas e onde os dados são desnormalizados para obter-se um tempo de latência menor (MPINDA; FERREIRA; SANTOS, 2015).

Segundo (CUNG; JEDIDI, 2014), um modelo de dados em grafo, apesar de ser algo relativamente novo, possui muitas soluções para melhorar aplicações que tenham uma enorme quantidade de dados com um elevado grau de correlação. A principal vantagem dos modelos de dados grafos é que o relacionamento/ligação entre os elementos/entidades é o núcleo do modelo. Além disso, grafos também tem um elevado grau de flexibilidade e escalabilidade. Isso significa que, em uma era, onde a quantidade de dados é tão grande e onde a internet está enfrentando uma sobrecarga, bases de dados grafos propõem algumas soluções para gerenciar e armazenar esses dados.

O modelo de dados grafo demonstra ser uma abordagem promissora para implementação de aplicações com grande volume de dados, tanto por sua flexibilidade e escalabilidade quanto pela sua capacidade de descrever relacionamentos, já que eles utilizam dados sem esquema e trocam a consistência tradicional por outras propriedades úteis.

O tempo de resposta menor é a meta que se quer alcançar em sistemas informatizados com grandes volumes de dados interligados e podemos dizer que isto é o “calcanhar de Aquiles” em um Banco de Dados Relacional.

De acordo com (PENTEADO et al., 2014) apesar de todos os benefícios em se utilizar um SGBDR, aplicações baseadas em modelos de dados complexos podem arcar

com problemas decorrentes desta adaptação ao modelo relacional, especialmente para bancos de dados que devem dar suporte a um grande volume de dados, de requisições e de usuários concorrentes. 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.

Assim sendo, este trabalho se propõe a realizar um estudo comparativo entre SGBD (Sistemas Gerenciadores de Banco de Dados) relacionais e grafos para aplicações que possuem alto grau de correlação entre os dados.

1.1 Objetivos

O objetivo geral desta monografia é realizar uma análise comparativa entre Banco de Dados em Grafos e Banco de Dados Relacionais, utilizando para o estudo os SGBD (Sistemas Gerenciadores de Banco de Dados) Relacional: MySQL e SGBD Grafo: Neo4j. Os objetivos específicos são:

- Apresentar os principais conceitos de modelagem relacional: construção, relacionamentos, propriedades, utilização, armazenamento e gerenciamento;
- Apresentar os principais conceitos de modelagem orientada a grafos: construção, relacionamentos, propriedades, utilização, armazenamento e gerenciamento;
- Analisar uma aplicação utilizando banco de dados relacional e sua diferenciação de um banco de dados orientado a grafos.

1.2 Organização do Trabalho

Este trabalho está organizado da seguinte forma: O Capítulo 2 apresenta os principais conceitos e tecnologias a cerca de banco de dados relacional e banco de dados NoSQL.O capítulo 3 apresenta um comparativo entre as principais características do Banco de Dados em Grafos e Banco de Dados Relacional, referente as ferramentas utilizadas durante este trabalho. Capítulo 4 apresenta um estudo de caso, realizando comparações entre as tecnologias temas desta monografia. O Capítulo 5 apresenta as conclusões sobre o estudo apresentado e propõe trabalhos futuros.

2 Referencial Teórico

Neste capítulo são descritos conceitos sobre Banco de Dados Relacional, Banco de Dados NoSQL e suas estruturas.

2.1 Banco de Dados Relacional

De acordo com (ELMASRI; NAVATHE, 2011) os bancos de dados e os sistemas de bancos de dados se tornaram componentes essenciais no cotidiano da sociedade moderna. No decorrer do dia, a maioria das pessoas se deparam com atividades que envolvem alguma interação com esses sistemas. Por exemplo, ao efetuar um depósito ou saque no banco, ao realizar reservas em um hotel, ao acessar o catálogo de uma biblioteca informatizada para consultar uma bibliografia, ao comprar produtos — como livros, brinquedos ou computadores — de um fornecedor por intermédio de sua página Web, muito provavelmente, essas atividades envolverão uma pessoa ou um programa de computador que acessará um banco de dados. Até mesmo, os produtos adquiridos em supermercados, em muitos casos, atualmente, incluem uma atualização automática do banco de dados que mantém o controle do estoque disponível nesses estabelecimentos.

(SADALAGE; FOWLER, 2013) acrescenta que os bancos de dados relacionais tornaram-se uma parte inerente à cultura computacional. E ele cita seus benefícios:

- Dados persistentes: capacidade de armazenar grandes quantidades de dados persistentes;
- Concorrência: controle de todo o acesso aos dados por meio de transações;
- Integração: tornar as atualizações acessíveis aos demais aplicativos e/ou usuários.

Pode-se dizer que os banco de dados relacionais foram bem sucedidos por trazer todos esses benefícios de uma forma padrão, permitindo assim aos desenvolvedores aprender o modelo relacional básico e aplicá-lo a muitos projetos.

Por outro lado, possui uma frustação que chama-se de incompatibilidade de impedância, que consiste na diferença entre o modelo relacional e as estruturas de dados na memória. Além disso, o modelo de dados relacional organiza os dados em uma estrutura de tabelas e linhas ou, mais apropriadamente, de relações e tuplas. Onde uma tupla é um conjunto de pares nome-valor e uma relação, um conjunto de

tuplas, como podemos visualizar na figura abaixo. Introduzindo assim limitações, em especial, os valores de uma tupla relacional têm de ser simples - eles não podem conter nenhum tipo de estrutura, como um registro aninhado ou uma lista.

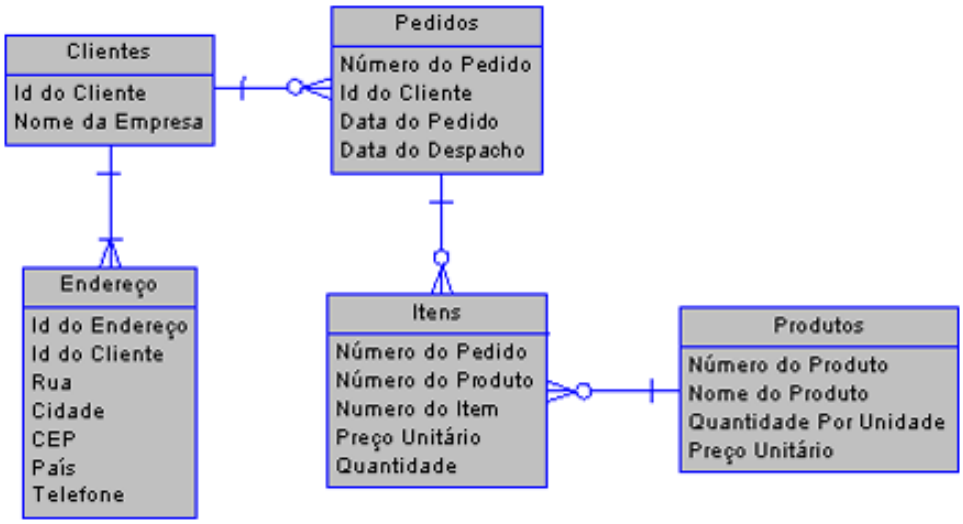

Figura 1 – Diagrama Entidade Relacionamento

Fonte: Autora

2.1.1 Problema do Banco de Dados Relacional

Segundo (SADALAGE; FOWLER, 2013), banco de dados relacionais triunfaram sobre aqueles orientados a objetos. Sob o ponto de vista dos autores, o principal fator foi a utilização da linguagem SQL como mecanismo de integração entre os aplicativos. Atuando assim, como um banco de dados de integração com múltiplos aplicativos.

Porém, existem também desvantagens na integração compartilhada de bancos de dados. A estrutura acaba tornando-se mais complexa e, consequentemente, lidar com o aumento de dados e do tráfego exige mais recursos computacionais. Dessa forma, sendo necessário a aquisição de máquinas maiores, com mais processadores e maior capacidade de armazenamento em disco e memória, ocasinando maior custo e ampliação dos limites físicos. Nesse contexto, a alternativa adotada, normalmente, é a utilização de clusters para minimizar os custos e melhorar a confiabilidade do sistema.

Contudo, de acordo com (SADALAGE; FOWLER, 2013) bancos de dados relacionais, como o Oracle RAC ou Microsoft SQL Server, não foram projetados para serem executados em clusters. Eles funcionam baseados no conceito de um subsistema de disco compartilhado. Ademais, bancos de dados relacionais também podem ser executados como servidores separados para conjuntos diferentes de dados, fragmentando efetivamente o banco. Embora tal procedimento divida a carga, toda a fragmentação tem de ser controlada pelo aplicativo, que precisa rastrear quais servidores de banco

de dados comunicam-se com quais partes dos dados. Além disso, perde-se quaisquer consultas, integridade referencial, transações ou controles de consistência que atravessarem os fragmentos.

