UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE SISTEMAS DE INFORMAÇÃO

Amanda Christoval da Veiga de Aquino

UMA ANÁLISE COMPARATIVA DE SISTEMAS DE GERENCIAMENTO DE BANCOS DE DADOS NoSQL MULTIMODELO

Florianópolis
2019

Amanda Christoval da Veiga de Aquino

UMA ANÁLISE COMPARATIVA DE SISTEMAS DE GERENCIAMENTO DE BANCOS DE DADOS NoSQL MULTIMODELO

Trabalho Conclusão do Curso de Graduação em Sistemas de Informação do Departamento de Informática e Estatística da Universidade Federal de Santa Catarina como requisito para a obtenção do título de Bacharel em Sistemas de Informação
Orientador: Prof. Dr. Ronaldo dos Santos Mello

Florianópolis

2019

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Aquino, Amanda Christoval da Veiga de
UMA ANÁLISE COMPARATIVA DE SISTEMAS DE GERENCIAMENTO DE BANCOS DE DADOS NoSQL MULTIMODELO / Amanda Christoval da Veiga de Aquino ; orientador, Ronaldo dos Santos Mello, 2019.
73 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Sistema de Informação, Florianópolis, 2019.

Inclui referências.

1. Sistema de Informação. 2. SGBD multimodelo. 3. NoSQL. 4. análise comparativa. I. Mello, Ronaldo dos Santos . II. Universidade Federal de Santa Catarina. Graduação em Sistema de Informação. III. Título.

Amanda Christoval da Veiga de Aquino

UMA ANÁLISE COMPARATIVA DE SISTEMAS DE GERENCIAMENTO DE BANCOS DE DADOS NoSQL MULTIMODELO

Este Trabalho Conclusão de Curso foi julgado adequado para obtenção do Título de "Bacharel em Sistemas de Informação" e aprovado em sua forma final pelo Curso Sistemas de informação.

Florianópolis, 15 de setembro de 2019.

_____

Prof. Cristian Koliver, Dr.
Coordenador do Curso

Banca Examinadora:

_____

Prof. Ronaldo dos Santos Mello, Dr.
Orientador

_____

Angelo Augusto Frozza
Avaliador

_____

Geomar André Schreiner
Avaliador

Este trabalho é dedicado a mim, amigos e a minha família que sempre estiveram comigo.

AGRADECIMENTOS

Agradeço primeiramente a Deus que sempre esteve comigo e me mostrou que há luz no fim do túnel em momentos que nem eu mesma acreditava. Também aos meus familiares que sempre incentivaram meus estudos, ressalto um agradecimento aos meus pais Rita e Márcio que nunca duvidaram da minha capacidade e foram fundamentais durante esse processo. Um agradecimento especial aos meus amigos que estiveram junto comigo nessa jornada da faculdade sempre me apoiando, seja por meio de vibrações positivas, palavras de honestidades, ou apenas ficando a meu lado no momentos bons e ruins. Agradeço também aos professores e mentores que tive durante essa experiência, que com paciência me ensinaram e tornaram a profissional que sou hoje. Por último, mas não menos importante, agradeço ao meu orientador que me ajudou e aconselhou durante todo o trabalho de conclusão de curso.

RESUMO

O aumento crescente do número de aplicações na Web, ou que utilizam dados da Web, como redes sociais e aplicações voltadas à Internet das coisas, teve um papel importante para o aumento exponencial de dados. Esses dados são obtidos das mais diversas fontes e assim se tornam mais complexos, possuem maior variedade e tem uma alta taxa de crescimento. Nesse sentido, é necessário desenvolver tecnologias de gerenciamento de dados que consigam suportar as distintas características de cada grupo de dados e ainda possuir um bom desempenho. Para obter esses resultados foram criados os sistemas de gerenciamento de banco de dados (SGBDs) chamados de NoSQL multimodelo. Eles oferecem mais funcionalidades e flexibilidade, podendo suportar vários modelos de dados em um único SGBD. Para os desenvolvedores que necessitam utilizar tais sistemas ainda não é claro qual solução combina e se adapta melhor ao seu projeto, pois essas tecnologias são consideradas novas no mercado. Na literatura é possível encontrar trabalhos comparando SGBDs, entretanto os que comparam SGBDs NoSQL multimodelo são escassos e não abrangem muitos sistemas. Sendo assim, esse trabalho de conclusão de curso tem como objetivo apresentar uma análise comparativa de SGBDs NoSQL multimodelo populares no mercado e não abordados por trabalhos relacionados.

Palavras-chave: SGBD multimodelo. NoSQL, análise comparativa.

ABSTRACT

The increasing number of web applications, or those using web data, such as social networks and IoT applications, has played an important role in exponentially increasing data. These data are obtained from various sources and thus become more complex, have greater variety and have a high growth rate. In this sense, it is necessary to develop data management technologies that can support the distinct characteristics of each data group and still have a good performance. To achieve these results, database management systems (DBMSs) called multi-model NoSQL have been created. They offer more functionality and flexibility and may support multiple data models in a single DBMS. For developers who need to use such systems it is not yet clear which solution best fits their project as these technologies are considered new to the market. In the literature it is possible to find out works comparing DBMSs, however those comparing multimodel NoSQL DBMSs are scarce and do not cover many systems. Thus, this undergraduate conclusion paper aims to present a comparative analysis of the most used multimodel NoSQL DBMS in the market that were not covered by the related work.

Keywords: Database. NoSQL multimodel. multimodel. comparative analysis

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SGBD – Sistema de Gerenciamento de Banco de Dados

NoSQL – Not Only SQL

SQL - Structured Query Language

DBMS - Database Management System

AQL - Arango Query Language

HTTP - Protocolo de Transferência de Hipertexto

MVCC - Multi-Version Concurrency Control

BD - Banco de Dados

OCC - Optimistic Concurrency Control

ACID - Atomicidade, Consistência, Isolamento e Durabilidade

SO - Sistema Operacional

IOT - Internet of Things

SUMÁRIO

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

Atualmente, uma grande quantidade de dados está sendo gerada a partir de fontes de dados diferentes como smartphones, computadores, sensores, câmeras, sistemas de posicionamento global, sites de redes sociais, transações comerciais e jogos (HASHEM et al., 2016). Com a ajuda de ferramentas de análise é possível extrair a partir desses dados inúmeras informações e aplicá-las no dia a dia. Contudo, essa não é uma tarefa fácil. Um dos desafios mais importantes para a comunidade de pesquisadores de banco de dados (BD) nos últimos anos tem sido o desenvolvimento de tecnologias para gerenciar essa grande quantidade de dados heterogêneos gerados em uma alta taxa de velocidade por aplicativos e pessoas (STONEBRAKER, 2012).

Ainda, a atual tecnologia de BD relacional não é adequada para lidar com essas grandes quantidades de dados altamente heterogêneos devido ao alto overhead com o controle da consistência dos dados. Isso incentivou a comunidade de BD a buscar novas soluções de armazenamento para esses dados. Uma dessas soluções são os Sistemas de Gerência de BDs (SGBDs) NoSQL (FOWLER; SADALAGE, 2013). Os sistemas NoSQL trabalham com armazenamento distribuído, não utilizando o modelo de dados relacional e nem sempre a linguagem SQL (ESTEFANI, 2017). Eles são uma família de modelos de dados, sendo os quatro principais o chave-valor, colunar, documento e grafo.

Como uma resposta direta a necessidade, sistemas de gerência de banco de dados (SGBDs) que suportam vários modelos surgiram para atender a necessidade de múltiplos formatos de dados, reduzindo a complexidade operacional e mantendo a consistência global dos dados (LU; HOLUBOVÁ, 2017). O surgimento desses SGBDs tem como motivação as aplicações cujos requisitos não se ajustam a apenas um modelo de dados. Aplicações com essas características, ou seja, que necessitam acessar dois ou mais SGBDs com modelos de dados diferentes são chamadas de “poliglotas”, uma vez que realizam persistência de dados em diversos formatos devido à natureza heterogênea dos dados que manipulam.

O casamento de SGBDs NoSQL com persistência poliglota é aqui denominado de SGBD NoSQL multimodelo. Este tipo de solução se caracteriza por ser um único SGBD que

persiste, indexa e consulta dados mantidos em vários modelos de dados NoSQL diferentes. Isso evita o acesso a vários SGBDs NoSQL para gerenciar os dados de uma aplicação.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

O objetivo deste trabalho de conclusão de curso é apresentar e realizar uma análise comparativa dos SGBDs NoSQL multimodelo mais conhecidos, ou seja, compará-los em termos de suporte à administração, manipulação e armazenamento de dados e, desta forma, auxiliar a tomada de decisão de profissionais na área quando se deparam com a escolha de um SGBD que atenda melhor as suas necessidades.

1.2.2 Objetivos Específicos

Os objetivos específicos deste trabalho são:

- Identificar e definir os principais SGBDs multimodelo que utilizam um ou mais modelos de dados NoSQL para fins de análise com base na documentação disponível;
- Definir critérios de análise dos SGBDs escolhidos;
- Avaliar os SGBDs estudados com base nos critérios de análise definidos;

1.3 JUSTIFICATIVA

Com o aumento de dados heterogêneos sendo criados diariamente aumentou-se a preocupação de como armazenar esses dados de forma a gerenciá-los de maneira mais eficiente. Isso acarreta no aumento do número de ferramentas para manipular e armazenar esses dados, tornando primordial o conhecimento dessas ferramentas.

No entanto, as demandas para simplificar a interação entre aplicativos e SGBDs com interfaces simples de armazenamento e manipulação de dados nem sempre é possível usando apenas o modelo relacional (LIU; LU, 2018). Para suprir essa necessidade surgiram os SGBDs NoSQL multimodelo, que têm a capacidade de incorporar vários modelos de dados e

permite os usuários manipular todos os modelos de dados declarativamente. Portanto, compreender as opções de SGBDs NoSQL multimodelo é essencial no processo de desenvolvimento de aplicações principalmente para o engenheiro de software ou o projetista de BD. Entender como esses sistemas funcionam é de alta importância para que se escolha o SGBD que mais se encaixa melhor às necessidades da aplicação que está sendo desenvolvida.

Com esse intuito, este trabalho de conclusão de curso apresenta e analisa alguns SGBDs NoSQL multimodelo para auxiliar no entendimento e na seleção de qual tecnologia pode ajudar melhor a um engenheiro de software ou projetista de BD. Cabe ressaltar que este trabalho não avalia esses SGBDs em termos de desempenho, utilizando, por exemplo, benchmarks. O foco aqui é avaliá-los com base nas documentações disponíveis sobre os mesmos. Uma avaliação de desempenho pode ser alvo de um trabalho futuro.

1.4 METODOLOGIA

O método de pesquisa definido para confecção deste trabalho está dividido em quatro etapas. A primeira etapa consiste no levantamento do estado da arte sobre "SGBDs NoSQL multimodelos". Para isso, foram realizadas pesquisas em livros, artigos e outras documentações sobre os conceitos relacionados ao trabalho visando adquirir subsídios para a organização desse trabalho.

Uma vez adquirido esse conhecimento, a segunda etapa é o levantamento dos SGBDs NoSQL multimodelos existentes no mercado e a escolha dos principais para realização deste trabalho. Na sequência, a terceira etapa constituiu no levantamento e definição de critérios de análise sobre os SGBDs escolhidos tendo como base documentações disponíveis e o conhecimento adquirido.

Por fim, na quarta etapa é realizada uma análise comparativa dos SGBDs escolhidos a partir dos critérios definidos na terceira etapa. Como resultado da execução dessas etapas, produziu-se a monografia deste trabalho de conclusão de curso.

1.5 ESTRUTURA DO TRABALHO

O trabalho está dividido em oito capítulos. Este primeiro apresenta a introdução ao assunto. O segundo apresenta a fundamentação teórica incluindo os SGBDs NoSQL, que é

um dos focos deste trabalho. No terceiro capítulo são detalhados os quatro SGBDs NoSQL multimodelo que são comparados neste trabalho. No capítulo seguinte são apresentados os trabalhos correlatos e como este se diferencia dos demais. O quinto capítulo apresenta os critérios de análise que são utilizados. No sexto é apresentado a análise dos quatro SGBDs NoSQL multimodelo conforme os critérios definidos. Por fim, o sétimo capítulo apresenta as conclusões e trabalhos futuros.

2 BANCO DE DADOS NOSQL

Este capítulo apresenta as principais características e modelos de dados dos BDs NoSQL, que é a família de BDs considerada neste trabalho. Os SGBDs NoSQL são uma classe de SGBD que não atendem ao modelo relacional (FOWLER; SADALAGE, 2013). Eles também são chamados de "Não Somente SQL", enfatizando que geralmente a SQL não é utilizada como única linguagem de consulta, como em BDs relacionais. Suas principais características são métodos de acesso simples, escaláveis para grandes volumes de dados e não exigem que um esquema seja definido a priori (schemaless).

Algumas das principais características almejadas nos BDs NoSQL são o alto desempenho e alta disponibilidade. Contudo, para obter esse resultado, eles não priorizam a consistência dos dados. Assim sendo, eles são capazes de processar um grande número de operações simples de leitura e gravação por segundo, mas não suportam todas as propriedades transacionais ACID (Atomicidade, Consistência, Isolamento e Durabilidade), ou seja, as atualizações são garantidas em algum momento futuro a sua execução, não havendo, assim, garantia de consistência imediata nas leituras (CATTEL, 2011). Conclui-se, assim, que a idéia dos SGBDs NoSQL é renunciar às restrições do ACID para alcançar alto desempenho e escalabilidade.

As propriedades de transação de um sistema de banco de dados ACID consistem em Atomicidade, Consistência, Isolamento e Durabilidade. A Atomicidade significa que em uma transação envolvendo duas ou mais partes de informações discretas, ou a transação será executada totalmente ou não será executada, garantindo assim que as transações sejam atômica. A Consistência, por sua vez, é quando uma transação cria um novo estado válido dos dados ou, em caso de falha, retorna todos os dados ao seu estado anterior ao início da transação. Isolamento significa que uma transação em andamento, mas ainda não validada, deve permanecer isolada de qualquer outra operação externa, ou seja, garante-se que a transação não será interferida por nenhuma outra transação concorrente. Por último, a Durabilidade indica que dados validados são registados pelo sistema de tal forma que, mesmo no caso de uma falha ou reinício do sistema, os dados estão disponíveis em seu estado correto.

É possível concluir que a intenção dos SGBDs NoSQL, ao renunciar às propriedades ACID, é poder alcançar desempenho e escalabilidade superiores. De acordo com Rick Carttell (2011), os diferenciais dos SGBDs NoSQL são os seguintes:

1. A capacidade de dimensionar horizontalmente os dados em muitos servidores;
2. A capacidade de replicar e distribuir (particionamento) dados em muitos servidores;
3. Uma interface ou protocolo simples de nível de chamada (em contraste com uma instrução SQL);
4. Um modelo de concorrência mais fraco que o tradicional ACID dos BDs relacionais;
5. Uso eficiente de índices distribuídos e RAM para armazenamento de dados;
6. A capacidade de adicionar dinamicamente novos atributos ao registros de dados.

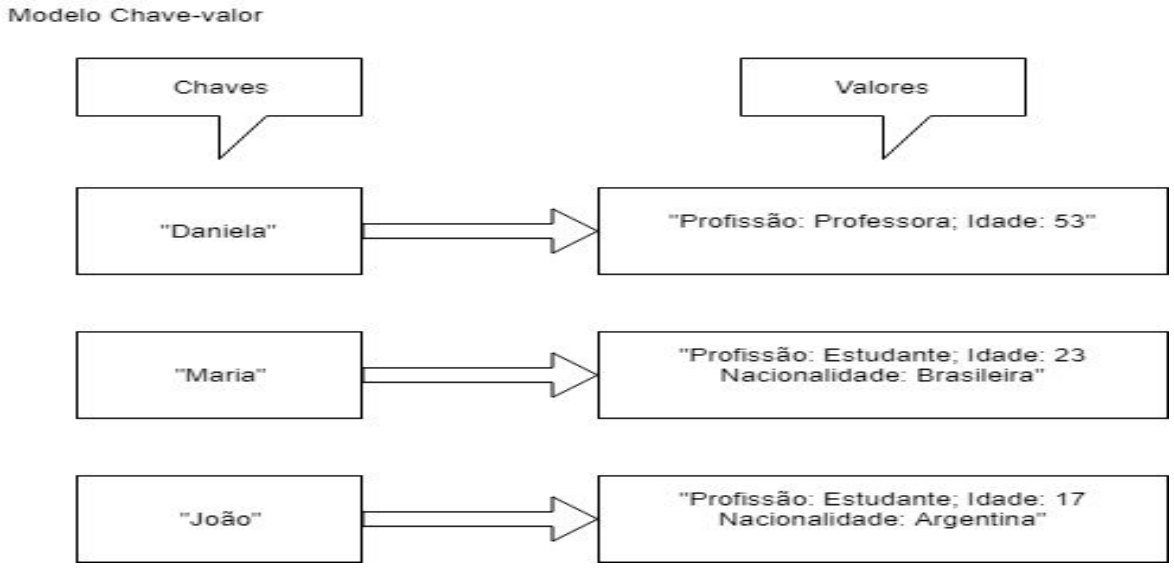
Os SGBDs NoSQL apresentam modelos de dados heterogêneos. Os modelos de dados mais comuns são o colunar, chave-valor, documento e grafo. Eles são detalhados nas próximas seções.

2.1 MODELO CHAVE-VALOR

O modelo chave-valor utiliza uma tabela hash ou uma tabela com varias chaves no qual cada chave referencia um valor. É o modelo de dados mais simples, possibilitando consultas a objetos apenas pela sua chave, sendo que essa chave pode suportar atributos simples ou multivalorados. Alguns dos SGBD utilizam esse modelo de dados são o Voldemort e o Redis.

Uma das vantagens do modelo chave-valor é a sua facilidade de implementação e de adição de novos dados em tempo de execução, uma vez que o valor do dado é visto como uma “caixa preta” pelo SGBD. No entanto, ele não suporta relacionamentos entre dados, definição de esquema e linguagem de consulta.

Figura 1 – Exemplo de modelo chave-valor.



