Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

NoSQL^2: Administrando Banco de Dados NoSQL com a Linguagem SQL

Jane Adriana Souza

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

Orientadora
Prof.ª Dr.ª Maristela Terto de Holanda

Brasília
2016

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Pós-graduação em Informática

Coordenadora: Prof.ª Dr.ª Celia Ghedini Ralha

Banca examinadora composta por:

Prof.ª Dr.ª Maristela Terto de Holanda (Orientadora) — CIC/UnB
Prof. Dr.ª Celia Ghedini Ralha — CIC/UnB
Prof. Dr.ª Edna Dias Canedo — FGA/UnB

CIP — Catalogação Internacional na Publicação

Souza, Jane Adriana.

NoSQL²: Administrando Banco de Dados NoSQL com a Linguagem SQL / Jane Adriana Souza. Brasília : UnB, 2016.
106 p. : il. ; 29,5 cm.

Dissertação (Mestrado) — Universidade de Brasília, Brasília, 2016.

1. SGBD, 2. Big Data, 3. DBA, 4. query

CDU 004

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

NoSQL^2: Administrando Banco de Dados NoSQL com a Linguagem SQL

Jane Adriana Souza

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

Prof.ª Dr.ª Maristela Terto de Holanda (Orientadora)
CIC/UnB

Prof. Dr.ª Celia Ghedini Ralha
CIC/UnB

Prof. Dr.ª Edna Dias Canedo
FGA/UnB

Prof.ª Dr.ª Celia Ghedini Ralha
Coordenadora do Programa de Pós-graduação em Informática

Brasília, 24 de novembro de 2016

Dedicatória

Eu dedico essa Dissertação de Mestrado aos meus pais, Noé e Efigênia, que sempre ao meu lado, me ajudaram a superar os obstáculos com força, fé e coragem, e me proporcionaram os conhecimentos da integridade, da perseverança e de procurar sempre em Deus à força maior para o meu desenvolvimento como ser humano.

Agradecimentos

Tenho tanto a agradecer! Existiram percalços no caminho, mas Deus me deu forças e sabedoria para superá-los com a cabeça erguida e com muita fé. Agradeço primeiramente a *Deus* pois sem ELE nada é possível! Agradeço especialmente ao meu pai e minha mãe, que sempre ao meu lado, me incentivaram a seguir em frente. Agradeço carinhosamente a minha irmã e meu irmão que me acompanharam nessa caminhada. Agradeço aos meus amigos da universidade, do IPHAN e do MPOG, aos amigos de coração, e a todos os amigos que sempre me ajudaram com preciosas dicas e sugestões para o engradecimento desse projeto. Agradeço à Universidade de Brasília e aos professores que contribuíram para a minha formação. E agradeço imensamente a minha orientadora Prof. Maristela, pela ajuda inestimável, pelos esclarecimentos e pelas valiosas sugestões. Obrigada!

Resumo

Nos últimos anos, novos modelos de banco de dados, chamados NoSQL (*Not Only SQL*) estão sendo considerados alternativas para a gestão de grandes volumes de dados - *Big Data*, pois gerenciam e armazenam os dados de forma eficiente, possuem alta escalabilidade, disponibilidade e desempenho satisfatório. A administração de bancos de dados implica na execução de tarefas, tais como criação de bases e objetos, atribuição de privilégios, realização de *backups*, dentre outras atividades. A execução dessas tarefas de administração em bancos de dados NoSQL exige um maior nível de conhecimento por parte dos administradores de bancos de dados (DBA), e expõe questões relacionadas à falta de familiaridade desses profissionais nos ambientes NoSQL. De forma a contribuir nesse campo de estudo, a presente dissertação apresenta a solução NoSQL^2 para execução de tarefas de administração, usando a linguagem SQL (*Structured Query Language*), que funciona em diferentes Sistemas Gerenciadores de Bancos de Dados (SGBD) NoSQL. O NoSQL^2 permite aos DBAs se desvincularem das particularidades de formas de acesso de cada NoSQL, pois disponibiliza recursos para conversão de comandos da sintaxe SQL para a sintaxe proprietária do banco de dados NoSQL.

Palavras-chave: SGBD, Big Data, DBA, query

Abstract

New database models, called NoSQL (Not Only SQL) are considered appropriate alternatives for managing and storing Big Data due to their efficiency, high scalability, availability and performance. Database administration effects tasks such as creating databases and objects, attributing priorities and performing backups. The execution of these tasks, in NoSQL databases, require that DBA (database administrators) have a high level of knowledge, and often exposes problems with the DBA unfamiliarity of the NoSQL environments. In order to contribute to the scholarship in this field, this paper presents the middleware NoSQL^2 to perform management tasks using the SQL language, which runs on different NoSQL databases. The NoSQL^2 allows DBA to disassociate themselves from the particularities of access forms of each NoSQL, since provides resources for converting SQL commands to the proprietary NoSQL database syntax.

Keywords: SGBD, Big Data, DBA, query

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Edgar Codd introduziu o modelo relacional na década de 70 com o objetivo principal de superar questões relativas ao suporte, à independência e integridade dos dados nos Sistemas Gerenciadores de Banco de Dados (SGBD) (Elmasri e Navathe, 2011). Naquela época, devido aos problemas de falta de integridade dos dados era necessário desenvolver programas complexos para responder a uma simples consulta. Assim, houve uma boa aceitação dos usuários e da comunidade acadêmica, devido a simplicidade e praticidade do modelo relacional e o seu embasamento teórico, que engloba conceitos como as propriedades de Atomicidade, Consistência, Isolamento e Durabilidade (ACID) das transações, as formas normais, e a linguagem de consulta padronizada SQL (*Structured Query Language*). Os SGBD relacionais disputam espaço na indústria de banco de dados com outros sistemas gerenciadores de banco de dados não-relacionais, tais como o SGBD Orientado a Objeto, baseado em documento XML e os NoSQL (*Not Only SQL*) (Silberschatz et al., 2010, Padhy et al., 2011; Corbellini, 2016).