Ademais, tem-se também o fato do banco de dados relacional possuir dois tipos de atributos, de acordo com o valor existente dentro de uma entidade. Segundo (ELMASRI; NAVATHE, 2011) a maioria dos atributos tem um valor único para uma dada entidade; esses atributos são chamados monovalorados. Por exemplo, Idade é um atributo monovalorado de uma pessoa. Em alguns casos, um atributo pode ter um conjunto de valores para a mesma entidade — por exemplo, um atributo Cor para um carro ou um atributo Titulação para uma pessoa. Os carros com uma cor têm um valor único, enquanto aqueles com dois tons contêm dois valores para Cor. Da mesma forma, uma pessoa pode não ter um título acadêmico, a outra pessoa pode possuir dois ou mais títulos, portanto, pessoas diferentes podem ter quantidades de valores diferentes para o atributo Titulação. Esses atributos são chamados multivalorados. Um atributo multivalorado deve ter limite inferior e superior para restringir a quantidade de valores permitidos a cada entidade individual. Por exemplo, o atributo Cor de um carro pode ter entre um e três valores, se presumir-se que um carro possa ter, no máximo, três cores.

Esse atributo multivalorado é chamado, segundo (ELMASRI; NAVATHE, 2011), de atributo complexo, já que observa-se que os atributos compostos e multivalorados podem ser aninhados de uma maneira arbitrária. Por exemplo, uma pessoa pode ter mais de uma residência e cada uma delas pode ter múltiplos telefones, um atributo EnderecoFone para uma pessoa pode ser especificado.

Como podemos observar no que foi dito acima, existem vantagens e desvantagens do uso do Banco de Dados Relacional. Temos como vantagens:

- Dados persistentes;
- Concorrência;
- Integração.

Já as desvantagens são:

- Incompatibilidade de impedância;
- Dificuldade em lidar com grande volume de dados

Podemos inferir que a quantidade de dados e a forma como ele são gravados tornam o banco de dados relacional limitado. Nesse cenário, os bancos de dados não baseado em SQL, mais conhecido como banco de dados NoSQL, podem surgir como uma solução para o problema.

2.2 Banco de dados NoSQL

Banco de dados NoSQL, ou não SQL, ou não relacionais é um termo para representar Banco de Dados Não Relacionais. Modelados de forma diferente da tabular, utilizada pelo Banco de dados relacional (vide 4) e utilizam linguagem de consulta diferente da SQL. Temos como vantagens:

- Flexibilidade: por não possuir necessidade de modelagem de dados ou normalização;
- Facilidade em lidar com Big Data e aplicações em tempo real.

No entanto também temos a desvantagem da imaturidade do Banco de Dados, já que é uma tecnologia nova existe maior possibilidade de suporte falho, pouca documentação, inexistência de ferramentas de integração com sistemas.

Segundo (SADALAGE; FOWLER, 2013), cada solução NoSQL possui um modelo diferente, os quais foram divididos em quatro categorias: chave-valor, documento, família de colunas e grafos. Dessas, as três primeiras compartilham uma característica comum em seus modelos de dados - que chamaremos de "orientação agregada", ou seja, valores que podem ser anhinhados dentro de si próprios.

Temos nas próximas sessões uma classificação desses bancos de dados.

2.2.1 Banco de Dados de chave-valor

Um Banco de Dados de chave-valor é uma simples base de dados que usa um vetor associativo como modelo de dados fundamental onde cada chave é associada a um e apenas um valor em uma coleção. Esse relacionamento é conhecido como par chave-valor. Em cada par, a chave é representada por uma string arbitrária como nome, URI. O valor pode ser um tipo de dado como uma string, imagem, documento, dentre outros e armazenado na forma de Blob (AEROSKIPE, 2011).

Na Tabela 1, temos um exemplo de um armazenamento de dados do tipo chave-valor, nesse caso trata-se de uma tabela de endereçamento de IP, onde a chave é o endereço IP da máquina e valor é o endereço MAC da mesma.

Tabela 1 – Exemplo de armazenamento chave-valor: Tabela de endereçamento de IP

Chave	**Valor**
202.45.12.34	01:23:36:0f:a2:33
202.45.123.4	00:25:33:da:4c:01
245.12.33.45	02:03:33:10:e2:b1
101.234.55.1	b8:67:a3:11:23:b1

Alguns dos bancos de dados chave-valor são o Riak (RIAK, 2018), Redis (REDIS, 2018), Memcached DB (MEMCACHED, 2018) e suas variedades, Berkeley DB (ORACLE, 2018), Hamster DB (DB, 2018), Amazon DynamoDB (AMAZON, 2018a) e Project Voldemort (VOLDEMORT, 2018).

Como objeto de estudo, será tratado sobre o Riak, um banco de dados Open Source que consiste em tabelas com itens e atributos. Cada item tem uma chave primária única e qualquer número de atributos. O item é similar a uma linha em banco de dados relacionais e o atributo, similar a uma coluna. Contudo, nem todos os itens precisam ter o mesmo número de atributos. A Tabela 2 apresenta uma correlação entre as propriedades do Riak e do MySQL.

Tabela 2 – Correlação entre SGBDR e Riak

MySQL	**Riak**
Instância de banco de dados	Cluster Riak
Tabela	Bucket
Linha	chave-valor
Chave	Chave

Os depósitos de chave-valor utilizam uma tabela hash simples e sempre fazem o acesso pela chave primária. Com isso, eles têm, geralmente, um ótimo desempenho e podem ser escaláveis facilmente.

Alguns dos problemas para os quais os armazenamentos de chave-valor se apresentam como uma solução apropriada são:

- Armazenamento de informações de sessão web, já que tudo a respeito da sessão pode ser armazenado ou recuperado por uma única solicitação, tornando o processo mais rápido;
- Perfis de usuários, preferências, tudo pode ser colocado em um objeto de modo que as preferências de um usuário sejam obtidas por meio de uma única operação;
- Dados de carrinho de compras, websites de comércio eletrônico possuem esse serviço, o qual deseja-se que esteja disponível todo o tempo. Todas as informações de compras podem ser colocadas no valor da chave.

(SADALAGE; FOWLER, 2013) cita situações nas quais os depósitos de chave-valor não são a melhor opção:

- Relacionamentos entre dados: se for preciso relacionar diferentes conjuntos de dados, ou então correlacionar os dados entre diferentes conjuntos de chaves,

armazenamentos de chave-valor não são a melhor opção, ainda que alguns deles forneçam recursos de navegação entre conexões (link-walking);

- Transações com múltipas operações: se for preciso salvar múltiplas chaves e ocorrer uma falha na gravação de alguma delas e quiser reverter ou desfazer o restante das operações;
- Consulta por dados: Não há como inspencionar o conteúdo dos pares, com raras exceções;
- Operações por conjuntos: já que as operações são limitadas a uma chave por vez, não há como trabalhar em múltiplas chaves ao mesmo tempo.

2.2.2 Banco de dados de documentos

De acordo com (HOWS; PLUGGE, 2015) banco de dados de documentos armazenam documentos em que o valor pode ser examinado. Além disso, permitem que novos atributos sejam criados sem a necessidade de definição prévia. Essa tecnologia não possui conceitos de tabelas, esquemas, SQL ou linhas.

Pode-se observar na tabela 3 um comparativo entre os Sistemas de Gerecia-mentos dos banco de dados de documentos e o banco de Dados Relacional.

Tabela 3 – Tabela comparativa do MySQL com o MongoDB

MySQL	MongoDB
Tabela	Coleção
Coluna	Campo
Linha	Documento

Um documento é a representação da informação em formato JSON e utiliza a linguagem JavaScript, como podemos observar na listagem abaixo. Temos como exemplo de bancos de dados de documentos MongoDB (MONGODB, 2018), CouchDB (APACHE, 2018a), Terrastore (CODE, 2018), OrientDB (ORIENTDB, 2018), RavenDB (RAVENDB, 2018), entre outros.

Listagem 1 – Exemplo de documento em formato JSON.

```
1 {
2
3     int id_titulo = 102;
4     titulo = "Como fazer uma monografia";
5     subtitulo = "Aprenda a fazer uma monografia em 90 dias";
6     post = "Dedique-se ao máximo...";
7     marcadores = "monografia, estudo, como fazer";
8
```

```
9 }
```

Fonte: Autora

(HOWS; PLUGGE, 2015) fala sobre o MongoDB e cita as seguintes características:

- Consistência: a consistência em bancos de dados MongoDB é configurada utilizando os conjuntos de réplicas e optando por esperar que as gravações sejam replicadas em todos os escravos ou em um determinado número de escravos. Cada gravação pode especificar o número de escravos na qual deve ser propagada antes de ser considerada bem-sucedida;
- Transações: comandos insert, uptade e delete;
- Disponibilidade: tentam melhorar a disponibilidade replicando os dados, utilizando a configuração mestre-escravo;
- Recursos de consulta: podemos consultar os dados dentro do documento sem ter de recuperar o documento inteiro;
- Escalabilidade: adicionar nodos ou alterar o armazenamento de dados sem simplesmente migrar o banco de dados para uma máquina mais potente.