Fonte: Autor (2018)

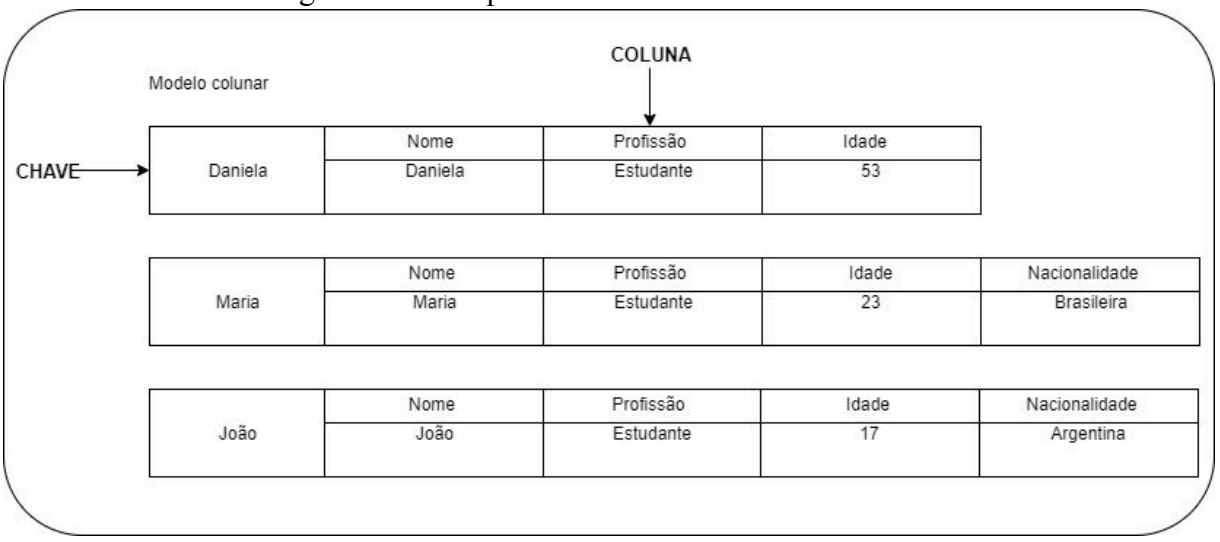
A Figura 1 apresenta um exemplo simples de uma modelagem chave-valor.

2.2 MODELO ORIENTADO A COLUNAS

O modelo orientado a colunas, também chamado de orientado a colunas, utiliza-se de tabelas para representação de entidades e os dados são mantidos agrupados por colunas (MATEI, 2010). Este modelo é muito utilizado para grande massas de dados devido a sua grande escalabilidade horizontal, contudo não suporta relacionamentos entre dados. Alguns SGBDs que utilizam esse modelo de dados são Cassandra e HBase.

Os conceitos utilizados neste modelo são o *column family*, semelhante a uma tabela no modelo relacional, e o conjunto de colunas que representa uma tupla. Neste caso, uma coluna é composta por um nome e um valor, e cada conjunto de colunas é acessado a partir do seu nome (chave).

Figura 2 – Exemplo de modelo orientado a colunas.



Fonte: Autor (2018)

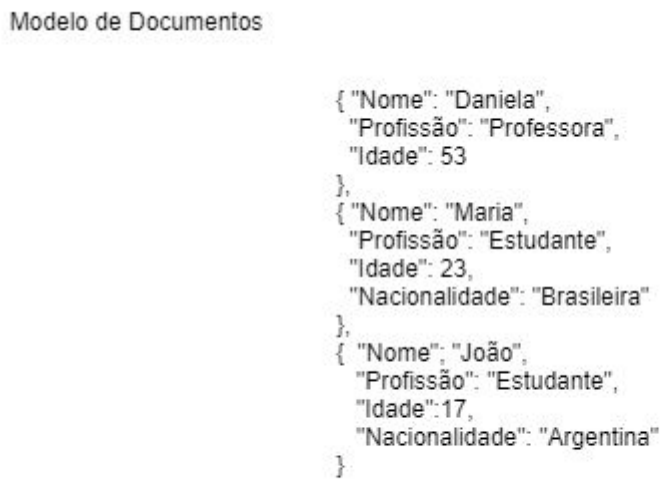
A Figura 2 apresenta um exemplo de uma modelagem colunar. Cada registro é composto por um conjunto de colunas, contudo não é necessário que todos os registros possuam as mesmas colunas, como é o caso dos registros “Daniela” e “Maria”.

2.3 MODELO ORIENTADO A DOCUMENTOS

O modelo de documento organiza os dados em uma coleção de documentos, sendo um documento um objeto com um código único e um conjunto de atributos. Essas atributos podem conter dados atômicos, documentos aninhados ou ainda listas. Em geral, os BDs orientados a documentos não possuem esquema, ou seja, os documentos armazenados não precisam ter uma estrutura em comum. Essa característica permite que sejam adicionados novos campos sem causar algum problema no BD. Assim sendo, este modelo é uma boa opção para o armazenamento de dados semiestruturados. As vantagens desse modelo são a possibilidade de indexar valores de atributos e possuir dados organizados em listas e subobjetos. Contudo, ele não suporta relacionamentos entre dados.

A maioria dos SGBDs baseados em documentos utilizam o JSON[1], uma notação em Java para representar objetos de forma simplificada, como estrutura básica. Entretanto, existem alguns SGBDs que escolhem o formato de armazenamento XML. Exemplos de SGBDs baseados em documentos são o MongoDB e o CouchDB.

Figura 3 – Exemplo de modelo orientado a documentos.



Fonte: Autor (2018)

A Figura 3 apresenta um exemplo de um documento no formato JSON, que é suportado pela grande maioria dos SGBDs baseados em documentos. Ele mostra vários objetos (pessoas) com atributos atômicos. Já a Figura 4 mostra um documento JSON mais complexo, composto por listas e subobjetos.

[1] http://json.org.

Figura 4 – Exemplo de modelo completo de documentos.

Modelo de Documentos

```
{  "Nome": "Daniela",
   "Profissão": "Professora",
   "Idade": 53,
   "Universidade": {
                "Chave": 001,
                "Nome": "Universidade Federal de Santa Catarina",
                "Sigla": "UFSC"
                  },
   "Disciplinas": ["INE5619", "INE5620", "INE5430"]
},
{  "Nome": "Maria",
   "Profissão": "Estudante",
   "Idade": 23,
   "Nacionalidade": "Brasileira",
   "Universidade": {
                "Chave": 001,
                "Nome": "Universidade Federal de Santa Catarina",
                "Sigla": "UFSC"
               }
},
{  "Nome"; "João",
   "Profissão": "Estudante",
   "Idade":17,
   "Nacionalidade": "Argentina"
   "Universidade": [
                {
                        "Chave": 001,
                        "Nome": "Universidade Federal de Santa Catarina",
                        "Sigla": "UFSC"
                 },
                {
                        "Chave": 002,
                        "Nome": "Universidade de Buenos Aires",
                        "Sigla": "UBA"
                 }
       ]
}
```

Fonte: Autor (2018)

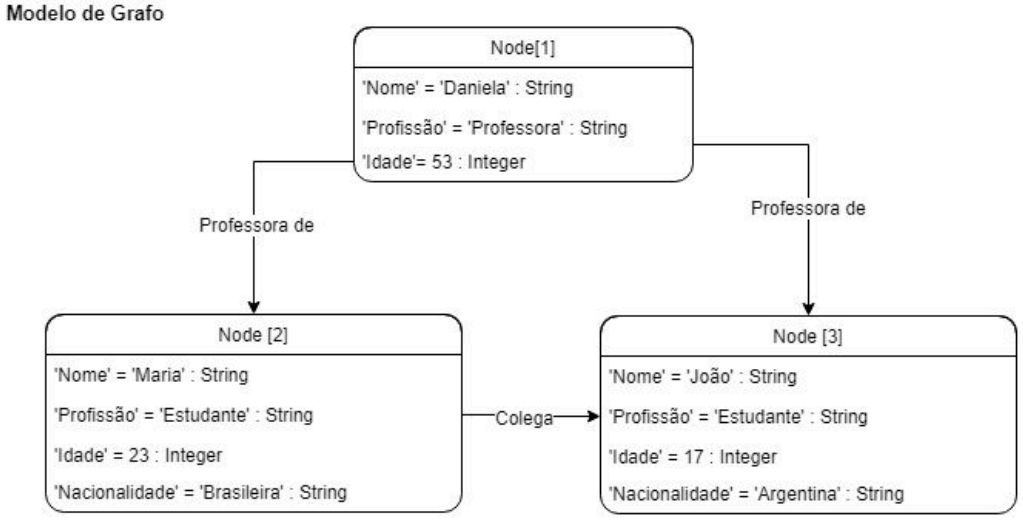
2.4 MODELO DE GRAFO

O processo de transformação de um modelo conceitual para um modelo lógico tem consequências como a perda ou diminuição da capacidade de representar o modelo conceitual em sua plenitude. O modelo de grafo, por sua vez, faz com que as entidades e também os seus relacionamentos sejam representados conforme o modelo conceitual praticamente anulando a perda de informações neste processo.

O modelo de grafo é composto basicamente por três componentes: os nós, representados pelos vértices, os relacionamentos, representados pelas arestas e as propriedades, ou seja os atributos das entidades e dos relacionamentos. Esse modelo é indicado quando informações sobre interconectividade são importantes. Alguns exemplos de SGBD NoSQL que utilizam esse modelo são Neo4J e Infinity.

Este modelo é vantajoso pois possibilita a criação de consultas mais complexas frente aos outros modelos. Seu diferencial é o ganho de desempenho pelo fato de não ter tanta replicação de dados levando em conta os outros modelos. Em modelos como o orientado a documentos, por exemplo, cada documento armazena todas as suas relações de forma atômica, sendo assim, se houver dois documentos, um representando a pessoa “X” e outro representando a pessoa “Y” e ambos conhecem a pessoa “Z”, tanto o documento “X” quanto “Y” teriam que guardar uma instância da pessoa “Z”. No caso do modelo de grafo, isso é resolvido apenas utilizando ligações com arestas que armazenam um valor que indica que a pessoa conhece “Z”. A Figura 5 mostra um exemplo de modelagem de grafo, mostrando um relacionamento através das setas “Professora de” e “Colega” entre os nós.

Figura 5 – Exemplo de modelo de grafo.



Fonte: Autor (2018)

3 SGBDs NOSQL MULTIMODELO

À medida que um sistema vai se tornando mais complexo, agregando novos dados com formatos distintos, volumes elevados e necessidades especiais de consulta ou transformação, surge naturalmente a (re)discussão sobre a camada de armazenamento da solução (LU;HOLUBOVÁ, 2017), surgindo à necessidade de um SGBD que suporte vários modelos ao mesmo tempo.

Diferente dos tradicionais SGBDs, que são estruturados sobre um único modelo de dados, os NoSQL multimodelos são concebidos com o intuito de integrar e suportar diferentes modelos de dados. Para a realização deste trabalho foram considerados os 4 SGBDs NoSQL multimodelo mais conhecidos e relevantes no mercado, conforme mostra a Tabela 1. Os quatros SGBDs selecionados são citados na DB Ranking, que é uma lista que classifica os SGBDs de acordo com sua popularidade[2]. É importante ressaltar que por serem tecnologias novas no mercado é natural que não estejam em posições altas. Outro motivo da escolha dos SGBDs foi as documentações disponibilizadas, que são mais abrangentes e detalhadas permitindo uma melhor análise para o trabalho atual. Suas principais características são descritas nas próximas seções.

Tabela 1 – SGBDs NoSQL multimodelo escolhidos

SGBD	**Link**
ArangoDB	https://www.arangodb.com/
OrientDB	https://orientdb.com/
CrateDB	https://crate.io/
Marklogic	https://www.marklogic.com/

Fonte: Autor (2019).

3.1 ARANGODB

O ArangoDB é um SGBD NoSQL de código aberto desenvolvido pela TRIAGENS GmbH, escrito na linguagem C++. Ele é um SGBD nativo multimodelo, sendo concebido para armazenar seus dados como pares chave/valor, grafos ou documentos, e acessar qualquer representação de dados usando uma única linguagem de consulta declarativa. Um dos benefícios do ArangoDB é utilizar um único *backend* para gerenciar seus diferentes

[2] *https://db-engines.com/en/ranking*

modelos de dados. Além disso, ele possui suporte a transações ACID, oferecendo forte consistência em uma única instância e operações atômicas ao operar no modo de cluster.

O SGBD é classificado como schemaless, não obrigando o usuário a nenhuma estrutura pré-definida. Uma de suas características distintas é possuir uma linguagem unificada própria chamada AQL (Arango Query Language), que permite consultar dados em diferentes modelos suportados pelo ArangoDB. Ela é inspirada na linguagem SQL, no entanto, a AQL não suporta operações de definição de dados, como criar e excluir bancos de dados, coleções e índices. As declarações são enviadas via HTTP a partir da aplicação ArangoDB, sendo retornada uma resposta no formato JSON. A Figura 6 apresenta um exemplo de inserção de um documento em uma coleção utilizando a linguagem AQL. As aspas em torno das chaves de atributo são opcionais.

Figura 6 – Exemplo de *insert* de documentos em AQL.

```
INSERT {
    "name": "Ned",
    "surname": "Stark",
    "alive": true,
    "age": 41,
    "traits": ["A","H","C","N","P"]
} INTO Characters
```

Fonte: ArangoDB (2018)

No ArangoDB o armazenamento do dados de grafo é feito no modelo de documentos, com documentos ligando outros documentos através de referências entre as coleções de documentos. Esses documentos especiais que armazenam as ligações entre os documentos vértices são chamados de documentos Edge (Arestas). Os Edges possuem atributos denominados *_from* e *_to* com os identificadores dos vértices. Os algoritmos de grafos foram adaptados para usar essas estruturas, realizando acesso a índices apropriados para acelerar o busca nos documentos (Weinberger, 2016).

Figura 7 – Exemplo de consulta em grafo

```
FOR c IN Characters
    FILTER c.name == "Ned"
    FOR v IN 1..1 INBOUND c ChildOf
        RETURN v.name
```

Fonte: ArangoDB (2019)

A Figura 7 apresenta uma consulta utilizando AQL para buscar os filhos (relações) de “Ned” no qual os dados foram modelados em grafos. A expressão INBOUND é utilizada para buscar as relações em direção do nó pai para os nós filhos, sendo ”1..1” o número de passos (relações) entre um nó e outro. Caso seja necessário a busca dos nós filhos para os pais a expressão a ser utilizada é OUTBOUND.

Embora a linguagem AQL seja a principal escolha para consultas, ela não é a única forma de acessar os dados. O ArangoDB possui outras duas formas, sendo elas uma interface REST que fornece medidas simples para criação, acesso e manipulação dos dados usando o identificador do documento ou consulta. A outra forma de acesso é o uso do JavaScript, que é incorporado ao ArangoDB permitindo que todas as partes da API sejam diretamente acessíveis a partir do JavaScript, sendo isso um grande diferencial. O JavaScript Driver permite a criação, manipulação e consulta de dados. Na Figura 8 é possível visualizar as operações de insert e update utilizando o Javascript. Já a Figura 9 apresenta uma consulta que retorna usuários maiores de 12 anos utilizando a linguagem AQL em Javascript.

Figura 8 – Exemplo das operações insert e update utilizando Javascript

```
const db = new Database();
const collection = db.collection("some-collection");
const doc = { number: 1, hello: "world" };
const doc1 = await collection.save(doc);
const doc2 = await collection.update(doc1, { number: 2 });
```

Fonte: ArangoDB (2019)

Figura 9 – Exemplo de consulta em Javascript

```
const db = new Database();
const action = String(function(params) {
  // This code will be executed inside ArangoDB!
  const db = require("@arangodb").db;
  return db
    ._query(
      aql`
    FOR user IN _users
    FILTER user.age > ${params.age}
    RETURN u.user
  `
    )
    .toArray();
});

const result = await db.transaction({ read: "_users" }, action, { age: 12 });
// result contains the return value of the action
```

Fonte: ArangoDB (2019)

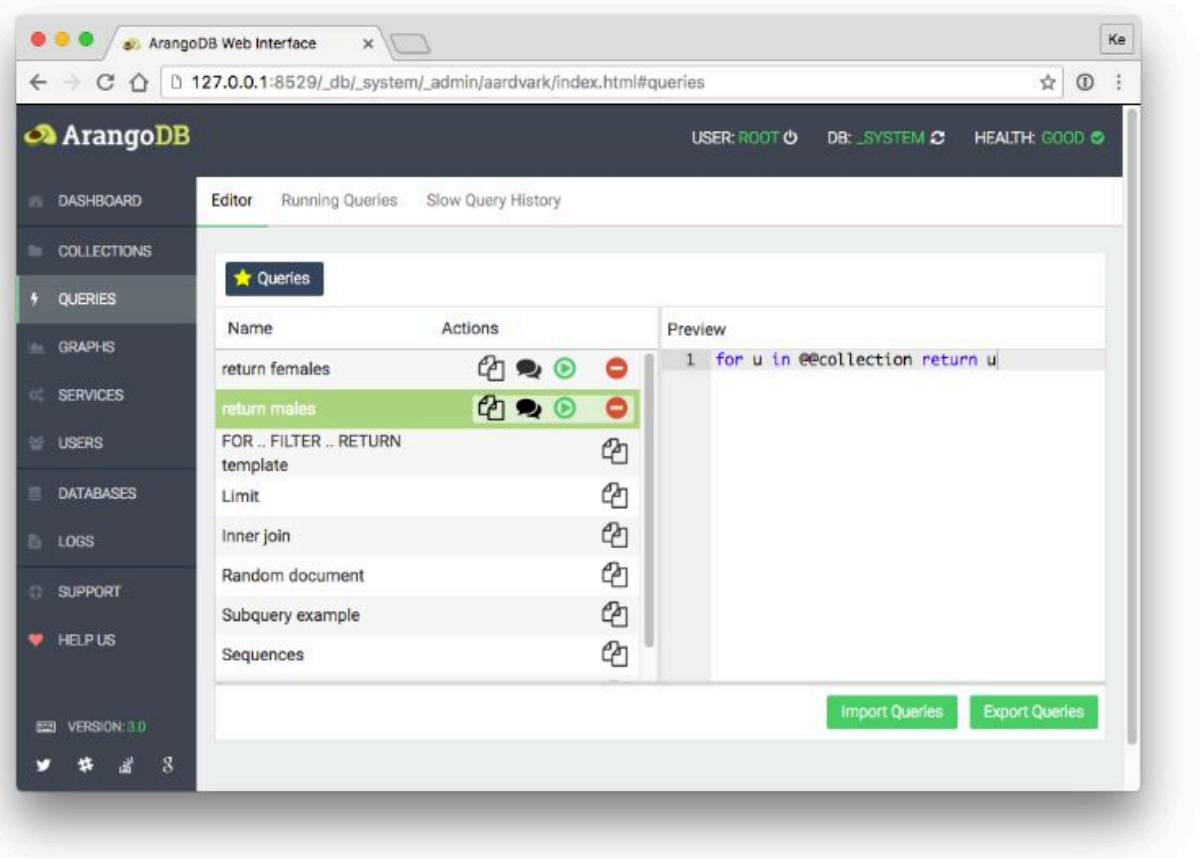
O ArangoDB possui também uma grande diversidade de índices, como os *hash index*, *skiplist index*, *fulltext index* e *geo index*. O *hash index* pode ser utilizado para localizar rapidamente documentos com valores de atributos específicos. Como ele não mantém os dados ordenados, ele é indicado para pesquisas por igualdade, mas não para pesquisas por intervalo de valores. O *skiplist index* é uma estrutura de dados ordenada, sendo indicado para pesquisas por intervalo e para conjuntos de dados que sofrem muita atualização. Ele garante boa eficiência sem a necessidade de balanceamento de estruturas de árvore, como exigido pelos índices B-tree ou AVL. O *fulltext index* pode ser utilizado para encontrar palavras ou prefixos em documentos. Já o *geo index* é utilizado para encontrar rapidamente lugares específicos na superfície da Terra. Ele armazena coordenadas bidimensionais e é indicado para operações de localização de documentos com coordenadas mais próximas de uma

determinada coordenada de comparação e documentos com coordenadas que estejam dentro de um raio “X” em torno de uma coordenada de comparação.

Por padrão, a estrutura de dados disponível no ArangoDB é o JSON, e ele possui 2 estratégias de armazenamento: MMFiles e RocksDB. O MMFiles persiste os dados em arquivos na memória e, para garantir consistência quando uma operação de alteração nos dados é invocada, primeiro ela é escrita em um log e depois aplicada aos dados, garantindo uma recuperação após alguma falha. Outra característica importante é que os arquivos são sempre incrementados, ou seja, quando um documento sofre alguma alteração, o conteúdo é copiado para um novo arquivo e inserido no fim dos arquivos de dados. O arquivo antigo é marcado como lixo e posteriormente é descartado. Quando o volume de dados é maior que a memória o método de armazenamento RocksDB é considerado uma opção melhor. Ele mantém um conjunto atualizado de dados na memória principal, mas carrega dados adicionais do disco, caso o dado não esteja no conjunto armazenado em cache. O método também grava todos os valores de índice no disco. Portanto, não há necessidade de reconstruir índices ao acessar pela primeira vez uma coleção após uma reinicialização (ArangoDB, 2019). Isso torna a inicialização do BD muito rápida. Após a inicialização, o RocksDB preenche gradualmente seus caches durante as operações, aumentando o seu desempenho com o tempo.

O servidor do ArangoDB possui uma interface web para administração, conhecida como Arangod, permitindo ao usuário gerenciar dados, coleções, usuários, bem como formular consultas em AQL. A Figura 10 apresenta a interface do Arangod na aba Queries.

Figura 10 – Exemplo da área de queries ArangoDB

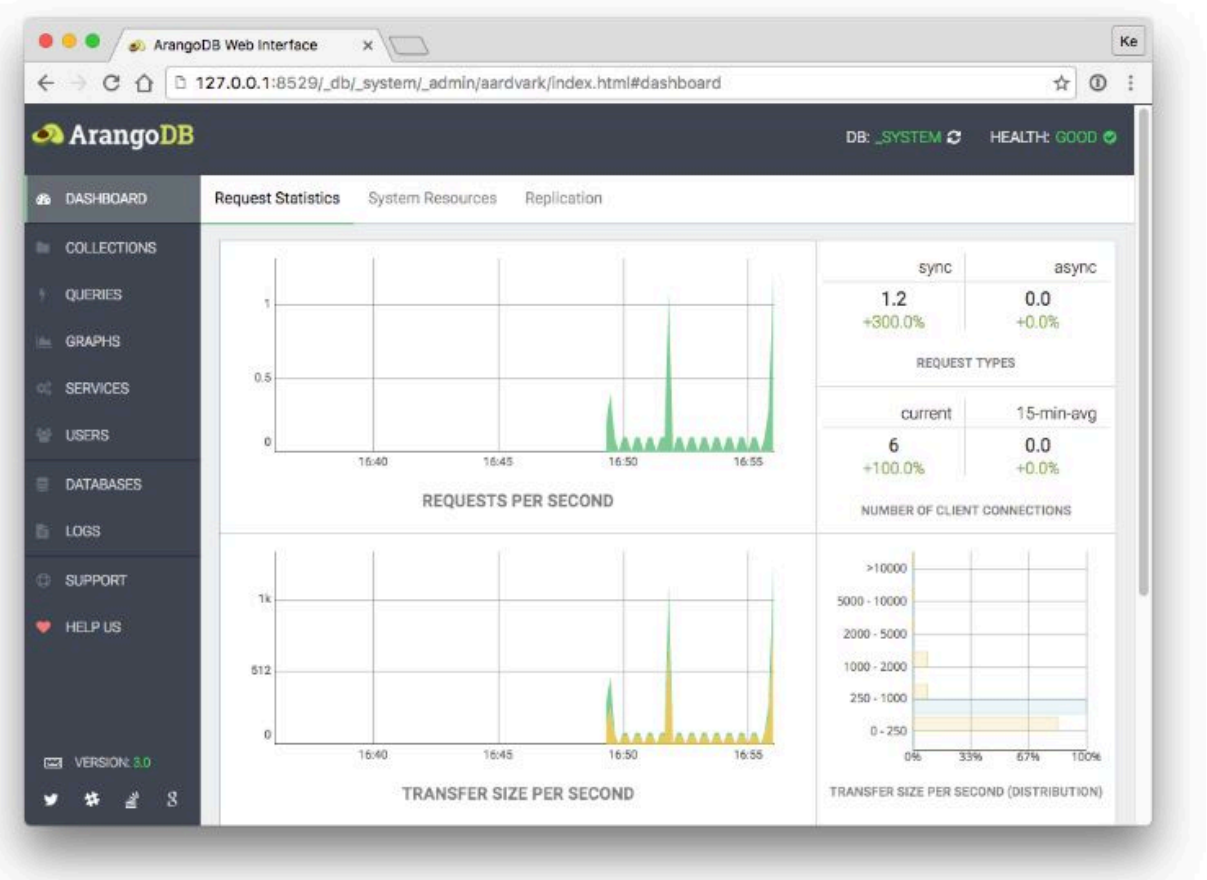


Fonte: ArangoDB (2019)

A Figura 11 apresenta a aba dashboard do Arangod no qual é possível visualizar dados analíticos e estatísticos sobre os dados. Exemplos de informações que podem ser obtidas através desta aba são:

- Solicitações feitas por segundo;
- Solicitações por segundo;
- Tipos de pedidos;
- Número de conexões do cliente;
- Tamanho de transferência;
- Tempo médio de solicitação;
- Número de processos sendo executados;
- Memória;
- Principais falhas de página;
- Tempo de CPU usado.

Figura 11 – Exemplo do dashboard ArangoDB



Fonte: ArangoDB (2019)

A interface web não é a única forma de gerenciamento dos servidores ArangoDB. Outra forma é o Arangosh, um shell V8 utilizado para interagir com qualquer servidor ArangoDB local ou remoto através de uma interface JavaScript, sendo ele muito usado para automatização de tarefas. Outra opção é utilizar uma API Restfull, uma interface HTTP que se comunica com as interfaces recém citadas.

3.2 ORIENTDB

O OrientDB é um SGBD NoSQL multimodelo de código aberto, utilizado por mais de 500 empresas, entidades governamentais e startups que desenvolvem aplicações inovadoras. Ele combina o modelo de grafo com os modelos de documento, chave-valor, orientados a objetos e geoespaciais em um único BD operacional escalável e de alto desempenho. O OrientDB possui duas edições: uma comunitária de código aberto e uma

edição paga. A edição paga, conhecida como ‘Enterprise’, foi construída a partir da edição aberta, contendo mais alguns recursos como *backup* e restauração ininterruptos, backups completos e incrementais agendados, gerenciador de consultas, configuração de clustering distribuído e gravação de métricas.

Para suportar todos os modelos de dados, o OrientDB possui algumas características próprias de como moldar esses dados. No modelo de documentos, os dados são guardados em documentos e suas relações entre documentos são feitas através de links. Esses links são utilizados para realizar junções. A Figura 12 apresenta uma consulta com múltiplos joins utilizando links. Esses links são especificados através de uma notação de ponto (“.”) para navegar nos relacionamentos.

Figura 12 – Exemplo de link

```
SELECT * FROM Employee WHERE city.country.name = 'Italy'
```

Fonte: OrientDB (2019)

A Tabela 2 apresenta uma comparação entre os conceitos utilizados no modelo de documentos e os utilizados no modelo de documentos do orientDB.

Tabela 2 – Comparação de modelos de documentos

Modelo de documentos	**Modelo de documentos do OrientDB**
Coleção	Classe
Documento	Documento
Par chave-valor	Campo do documento
-	Link

Fonte: Autor (2019).

O modelo de grafo no OrientDB apresenta uma estrutura composta por Vértices (também conhecidos como Nós) interconectados por Arestas (também conhecidos como Arcos). Os vértices são entidades que podem ser vinculados a outros vértices e possuem como propriedades obrigatórias um identificador único, um conjunto de arestas de entrada e um conjunto de arestas de saídas. As arestas são entidades que ligam dois vértices e possuem como propriedades obrigatórias um identificador único, um link para um vértice de entrada(conhecido como head), um link para um vértice de saída(conhecido com tail) e uma *label* que define o tipo de relação entre *head* e *tail*. Além das propriedade obrigatórias o vértice e aresta podem ter propriedades customizadas definidas pelo usuário o que as tornam

similares a documentos. A Tabela 3 apresenta uma comparação entre conceitos do modelo de grafo e o modelo de grafo do OrientDB.

Tabela 3 – Comparação de modelos de grafos

Modelo de grafo	**Modelo de grafo do OrientDB**
Conjunto de vértices e arestas	Classe que estende "V" (para Vértice) e "E" (para Aresta)
Vértice	Vértice
Propriedade de vértices e arestas	Propriedades de vértices e arestas
Arestas	Arestas

Fonte: Autor (2019).

O modelo chave/valor utilizado pelo OrientDB é um pouco mais robusto que o modelo chave/valor clássico. Ele suporta elementos de documentos e grafos como valores que permitem um modelo mais rico. A Tabela 4 apresenta uma comparação entre o modelo de chave/valor e o modelo chave/valor no OrientDB.

Tabela 4 – Comparação de modelos de chave/valor

Modelo de chave/valor	**Modelo de chave/valor do OrientDB**
Intervalo	Classe
Par chave/valor	Documento
-	Campo do documento ou propriedades de vértices e arestas
-	Link

Fonte: Autor (2019).

O modelo de objeto foi herdado pela programação Orientada a Objetos e suporta herança entre tipos (subtipos estendem os supertipos). Ele suporta também polimorfismo quando você se refere a uma classe. A Tabela 5 apresenta uma comparação entre alguns conceitos utilizados no modelo de objeto e no modelo de objetos do OrientDB.

Tabela 5 – Comparação de modelos de objetos

Modelo de objeto	**Modelo de objetos do OrientDB**
Classe	Classe
Objeto	Documento ou vértice
Propriedade do objeto	Campo do documento ou propriedades de vértices e arestas
Ponteiro	Link

Fonte: Autor(2019).

Uma das vantagens do OrientDB é que ele pode ser utilizado nas formas *schemaless*, *schema-full* e *schema* híbrido. O *schemaless* não obriga o usuário a definir uma estrutura pré-definida para os dados, ao contrário do *schema-full*. Já o *schema* híbrido é uma mistura dos dois anteriores, permitindo algumas partes dos dados tenham estrutura pré-definida e outras partes não.

O OrientDB disponibiliza diversas interfaces para acesso aos dados. Uma delas é a linguagem SQL com extensão para manipular árvores e grafos, API Nativa e Gremlin. Com relação a índices, o OrientDB suporta quatro tipos. Um deles é o SB-Tree, um índice padrão, que é durável, transacional e suporta consultas de intervalo. Ele é baseado no índice B-tree, porém, adaptado com otimizações relacionadas à inserção de dados. Ele fornece um mix de recursos disponíveis de outros tipos de índices para uso geral sendo eles *Unique*, *NotUnique*, *FullText* e *Dictionary*.

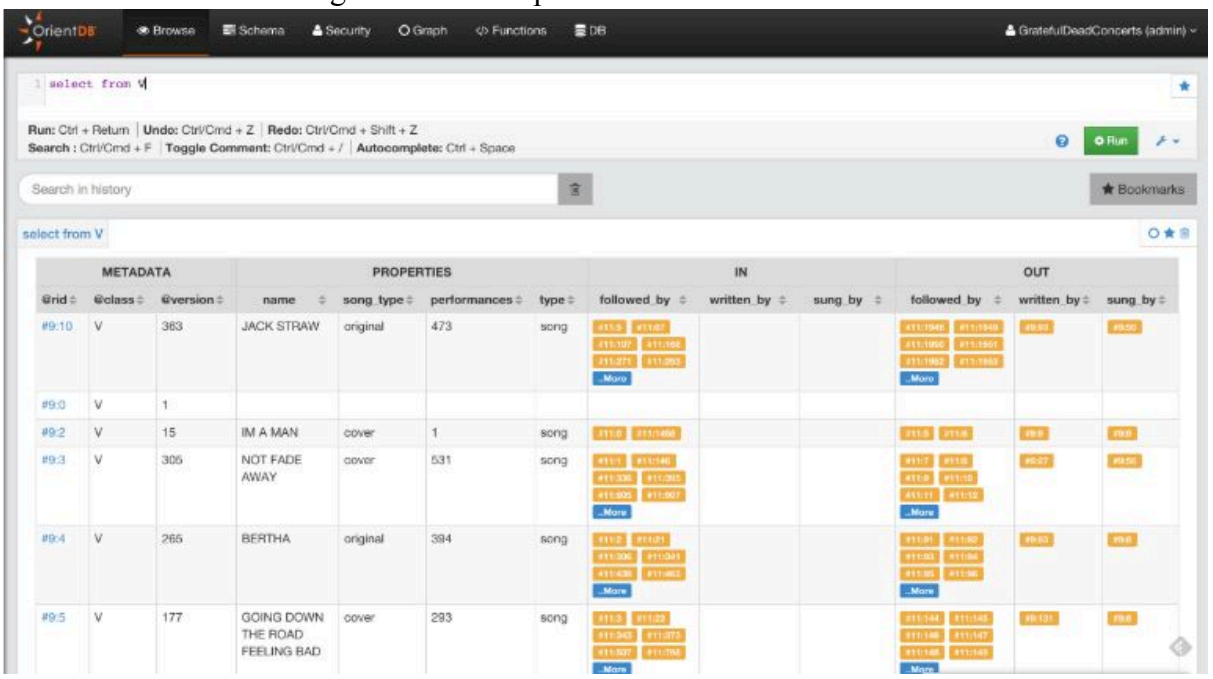
OrientDB também suporta o índice *Hash*, que fornece pesquisa rápida, durável e transacional, mas não oferece suporte a consultas de intervalo. Ele funciona como um *HashMap*, o que o torna mais rápido pesquisas pontuais e consome menos recursos do que outros tipos de índice. Outro recurso suportado pelo OrientDB é o índice *Auto Sharing*, uma implementação de uma DHT na qual as chaves são guardadas em uma partição separada. Esse índice é durável e transacional, mas também não oferece suporte a consultas de intervalo.

Outra vantagem do OrientDB é a capacidade de garantir as propriedades ACID mesmo em modo distribuído. Em relação a sua estratégia de armazenamento, o OrientDB consegue armazenar os dados em disco ou em memória. No armazenamento em memória, todos os dados ficam em memória virtual Java. Nesta opção de armazenamento, o ‘D’ do ACID não é garantido. Já o armazenamento em disco garante todas as propriedades ACID.

O controle de transações é feito pela abordagem MVCC, ou seja, Controle de Concorrência de Múltiplas Versões. Nesta estratégia, as operações geram uma nova versão dos dados que fica disponível assim que a transição terminar com sucesso, porém enquanto a transação não termina a última versão do dado é disponível somente para leitura (YU,2015).

O OrientDB possui uma interface gráfica para interação com o usuário conhecida por OrientDB Studio. A Figura 13 apresenta a tela de consulta do OrientDB *Studio*, onde é possível realizar consultas aos dados e ver os resultados.

Figura 13 – Exemplo de consulta no OrientDB



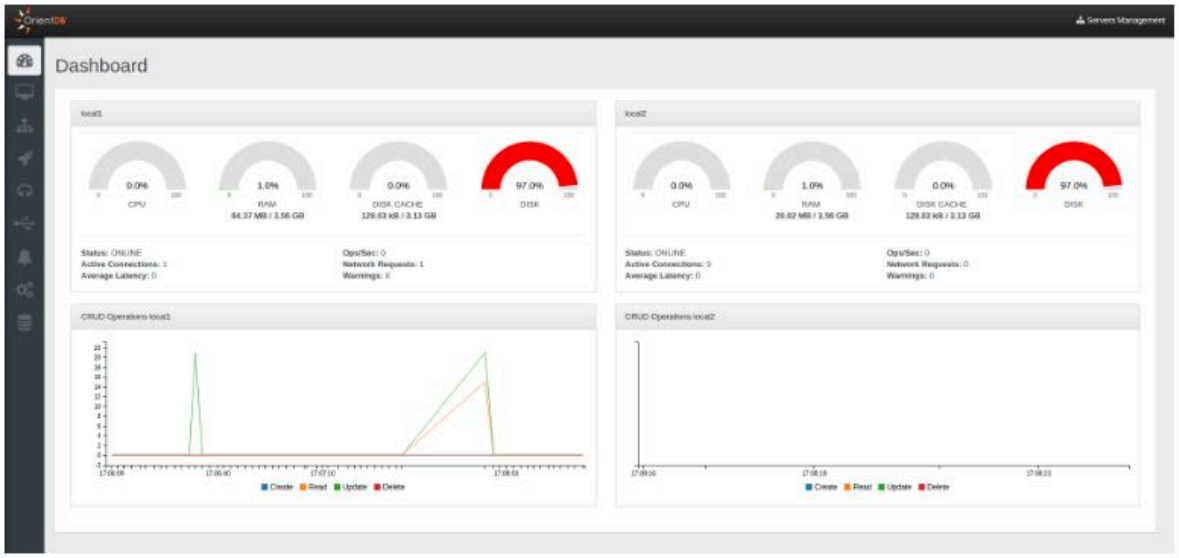
Fonte: OrientDB (2019)

O OrientDB *Studio* possui várias outras funcionalidades, como o gerenciamento de usuários e papéis, visualização de esquemas e ainda possui um painel de informações. O painel de informações apresenta o estado atual do BD, indicadores de desempenho e algumas informações como:

- CPU, RAM, DISCO CACHE e DISK usado;
- Status do nó;
- Operações por segundo;
- Conexões Ativas;
- Pedido de Rede;
- Latência Média.