As situações em que o uso do banco de dados de documentos é apropriado são: registro de eventos (log), sistema de gerenciamento de conteúdo (CMS), plataformas de blog, análises web ou em tempo real (analytics), aplicativos de comércio eletrônico. Por outro lado, ele não é recomendado para situações que exijam transações complexas que abranjam diferentes operações e consultas em estruturas agregadas variáveis.

2.2.3 Banco de dados armazenamento em famílias de colunas

Banco de dados de família de colunas permite o armazenamento de dados com chaves mapeadas para os valores, e os valores são agrupados em múltiplas famílias de colunas, cada uma delas, funcionando como um mapa de dados. Segundo (MARQUESONE, 2016), buscando assim, resolver problemas de escalabilidade e flexibilidade na armazenagem de dados.

Já existem diversas soluções para o armazenamento de dados em famílias de colunas, sendo alguma delas, o Cassandra (APACHE, 2018c), Hbase (APACHE, 2018b), Hypertable (DB-ENGINES, 2018b), Amazon SimpleDB (AMAZON, 2018b). Neste estudo, será abordado o Cassandra.

A modelagem de dados no Cassandra é feita da seguinte forma:

- Coluna: par de chave e valor;
- Linha: coleção de colunas rotuladas e classificadas por um nome;
- Família de colunas: coleção de linhas rotuladas por um nome;
- Chave: agrupamento de família de colunas.

Pode-se observar na Tabela 4, um comparativo dos Sistemas de Gerenciamento de Banco de Dados Relacional e o Banco de dados de família de colunas, mais especificamente o Cassandra.

Tabela 4 – Tabela comparativa do MySQL e o Cassandra

MYSQL	Cassandra
Instância de Banco de Dados	Cluster
Banco de dados	Chave
Tabela	Família de Colunas
Coluna (a mesma para todas as linhas)	Colunas (podem ser diferentes por linha)

(SADALAGE; FOWLER, 2013) explana o Cassandra e algumas de suas características coincidentes com os demais banco de dados já citados:

- Consistência: os dados são gravados primeiro em um commit log (registro de operações), depois em uma estrutura na memória conhecida como memtable. E separa em 3 níveis de consistência: ONE (o nível mais baixo), QUORUM (nível intermediário) e o ALL (o qual todos os nodos terão que responder a leituras ou gravações, tornando o cluster intolerante a falhas). Dados são distribuídos para nodo com intervalos de dados adjacentes, se um nodo for removido, o dado anterior passaria a pertencer ao próximo nodo;
- Transações: não existe, já que uma gravação é atômica em relação a linha;
- Disponibilidade: altamente disponível, uma vez que não há um mestre no cluster e todos os nodos têm, o mesmo status;
- Consultas básicas: incluem GET, SET e DEL;
- Consultas avançadas e indexação: permite que você indexe outras colunas além das chaves da família de colunas;
- Escalabilidade: para torná-lo escalável basta adicionar mais nodos.

Existem aplicações que são apropriadas para tal uso do banco de dados. Nesse ínterim, (MARQUESONE, 2016) cita as aplicações que necessitam armazenar informações sobre eventos, como, por exemplo, os estados do aplicativo ou os erros

encontrados por ele. E que não deve ser utilizado em sistemas que requerem transações ACID (acrônimo de Atomicidade, Consistência, Isolamento e Durabilidade) para gravações e leituras.

Onde ACID é um conjunto de propriedades de transações para o Banco de Dados. No qual, Atomicidade trata a transação como parte indivisível, devendo ter todas as suas operações realizadas. A Consistência leva a transação respeitar todas as regras de integridade de dados, como a unicidade de chaves. Já o Isolamento serve para o controle de concorrência. E por fim a Durabilidade, garante que os dados estarão disponíveis em definitivo.

2.2.4 Banco de Dados de Grafos

Banco de Dados Grafos permite que seja armazenado entidades e também relacionamentos entre essa entidades. As entidades são também conhecidas como nodos, os quais possuem propriedades. Os relacionamentos são conhecidos como arestas que podem ter propriedades.

Na Figura 2, temos um exemplo de modelo de dados em grafos de uma Rede Social de recomendação de livros onde, vários nodos estão relacionados entre si. Os nodos são entidades que têm propriedade, como, por exemplo, nomes como a "Anna", "Barbara", "Refactoring". Nota-se também que as arestas possuem tipos, como "likes"e "author", e podem ter múltiplas propriedades. Além disso, tipos de relacionamentos têm significância direcional, assim pode-se consultar o grafo de muitas formas.

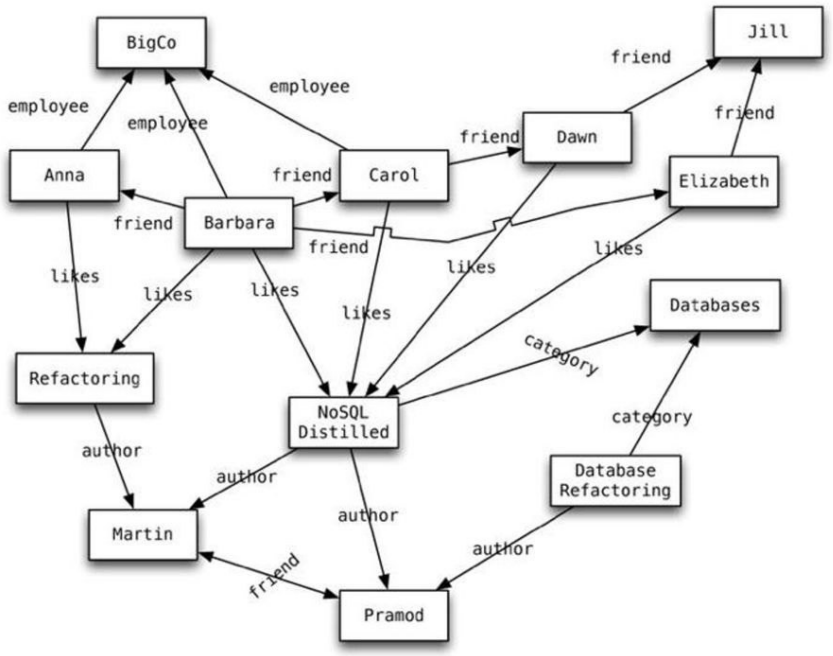

Figura 2 – Modelo de dados em grafo: Rede Social

Fonte: Sadalage e Fowler (2013)

Uma das vantagens dos banco de dados grafos é a possibilidade de alterar os requisitos de uma consulta, também chamada de travessia, sem precisar alterar os nodos ou arestas.

Existem vários banco de dados de grafos, tais como, Neo4j (NEO4J, 2018), Infinite Graph (OBJECTIVITY, 2018), OrienteDB (ORIENTDB, 2018) ou FlockDB (DB-ENGINES, 2018a). Algumas características dessa modelagem são citadas por (SADALAGE; FOWLER, 2013), dentre elas, destacam-se:

- Consistência: banco de dados de grafos garantem consitência por meio de transações;
- Transações: antes de alterar ou adicionar algo deve-se iniciar uma transação e a mesma tem que ser marcada como bem-sucedida, isso precisa ser lembrado durante o desenvolvimento, diferente do modo de execução do SGBDR;
- Disponibilidade: alta disponibilidade por permitir escravos replicados;
- Escalabilidade: pode ser adquirido adicionando mais memória RAM; adicionando mais escravos com acesso apenas de leitura ou fragmentando esses dados a partir do lado do aplicativo.

De acordo com (PANIZ, 2016) exitem diversos casos de uso apropriado para banco de dados de grafos, entre eles: dados conectados, como, por exemplo, as

redes sociais; roteamento, serviços baseados em localização, já que o local pode ser considerado um nodo; e também nos sistemas de recomendação.

(PANIZ, 2016) também cita mais questões que atraem o interesse para o uso do banco de dados grafos:

- Perfomace: elevação da perfomace quando utilizado com dados conectados, contrastando com BDR onde a performace de consulta com intensas junções deteriora com o crescimento de conjunto de dados;
- Flexibilidade: grafos são naturalmente adicionados, permitindo assim, a qualquer momento do desenvolvimento do projeto, adicionar novos tipo de relacionamentos, novos nós, novos rótulos e novos subgrafos sem prejudiciar consultas já existentes;
- Agilidade: devido a flexibilidade relatada acima o desenvolvimento do projeto se dá de forma mais ágil.