Ele também apresenta um gráfico ‘ao vivo’ das operações CRUD em tempo real, como mostrado na Figura 14.

Figura 14 – Exemplo do gráfico de operações CRUD OrientDB



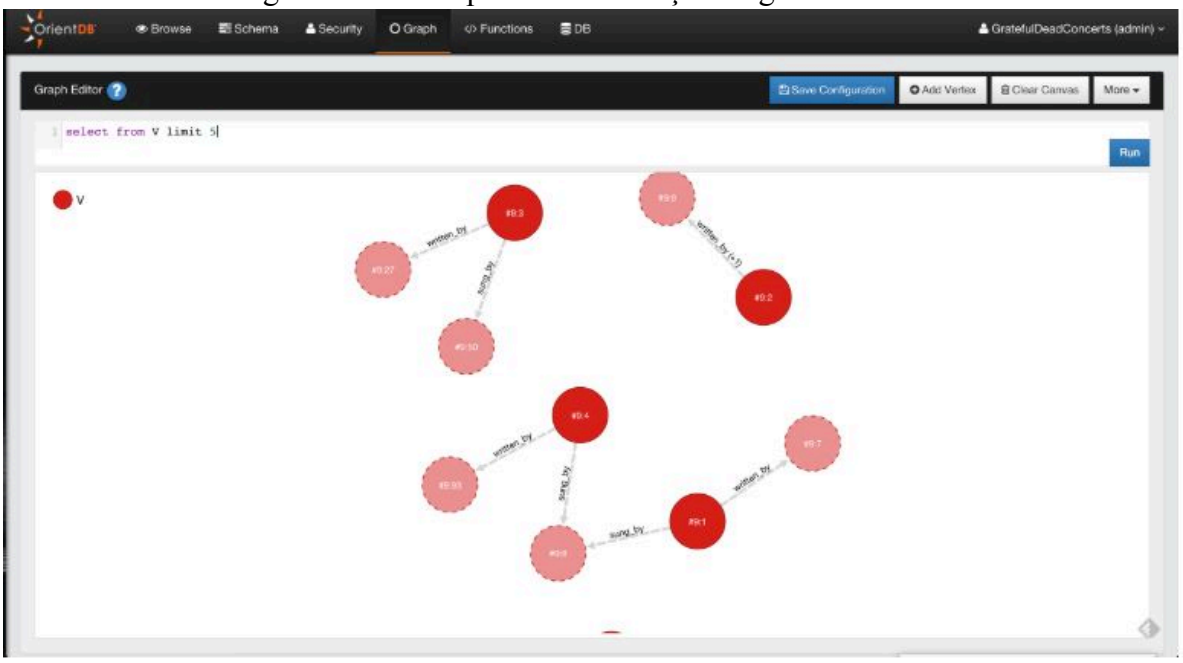
Fonte: OrientDB (2019)

Outro grande diferencial do OrientDB é um editor de grafos no qual é possível visualizar grafos de maneira gráfica e também interagir com eles. Algumas das operações que são suportadas por essa funcionalidade são:

- Adicionar Vértices;
- Salvar a configuração de renderização de gráfico;
- Limpar a tela de renderização de gráfico;
- Excluir vértices;
- Remover Vértices da Tela;
- Editar vértices;
- Inspecionar Vértices;
- Alterar a configuração de renderização de vértices;
- Navegar nos Relacionamentos;
- Criar bordas entre vértices;
- Excluir bordas entre vértices;
- Inspecionar bordas;
- Editar bordas.

A Figura 15 apresenta um exemplo desse editor gráfico.

Figura 15 – Exemplo da visualização de grafos OrientDB



Fonte: OrientDB (2019)

3.3 CRATEDB

O CrateDB é um SGBD distribuído de código aberto, escrito em Java, que integra o modelo chave-valor com o modelo de documento com foco em escalabilidade. Diferente de outros SGBDs citados neste trabalho, o CrateDB não oferece suporte à todas as propriedades ACID, suportando apenas atomicidade, isolamento e durabilidade das alterações realizadas.

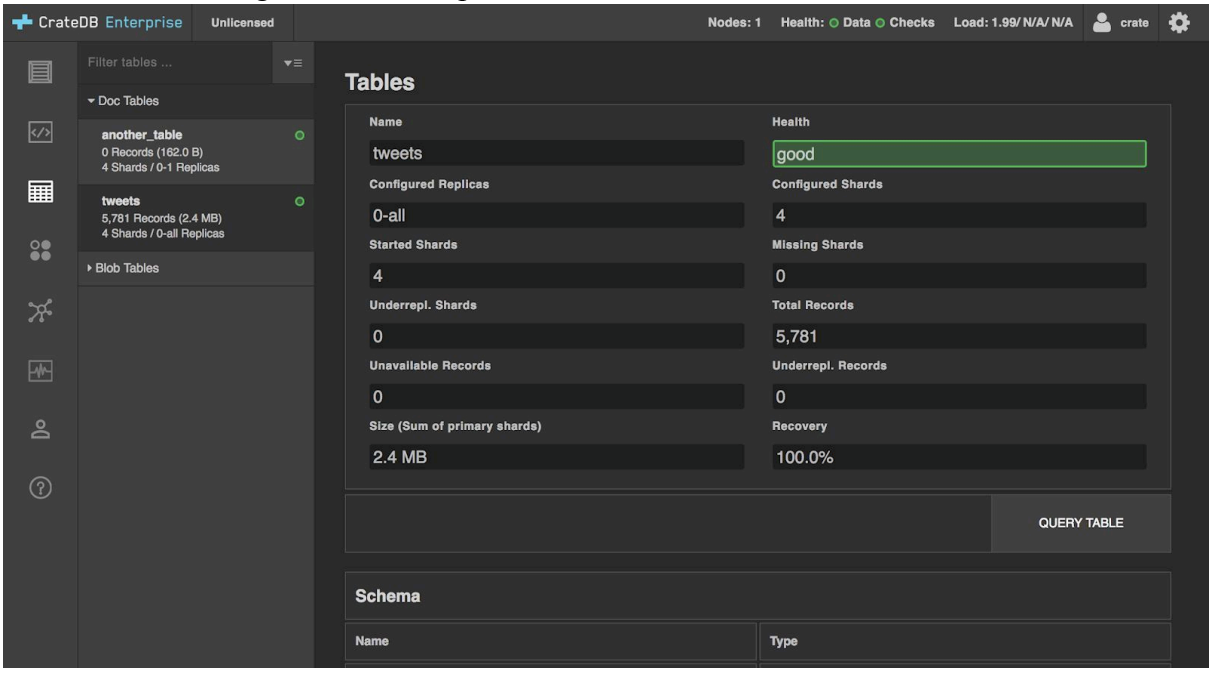
A estratégia de armazenamento do CrateDB é a utilização de documentos no formato JSON, que são guardados em sistemas de arquivos distribuídos pelos nós de um cluster. No requisito de gerenciamento de esquemas, ele não requer um esquema prévio. Entretanto, durante a criação dos dados é registrado a estrutura dos documentos em um catálogo de metadados, permitindo que se conheça a organização interna dos documentos.

O CrateDB utiliza a linguagem SQL para realizar consultas em seus dados. Ele também oferece três opções de indexação para os dados: o índice simples, índice fulltext e o índice geoespacial. O acesso aos dados pode ocorrer de diversas maneira, como ANSI SQL, via JDBC para acesso ao BD em java, ODBC, Rest e PostgresSQL. Ele também apresenta

uma interface de administração web na qual é possui executar consultas, administrar os usuários e seus privilégios, bem como schemas e tabelas.

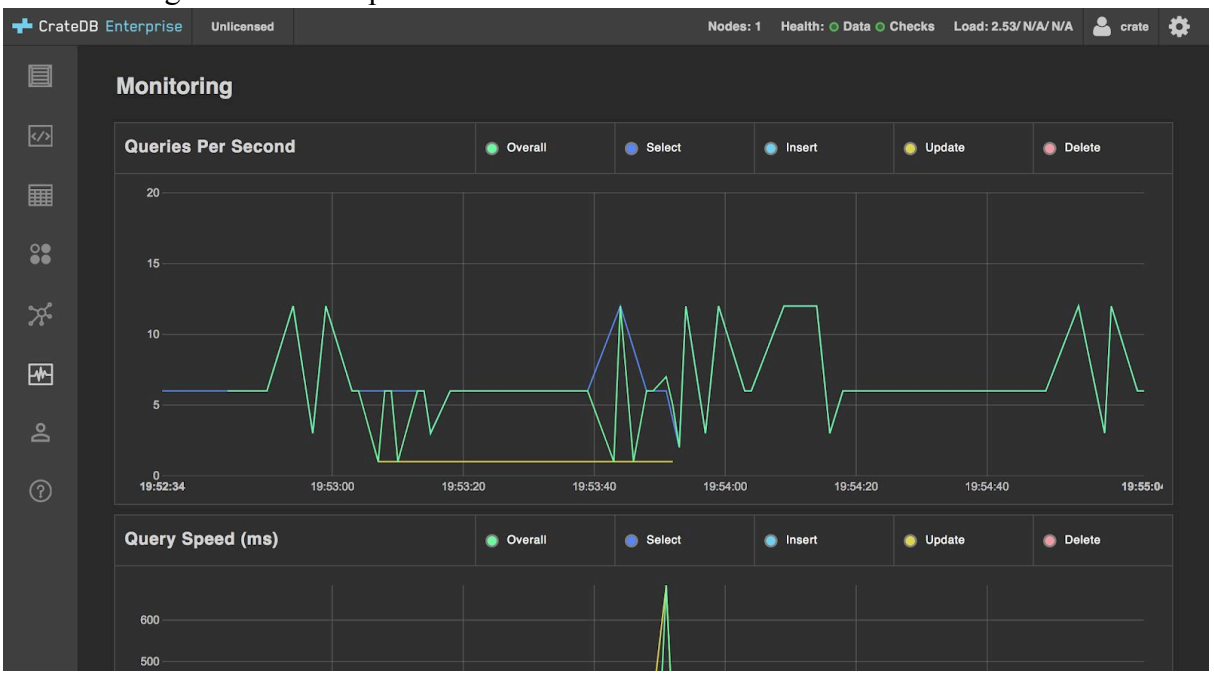
A Figura 16 apresenta um exemplo da interface do usuário visualizando as características de uma tabela exemplo. A interface também permite monitorar estatísticas do BD, como a execução de consultas por segundo, conforme mostra a Figura 17.

Figura 16 – Exemplo da interface de usuário web CrateDB



Fonte: CrateDB (2019)

Figura 17 – Exemplo da tela de monitoramento de consultas do CrateDB



Fonte: CrateDB (2019)

3.4 MARKLOGIC

O Marklogic é um SGBD que fornece armazenamento nativo para binários JSON, XML, RDF, geoespaciais e dados grandes (por exemplo, PDFs, imagens e vídeos). Sua estratégia de armazenamento gira em torno de XML e JSON, sendo dividida em dois agentes: os processadores de consulta e o de gerenciamento de dados.

Seus dados podem ser acessados de diversas maneiras, como por exemplo, através de JavaScript, XQuery, SQL ou SPARQL. A linguagem XQuery é um dos meios de acesso mais utilizados. Na Figura 18 é possível ver uma consulta buscando, no arquivo "family.xml", os elementos com idade maior de 50 anos e retornando os nomes em ordem alfabética.

Figura 18 – Exemplo de XQuery do Marklogic

```
for $x in doc("family.xml")/generation/family/family.xml
where $x/age>50
order by $x/name
return $x/name
```

Fonte: Marklogic (2019)

Uma das características mais relevantes do Marklogic é o desempenho das buscas, que está diretamente ligado às suas capacidades de indexação de dados. As principais abordagens de indexação são o índice universal, modo palavra, modo frase, modo frase com ordenação, índice de relacionamentos, faixas, geoespacial e triplas semânticas. O índice universal, em particular, indexa elementos XML e propriedades JSON dos documentos. Por default, o MarkLogic *Server* cria um conjunto de índices projetados para gerar o desempenho rápido da consulta em cenários gerais de uso (MARKLOGIC, 2018).

O MarkLogic garante as propriedades ACID ao mesmo tempo que permite escalabilidade, elasticidade, segurança certificada, alta disponibilidade e recuperação de falhas. Para o controle de transações, é adotado também a estratégia MVCC, que garante leituras livres de bloqueios, consultas temporais e rollbacks rápidos. O Marklogic pode ser classificado como um SGBD de esquema flexível, ou seja, o esquema pode mudar ou vários esquemas podem coexistir no mesmo BD. Esse recurso facilita e agiliza a integração de

dados, pois, diferentemente do relacional, não é preciso mapear todos os seus dados de volta para um determinado esquema.

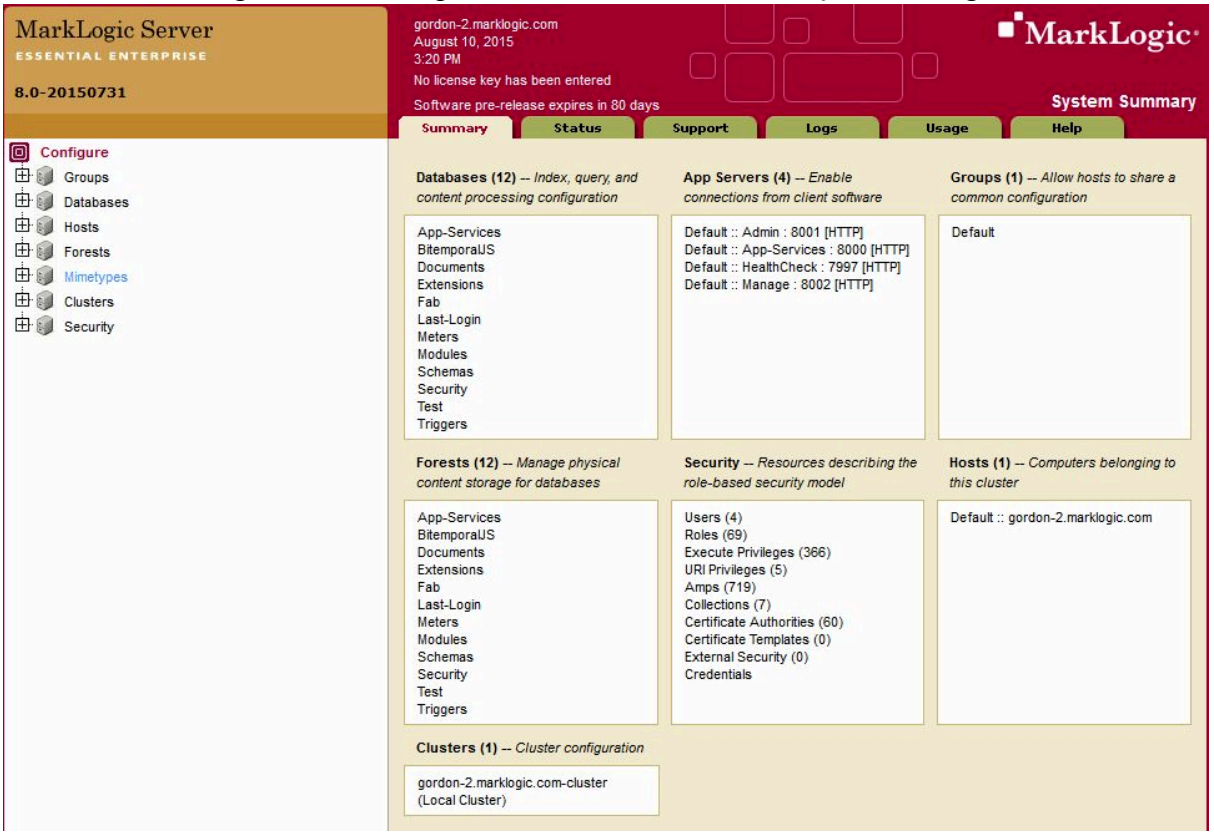
O Marklogic possui algumas ferramentas para administração e monitoramento de BDs, sendo elas uma interface de administração e APIs de administração. A interface de administração é uma interface gráfica implementada como uma aplicação web, conforme mostra a Figura 19. Suas funções são:

- Gerenciar configuração básica de software;
- Criar e configurar grupos;
- Criar e gerenciar BDs;
- Fazer backup e restaurar o conteúdo do BD;
- Criar e gerenciar novos caminhos de acesso ao servidor da Web e à linguagem Java;
- Criar e gerenciar configurações de segurança;
- Ajustar o desempenho do sistema;
- Configurar namespaces e esquemas;
- Verificar o status dos recursos em seus sistemas.

As APIs de administração fornecem um conjunto de ferramentas flexíveis para administrar e gerenciar BDs utilizando funções XQuery, funções de Javascript do lado do servidor e recursos REST. Elas são muito utilizadas para automatização do servidor, como por exemplo, para criar backup, para gerar notificações (log ou email) para eventos específicos do servidor, para fazer atualizações em massa para a configuração do servidor, além de gerenciar recursos do servidor como exemplo, a rotação de floresta somente de exclusão.