(SADALAGE; FOWLER, 2013) alertam que o uso desse banco de dados não é recomendado para situações, em que seja necessário atualizar todas as entidades ou um subconjunto delas, já que alterar uma propriedade em todos os nodos não é uma operação direta.

Nessa modelagem encontrar nodos e suas relações imediatas é fácil e isso também pode ser feito em bancos de dados relacionais. Desse modo, BDG só se destacam quando deseja-se percorrer os grafos em qualquer profundidade e especificar um nodo inicial para a travessia.

O BDG utiliza *index free adjacency* que é importante para alcançar alta performance ao percorrer o grafo. No banco de dados todo nó mantém referência direta para os nós adjacentes. Ele é referenciado como um micro índice para outros nós e é mais barato que usar índices globais. Isso significa que o tempo de consulta é independente do tamanho total do grafo e simplesmente diretamente proporcional ao tamanho do subgrafo pesquisado. Banco de dados grafos produzem resultados muito rápido em termos de tempo de consulta e também podem armazenar grandes quantidades de dados (KALIYAR, 2015).

Baseado no que foi apresentado até aqui o banco de dados Grafos foi escolhido para este trabalho por ser mais próximo do banco de dados relacional, possuindo as propriedades ACID, controle de transações e gerenciador de bloqueio (ALMEIDA, 2011). Como SGBDG, foi escolhido o Neo4j, por utiizar o algoritmo de Dijktras para manipulação de dados e outras características que serão melhor explanadas na próxima seção.

2.2.4.1 Neo4j

(MARQUESONE, 2016) afirma que apenas o Neo4J e OrientDB se preocupam em garantir as propriedades da ACID (atomicidade, consistência, integridade e durabilidade). (ALVAREZ; CECI; GONçALVES, 2016) realizou testes de desempenho entre o Neo4J e o OrientDB, onde foi possível identificar diferenças relevantes entre eles. Os resultados obtidos são apresentados na Tabela 5.

Tabela 5 – Comparativo entre Neo4j e OrientDB

Profundidade	**Neo4J**	**OrientDB**
2	16ms	15ms
3	358ms	5439ms
4	2497ms	Falha de memória
5	11218ms	Falha de memória

O Neo4j permite que dados sejam percorridos da mesma forma que os grafos, mantendo algumas características que se tornaram bastante comuns em banco de dados relacionais, como controle de transações e suporte completo as propriedades ACID, por possuir na memória logs de transações e gerenciador de bloqueio (ALMEIDA, 2011).

Segundo (EDUARDO, 2015), algumas características particulares fazem o Neo4j ser muito popular entre os usuários, desenvolvedores e adminstradores de banco de dados, sendo elas:

- Todos os relacionamentos em Neo4j são igualmente importantes e rápido, fazendo o possível para materializar e usar novas relações, mais tarde, para “atalho” e acelerar os dados de domínio quando surgem novas necessidades;
- Armazenamento compacto e cache de memória para grafos, resultando em aumento de escala e eficiente bilhões de nós em um banco de dados em hardware moderado;
- Escrito em Java, rodando em JVM(Java Virtual Machine).

O Neo4J possibilita que as propriedades dos nodos do grafo sejam consultadas e os relacionamentos, entre eles, percorridos, utilizando bindings da linguagem. Desse modo, os nodos podem ser indexados, à medida que são adicionados ao banco de dados ou todos posteriormente. Logo, precisa-se criar um índice para os nodos utilizando o **IndexManager**.

Para tanto, o Neo4J utiliza o Lucene como seu serviço de indexação. Dessa forma, possibilitando a existência de um recurso bem interessantes nos bancos de

dados de grafos, que é o de encontrar os caminhos entre dois nodos. No grafo da figura 1, sabemos que *Barbara* está conectada a *Jill* por dois caminhos distintos; para encontrar todos esses caminhos e também a distância entre eles pode-se utilizar:

Listagem 2 – Exemplo de construção de grafo Neo4J.

```
1 Node barbara = nodeIndex.get("name", "Barbara").getSingle();
2 Node jill = nodeIndex.get("name", "Jill").getSingle();
3 PathFinder<Path> finder = GraphAlgoFactory.allPaths(Tranversal.
    expanderForTypes(FRIEND, Direction.OUTGOING), MAX_DEPTH);
4 Iterable<Path> paths = finder.findAllPaths(barbara,jill);
```

Fonte: Sadalage e Fowler (2013)

O Neo4J também fornece a linguagem de consulta **Cypher** para realizar consultas no grafo. Essa ferramenta será explicada na seção seguinte.

2.2.4.2 Cypher

De acordo com (ROBINSON; WEBBER; EIFREM, 2015) Cypher contém um poder expressivo de um grafo de propriedades combinado com uma linguagem de consulta declarativa, o que a torna uma linguagem de fácil entendimento por desenvolvedores.

Cypher necessita de um nodo START para iniciar a consulta que é identificado pelo seu ID de nodo. Utiliza a palavra-chave MATCH para buscar padrões em relacionamentos; a palavra-chave WHERE para filtrar as propriedades em um nodo ou relacionamento e a paravra-chave RETURN para especificar o que é retornado pela consulta.

O Cypher fornece também alguns métodos ORDER, AGGREGATE, SKIP E LIMIT para ordenar, agregar, pular e limitar os dados. Além desses, existem muitos outros recursos de consulta nessa linguagem que podem ser explorados.

A Figura 3 apresenta o modelo elaborado para o exemplo.

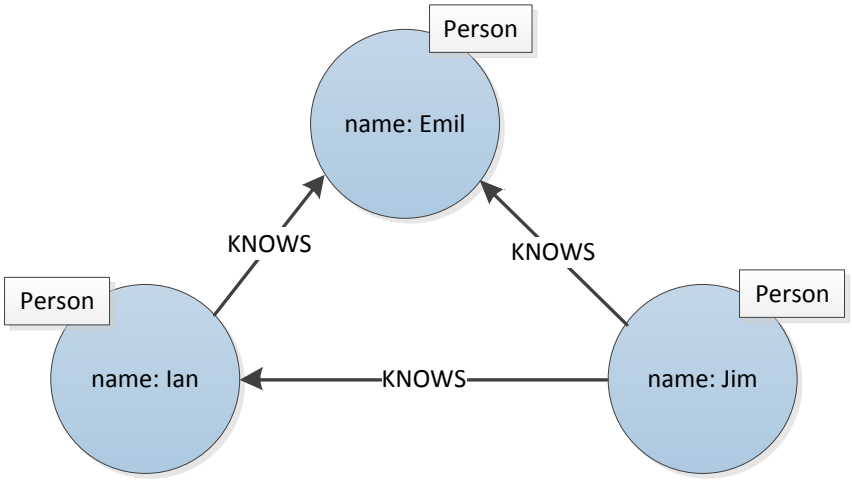

Figura 3 – Modelo de grafo simples, expressado na forma de diagrama

Fonte: Robinson, Webber e Eifrem (2015)

O Cypher representa os grafos em forma de desenhos utilizando ASCII, o modelo apresentado na figura é representado na linguagem da seguinte forma:

Listagem 3 – Exemplo de contrução de grafo na linguagem Cypher.

```
1 (emil)<-[:KNOWS]-(jim)-[:KNOWS]->(ian)-[:KNOWS]->(emil)
```

Fonte: Robinson, Webber e Eifrem (2015)

O padrão descreve caminhos que conectam *Jim* a *Emil* e *Ian*, assim como, *Ian* a *Emil*.

Com o Cypher as consultas ficam mais simples e intuitivas, facilitando assim, futuras manutenções de sistema, já que fica mais claro compreender a função de cada uma delas.

3 Análise Comparativa do BDR e BDG

Neste capítulo serão elencadas algumas comparações entre o Banco de Dados Relacional e o Banco de Dados Grafos. É importante ser capaz de identificar como o aumento de cada nível de informação resulta em um desempenho de banco de dados, bem como características gerais de ambos.

Tabela 6 – Tabela de Correspondências entre BDR e BDG

Características	BDR	BDG
Estrutura	Tabular	Grafos
Relacionamentos	Chave estrangeira	Arestas
Dados	Tuplas	Nós
Normalização	Sim	Não

Como pode-se observar na Tabela 6 as principais características que diferem o banco de dados relacional do banco de dados orientado a grafos estão relacionadas a forma de organizar e estruturar os dados. Essa forma de estruturar os dados reflete na principal diferença entre as duas modelagens: a forma de lidar com os relacionamentos.

Os Banco de dados Relacionais não possuem uma estrutura explícita de representar relacionamento, para isso recorre a artifícios como inserções de chaves estrangeiras para poder representá-los. Diferentemente, os bancos de dados grafos consideram os relacionamentos tão importante quanto os dados e possuem uma estrutura própria para representá-los, as arestas, dando mais clareza e semântida as relações no banco de dados.