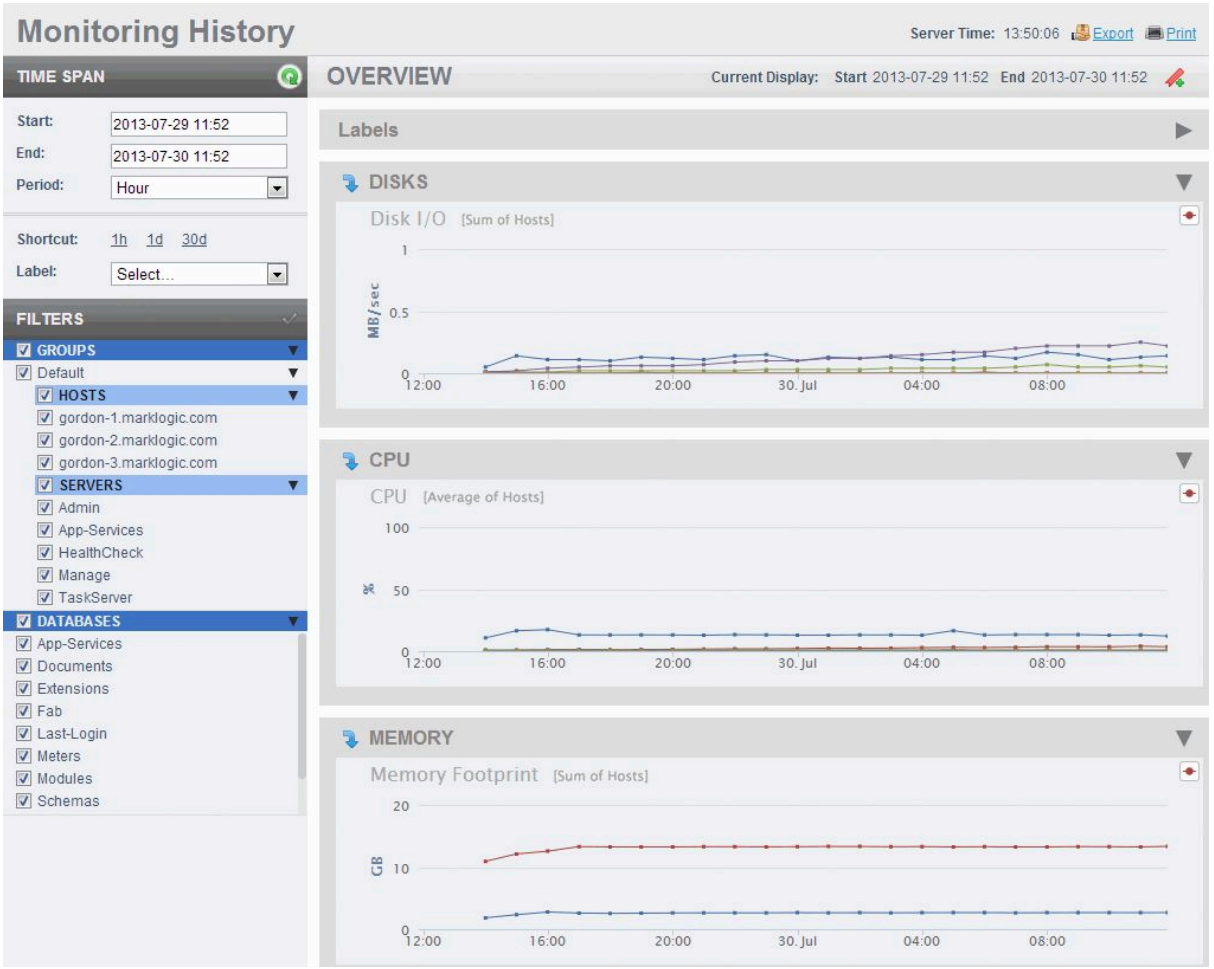
O Marklogic também possui ferramentas de monitoramento, sendo uma delas um painel de monitoramento que fornece gráficos baseados em tarefas das métricas de desempenho do servidor em tempo real. Outra ferramenta é o histórico de monitoramento, que permite capturar e visualizar dados críticos de desempenho do seu cluster, fornecendo uma visão geral das métricas de desempenho de todos os seus principais recursos. Para cada recurso é possível visualizar mais detalhes, ajustar período de tempo dos dados visualizados e aplicar filtros para exibição de dados de recursos selecionados para comparação. Um exemplo desta ferramenta de histórico de monitoramento é exposto na Figura 20, onde cada linha em um gráfico representa uma métrica para um recurso diferente.

Figura 19 – Exemplo da interface de administração Marklogic



Fonte: Marklogic(2019)

Figura 20 – Exemplo de histórico de monitoramento Marklogic



Fonte: Marklogic(2019)

4 TRABALHOS CORRELATOS

Esse capítulo apresenta alguns trabalhos correlatos à pesquisa realizada neste trabalho.

4.1 TRABALHO DE OLIVEIRA E CURA

O artigo de Fábio e Luís (OLIVEIRA, CURA, 2015) tem o propósito de avaliar dois NoSQL multimodelos SGBD em um sistema com persistência poliglota. Para os testes, eles utilizaram os SGBDs ArangoDB e OrientDB, e realizaram uma comparação desses com os SGBD NoSQL Neo4J e MongoDB.

Os testes realizados consideraram operações inserção e de consulta que buscam dados nos modelos de documento e de grafo simultaneamente nos SGBDs ArangoDB e OrientDB. Para fins de comparação, eles também realizaram os mesmos testes usando o MongoDB para a manipulação de documentos e o Neo4J para a manipulação de grafos.

A partir dos testes de carga de documentos, foi constatado que o ArangoDB possui melhor desempenho que o MongoDB, porém quando os documentos possuem muitos campos, o MongoDB supera o ArangoDB. Nos testes de carga de grafos o ArangoDB apresentou melhor desempenho que o Neo4J.

Nos testes de consulta, os SGBD multimodelo retornaram os documentos procurados, bem como documentos conectados a grafos. Já nos SGBDs monomodelo, cada consulta foi executada em duas etapas: primeiro no Neo4J para obter as chaves dos dados relacionados e, posteriormente, no MongoDB para recuperar os documentos. Nos testes foi constatado que, entre os SGBDs multimodelo, o de melhor desempenho é o OrientDB.

Após todos os testes realizados, os autores constataram que os cenários envolvendo os SGBDs multimodelo obtiveram um desempenho similar aos SGBDs monomodelo. Entretanto, o SGBDs Neo4J e MongoDB obtiveram melhores resultados quando uma aplicação precisa de consultas com maior nível de grafos e relacionamentos. Porém, se não são necessárias muitas navegações em relacionamentos na consulta, é recomendado o ArangoDB por causa de sua camada de grafos de documentos.

4.2 TRABALHO DE FERNANDOS E BERNADINHO

Os autores Diogo e Jorge (FERNANDES, BERNARDINHO, 2018) realizaram um estudo sobre as principais características, vantagens e usos de SGBDs baseados no modelo de grafos. Eles optaram pela análise dos SGBDs AllegroGraph, ArangoDB, InfiniteGraph, Neo4J e OrientDB, alguns deles sendo multimodelo. Foi considerado um conjunto de características consideradas mais relevantes e que impactam no cotidiano de um administrador de BD para a escolha de ferramentas de gerenciamento de dados, sendo elas:

- Possuir um esquema flexível;
- Possuir uma linguagem de pesquisa poderosa;
- Possuir funções para planejamento, execução e restauração de um *backup* de um banco de dados;
- Ser multimodelo;
- Possuir escalabilidade;
- Ser implementado na nuvem.

Conforme mostra a Figura 21, os autores apresentaram a comparação feita entre os SGBDs avaliando cada característica em uma escala de 0 a 4, no qual 4 significa que a característica foi bem implementada e 1 significa que a característica não foi bem implementada e deve ser melhorada. A nota 0 significa que a característica não é suportada pelo SGBD.

Figura 21 – Tabela de comparação Diego e Jorge

	AllegroGraph	ArangoDB	InfiniteGraph	Neo4J	OrientDB
Flexible Schema	1	3	3	4	3
Query Language	3	3	3	4	3
Sharding	3	3	0	0	3
Backups	3	2	3	4	3
Multimodel	4	4	2	2	4
Multi Architeture	3	4	3	4	3
Scalability	3	4	3	4	3
Cloud Ready	3	3	4	4	3
Total	23	26	21	26	25

Fonte: Diego et al..(2018)

A análise dos autores demonstrou que o SGBD AllegroGraph atendeu a todas as características, porém em relação aos outros SGBDs, algumas de suas características não são tão bem implementadas quanto poderiam ser. A análise também demonstrou que o ArangoDB atendeu a todas as características, porém, algumas funcionalidades, como o backup, poderiam melhorar. Os autores apontam o Neo4J como melhor opção de SGBD de grafo. Por fim, eles concluem que Neo4J e ArangoDB se sobressaem em termos de funcionalidades, sendo as melhores opções de SGBD de grafo analisadas.

4.3 TRABALHO DE ZHANG E LU

Os autores Chan e Jiaheng (ZHANG, LU, 2018) realizaram um *benchmark* sobre os SGBDs NoSQL multimodelos ArangoDB e OrientDB. Eles utilizaram um modelo de dados misto, proveniente de dados sintéticos, imitando os dados provenientes de uma aplicação web que combina comércio eletrônico com mídias sociais. Esses dados são de diversos formatos, incluindo JSON, XML, chave-valor e grafos.

Um dos critérios de comparação escolhidos foi o tempo de importação do conjunto de dados. Percebeu-se que tanto o ArangoDB quanto o OrientDB executaram em uma única thread, porém, o tempo de importação no ArangoDB foi em média 3,8x mais rápido que o OrientDB. A maior diferença foi na importação de dados de grafos, onde o ArangoDB obteve tempo de importação 7,5x mais rápido devido a utilização da estratégia de armazenamento de documentos com atributos “_from” e “_to”.

Outro critério escolhido foi o desempenho de consulta de dados utilizando as linguagens AQL e SQL suportado pelo OrientDB. Percebeu-se que o ArangoDB apresentou um desempenho melhor. Por último, realizaram uma avaliação da desempenho de execução de transações utilizando APIs java e node.js. Neste caso, o OrientDB utiliza comandos semelhantes aos de um SGBD relacional como begin, rollback, commit para lidar com as transações. No entanto, o ArangoDB não possui esses comandos, executando a transação via funções Javascript. Após os testes, os dois SGBDs conseguiram reverter as transações inválidas e confirmar as válidas. O ArangoDB obteve melhor desempenho nas transações de escrita pesadas, enquanto o OrientDB foi melhor em transações de leitura pesada. Os autores acreditam que isso se deve à diferença de mecanismos de armazenamento, ou seja,

armazenamento baseado em indexação por árvore LSM para o ArangoDB e baseado em indexação por árvore B para o OrientDB.

Por fim, os autores concluem que os dois SGBDs são capazes de ingerir uma variedade de dados no armazenamento sem muito esforço adicional. Eles também são capazes de suportar junções sobre vários modelos, como grapfo-JSON, JSON-relacional e grafo-relacional. No entanto, eles não possuem algoritmos específicos para otimizar seus planos de execução.

4.4 ANÁLISE DOS TRABALHOS CORRELATOS

Este capítulo apresenta a diferença deste trabalho em relação aos trabalhos similares apresentados anteriormente. O trabalho de Fábio e Luiz (OLIVEIRA, CURA, 2015) avalia o desempenho de alguns SGBDs multimodelos, realizando testes de carga no MongoDB, ArangoDB, Neo4J e OrientDB. O trabalho de Diogo e Jorge (FERNANDES, BERNARDINHO,2018) enfatiza análises comparativas em SGBDs de grafo. Já Chan e Jiaheng (ZHANG, LU,2018) apresentam um *benchmark* e testes de importação, consulta e utilização de APIs nos SGBDs multimodelo ArangoDB e OrientDB.

Este trabalho se diferencia desses trabalhos correlatos ao comparar apenas SGBD NoSQL multimodelos e não focando em um modelo de dado específico. Além disso, os quatros SGBDs multimodelo considerados neste trabalho não foram analisados e comparados entre si em nenhum dos trabalhos correlatos, bem como o conjunto de critérios de comparação.

5 ANÁLISE COMPARATIVA

Este capítulo apresenta a análise comparativa dos SGBDs NoSQL multimodelo ArangoDB, OrientDB, CrateDB e Marklogic. Ele também apresenta quadros comparativos salientando as semelhanças e diferenças dos SGBDs escolhidos. Inicialmente apresentam-se os critérios de comparação, que foram definidos com base nos trabalhos correlatos e na documentação dos SGBDs escolhidos. Os critérios definidos são os seguintes:

1. Modelo de dados;
2. Licença de uso;
3. Sistemas operacionais;
4. Gerenciamento de esquema;
5. Formas de acesso;
6. Operações;
7. Índices;
8. Recuperação de falhas;
9. Controle de concorrência;
10. Ferramentas de administração de dados.

Cabe ressaltar que este trabalho não analisa a documentação disponibilizada por cada SGBD multimodelo como um critério de análise, tendo em vista que sempre se espera uma boa documentação de um produto, pois é a partir dele que o DBA conseguirá aprender a utilizá-lo em seu benefício. Neste trabalho, opta-se por avaliar critérios que estão presentes nas documentações disponíveis, ou seja, a qualidade da documentação é implicitamente avaliada a partir da análise dos demais critérios. Ainda, o suporte às propriedades ACID pelos SGBDs também não é um critério específico de análise, pois o mesmo já está contemplado nos critérios controle de concorrência e recuperação de dados.

Outro critério que não é considerado neste trabalho é uma avaliação experimental de desempenho dos SGBDs multimodelo utilizando, por exemplo, algum *benchmark*. Este tipo de análise já foi considerado por alguns trabalhos correlatos, sendo o foco deste trabalho comparar os suportes oferecidos por cada produto com base nas suas documentações.

Os critérios de comparação definidos neste trabalho são detalhados a seguir.

5.1 MODELO DE DADOS

O primeiro critério diz respeito aos modelos de dados que os SGBDs suportam. A modelagem de dados é parte fundamental do processo do desenvolvimento de um BD. Ela consiste em abstrair os dados utilizados na aplicação e representá-los em um modelo que traga mais benefícios no seu armazenamento e na manipulação. Por isso, é de extrema importância saber quais modelos de dados o SGBD escolhido suporta e se esses modelos são adequados aos dados que se deseja armazenar. Este critério também é abordado em outros trabalhos, como o trabalho de Fábio e Luís (OLIVEIRA, CURA, 2015).

O suporte a diferentes modelos de dados é investigado primeiramente no SGBD ArangoDB, que possui uma grande documentação a respeito. O mesmo suporta três modelos de dados (chave-valor, documento e grafo) e guarda em um único C++ core. Para suportar isso, ele utiliza a flexibilidade do formato JSON. Nele é possível armazenar dados complexos e criar até mesmo estruturas similares a tabelas de um BD relacional através do agrupamento de documentos em coleções.

O suporte ao modelo chave-valor é feito da forma que todo documento possui uma chave primária "*_key*". Neste modo na ausência de índices secundários a coleção de documento se comporta como o modelo chave-valor. Sendo as únicas operações possíveis nesse contexto as pesquisas utilizando a chave única e inserções e atualizações utilizando os pares chave-valor.

Para lidar com o modelo de grafo, cada documento JSON corresponde a um vértice que possui um único atributo "*_id*". Para definir uma relação entre 2 documentos, isto é, uma aresta, define-se atributos especiais *_from* e *_to*. As arestas são armazenadas como documentos JSON em uma coleção especial de arestas. Ainda, para prover um desempenho eficiente e escalável para buscas em grafos, utiliza-se um índice *hash* nos atributos *_from* e *_to*. Vale ressaltar que, por causa dos vértices e arestas serem documentos JSON, o ArangoDB é capaz de ter grande escalabilidade horizontal devido a grande facilidade de inserção de novos dados.

O OrientDB suporta os modelos grafo, documento, chave-valor e objeto. De modo que possui um único núcleo. No modelo de documento, os dados são guardados em documentos que podem ser mantidos em coleções, utilizando os conceitos de classes e clusters.

O modelo de grafo no OrientDB é representado através do conceito de propriedades de grafo, comentado na sessão sobre o OrientDB. No caso do modelo chave-valor, assume-se que os dados no BD são alcançados a partir de uma chave no qual os valores podem ser simples ou complexos. Por fim, o modelo de objeto advém da programação orientada a objetos e suporta herança e polimorfismo quando se trata de uma classe base, tendo uma ligação direta com os objetos utilizados na linguagem de programação da aplicação.

Em relação a modelos de dados, o CrateDB suporta apenas os modelos chave-valor e documento. Sua documentação deixa a desejar na especificação de como os modelos são tratados, informando apenas que sua unidade de armazenamento é o documento. Assim sendo, provavelmente os dados do tipo chave-valor são mapeados para documentos.

O Marklogic suporta o modelo de grafo e de documento, sendo possível utilizar os formatos JSON para manter documentos que representam entidades, XML para dados em texto, bem como RDF para dados relacionados. No caso de sistemas utilizando dados destes 3 modelos, é possível utilizar simultaneamente os 3 formatos: documentos podem conter triplas, triplas podem anotar documentos e grafos de triplas podem conter documentos.

Tabela 6 – Comparação dos modelos de dados suportados

Modelo	ArangoDB	OrientDB	CrateDB	Marklogic
Colunar	Não suporta	Não suporta	Não suporta	Não suporta
Chave-valor	Suporta	Suporta	Suporta	Não suporta
Documento	Suporta	Suporta	Suporta	Suporta
Grafo	Suporta	Suporta	Não suporta	Suporta

Fonte: Autor(2019).

A Tabela 6 compara os modelos de dados adotados pelos SGBDs escolhidos. O único modelo de dados suportado por todos os SGBDs é o modelo de documento, e o modelo colunar não é suportado por nenhum. É possível perceber também que o ArangoDB e OrientDB são os produtos que consideram mais modelos de dados NoSQL, levando vantagem sobre os tipos de dados que podem ser mantidos no BD em relação ao CrateDB e ao Marklogic. Outro ponto a ser destacado é que todos os SGBDs possuem um único core, ou seja, foram construídos para ser realmente multimodelo. O que os torna mais atrativos frente a outros SGBDs que possuem cada modelo de dados em um core e uma camada de aplicação unificado-os, necessitando acessos a vários cores para buscar diferentes tipos de dados.

5.2 LICENÇA DE USO

O segundo critério escolhido é a licença de uso. Neste caso, o SGBD NoSQL multimodelo pode ter a contratação do seu software paga, ser de código aberto, ou seja, ser disponibilizado gratuitamente, ou ainda possuir uma versão paga e uma versão gratuita. Essa característica também é importante na escolha de um SGBD pois impacta no orçamento do projeto e, ainda, no caso de ser código aberto, permite extensões.

O ArangoDB possui duas edições: *community* e *enterprise*. A edição *community* é uma versão de código aberto da ferramenta ArangoDB voltada para a comunidade. A edição *enterprise* é uma versão paga voltada para projetos que requerem armazenamento altamente disponível, escalabilidade exclusiva e suporte incluído.

Similar ao ArangoDB, o OrientDB possui edições *community* e *enterprise*. A *community* é a edição de código aberto, enquanto a *enterprise* é a edição comercial para aplicações que buscam segurança, suporte 24 horas por dia, ferramenta visual studio e de console, *backup* e restauração incremental, dentre outras facilidades. A edição comercial possui um período de teste 30 dias. Após esse período é necessário pagar.

O CrateDB oferece também edições *community* e comercial. A primeira, de código aberto, é ideal para aprendizado e teste de novas ideais. Já para utilizar a edição comercial, é necessária a compra da licença que inclui alguns benefícios como acesso autenticado, monitoramento de cluster e gerenciamento de usuários. É possível solicitar a licença sem custos para a edição comercial caso o usuário for uma instituição sem fins lucrativos ou educacional.

Por fim, o Marklogic também oferece duas edições similares: a *developer* e a *enterprise*. Deste modo, é possível constatar que todos os SGBDs escolhidos possuem uma edição gratuita e uma edição paga. A versão gratuita em todos os casos possui menos benefícios, porém é ideal para que os administradores de BD e desenvolvedores consigam se familiarizar com a ferramenta e avaliem se ela se adapta à realidade da sua aplicação. Também é válido ressaltar que o OrientDB, diferente dos demais, apresenta a oportunidade de um período de teste na edição paga. O CrateDB também se destaca por disponibilizar uma versão paga para instituições sem fins lucrativos ou educacionais.

5.3 SISTEMAS OPERACIONAIS

O terceiro critério escolhido indica quais sistemas operacionais (SOs) suportam o uso dos SGBDs escolhidos. Esse critério é relevante pois quanto maior o número de SOs suportados maior a abrangência e aderência ao mercado, além de ser considerado também em trabalhos relacionados que analisam SGBDs comerciais, como o trabalho de Estefani (2017).

O ArangoDB e o CrateDB suportam os três sistemas operacionais mais comuns: Windows, Linux e MacOS. O OrientDB executa em sistemas operacionais que implementam o Java Virtual Machine(JVM), como Linux, Mac OS, Windows (após a versão 95), Solaris, HP-UX e IBM AIS. Já o Marklogic suporta os sistemas operacionais Windows (versão 7 ou posterior), Red Hat Enterprise Linux 7, CentOS 7, MacOS (versão 10.14 ou posterior) e Linux.

Em suma, todos os 4 SGBDs suportam os 3 sistemas computacionais mais comuns (Windows, Linux e MacOS), o que abrange a maior parte das necessidades do mercado atual. OrientDB é o SGBD que executa em um número maior de sistemas operacionais, o que o torna bem abrangente neste critério. Contudo, o Marklogic se diferencia pelo nível de detalhamento de sua documentação em relação aos sistemas operacionais suportados e boas práticas para a ferramenta apresentar um bom desempenho.

5.4 GERENCIAMENTO DE ESQUEMA

O gerenciamento de esquema é uma das características marcantes em SGBDs NoSQL e um critério utilizado em diversas análises de SGBDs desta família, uma vez que oferecem um controle da estrutura de um BD além da natureza default de serem schemaless. Portanto, ele foi escolhido como o quarto critério de análise para este trabalho.

O ArangoDB é classificado com schemaless, ou seja, não é necessário especificar nenhuma estrutura pré-definida. Todos os documentos podem apresentar uma estrutura integralmente diferente do outro em uma mesma coleção.

O OrientDB funciona como schemaless, porém o mesmo suporta também abordagens *schema-full* e *schema-hybrid*. O *schema-full* obrigando o usuário a especificar uma estrutura de dados pré-definida e *schema-hybrid* no qual é a mistura dos dois tipos anteriores,

permitindo algumas partes dos dados tenham estrutura pré-definida e outras partes não. Sua documentação também preconiza que as alterações nos esquemas não são transacionais, ou seja, é necessário executar os comandos de alterações fora da transação. É possível ainda visualizar o esquema criado através de instruções SQL ou Java API. O Marklogic também suporta os modos *schemaless* e *schema-full*.

Por fim, o CrateDB também funciona como *schemaless*, mas possui um esquema especial, chamado de information *schema*, no qual é possível conhecer a organização interna dos documentos sem que se obrigue os dados a serem salvos da mesma maneira.

Tabela 7 – Tipos de esquemas suportados

Esquemas	**ArangoDB**	**OrientDB**	**CrateDB**	**Marklogic**
Schemaless	Suporta	Suporta	Suporta	Suporta
Schema-full	Não suporta	Suporta	Não suporta	Suporta
Schema-hybrid	Não suporta	Suporta	Não suporta	Não suporta

Fonte: Autor(2019).

É possível perceber que uma das características marcantes no movimento NoSQL, ou seja, o fim do uso de uma estrutura pré-definida, ainda permanece nos sistemas NoSQL multimodelo. Todos os SGBDs escolhidos não impõem nenhuma estrutura a priori, conforme apresentado na Tabela 7. Neste critério, o ArangoDB acaba sendo ofuscado pelos demais SGBDs por suportar apenas o modo *schemaless*.

Mesmo assim, são muitas as possibilidades de aplicações que desejam persistir dados, e uma porcentagem delas pode achar necessário utilizar um esquema para os seus dados. Neste caso, o OrientDB se destaca entre os demais, pois ele suporta os três tipos de esquema. Isso agrega a ele um maior potencial de utilização no mercado.

5.5 FORMAS DE ACESSO

O quinto critério diz respeito às formas de acesso suportadas por cada SGBD. Uma grande diversidade de formas de acesso torna o SGBD mais atrativo ao mercado, pois pode não ser necessário realizar muitas adaptações em termos de acesso, podendo estar já disponível a melhor forma de acesso de acordo com a linguagem de programação utilizada no desenvolvimento da aplicação.

O método de acesso mais conhecido do ArangoDB é a AQL, uma linguagem de consulta especificamente desenvolvida para o mesmo. Além dela, existem outras formas de acesso como por shell e *drivers* oficiais para diversas linguagens de programação: Java, Java assíncrono, JavaScript, PHP e Go. Outra forma de acesso é a API HTTP, que possui uma vasta documentação sobre como utilizá-la e boas práticas. Esta API HTTP incluir os métodos padrões, como GET, POST, PUT e DELETE, além do método PATCH. O método PATCH solicita que um conjunto de alterações descrito na entidade solicitante seja aplicada em um determinado recurso.

O OrientDB tem como forma de acesso nativo aos dados o Java, a comunicação diretamente com o *socket* TCP/IP. Neste caso é necessário utilizar uma API socket apropriada a linguagem de programação utilizada para conversar e solicitar dados via socket.

É possível também manipular os dados através da linguagem SQL. Com o propósito de acessar os dados a partir de diversas linguagens de programação, o OrientDB disponibiliza uma extensa lista de drivers de acesso, sendo elas o Java, Node, PHP, Net, Go, Gremlin, C, JavaScript, Ruby, Scala, R, Elixir e Perl.

A documentação do CrateDB deixa a muito a desejar neste critério. Entretanto, é possível concluir que os seus métodos de acesso são via interface de linha de comando, e drivers para as linguagens Java, PHP e Phyton. Por fim, as formas de acessos do Marklogic são via XQuery, SPARQL, API REST e drivers para Java, Node e Javascript. Também permite a manipulação dos dados através da linguagem SQL. Sua documentação não possui uma seção específica de métodos de acesso, dificultando a compilação de informações a esse respeito.

Ao analisar esse critério, fica evidente os diversos métodos de acesso que o OrientDB possui, suportando mais linguagens de programação que os demais SGBDs. Todavia, o ArangoDB e Marklogic também possuem um número considerável de opções. O menos robusto neste critério é o CrateDB, o que o torna menos atrativo.

5.6 OPERAÇÕES DE ACESSO

As operações de manipulação dos dados também são uma característica importante de ser analisada, pois ela pode ser um critério de exclusão de um SGBD se ele não oferecer

determinada operação que uma aplicação deseja. Este critério é bastante utilizado em trabalhos correlatos que analisam SGBD. Neste trabalho pretende-se analisar um grande número de operações, além do básico CRUD (Criação, Leitura, Atualização e Exclusão), como agrupamento, ordenação e junção.

O ArangoDB suporta as operações básicas CRUD. Caso seja necessário alterar um documento inteiro é disponibilizada a operação *REPLACE*, que substitui um documento. Ele também oferece a operação *UPSERT*, utilizada para inserir ou atualizar condicionalmente documentos em uma coleção. Entretanto, sua linguagem AQL não possui um comando para realizar junções. Em relação a operação de agrupamento, a AQL oferece o comando *COLLECT*, que executa um agrupamento. Caso seja necessária a utilização de agregação, deve ser utilizado o comando *AGGREGATE* junto com o *COLLECT*. Por fim, a documentação disponibilizada sobre as operações suportadas é bem ampla. Outros exemplos de operações oferecidas são *FOR, RETURN, FILTER, SORT, LIMIT, LET* e *WITH*.

Por utilizar o SQL como linguagem de consulta, o OrientDB suporta as operações básicas CRUD, além da operação *MATCH* e agregação/agrupamento. A operação *MATCH* é semelhante ao *SELECT*, entretanto é utilizada para buscar dados com correspondência de padrões. O OrientDB oferece operação de junção. As relações entre os dados são chamadas de link e acabam sendo mais diretas e poderosas as junções. Os link são um relacionamento gerenciado, armazenando o RID(id) de destino no registro de origem, ou seja, armazenar um ponteiro entre 2 objetos na memória virtual. Desta forma é como se o banco de dados estivesse na memória. Portanto o criar links é possível navegar entre os objetos de maneira rápida e fácil.

No que se diz respeito a operações, o CrateDB provê as operações básicas CRUD. Além disso também uma similar da operação *UPSERT* através da cláusula *ON CONFLIT DO UPDATE SET*. As operações de junção e agregação, bem como subconsultas também são oferecidas. Diferente dos demais critérios analisados, a documentação disponibilizada pelo CrateDB sobre operações é boa, apresentando diversos exemplos e boas práticas.

Ainda, o Marklogic, como os outros SGBDs, também suporta as operações CRUD e agregação. Entretanto, não suporta operações de junções e operações de subconsultas que possuem o agrupamento junto. A Tabela 8 compara esse critério para os SGBDs escolhidos, considerando as operações mais comuns de manipulação de dados

Tabela 8 – Comparação do suporte de operações

Operação	ArangoDB	OrientDB	CrateDB	Marklogic
CRUD	Suporta	Suporta	Suporta	Suporta
Junção	Suporta	Suporta	Suporta	Não suporta
Agregação	Suporta	Suporta	Suporta	Suporta
Agrupamento	Suporta	Suporta	Suporta	Suporta
Upsert	Suporta	Não suporta	Suporta	Não suporta

Fonte: Autor(2019).

De acordo com a Tabela 8, os SGBDs ArangoDB, OrientDB e CrateDB oferecem um amplo conjunto de operações. O Marklogic fica em desvantagem em relação por ser o único a não suportar junções e *Upserts*. Entretanto, é evidente que o mercado atual valoriza e considera de grande importância os meios de manipulação dos dados e como operá-los, e isso é refletido através das operações suportadas pelos SGBDs escolhidos.

5.7 ÍNDICES

Os índices são um recurso poderoso no sentido de melhorar o desempenho de consultas de dados. Eles são estruturas de dados especiais que armazenam chaves de acesso a conjuntos, sendo percorridos de forma eficiente. Por esta razão, os tipos de índices suportados pelos SGBDs multimodelo é o sétimo critério de análise escolhido.

O ArangoDB possui uma documentação detalhada sobre seus diversos tipos de índices. O *primary index* é um índice para coleções. Cada coleção sempre possui este índice para as chaves dos seus documentos. Já o *edge index* é utilizado nas coleções de aresta, indexando os campos "*_from*" e "*_to*". O *hash index* é utilizado para buscas rápidas de documentos com base em valores de atributos específicos. O TTL index (*Time-To-Live*) é muito utilizado internamente para automaticamente expirar documentos de uma coleção. Para utilização deste índice é necessário setar uma variável "*expireAfter*" definindo um período de tempo que o documento ficará ativo ou uma data limite, após isso ele será expirado e posteriormente deletado. Além disso, o ArangoDB suporta também os índices *Skiplist index*, *Geo index* e *Fulltext index*. O *Skiplist index* possui um comportamento semelhante ao indíce B-tree, no qual uma estrutura de dados ordenada, sendo indicado para pesquisas por intervalo e para conjuntos de dados que sofrem muita atualização. Ele garante boa eficiência

sem a necessidade de balanceamento de estruturas de árvore, como exigido pelos índices B-tree ou AVL.

O OrientDB suporta os seguintes índices também oferecidos pelo ArangoDB: *Fulltext index*, *Spatial index* (semelhante a um *Geo index*) e *Hash index*. Além disso, ele oferece o *SB-Tree index*, que é baseado no tradicional índice B-Tree, com várias otimizações.

O CrateDB suporta os índices *Geo index* e *Fulltext index*. Por fim, o Marklogic oferece o *Universal index*, que é utilizado para indexar palavras, frases, relações e valores. Além desses, disponibiliza o *Range index*, *Reverse index* e *Triple index*. O *Range index*, funciona para pesquisas baseadas em desigualdade em valores numéricos, datas e outras informações digitadas. A especificação de índices de intervalo para esses elementos e/ou atributos acelera substancialmente a avaliação dessas consultas. O *Triple index* é um índice de três níveis, no qual o índice primário é um índice normal com a adição de um índice secundário para cada entrada no índice primário. O índice secundário contém palavras ou relações que seguem a palavra ou relação no correspondente índice primário, o índice terciário segue o mesmo fluxo. Por fim o *Reverse index* reúne o conjunto distinto de termos de consulta do nó folha (ou seja, os termos de consulta não compostos) em todos os documentos da consulta. Para cada nó folha, o MarkLogic mantém um conjunto de IDs de documento para nomear como uma possível correspondência inversa de consulta quando esse termo está presente em um documento, e outro conjunto de IDs para nomear quando o termo não está explicitamente presente.

Tabela 9 – Índices mais conhecidos suportados

Índices	**ArangoDB**	**OrientDB**	**CrateDB**	**Marklogic**
Edge index	Suporta	Não suporta	Não suporta	Não suporta
Hash index	Suporta	Suporta	Não suporta	Não suporta
SkipList index	Suporta	Não suporta	Não suporta	Não suporta
Fulltext index	Suporta	Suporta	Suporta	Suporta
Geo/Spatial index	Suporta	Suporta	Suporta	Não suporta
B-tree index	Não suporta	Suporta	Não suporta	Não suporta

Fonte: Autor(2019).

No que se diz respeito a suporte a índices todos os SGBDs têm uma abrangência relativamente boa e algumas peculiaridades. O ArangoDB se destaca por oferecer um maior número de tipos de índices conforme apresentado na Tabela 9. O ArangoDB e OrientDB

também se destacam por suportar índices semelhantes ao B-tree index. Em relação a documentação disponibilizada sobre os índices, todos os SGBDs se destacam ao apresentar, explicar e exemplificar o uso dos índices.

5.8 RECUPERAÇÃO DE FALHAS

Uma das funções de um administrador de banco de dados (DBA) é gerenciar cópias dos dados armazenados em um BD, também chamado de *backup*, além de ser capaz de realizar operações de recuperação total ou parcial de um BD no caso da ocorrência de uma falha. Por ser uma característica fundamental de um SGBD, os processos de recuperação de falhas foram escolhidos como oitavo critério de análise. Eles podem ter um impacto significativo em termos de desempenho considerando um SGBD que lida com múltiplos modelos de dados.

O ArangoDB suporta 3 métodos de *backups*: físico, lógico e quente. O *backup* físico somente pode ser feito quando o servidor não estiver rodando, pois ele realiza uma cópia bruta do diretório de dados (*backup* offline*)*. O backup quente, por sua vez, pode ser criado ou restaurado com o servidor rodando, através da ferramenta Arangobackup, que só é disponível na edição paga (*backup online*). O *backup* lógico é utilizado para mover ou arquivar um BD (registros e esquemas). Ele copia os dados, porém não os arquivos físicos.

O ArangoDB utiliza a ferramenta Arangostore para o restore de *backups* lógicos. Através dela é possível escolher quais coleções ou estrutura se deseja restaurar. Para os demais tipos de *backup* é utilizada uma procedure específica para restaurar os dados.

No caso do OrientDB, o *backup* executa uma cópia de todo o BD, que depois é comprimido utilizando o algoritmo ZIP. Essa rotina pode ser feita automaticamente habilitando o plugin de *backup* automático do OrientDB. Os processos de *backup* e restauração podem ser executados via console, ou seja, executando manualmente os comandos, ou através de uma API. O OrientDB suporta o *backup* e restauração incremental, ou seja, é salvo somente a diferença entre duas versões do BD. Esse método é muito utilizado quando o *backup* é feito regularmente, não sendo necessário salvar o BD por inteiro. Entretanto, este método somente está disponível na edição paga da ferramenta.

O CrateDB permite a execução de processos de *backup* e de restauração de dados através de linha de comando. Outro ponto positivo é a possibilidade de realizar *backups* incrementais. Porém, sua documentação não fornece detalhes de como esses processos funcionam.

Marklogic possui duas formas de executar *backups*: imediato ou programado. O usuário pode escolher a melhor hora para realizar o *backup* e quais são os documentos e esquemas que ele deseja realizar o *backup*. As duas formas podem ser feitas sem a necessidade do sistema estar *offline* ou interromper consultas e atualizações, o tornando-o mais atrativo. Como o OrientDB e CrateDB, o Marklogic também disponibiliza a funcionalidade de *backup* incremental. É possível realizar esses processos através do console web ou via script XQuery.