Outra diferença entre o BDR e o BDG está relacionada a necessidade de normalização. Diagramas de entidades e relacionamentos são necessários no banco de dados relacional, enquanto no banco de dados grafos é dispensável, pois o grafo da forma que é modelado e idealizado pode ser inserido no banco de dados sem a necessidade de transformações para a representação desses dados em tabela.

Para facilitar as comparações iremos examinar um simples domínio de perfil de alunos universitários do Brasil. Nesse domínio, cada aluno do Brasil possui um perfil: nome, o seu estado de origem, cor da pele e a universidade em que cursa. A partir desse domínio, será realizada a modelagem de forma relacional, mostrando sua normalização, apontando também impasses nesse modelo e apresentando como a modelagem baseada em grafos afeta a performace e a forma de leitura.

Desenvolvedores de banco de dados relacional trabalham com uma tabela igual a Tabela7, a qual relaciona pessoas, estado de origem, cor da pele e onde estudou.

Todos esses dados estão em uma tabela e é, realmente, fácil de ler, o que torna mais propício a inserção de novas pessoas na equipe de desenvolvimento. Porém, da forma apresentada é difícil manter a consistência dos dados, pois percebe-se no exemplo, que os valores Maranhão e UFMA são utilizados várias vezes e caso o nome de um ou mais desses estados ou universidades mudar deve-se manter a consistência, ou seja, todos as tuplas correspondentes a eles deverão ser alteradas com o valor atualizado.

Tabela 7 – Tabela Relacional

Nome	Estado	Cor	Universidade
Aitan	MA	Índio	UFMA
Flaviane	PA	Branca	IFPA
Alice	BH	Parda	UFMA
Beatriz	MA	Negra	UFMS
Miguel	SP	Negro	PUC

Para facilitar esse processo, é criada outra tabela com os nomes de cada um dos estados, igual a Tabela 8, permitindo inserir informações adicionais, como o nome do governador e as chaves primárias que podem ser referenciadas como estrangeiras nas outras tabelas, ficando igual a Tabela 9 e tornando a Tabela 9 um pouco mais difícil de ler.

Tabela 8 – Tabela Relacional Cidade

ID	Estado	Governador
17	MA	Flávio Dino
12	PA	Simão Jatene
19	BH	Fernando Pimentel
17	MA	Flávio Dino
21	SP	Geraldo Alckmin

Tabela 9 – Tabela Relacional Atualizada

Nome	Estado	Cor	Universidade
Aitan	17	Índio	UFMA
Flaviane	12	Branca	IFPA
Alice	19	Parda	UFMA
Beatriz	17	Negra	UFMS
Miguel	21	Negro	PUC

Esse mesmo procedimento deverá ser realizado em várias tabelas, como por exemplo universidade, o que vai tornando mais difícil a compreensão e mais dados poderão ser adicionados como podemos observar na Tabela 10.

Tabela 10 – Tabela Relacional Universidade

ID	Nome	Estado
92	UFMA	MA
34	IFPA	PA
1	PUC	SP
25	UFMS	MS

Com isso, a manutenção do banco de dados e a inserção de novos membros na equipe vai sendo prejudicada, pois a compreensão dos relacionamentos se torna mais complexa. Como pode ser observado na Tabela 11 resultante da Tabela 7 após as transformações.

Tabela 11 – Tabela Relacional Final

Nome	Estado	Cor	Universidade
Aitan	17	Índio	92
Flaviane	12	Branca	34
Alice	19	Parda	92
Beatriz	17	Negra	25
Miguel	21	Negro	1

Todos os passos descritos são chamados de normalização e são necessários para que não haja redundância dos dados. O próximo passo é elaborar o diagrama UML para apresentar como as tabelas se relacionam.

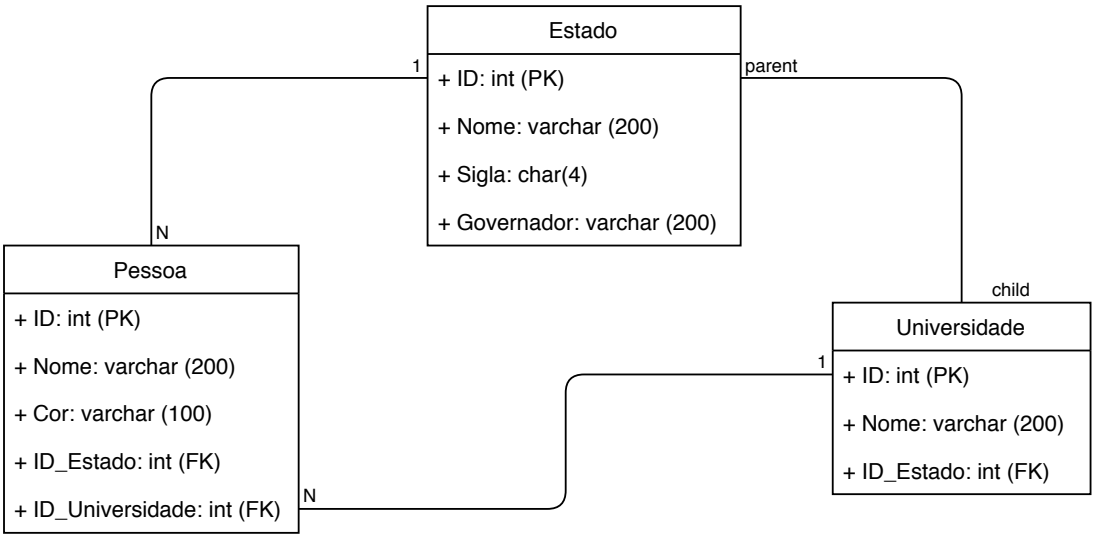

Figura 4 – Diagrama UML das relações entre as tabelas

Fonte: Elabarado pela autora

Como podemos observar no diagrama da Figura 4, à medida que avança-se na normalização dos dados mais complexa se torna a interpretação para novos

desenvolvedores e mais difícil para realizar manutenção do sistema.

Além disso, existe também diferença nas buscas. Os exemplos das listagens 4 e 5 têm a mesma finalidade, obter listas de alunos (retornando seu nome, estado e cor) das universidades do estado do Maranhão.

A listagem 4 apresenta uma consulta SQL com instruções JOIN, onde as linhas 7 e 8 fazem uma junção entre as três tabelas: pessoa, estado e universidade.

Listagem 4 – Exemplo de consulta SQL

```
1  SELECT
2  p.nome,
3  e.estado, p.cor,
4  u.nome, u.estado,
5  FROM
6   pessoa p
7   LEFT JOIN estado e ON e.ID=p.estado
8   LEFT JOIN uni u ON p.uni=u.id
9  WHERE
10  u.estado="MA"
```

Fonte: Autor

Nesse exemplo, com poucas tabelas, foi fácil descrever a consulta. Contudo, em sistemas maiores as consultas se tornam complexas. Logo, não raro, as equipes de desenvolvimento possuem membros que se dedicam apenas a tornar as consultas mais eficazes. Entretanto, como no BDR as consultas precisam percorrer toda a tabela, sua performace fica prejudicada. Para contornar esse problema, os SGBDR oferecem o recurso de criação de índices, que facilitam as buscas. Porém, vale salientar, que esse recurso consome muito espaço em disco e dependendo da consulta pode até reduzir o desempenho da mesma.

Com a finalidade de lidar com a performace reduzida e para permitir uma consulta mais rápida, muitas vezes, os desenvolvedores precisam desnormalizar seus dados, levando para fora da forma normal (Tabela 12), o que permite consultá-los mais rapidamente.

Tabela 12 – Tabela Relacional Desnormalizada

Nome	Estado	Governador	Cor	Universidade	Estado-Uni
Aitan	MA	Flávio Dinho	Índio	UFMA	MA
Flaviane	PA	Simão Janete	Branca	IFPA	PA
Alice	BH	Fernando Pimentel	Parda	UFMA	MA
Beatriz	MA	Flávio Dino	Negra	UFMS	MS
Miguel	SP	Geraldo Alckmin	Negro	PUC	SP

Pórem, isso pode causar problemas de inconsistências e de desempenho de tempo de gravação, já que terá que ser atualizado cada linha. Ademais, a manutenção fica difícil porque não se sabe como os dados estão funcionando e se é necessário adicionar novas tabelas para as instruções JOIN, tornando-se numa tarefa desafiadora para banco de dados com grandes volumes.

Todavia, essa tarefa se torna menos exaustiva ao se converter o modelo de tabela para um em um grafo, onde as relações ficam muito mais intuitivas, como pode-se observar na Figura 5. Percebe-se claramente os nós que representam as entidades no modelo relacional e as arestas que são os relacionamentos.

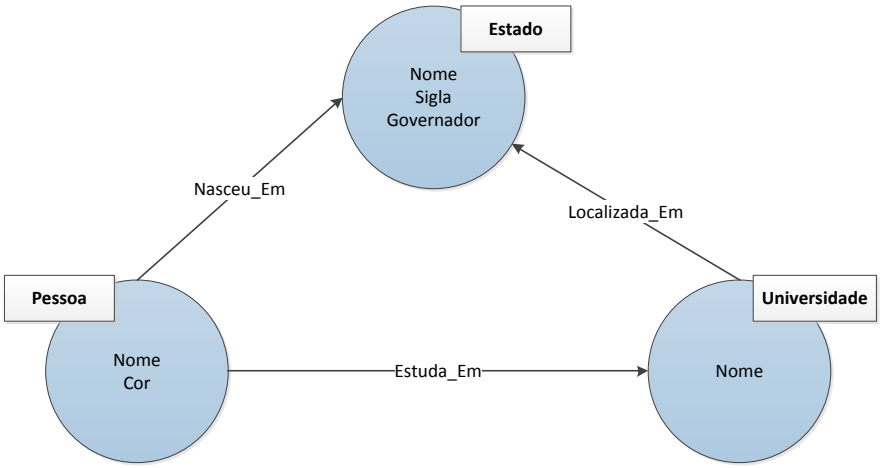

Figura 5 – Modelo de grafo do domínio

Fonte: Autor

Utilizando Cypher fica muito mais fácil ler e percorrer o grafo. Lembrando que Cypher é uma linguagem de consulta declarativa semelhante ao SQL, mas otimizada para grafos e para facilitar a leitura e a performace de busca. Tem-se como exemplo a listagem 5 onde é feito uma busca no grafo que irá partir do nó inicial, representando o Estado do MA para os nós adjacentes, que corresponde aos alunos. Desta forma a busca será restrita a um subgrafo dos nós adjacentes ao estado procurado, otimizando a performace do banco de dados, visto que não será necessário percorrer todo o banco de dados.

Listagem 5 – Exemplo de consulta Cypher

```
1 MATCH
2  (p:pessoa)-[:ESTUDA_EM]->(u:Uni),
3  (p)-[:VIVE_EM]->(c:Estado),
4  (u)-[:LOCALIZADO_EM]->(e:Estado-Uni)
5 WHERE
```

```
6   e.sigla="MA"
7 RETURN
8   p.nome,
9   c.estado, p.cor,
10  u.nome, e.sigla
```

Fonte: Elaborado pela autora

Em aplicações com dados altamente conectados, como foi possível observar no exemplo, é preferível utilizar banco de dados grafos, ao invés de persistir em BDR, tendo para isso que recorrer a consultas com toneladas de JOINS em SQL. Também é possível, realizar a migração de apenas um subconjunto de dados, mantendo parte dos dados em estruturas tabulares ou chaves.

O conceito de persistência poliglota, isto significa que o desenvolvedor irá acessar o banco de dados que melhor se adequa ao tipo de dados que precisa acessar. Podendo ser melhor armazenado em um BDR se houver dados tabulares de colunas e linhas. Mas, em casos em que os dados são relacionais e estão altamente conectados, poderá ser melhor armazenado em um BDG. Ou seja, os BDR e BDG não são, necessariamente, concorrentes, os dois podem ser utilizados em conjunto para melhorar a performace das aplicações.

4 Estudo de Caso

Neste capítulo é realizado um estudo de caso, com a finalidade de apresentar um comparativo entre as consultas realizadas em Banco de dados Relacionais e em Banco de Dados Grafos em um domínio altamente conectado.

A comparação dos bancos de dados utilizados consiste em um parâmetro objetivo: tempo de resposta das consulta, e um subjetivo: facilidade de codificação.

4.1 Conjunto de Dados

Neste trabalho buscou-se utilizar um conjunto de dados consolidado na literatura e que representasse um domínio altamente correlacionado. Desta forma, o conjunto de dados escolhido foi o MovieLens, disponibilizado pelo projeto de pesquisa GroupLens Research Project da Universidade de Minnesota, que consiste em dados de perfil de usuários e suas respectivas avaliações a filmes assistidos. O MovieLens é muito utilizado para estudo e desenvolvimento de sistemas de recomendação.

Várias versões do conjunto de dados são disponibilizadas no site da MovieLens, a versão escolhida foi a ML-100K que possui 100 mil avaliações para 1.682 filmes por 943 usuários. Os dados deste conjunto foram coletados a partir do site da MovieLens durante o período de 19 de Setembro de 1997 a 22 de abril de 1998. Estes dados foram tratados e usuários com menos de 20 avaliações foram removidos do conjunto (HARPER; KONSTAN, 2015) para obter maior aumento de dados interconectados e assim simular um sistema de Big Data. Os dados disponíveis neste conjunto são listados a seguir:

- 100 mil avaliações. Onde cada avaliação varia de 1 a 5, onde 1 refere-se a ruim e 5 ótimo;
- São 1.682 filmes avaladiados por 943 usuários ;
- Cada usuário avaliou pelo menos 20 filmes;
- Informações demográficas simplificadas sobre os usuários (idade, sexo, ocupação, CEP e um id);
- Informações sobre o filme, como título, data de lançamento, IMDb URL, gênero (pode ser vários ao mesmo tempo), além de um identificador único.

Neste trabalho foram utilizadas apenas as informações referentes as avaliações dos usuários, título dos filmes com seu respectivo ID e ID do usuário. Desta forma, simplificando a modelagem dos dados para tornar o estudo de caso mais claro e objetivo.

Para a modelagem de dados grafos o conjunto foi representado, segundo a seguinte estrutura:

- Dois tipos de nós:

  User: usuário;

  Movie: filme.

- Um tipo de relacionamento:

  Has_rated: se o usuário u_i avaliou o filme m_j, então u_i está explicitamente conectado a m_j, com uma relação do tipo Has_rated. A avaliação $r(m_j, u_i)$ é armazenada como uma propriedade do relacionamento e seu valor varia entre 1 e 5.

Um modelo do grafo extraído do conjunto de dados instaciado no Neo4j é apresentado na Figura 6, que demonstra parte do grafo, selecionando apenas o usuário 90 e os respectivos filmes avaliados por ele.

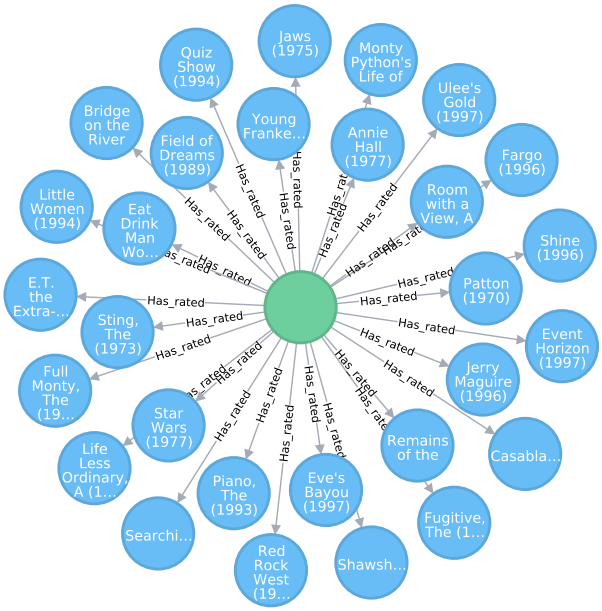

Figura 6 – Usuário 90 e todos os filmes que ele avaliou

Fonte: *Print Screen* do Neo4j

Para a modelagem relacional a estrutura utilizada foi a seguinte:

- Três tabelas:

  User: usuário;

  Movie: filme;

  Rating: o usuário avaliou.

- Relacionamentos:

  Os relacionamentos serão expressos, a partir de duas chaves estrangeiras na tabela rating, sendo uma correspondente ao id do usuário (idUser) e outra correspondente ao id do filme (idMovie), a tabela rating também possuíra o atributo rating que armazenará a nota dado pelo usuário.

A Figura 7 representa a estrutura da base de dados no modelo relacional.

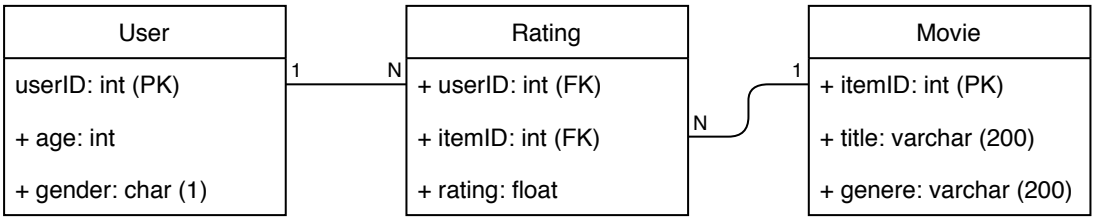

Figura 7 – Diagrama UML da base de dados MovieLens

Fonte: Elaborado pela autora

4.2 Consultas

.