Tabela 10 – Formas de recuperação de dados suportadas

Recuperação	**ArangoDB**	**OrientDB**	**CrateDB**	**Marklogic**
Backup online	Suporta	Não suporta	Não suporta	Suporta
Backup incremental	Não suporta	Suporta	Suporta	Suporta
Backup programado	Não suporta	Não suporta	Não suporta	Suporta
Restore	Suporta	Suporta	Suporta	Suporta

Fonte: Autor(2019).

A Tabela 10 compara esse critério de análise entre os SGBDs considerados neste trabalho. Em relação à este critério de análise, todos os SGBDs apresentados possuem suporte à *backup* e restauração de dados, suprindo a maioria das necessidades de uma aplicação. Em relação a documentação sobre esse processo, o CrateDB fica em desvantagem visto que a documentação é escassa e não é muito clara comparada às demais documentações. O ArangoDB deixa a desejar na questão de ser o único dos SGBDs a não suportar o *backup* incremental. A vantagem do Marklogic para os demais é que ele suporta diversos tipos de *backup*, o que o torna bem atrativo no mercado.

5.9 CONTROLE DE CONCORRÊNCIA

O nono critério de análise escolhido foi o suporte à controle de concorrência. Da mesma forma que o critério anterior, esta também é uma característica fundamental de gerenciamento por parte de um SGBD, sendo interessante entender como o SGBD se comporta nos casos de transações que acessam concorrentemente dados mantidos em diversos modelos e se as técnicas utilizadas são eficientes para os propósitos de uso pela aplicação.

No caso do ArangoDB, este delega o controle de concorrência para 2 produtos que controlam as transações: MMFiles e o RocksDB. No primeiro caso, uma coleção é bloqueada, no início da transação, no modo solicitado (leitura ou escrita). Caso várias coleções forem manipuladas, é realizado o bloqueio em ordem alfabética. Assim, quando a transação termina, os bloqueios são liberados na ordem inversa. Durante um bloqueio de escrita, outras transações que desejam modificar as mesmas coleções são impedidas, ou seja, cada transação executa de forma isolada para fins de escrita: ela não é interrompida por outras transações ou intercalada com outras transações.

O outro mecanismo disponibilizado pelo ArangoDB é o RocksDB. Ele não realiza o bloqueio de nenhuma coleção que participe de uma transação de leitura. As operações de leitura podem ser executadas paralelamente a outras operações. Contudo, no caso de duas transações tentarem modificar o mesmo documento, as transações são abortadas. A propriedade de isolamento é disponibilizada pelo RocksDB através do snapshot isolation, ou seja, uma transação enxerga a última versão consistente do BD no momento que inicia.

O OrientDB utiliza a abordagem OCC (*Optimistic Concurrency Control*). Ela pressupõe que várias transações possam executar concorrentemente sem interferir umas nas outras. Se ocorrer algum conflito de acesso, então algumas transações são abortadas. Para operações atômicas é utilizada a abordagem MVCC (*Multi-Version Concurrency Control*), que evita o bloqueio de recursos ao servidor. Neste caso, ao executar uma transação é verificado se a versão do BD é igual à contida na transação. Caso seja, a operação é bem sucedida. Caso não seja, é gerada uma exceção pois outra transação já atualizou o mesmo registro.

O CrateDB também utiliza a abordagem OCC. Ela é controlada através da variável internas do sistema "*_version*". No qual cada novo documento tem um número de sequência

inicial de 1. Esse valor é aumentado em 1 em cada operação de inserção, exclusão ou atualização que o documento executa, cabe ao programador e administrador utilizar da variável para controle.

Por fim, o Marklogic utiliza a abordagem MVCC através do versionamento dos documentos após cada criação, alteração ou exclusão dos mesmos. Cada transação gera uma nova versão que é disponibilizada após o término da transação. Caso outra transação deseje ler o mesmo documento, uma última versão estará disponível.

Tabela 11 – Comparação de controle de concorrência

Abordagens	**ArangoDB**	**OrientDB**	**CrateDB**	**Marklogic**
Bloqueio (R/W) e aborto (W)	Utiliza	Não utiliza	Não utiliza	Não utiliza
MVCC	Não utiliza	Utiliza	Não utiliza	Utiliza
OCC	Não utiliza	Utiliza	Utiliza	Não utiliza

Fonte: Autor(2019).

Do ponto de vista de controle de concorrência, o Marklogic e OrientDB utilizam a abordagem MVCC, que é simples e garante as propriedades ACID conforme é apresentado na Tabela 11. O CrateDB possui a abordagem mais simples de controle de concorrência (OCC) que, diferente das demais, é recomendado para ambiente de baixa contenção de dados, onde conflitos são raros. O ArangoDB se destaca por apresentar 2 mecanismos de controle de concorrência: um baseado em bloqueio e outro não, deixando a cargo do usuário escolher qual mecanismo melhor se adequa à sua aplicação. Outra questão é a documentação disponibilizada pelos SGBDs em relação à controle de concorrência. Marklogic e CrateDB deixam a desejar nesse aspecto, pois não são muitos específicos.

5.10 FERRAMENTAS DE ADMINISTRAÇÃO

Por fim, o décimo critério escolhido é a ferramenta de administração de dados dos SGBDs. Aqui pretende-se investigar quais as principais funcionalidades oferecidas por elas, como por exemplo, monitoramento de desempenho, atividades, status e gerenciamento de usuários. Uma ferramenta de administração de dados é um poderoso aliado para todo DBA, podendo facilitar ou mesmo dificultar o seu trabalho. Por isso, a escolha por deste critério de análise.

Esta seção aborda as ferramentas de administração disponibilizadas por cada SGBD e suas funcionalidades. O ArangoDB possui uma ferramenta web para administração denominada Arangod. Ela oferece funcionalidades básicas e avançadas para interagir com o servidor e os dados. A Arangod possui um painel de controle dividido em 3 partes. A primeira delas apresenta graficamente estatísticas sobre as seguintes solicitações:

- Pedidos por segundo;
- Tipos de solicitação;
- Número de conexões do cliente;
- Tamanho de transferência;
- Tamanho da transferência (distribuição);
- Tempo médio de solicitação;
- Tempo médio de solicitação (distribuição).

A segunda parte apresenta os recursos do sistema, contendo o número de processos, memória, principais falhas de página e tempo de CPU utilizado. A terceira parte foca nas replicações apresentando o número de replicações, seus estados e progressos. Outro painel disponível é sobre clusters. Ele apresenta estatísticas sobre os mesmos, como servidores de BD disponíveis e ausentes, uso de memória (em porcentagem) e conexões atuais. É disponibilizado também um painel de coleções que apresenta todas as coleções do BD, sendo possível aplicar filtros e manipular (por exemplo criar e alterar) documentos. Possui também um painel editor de AQL, onde é possível escrever e executar consultas utilizando a linguagem. Outro painel apresentado é o de grafos. Ele fornece uma fácil visão dos dados de grafos, além de permitir a manipulação dos dados. Existe também o painel de usuários para controle e gestão dos mesmos, bem como um painel de log no qual apresenta os log disponíveis e podem ser filtrados pelos seus níveis de logs(info, error, warning, debug).

O SGBD OrientDB possui uma ferramenta web para administração de dados chamada Orient Studio. Ela oferece diversas opções de interação, que são chamadas sessões. O objetivo de cada sessão é a seguinte:

- Sessão "*Browser*": permite escrever e executar consultas, sendo possível também salvar consultas criadas;

- Sessão "*Schema*": permite gerenciar esquemas, incluindo uma lista de todas as classes e índices do BD atual, sendo possível criar, editar ou eliminar classes, suas propriedades e índices;
- Sessão "*Security*": permite visualizar e gerenciar todos os usuários e papéis do BD;
- Sessão "*Graph*": permite manipular os dados graficamente e visualizá-los na forma de grafos;
- Sessão "*Functions*": permite gerenciar as funções existentes no BD;
- Sessão "*Backup*": inclui um gerenciador de *backup* permitindo agendar e executá-los. É oferecida apenas na edição paga;
- Sessão "*Database Management*": contém todas as informações sobre o BD, apresentando a estrutura dos *clusters*, as configurações do BD e uma opção de exportar o BD corrente no formato JSON GZipped;
- Sessão "*Dashboard*": consiste em um painel de informações com apresentação gráfica do status atual e das tendências históricas de cada nó que integra o cluster. É oferecida apenas na edição paga e apresenta gráficos para as seguintes informações:
    - CPU, RAM, CACHE DE DISCO e DISCO usados;
    - Status do nó;
    - Operações por segundo;
    - Conexões ativas;
    - Solicitação de rede;
    - Latência média.

  Esta sessão também oferece um gráfico que apresenta a execução de operações CRUD em tempo real.

O CrateDB, como os demais SGBDs, também possui uma ferramenta web para administração de dados chamada *Crate admin UI*. A mesma é dividida em abas:

- "*Console*": permite escrever e executar consultas sobre os dados;
- "Tables": permite a visualização de documentos e suas informações, como por exemplo, os seus esquemas;
- "*Clusters*": apresenta os clusters e suas informações;

- "*Privileges*": permite a gestão dos usuários e privilégios de um BD;
- "*Monitoring*": apresenta estatísticas sobre os BDs, como consultas executadas por segundo e velocidade média de consultas.

Por fim, o Marklogic também possui uma ferramenta web para administração dos dados que disponibiliza as seguintes funcionalidades:

- Gerenciar a configuração básica do software;
- Criar e configurar grupos;
- Criar e gerenciar BDs;
- Criar e gerenciar coleções;
- Executar backups e restaurar o conteúdo de coleções;
- Criar e gerenciar novos caminhos de acesso ao servidor da Web e à linguagem Java;
- Criar e gerenciar configurações de segurança;
- Ajustar o desempenho do sistema;
- Configurar namespaces e esquemas;
- Verificar o status dos recursos em seus sistemas.

A documentação disponibilizada sobre a ferramenta é vaga e não apresenta muitos detalhes.

É possível perceber que o ArangoDB possui uma ferramenta completa, com vários dados estatísticos sobre seus BDs, o que torna atraente ao mercado para administradores de BDs. Os SGBDs OrientDB e CrateDB também apresentam diversos dados estatísticos sobre seus BDs, entretanto, em menor quantidade se comparado ao ArangoDB. O Marklogic fica em desvantagem em comparação aos demais SGBDs por não apresentar estatísticas sobre seus BDs do mesmo modo que os demais. Em relação a apresentação de dados, o ArangoDB e o OrientDB se destacam pela visualização diferenciada dos dados de grafos.

5.11 COMPARATIVO GERAL

Esta seção apresenta um comparativo geral dos 4 SGBDs analisados, levando em conta todos os critérios analisados, conforme mostra a Tabela 12.

O ArangoDB se destaca nos critérios *modelos de dados, formas de acesso, índices e ferramentas de administração*. Ele é o SGBD que suporta o maior número de modelos de dados, juntamente com o OrientDB, além de disponibilizar diferentes formas de acesso para usuários e aplicações (linguagens, drivers e APIs). Ele também oferece as principais operações de acesso e o maior número de tipos de índices. Entretanto, seus pontos fracos são o *gerenciamento de esquemas*, pois ele suporta apenas o modo *schemaless*, e *controle de concorrência*, uma vez que utiliza técnicas de bloqueio, que são as mais custosas.

O OrientDB se destaca nos critérios *modelos de dados, licenças de uso, sistemas operacionais, gerenciamento de esquemas, controle de concorrência* e *ferramenta de administração*. Além de suportar o maior número de modelos de dados, ele também oferece um período de teste para sua versão paga, executa no maior número de sistemas operacionais, gerencia o maior número de tipos de esquemas, lida com mais de uma técnica de controle de concorrência, sendo que ambas são técnicas com bom desempenho, e sua ferramenta de administração é uma das que oferece maior número de recursos.

O CrateDB se destaca no critério *ferramenta de administração*, oferecendo diversos tipos de interações e informações úteis aos usuários. Entretanto, ele apresenta diversos pontos fracos, como o fato de suportar poucos modelos de dados, não controlar esquemas para os dados e disponibilizar poucas formas de acesso.

Por fim, o Marklogic se destaca nos critérios *licenças de uso, formas de acesso* e *recuperação de dados*. No primeiro caso, ele disponibiliza versões pagas (mais completas) para instituições sem fins lucrativos. Suas formas de acesso são bastante variadas, incluindo linguagens, APIs e drivers. Além disso, ele é o SGBD que oferece o maior número de tipos de backup de dados. Porém, o Marklogic suporta poucos modelos de dados, é o mais limitado em termos de operações de acesso e índices, e também a sua ferramenta de administração apresenta recursos limitados.

Tabela 12 – Comparativo geral dos SGBDs multimodelo

Critério	ArangoDB	OrientDB	CrateDB	Marklogic
Modelos de Dados	Chave-Valor, Documento, Grafo	Chave-Valor, Documento, Grafo	Chave-Valor, Documento	Documento, Grafo
Licenças de Uso	Gratuita e Paga	Gratuita e Paga. Período de teste para a versão Paga	Gratuita e Paga	Gratuita e Paga. Versão Paga para instituições sem fins lucrativos
Sistemas Operacionais	Windows, Linux e MacOS	Windows, Linux, MacOS, Solaris, HP-UX e IBM AIS	Windows, Linux e MacOS	Windows, Linux, MacOS, Red Hat Enterprise Linux 7 e CentOS 7
Gerenciamento de Esquemas	*schemaless*	*schemaless, schema-full, schema-hybrid*	*schemaless*	*schemaless, schema-full*
Formas de Acesso	AQL; *drivers* Java, Java assíncrono, JavaScript, PHP e Go; API HTTP	SQL; acesso binário remoto; *drivers* Java, Node, PHP, Net, Go, Gremlin, C, JavaScript, Ruby, Scala, R, Elixir e Perl	*drivers* para as linguagens Java, PHP e Phyton	XQuery; SPARQL; API REST; *drivers* para Java, Node e Javascript
Operações de Acesso	CRUD, junção e agregação	CRUD, junção e agregação	CRUD, junção e agregação	CRUD e agregação
Índices	Edge, Hash, SkipList, Geo, Fulltext	Hash, Fulltext, Spatial, B-Tree	Geo, Fulltext	Fulltext
Recuperação de Dados	Backup offline e online	Backup full e incremental	Backup full e incremental	Backup offline, online, full, incremental, imediato e programado
Controle de Concorrência	Bloqueio (R/W) e aborto (W)	OCC e MVCC	OCC	MVCC
Ferramenta de Administração	estatísticas, recursos, esquemas e dados, operações, usuários	estatísticas, recursos, esquemas e dados, operações, backup e usuários	estatísticas, recursos, esquemas e dados, operações e usuários	recursos, esquemas e dados, operações e usuários

Fonte: Autor(2019).

Pode-se concluir, a partir desta análise geral, que o SGBD OrientDB é o que apresenta o maior número de pontos fortes em vários critérios importantes, como *modelos de dados* e *gerenciamento de esquemas*. Por outro lado, o SGBD que menos se destaca é o CrateDB, pois é fraco em vários critérios importantes. Isso se deve em parte a sua documentação, que é superficial no detalhamento de muitos critérios.

6 CONCLUSÃO

Este trabalho de conclusão de curso teve como objetivo principal a análise dos SGBDs NoSQL multimodelo mais conhecidos no mercado. Os seguintes produtos foram escolhidos: ArangoDB, OrientDB, CrateDB e Marklogic. Este trabalho se distingue dos trabalhos relacionados por comparar apenas SGBDs multimodelo e empregar um conjunto de critérios de comparação diferente.

A principal motivação deste trabalho foi um melhor entendimento de SGBDs multimodelo e também apresentar uma análise que facilite o trabalho de administradores de BD e profissões de Informática que desejam utilizar novas tecnologias de gerenciamento de dados em seus projetos. Para alcançar o objetivo proposto neste trabalho, foi necessário levantar o estado da arte sobre SGBDs NoSQL multimodelo, realizar um estudo dos produtos atuais do mercado, bem como outras análises deste tipo de SGBD já realizadas. Foi necessário também o estudo e levantamento de critérios de análise e, por fim, aplicar esses critérios em uma análise comparativa. Com isso, tanto o objetivo geral quanto os objetivos específicos, descritos nas seções 1.2.1 e 1.2.2, podem ser considerados atingidos.

Como conclusão deste trabalho é possível determinar que, a partir dos critérios analisados sobre os produtos, o SGBD NoSQL multimodelo que mais se destacou foi o OrientDB. Ele se sobressai em vários critérios considerados mais importantes, como modelos de dados, gerenciamento de esquemas e ferramenta de administração. Também considera-se o ArangoDB como um bom produto, com destaque para modelos de dados suportados, formas de acesso e indexação. De todos os SGBDs avaliados, o que obteve a pior avaliação é o CrateDB, o que o torna menos atraente.

Por fim, a partir da análise feita neste trabalho pode-se dizer que, em termos de ranking dos SGBDs analisados, o primeiro lugar fica com o OrientDB, em segundo lugar o ArangoDB, em terceiro lugar o Marklogic e em último lugar o CrateDB. Mesmo assim, vale lembrar que não existe uma ferramenta perfeita, e sim a que melhor se encaixa nas necessidades de cada aplicação a ser desenvolvida. Assim sendo, espera-se que este trabalho sirva como um guia para facilitar a escolha do melhor produto para uma certa demanda.

Como trabalhos futuros, sugere-se uma análise de SGBDs utilizando outros critérios, sendo um deles o de desempenho, que inclua o uso de um *benchmark* para realizar testes de

carga e de consultas. Também sugere-se a análise e comparação de SGBDs com os mesmos modelos de dados, considerando necessidades específicas de mercado. Por fim, sugere-se uma comparação entre SGBDs NoSQL multimodelo e monomodelo.

REFERÊNCIAS

ÁVILA, Michel Leite de. Um estudo sobre sistemas de gerenciamento de dados multimodelos. Dissertação - Programa de Pós Graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis, 2017.

BREWER, E. A. CAP twelve years later: How the "rules"have changed. Computer, v. 45, n. 2, p. 23–29, 2012. ISSN 0018-9162. Disponível em: <http://ieeexplore.ieee.org-/document/6133253/>. Acessado em 20 Julho 2019.