As consultas foram idealizadas de forma a evidenciar os pontos positivos e negativos de ambas modelagens. Para este trabalho elas foram divididas em dois grupos: estruturais e analíticas.

As estruturais são mais focadas na forma como os dados estão relacionados, ou seja, os caminhos entre as entidades, por meio dos relacionamentos, nelas não existe a necessidade de fazer buscas em todo o banco de dados mas apenas em um parcela do mesmo, em subgrafos. A analítica está relacionada a consulta por entidades que possuam um determinado valor, nesse tipo de consulta é necessário percorrer todo o banco para encontrar o resultado desejado.

As consultas estruturais são as seguintes:

- E1: Tendo um filme como origem, a partir dos usuários imediatamente conectados a eles. Quais filmes são alcançáveis(Nível 2)?
- E2: Quais filmes são alcançáveis tendo como origem os filmes resultantes da consulta E1(Nível 3)?
- E3: Quais filmes são alcançáveis tendo como origem os filmes resultantes da consulta E2(Nível 4)?

Para as analíticas foram realizados as seguintes quesitações:

- A1: Quantos filmes possuem avaliações com valor superior a 4?
- A2: Qual o ranking dos filmes, classificados por número de visualizações?

4.2.1 Consultas Estruturais

Nesta subseção são apresentados os algoritmos que traduzem as consultas estruturais apresentados anteriormente. As linguagens de consultas utilizadas foram SQL, para Banco de Dados Relacional, e Cypher, para Banco de Dados Grafos.

Para não tornar esta subseção muito longa e sua leitura massante, optou-se por demonstrar apenas os códigos para a consulta estrutural E1.

A Listagem 6, apresenta o código em Cypher para a consulta E1, onde na linha 1 é realizado um MATCH(busca ou travessia) partindo do filme(m1) de id 50, encontrando todos os usuários que possuem um relacionamento com este filme, em seguida, na linha 2 são encontrados todos os filmes(m2) que foram avaliados por cada usuário(u1) conectado ao filme(m1). Por fim, é retornado os títulos dos filmes encontrados. Como pode-se observar com poucas linhas de códigos foi possível construir a consulta.

Listagem 6 – Código em Cypher para consulta E1

```
1 MATCH (m1:Movie{movie_id: 50})<-[:Has_rated]-(u1:User)
2 MATCH (u1)-[:Has_rated]->(m2:Movie)
3 RETURN m2.title
```

Fonte: Elaborado pela autora

A Listagem 7, apresenta o código em SQL para a consulta E1, pode-se observar que na linha 1 é selecionado como informação de retorno o título dos filmes que se deseja encontrar, em seguida é realizado uma junção entre as tabelas ‘movie’, ‘ratings’ e ‘users’ para encontrar os usuários que assistiram o filme de id 50.

Encontrados esse usuários é realizado uma nova junção na linha 5 entre as tabelas ‘user’, ‘rating’ e ‘movie’ para encontrar os filmes que esses usuários assistiram além do filme de id 50. Como pode-se observar, já existe um aumento considerável de linhas e complexidade em comparação com a consulta em Cypher.

Listagem 7 – Código em SQL para consulta E1

```
1 SELECT M2.title
2 FROM movies AS M1 INNER JOIN ratings AS R1
3 ON M1.itemID = R1.itemID INNER JOIN users AS U
4 ON U.userID = R1.userID INNER JOIN ratings AS R2
5 ON U.userID = R2.userID INNER JOIN movies AS M2
6 ON M2.itemID = R2.itemId
7 WHERE M1.itemID = 50
```

Fonte: Elaborado pela autora

A cada nível que a consulta SQL terá que descer deverá ser acrescentado mais quatro INNER JOINs.

4.2.2 Consultas Analíticas

Nesta subseção são apresentados os algoritmos que traduzem as consultas analíticas. De forma semelhante a subseção 4.2.1, é apresentado apenas os códigos paras a consulta analítica A2.

Na liguagem Cypher o algoritmo para a consulta A2 é demostrado na Listagem 8.

Listagem 8 – Código em Cypher para Consulta A2

```
1 MATCH(m:movie)<-[r:Has_rated]-(u:User)
2 with m, count(r) as countr
3 RETURN DISTINCT m.movie_id, countr ORDER BY countr DESC
```

Fonte: Elaborado pela autora

Na linha 1 é percorrido todos os nós 'movie' encontrando cada uma de suas relações com usuários, na linha 2 é armazenado em um vetor o filme e seu correspondente número de visualizações.

Na linha 3 é realizado o retorno da consulta, apresentando o ID dos filmes e seus respectivos número de visualizações em ordem decrescente.

A consulta em SQL é apresentada na listagem 9, onde é percorrida apenas a tabela ratings, pois todos os filmes que se encontram nela foram visualizados. Os filmes são agrupados pelos seus respectivos ID, que posteriormente são contabilizados para a obtenção do número de visualizações que cada um possui. Por fim, é apresentada uma lista dos filmes com suas respectivas visualizações, ordenados de forma decrescente.

Listagem 9 – Código em SQL para Consulta A3 - Etapa 2

```
1 SELECT itemID, count(*) as view_count
2 FROM ratings
3 GROUP BY itemID ORDER BY view_count DESC
```

Fonte: Elaborado pela autora

4.3 Resultados

Para avaliação das duas metodologias, relacional e grafo, o parâmetro objetivo utilizado foi o tempo de resposta. Para a obtenção desses tempos cada consulta estrutural foi executada 20 vezes e em cada uma delas com um filme de 'id' diferente. Por fim, foi realizado a média aritmética dos tempos de resposta obtidos.

Com os resultados obtidos foram elaboradas duas tabelas, uma para consultas estruturais e outra para consultas analíticas, que apresentam os tempos de resposta

em milisegundos. A Tabela 13 apresenta os tempos obtidos com a consulta estrutal.

Tabela 13 – Comparativo entre os tempos(em milisegundos) de respostas do MySQL e do Neo4j para as consultas estruturais

	Tempos para cada consulta Estrutural		
	E1	**E2**	**E3**
MySQL	32.1	75.3	597.5
Neo4j	28.3	33.4	38.9

Fonte: Elaborado pela autora

Analisando a Tabela 13, pode-se observar que houve uma pequena diferença entre os dois bancos de dados na Consulta E1, por ser menos complexa, à medida que a complexidade da consulta aumenta as diferenças se destacam, na consulta E3 o MySQL conclui a consulta em 597.5 milissegundos de execução, enquanto que no Neo4j foi concluída em 38 milissegundos.

Uma hipótese para a melhora dos resultados do Neo4j em relação ao MySQL nessas consultas, se dá pelo fato de que o MySQL precisa percorrer cada registro das tabelas do banco de dados, enquanto que o Neo4j só percorre a parcela dos dados que estão diretamente ligadas ao nó de partida, utilizando menos recursos de memória e retornando o resultado mais rápido.

Os resultados obtidos com as consultas analíticas é apresentado na Tabela 14. Percebe-se uma vantagem dos tempos de respostas do MySQL em relação ao Neo4j, provavelmente porque nesses tipos de consultas, toda a base de dados precisa ser percorrida e houve a necessitade de realização de operações de contagem e soma.

Tabela 14 – Comparativo entre os tempos(em milisegundos) de respostas do MySQL e do Neo4j para as consultas analíticas

	Tempo para cada consulta Analítica	
	A1	**A2**
MySQL	3.1	3.3
Neo4j	33.5	24.8

Fonte: Elaborado pela autora

Com relação ao aspecto subjetivo de facilidade de programação, os bancos de dados relacionais utilizam a linguagem SQL, amplamente conhecida entre os programadores e que possui API's para implementação nas mais variadas linguagens de programação, em contraste com os bancos de dados Grafos que ainda não possuem

uma linguagem padrão, pois apesar de utilizarmos Cypher neste trabalho, ela não é a única linguagem de consulta em Grafos, com isso, essas linguagens de consulta, ainda possuem poucas API's.

Comparando em específico o Cypher com o SQL, foi possível notar nas consultas realizadas, que com Cypher as consultas ficam mais claras e compreensivas, constratando com as consultas em SQL, onde é necessário fazer várias deduções por causa das junções.

5 Conclusão

O objetivo deste trabalho foi apresentar os principais conceitos de modelagem relacional e orientada a grafos, demonstrando a construção, relacionamentos, propriedades, utilização, armazenamento e gerenciamento de cada uma. Além de realizar uma análise comparativa entre os dois modelos, por meio de um estudo de caso.

Um dos grandes desafios para o desenvolvimento deste trabalho foi encontrar materiais de estudo de qualidade sobre a modelagem de dados orientada a grafos, principalmente em português, pois a mesma é uma tecnologia relativamente recente.

Com as comparações realizadas entre as modelagens, percebeu-se que existem vantagens em se utilizar banco de dados orientado a grafos para buscas em conjunto de dados altamente relacionados e quando a pesquisa realizada envolve buscas em ramos dos grafos ou em profundidade, as quais foram chamadas de consultas estruturais. Entretanto, em consultas que envolvem buscas, obrigatoriamente, em todas as entidades do conjunto de dados, chamadas neste trabalho de analíticas, o modelo relacional foi mais eficiente.

No ínicio do projeto é difícil avaliar a complexidade, devido o nível de abstração. Porém deve ser levado em conta a produtividade do programador e o desempenho no acesso aos dados. Banco de dados de grafos oferecem uma simplificação diferente, visto que o relacional possuem muitos relacionamentos. Os bancos de dados grafos também oferecem uma API de armazenamento mais natural, rapidez na recuperação de dados altamente conectados e acesso ráido a grandes quantidades de dados.

Por conseguinte, acredita-se que a integração entre os dois modelos é a melhor solução para as demandas de consultas de dados atuais, já que geralmente utilizar um único mecanismo de banco de dados para todas as necessidades poderá levar a soluções de baixo desempenho, tendo visto que mecanismos dos bancos de dados são desenvolvidos para executar apenas certas operações em determinadas estruturas e quantidades de dados. Desse modo, a programação poliglota, ou seja, a mistura de modelos, levará ao maior aproveitamento para aplicações complexas que combinem diferentes tipos de problemas. Algumas ferramentas já estão sendo desenvolvidas para facilitar a capacidade de comunicação entre os bancos de dados NoSQL e relacional.

5.1 Trabalhos Futuros

Os resultados obtidos foram satisfatórios para este trabalho, mas podem ser melhorados. Dessa forma, como trabalhos futuros propõe-se a utilização de maior

variedade de consultas e conjunto de dados para a apresentação de resultados mais expressivos entre as duas modelagens.

Além disso, propõe a implementação da integração entre as modelagens relacionais e orientadas a grafo para a análise e verificação das vantagens da utilização de uma modelagem híbrida, bem como, o estudo de ferramentas com capacidade de integração entre os diversos Banco de Dados NoSQL e Banco de Dados Relacional.

Referências

AEROSKIPE. What is a key-value store? 2011. Acessado em: 2017-07-12. Disponível em: <http://www.aerospike.com/what-is-a-key-value-store/>. Citado na página 18.

ALMEIDA, A. Neo4j na prática. 2011. Acessado em: 2017-07-12. Disponível em: <http://www.univale.com.br/unisite/mundo-j/artigos/51Neo4j.pdf>. Citado 2 vezes nas páginas 25 e 26.

ALVAREZ, G. M.; CECI, F.; GONçALVES, A. L. Análise comparativa dos bancos orientados a grafos de primeira e segunda geração – uma aplicação na análise social. *III Encontro de Inovação em SI, Florianópolis, Santa Catarina*, 2016. Citado na página 26.

AMAZON. Amazon dynamodb. 2018. Acessado em: 2018-07-12. Disponível em: <https://aws.amazon.com/pt/dynamodb/>. Citado na página 19.

AMAZON. Amazon simpledb. 2018. Acessado em: 2018-07-28. Disponível em: <https://aws.amazon.com/pt/simpledb/>. Citado na página 21.

APACHE. Data where you need it. 2018. Acessado em: 2018-07-22. Disponível em: <http://couchdb.apache.org/>. Citado na página 20.

APACHE. Welcome to apache hbase™. 2018. Acessado em: 2018-07-28. Disponível em: <https://hbase.apache.org/>. Citado na página 21.

APACHE. What is cassandra? 2018. Acessado em: 2018-07-28. Disponível em: <http://cassandra.apache.org/>. Citado na página 21.

CODE, G. Scalable, elastic, consistent document store. 2018. Acessado em: 2018-07-22. Disponível em: <https://code.google.com/archive/p/terrastore/>. Citado na página 20.

CUNG, H.-Q.; JEDIDI, M. Implementing a recommender system with graph database. *Universite de Fribourg*, Universite de Fribourg, p. 3–24, 2014. Citado na página 13.

DB-ENGINES. Flockdb system properties graph. 2018. Acessado em: 2018-07-29. Disponível em: <https://db-engines.com/en/system/FlockDB>. Citado na página 24.

DB-ENGINES. Hypertable system properties. 2018. Acessado em: 2018-07-28. Disponível em: <https://db-engines.com/en/system/Hypertable>. Citado na página 21.

DB, H. Moving on. 2018. Acessado em: 2018-07-12. Disponível em: <https://projecthamster.wordpress.com/>. Citado na página 19.

EDUARDO, N. *Bem vindo ao Neo4j*. [S.l.]: Nicholasess, 2015. <http://nicholasess.com.br/neo4j-2/bem-vindo-ao-neo4j/>. Citado na página 26.

ELMASRI, R.; NAVATHE, S. B. *Sistemas de banco de dados*. Sexta. [S.l.]: Pearson Education, 2011. Citado 2 vezes nas páginas 15 e 17.

HARPER, F. M.; KONSTAN, J. A. The movielens datasets: History and context. In: *ACM Transactions on Interactive Intelligent Systems (TiiS)*. [S.l.]: ACM, 2015. v. 4, n. 19. Citado na página 35.

HOWS, P. M. D.; PLUGGE, E. *Introdução ao MongoDB*. [S.l.]: Novatec, 2015. Citado 2 vezes nas páginas 20 e 21.

KALIYAR, R. kumar. Graph databases: A survey. *International Conference on Computing, Communication and Automation*, ACM, 2015. Citado na página 25.

MARQUESONE, R. *Big Data: Técnicas e tecnologias para extração de valor dos dados*. [S.l.]: Casa do Código, 2016. Citado 3 vezes nas páginas 21, 22 e 26.

MEMCACHED. What is memcached? 2018. Acessado em: 2018-07-12. Disponível em: <https://memcached.org/>. Citado na página 19.

MONGODB. The database for modern applications. 2018. Acessado em: 2018-07-22. Disponível em: <https://www.mongodb.com/>. Citado na página 20.

MPINDA, S. A. T.; FERREIRA, L. C.; SANTOS, M. T. P. Evaluation of graph databases performace through indexing techniques. *International Journal of Artificial Intelligence and Applications*, v. 6, n. 5, 2015. Citado na página 13.

NEO4J. What is neo4j? 2018. Acessado em: 2018-07-29. Disponível em: <https://neo4j.com/>. Citado na página 24.

OBJECTIVITY. Infinite graph. 2018. Acessado em: 2018-07-29. Disponível em: <https://www.objectivity.com/products/infinitegraph/>. Citado na página 24.

ORACLE. Oracle berkeley db downloads. 2018. Acessado em: 2018-07-12. Disponível em: <https://www.oracle.com/technetwork/database/database-technologies/berkeleydb/downloads/index.html>. Citado na página 19.

ORIENTDB. The database designed for the modern world. 2018. Acessado em: 2018-07-22. Disponível em: <https://orientdb.com/>. Citado 2 vezes nas páginas 20 e 24.

PANIZ, D. *NoSQL: Como armazenar os dados de uma aplicação moderna*. [S.l.]: Casa do Código, 2016. Citado 2 vezes nas páginas 24 e 25.

PENTEADO, R. R. M.; SCHROEDER, R.; HOSS, D.; NANDE, J.; MAEDA, R. M.; COUTO, W. O.; HARA, C. S. Um estudo sobre bancos de dados em grafos nativos. Escola Regional de Banco de Dados (ERBD 2014), 2014. Citado na página 13.

RAVENDB. Ravendb nosql document databased. 2018. Acessado em: 2018-07-22. Disponível em: <https://ravendb.net/>. Citado na página 20.

REDIS. Redis nosql database. 2018. Acessado em: 2018-07-12. Disponível em: <https://redis.io/>. Citado na página 19.

RIAK. Enterprise nosql database. 2018. Acessado em: 2018-07-12. Disponível em: <https://riak.com/>. Citado na página 19.

ROBINSON, I.; WEBBER, J.; EIFREM, E. *Graph Databases*. Second. [S.l.]: O'Reilly Media, 2015. Citado 3 vezes nas páginas 13, 27 e 28.

SADALAGE, P. J.; FOWLER, M. *NoSQL Essencial*. [S.l.]: Novatec, 2013. Citado 8 vezes nas páginas 15, 16, 18, 19, 22, 24, 25 e 27.

VOLDEMORT. Voldemort is a distributed key-value storage system. 2018. Acessado em: 2018-07-12. Disponível em: <https://www.project-voldemort.com/voldemort/>. Citado na página 19.