CATTELL, R. Scalable SQL and NoSQL Data Stores. 2011. Disponível em <http://www.cattell.net/datastores/Datastores.pdf> Acessado em 30 de setembro de 2019.

ESTEFANI, Roger Januário. Um comparativo de tecnologias de gerenciamento de big data nosql e newsql com foco nas ferramentas de administração de banco de dados para um dba. Dissertação - Programa de Pós Graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis, 2017.

FERNANDES, Diogo; BERNARDINHO, JORGE.Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB . 2018. Disponível em <https://www.scitepress.org/papers/2018/69102/69102.pdf> Acessado em 10 Ago 2018.

FOWLER, Martin; SADALAGE, Pramod : NoSQL Distilled. Addison-Wesley Professional, Boston, MA, Estados Unidos.

GUIDINAVA, Venkat N, RAO, DHANA, Raghavan, Vijay. Renaissance in Data Management Systems: SQL, NoSQL, and NewSQL. 2016. Disponível em: <https://pdfs.semanticscholar.org/31d4/cd04a6492186e89a4af98513784729bf9598.pdf>. Acesso em: 21 abr. 2018.

HASHEM, A. ; CHANG. V. ; ANUAR, B. ; ADEQOLE, K.; YAQOOB, I.; GANI, A.; AHMED, E.; CHROMA, H. :The role of big data in smart citiy, 2016

LIU, Zhen; LU, Jiaheng; GAWLICK, Dieter; HELSKYAHO, Heli. 2018. Multi-Model Database Management Systems - a Look Forward. Disponível em <https://www.cs.helsinki.fi/u/jilu/documents/Poly.pdf> Acessado em 22 de Setembro 2019.

LU, Jiaheng; HOLUBOVÁ, Irena. Multi-model Data Management: What's New and What's Next?. In: INTERNATIONAL CONFERENCE ON EXTENDING DATABASE TECHNOLOGY, 20., 2017, Venice. Proceedings.... Venice: Edbt, 2017. p. 602-605. Disponível em: <https://www.cs.helsinki.fi/u/jilu/paper/multi-model-data_EDBT.pdf>. Acesso em: 30 nov. 2017.

MARKLOGIC, 2018. Disponível em: <https://docs.marklogic.com/guide/concepts/indexing> Acessado em 02 Jun 2018.

MATEI, G. Column-Oriented Databases, an Alternative for Analytical Environment. Database Systems Journal, Bucharest, Romania, v. 1, n. 2, p. 3-16, Fevereiro, 2010

OLIVEIRA, Fábio Roberto; CURA, Luis Mariano del Val. Avaliação do desempenho de gerenciadores de bancos de dados multi modelo em aplicações com persistência poliglota. São Paulo: 2015

ORIENTDB, 2018. Disponível em <https://orientdb.com/multi-model-database/ > Acessado em 20 Jun 2018.

SAS, 2018. Disponível em <https://www.sas.com/pt_br/insights/big-data/what-is-big-data.html#historia> Acessado em 10 Maio 2018.

STONEBRAKER, Michel, 2012. What Does 'Big Data' Mean? Disponível em <http://cacm.acm.org/blogs/blog-cacm/155468-what-does-big-data-mean/fulltext > Acessado em 14 Maio 2018.

WEINBERGER , C 2016. Index Free Adjacency or Hybrid Indexes for Graph Databases. Disponível em: <https://www.arangodb.com/2016/04/index-freeadjacency-hybrid-indexes-graph-databases> Acessado em 03 de Julho 2018.

YU, X. 2015. An evaluation of concurrency control with one thousand cores. Disponível em <http://www.vldb.org/pvldb/vol8/p209-yu.pdf> Acessado em 30 de setembro de 2019.

ZHANG, Chao; LU, Jiaheng; XU, Pengfei; CHEN, Yuxing. 2018. UniBench: A Benchmark for Multi-Model Database Management Systems. Disponível em <https://www.cs.helsinki.fi/u/jilu/documents/UniBench.pdf> Acessado em 22 de Setembro 2019

APÊNDICE A - ARTIGO SOBRE O TRABALHO

Análise Comparativa de Sistemas de Gerenciamento de dado NoSQL multimodelos

Amanda C. V. Aquino[1]

[1]Departamento de Informática e Estatística
Universidade Federal de Santa Ctarina (UFSC) - Florianópolis , SC - Brasil

amandacveiga01@gmail.com

Abstract. *With the exponential increase in the number of web applications such as social networking and IoT applications. The data are obtained from many different sources and thus become more complex, have greater variety and have a high growth rate. In this sense, database management systems (DBMSs) called multi-model NoSQL can support the distinct characteristics of each data group and still perform well. They offer more functionality and flexibility and can support multiple data models in a single DBMS. This article studies and presents a comparative analysis of some of these most popular systems on the market.*

Resumo. *Com o aumento exponencial do número de aplicações na Web, como redes sociais e aplicações voltadas à Internet das coisas. Os dados são obtidos das mais diversas fontes e assim se tornam mais complexos, possuem maior variedade e tem uma alta taxa de crescimento. Nesse sentido, os sistemas de gerenciamento de banco de dados (SGBDs) chamados de NoSQL multimodelo conseguem suportar as distintas características de cada grupo de dados e ainda possuir um bom desempenho. Eles oferecem mais funcionalidades e flexibilidade, podendo suportar vários modelos de dados em um único SGBD. Este artigo estuda e apresenta uma análise comparativa de alguns desses sistemas mais populares no mercado.*

1. Introdução

Atualmente, uma grande quantidade de dados está sendo gerada a partir de fontes de dados diferentes como smartphones, computadores, sensores, câmeras, sistemas de posicionamento global, sites de redes sociais, transações comerciais e jogos (HASHEM et al., 2016). Com a ajuda de ferramentas de análise é possível extrair a partir desses dados inúmeras informações e aplicá-las no dia a dia. Contudo, essa não é uma tarefa fácil. Um dos desafios mais importantes para a comunidade de pesquisadores de banco de dados (BD) nos últimos anos tem sido o desenvolvimento de tecnologias para gerenciar essa grande quantidade de dados heterogêneos gerados em uma alta taxa de velocidade por aplicativos e pessoas (STONEBRAKER, 2012).

Ainda, a atual tecnologia de BD relacional não são adequados para lidar com essas grandes quantidades de dado altamente heterogêneos. Isso incentivou a comunidade de BD a buscar novas soluções de armazenamento para esses dados. Uma dessas soluções são os SGBDs NoSQL (FOWLER; SADALAGE, 2013). Os sistemas NoSQL trabalham com armazenamento distribuído, não utilizando o modelo de dados relacional e nem sempre a linguagem SQL (ESTEFANI, 2017). Eles são uma família de modelos de dados, sendo os quatro principais o chave-valor, colunar, documento e grafo.

Como uma resposta direta a necessidade, sistemas de gerência de banco de dados (SGBDs) que suportam vários modelos surgiram para atender a necessidade de múltiplos formatos de dados, reduzindo a complexidade operacional e mantendo a consistência global dos dados (LU; HOLUBOVÁ, 2017). O surgimento desses SGBDs, também conhecidos como SGBDs NoSQL multimodelo é uma inovação no mercado, por isso o foco deste trabalho é definir quatro ferramentas mais conhecidas e realizar uma análise comparativa sobre eles. O resto desse trabalho está dividido em três partes, sendo elas apresentação dos SGBDs escolhidos, análise comparativa dos SGBDs escolhidos e conclusão.

2. SGBDs escolhidos

À medida que um sistema vai se tornando mais complexo, agregando novos dados com formatos distintos, volumes elevados e necessidades especiais de consulta ou transformação, surge naturalmente a (re)discussão sobre a camada de armazenamento da solução (LU;HOLUBOVÁ, 2017), surgindo à necessidade de um SGBD que suporte vários modelos ao mesmo tempo.

Diferente dos tradicionais SGBDs, que são estruturados sobre um único modelo de dados, os NoSQL multimodelos são concebidos com o intuito de integrar e suportar diferentes modelos de dados. Para a realização deste trabalho foram considerados os 4 SGBDs NoSQL multimodelo mais conhecidos e relevantes no mercado, conforme mostra a Tabela 1. Os quatros SGBds selecionados são citados na DB Ranking, que é uma lista no qual classifica os SGBDs de acordo com sua popularidade. É importante ressaltar que por serem tecnologias novas no mercado é natural que não estejam em posições altas. Outro motivo da escolha dos SGBDs foi as documentações disponibilizadas, que são mais abrangentes e detalhadas permitindo uma melhor análise para o trabalho atual.

Tabela 1 – SGBDs NoSQL multimodelo escolhidos

SGBD	**Link**
ArangoDB	https://www.arangodb.com/
OrientDB	https://orientdb.com/
CrateDB	https://crate.io/
Marklogic	https://www.marklogic.com/

Fonte: Autor (2019).

O ArangoDB é um SGBD NoSQL de código aberto, sendo concebido para armazenar seus dados como pares chave/valor, grafos ou documentos, e acessar qualquer representação de dados usando uma única linguagem de consulta declarativa. O segundo SGBD escolhido é o OrientDB, um SGBD NoSQL multimodelo de código aberto, utilizado por mais de 500 empresas, entidades governamentais e startups que desenvolvem aplicações inovadoras. Ele combina o modelo de grafo com os modelos de documento, chave-valor, orientados a objetos e geoespaciais em um único BD operacional escalável e de alto desempenho. O CrateDB é um SGBD distribuído de código aberto, escrito em Java, que integra o modelo chave-valor com o modelo de documento com foco em escalabilidade. O Marklogic é um SGBD que fornece armazenamento nativo para binários JSON, XML, RDF, geoespaciais e dados grandes (por exemplo, PDFs, imagens e vídeos). Sua estratégia de

armazenamento gira em torno de XML e JSON, sendo dividida em dois agentes: os processadores de consulta e o de gerenciamento de dados.

3. Análise Comparativa

Após realizada o estudo e análise de cada SGBD escolhido, uma análise comparativa foi feita entre eles levando em consideração alguns critérios escolhidos. Os critérios escolhidos foram:

1. Modelo de dados;
2. Licença de uso;
3. Sistemas operacionais;
4. Gerenciamento de esquema;
5. Formas de acesso;
6. Operações;
7. Índices;
8. Recuperação de falhas;
9. Controle de concorrência;

Com a análise comparativa feita formou-se a Tabela 2, que apresenta os resultados da análise.

Tabela 12 – Comparativo geral dos SGBDs multimodelo

Critério	ArangoDB	OrientDB	CrateDB	Marklogic
Modelos de Dados	Chave-Valor, Documento, Grafo	Chave-Valor, Documento, Grafo	Chave-Valor, Documento	Documento, Grafo
Licenças de Uso	Gratuita e Paga	Gratuita e Paga. Período de teste para a versão Paga	Gratuita e Paga	Gratuita e Paga. Versão Paga para instituições sem fins lucrativos
Sistemas Operacionais	Windows, Linux e MacOS	Windows, Linux, MacOS, Solaris, HP-UX e IBM AIS	Windows, Linux e MacOS	Windows, Linux, MacOS, Red Hat Enterprise Linux 7 e CentOS 7
Gerenciamento de Esquemas	*schemaless*	*schemaless, schema-full, schema-hybrid*	*schemaless*	*schemaless, schema-full*
Formas de Acesso	AQL; *drivers* Java, Java assíncrono, JavaScript, PHP e Go; API HTTP	SQL; acesso binário remoto; *drivers* Java, Node, PHP, Net, Go, Gremlin, C, JavaScript, Ruby, Scala, R, Elixir e Perl	*drivers* para as linguagens Java, PHP e Phyton	XQuery; SPARQL; API REST; *drivers* para Java, Node e Javascript

Operações de Acesso	CRUD, junção e agregação	CRUD, junção e agregação	CRUD, junção e agregação	CRUD e agregação
Índices	Edge, Hash, SkipList, Geo, Fulltext	Hash, Fulltext, Spatial, B-Tree	Geo, Fulltext	Fulltext
Recuperação de Dados	Backup offline e online	Backup full e incremental	Backup full e incremental	Backup offline, online, full, incremental, imediato e programado
Controle de Concorrência	Bloqueio (R/W) e aborto (W)	OCC e MVCC	OCC	MVCC
Ferramenta de Administração	estatísticas, recursos, esquemas e dados, operações, usuários	estatísticas, recursos, esquemas e dados, operações, backup e usuários	estatísticas, recursos, esquemas e dados, operações e usuários	recursos, esquemas e dados, operações e usuários

O ArangoDB se destaca nos critérios modelos de dados, formas de acesso, índices e ferramentas de administração. Ele é o SGBD que suporta o maior número de modelos de dados, juntamente com o OrientDB, além de disponibilizar diferentes formas de acesso para usuários e aplicações (linguagens, drivers e APIs). Ele também oferece as principais operações de acesso e o maior número de tipos de índices. Entretanto, seus pontos fracos são o gerenciamento de esquemas, pois ele suporta apenas o modo schemaless, e controle de concorrência, uma vez que utiliza técnicas de bloqueio, que são as mais custosas.

O OrientDB se destaca nos critérios modelos de dados, licenças de uso, sistemas operacionais, gerenciamento de esquemas, controle de concorrência e ferramenta de administração. Além de suportar o maior número de modelos de dados, ele também oferece um período de teste para sua versão paga, executa no maior número de sistemas operacionais, gerencia o maior número de tipos de esquemas, lida com mais de uma técnica de controle de concorrência, sendo que ambas são técnicas com bom desempenho, e sua ferramenta de administração é uma das que oferece maior número de recursos.

O CrateDB se destaca no critério ferramenta de administração, oferecendo diversos tipos de interações e informações úteis aos usuários. Entretanto, ele apresenta diversos pontos fracos, como o fato de suportar poucos modelos de dados, não controlar esquemas para os dados e disponibilizar poucas formas de acesso.

Por fim, o Marklogic se destaca nos critérios licenças de uso, formas de acesso e recuperação de dados. No primeiro caso, ele disponibiliza versões pagas (mais completas) para instituições sem fins lucrativos. Suas formas de acesso são bastante variadas, incluindo linguagens, APIs e drivers. Além disso, ele é o SGBD que oferece o maior número de tipos de backup de dados. Porém, o Marklogic suporta poucos modelos de dados, é o mais limitado em termos de operações de acesso e índices, e também a sua ferramenta de administração apresenta recursos limitados.

4. Conclusão

Este trabalho de conclusão de curso teve como objetivo principal a análise dos SGBDs NoSQL multimodelo mais conhecidos no mercado. Os seguintes produtos foram escolhidos: ArangoDB, OrientDB, CrateDB e Marklogic. Este trabalho se distingue dos trabalhos relacionados por comparar apenas SGBDs multimodelo e empregar um conjunto de critérios de comparação diferente.

A principal motivação deste trabalho foi um melhor entendimento de SGBDs multimodelo e também apresentar uma análise que facilite o trabalho de administradores de BD e profissões de Informática que desejam utilizar novas tecnologias de gerenciamento de dados em seus projetos. Para alcançar o objetivo proposto neste trabalho, foi necessário levantar o estado da arte sobre SGBDs NoSQL multimodelo, realizar um estudo dos produtos atuais do mercado, bem como outras análises deste tipo de SGBD já realizadas. Foi necessário também o estudo e levantamento de critérios de análise e, por fim, aplicar esses critérios em uma análise comparativa. Com isso, tanto o objetivo geral quanto os objetivos específicos, descritos nas seções 1.2.1 e 1.2.2, podem ser considerados atingidos.

Como conclusão deste trabalho é possível determinar que, a partir dos critérios analisados sobre os produtos, o SGBD NoSQL multimodelo que mais se destacou foi o OrientDB. Ele se sobressai em vários critérios considerados mais importantes, como modelos de dados, gerenciamento de esquemas e ferramenta de administração. Também considera-se o ArangoDB como um bom produto, com destaque para modelos de dados suportados, formas de acesso e indexação. De todos os SGBDs avaliados, o que obteve a pior avaliação é o CrateDB, o que o torna menos atraente. Por fim a partir da análise feita neste trabalho pode-se dizer que o ranking dos SGBDs escolhidos ficaram em primeiro lugar o OrientDB, o segundo lugar o ArangoDB, em terceiro lugar o Marklogic e em último lugar o CrateDB. Mesmo assim, vale lembrar que não existe uma ferramenta perfeita, e sim a que melhor se encaixa nas necessidades de cada aplicação a ser desenvolvida. Assim sendo, espera-se que este trabalho sirva como um guia para facilitar a escolha do melhor produto para uma certa demanda.

Como trabalhos futuros, sugere-se uma análise de SGBDs utilizando outros critérios, sendo um deles o de desempenho, que inclua o uso de um benchmark para realizar testes de carga e de consultas. Também sugere-se a análise e comparação de SGBDs com os mesmos modelos de dados, considerando necessidades específicas de mercado. Por fim, sugere-se uma comparação entre SGBDs NoSQL multimodelo e monomodelo.

Referências

ESTEFANI, Roger Januário. Um comparativo de tecnologias de gerenciamento de big data nosql e newsql com foco nas ferramentas de administração de banco de dados para um dba. Dissertação - Programa de Pós Graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis, 2017.

FOWLER, Martin; SADALAGE, Pramod : NoSQL Distilled. Addison-Wesley Professional, Boston, MA, Estados Unidos.

LU, Jiaheng; HOLUBOVÁ, Irena. Multi-model Data Management: What's New and What's Next?. In: INTERNATIONAL CONFERENCE ON EXTENDING DATABASE

TECHNOLOGY, 20., 2017, Venice. Proceedings.... Venice: Edbt, 2017. p. 602-605. Disponível em: <https://www.cs.helsinki.fi/u/jilu/paper/multi-model-data_EDBT.pdf>. Acesso em: 30 nov. 2017.

HASHEM, A. ; CHANG. V. ; ANUAR, B. ; ADEQOLE, K.; YAQOOB, I.; GANI, A.; AHMED, E.; CHROMA, H. :The role of big data in smart citiy, 2016

STONEBRAKER, Michel, 2012. What Does 'Big Data' Mean? Disponível em <http://cacm.acm.org/blogs/blog-cacm/155468-what-does-big-data-mean/fulltext > Acessado em 14 Maio 2018.