Em (Abadi et al., 2016), um conjunto de renomados pesquisadores apresenta apontamentos, direções e as principais tendências de pesquisa na área de banco de dados: custo baixo de infraestrutura para gerar uma variedade de dados; custo baixo para processar grandes quantidades de dados; e o fato do gerenciamento de dados ter se tornado democratizado, não sendo necessário que um usuário se torne um profissional na área de banco de dados para utilizar as ferramentas associadas. Entre os apontamentos, destaca-se o fato da geração *Big Data* está atenta ao valor dos princípios aplicados usualmente nos SGBD relacionais, tais como consistência transacional, conceitos e métodos adotados comumente pelos especialistas da área, e formas de acesso e linguagens de consulta para bancos de dados.

Com o surgimento da Web 2.0 e o crescimento do volume de dados, o modelo de banco de dados, conhecido como NoSQL, tem se destacado no mercado, propondo alto desempenho, e processamento de grandes volumes de dados não-estruturados. O interesse

das organizações em adotar os bancos de dados NoSQL está movimentando as áreas de pesquisas sobre o assunto, pois os princípios fundamentais do modelo relacional, tais como definição de esquema, relacionamentos, formas normais, propriedades ACID, linguagens de consulta, na maior parte das vezes, não se mantém nos modelos enquadrados como NoSQL.

Os bancos de dados NoSQL utilizam formas diversificadas de acesso aos dados, tais como API *(Application Programming Interface)*, *drivers* JDBC (*Java Database Connectivity*)/ODBC (*Open Database Connectivity*), e linguagens de consulta, sendo que estas, quando existem, possuem uma sintaxe de comandos própria e não padronizada, exigindo que o usuário se especialize na linguagem em questão. A possibilidade de fornecimento de linguagens de consulta padronizada e de fácil utilização tem sido objeto de estudo acadêmico em banco de dados NoSQL, sendo considerado um assunto desafiante, uma vez que existem muitos SGBD NoSQL. A complexidade desse assunto aumenta ao se tratar de tarefas de gerenciamento executadas pelos administradores de bancos de dados (DBA), tais como comandos para concessão de privilégios, criação de tabelas, criação de índices, dentre outros. Os trabalhos de (Nasholm, 2012), (Bach e Werner, 2014), (Lawrence, 2014), (Lee e Zheng, 2015), (Burdakov et al., 2016), (Corbellini et al., 2016), e (Silva et al., 2016) tratam de abordagens de execução de consultas nas bases de dados NoSQL, usando a sintaxe SQL, ou mecanismos amigáveis para recuperação de dados.

A ausência de uma linguagem de consulta padrão contribui para que a comunidade de usuários considere os bancos de dados NoSQL mais complicados no quesito de realização de consultas, e exige mais especialização dos DBA na execução de tarefas de administração nesses bancos de dados.

Neste contexto, a presente dissertação propõe a solução NoSQL^2 para execução de comandos na sintaxe SQL em banco de dados NoSQL, com ênfase nos comandos básicos relacionados às tarefas de administração em banco de dados, tais como criação de objetos e concessão de privilégios.

1.1 O Problema

A realização de consultas em banco de dados NoSQL é considerada complexa para os utilizadores em comparação com as bases de dados relacionais, visto que os bancos de dados NoSQL utilizam formas diversificadas e particulares de acesso aos dados, via acesso nativo ou linguagens de consulta própria, com sintaxe de comandos não padronizada, exigindo especialização para o uso. A complexidade aumenta ao se tratar de tarefas de administração em banco de dados, tais como comandos de definição de estruturas, concessão de privilégios, gerenciamento de transações, dentre outros.

A administração de banco de dados tem um papel essencial na gestão das novas tecnologias de banco de dados, e os bancos de dados NoSQL exigem um maior nível de especialização por parte do DBA, tanto para tarefas de administração, quanto na necessidade de se especializar em tarefas de alto nível, como *backups* e análise de desempenho. Isso expõe questões relacionadas à falta de familiaridade dos administradores de bancos de dados nos ambientes NoSQL, e contribui para aumentar a complexidade no uso desses modelos, que já envolve uma quebra de paradigma relacionado à flexibilidade de esquemas e ao tratamento de dados não estruturados.

As dificuldades dos DBA na execução de tarefas de administração em banco de dados NoSQL, usando um meio ou ferramenta facilitadora, implica em que o DBA necessita conhecer as características das formas de acesso ou linguagem de consulta específicas de cada banco de dados NoSQL sendo administrado.

A questão de pesquisa que orienta esta monografia é desencadeada pelo fato que os bancos de dados NoSQL exigem uma maior expertize por parte dos DBA, para execução das tarefas de gerenciamento e administração atribuídas a eles. O problema deriva da ausência de uma linguagem declarativa de consulta padrão para execução de tarefas de administração em bancos de dados NoSQL. Esse problema é ressaltado na observação de práticas de gerenciamento de banco de dados de profissionais da área, e em artigos científicos e pesquisas acadêmicas sobre o assunto podendo-se citar Corbellini et al. (2016), Burdakov et al. (2016), Abadi et al. (2016), Bugiotti et al. (2014) e Porkorny (2013).

1.2 Motivação

Existem estudos sobre formas de consultas para bancos de dados NoSQL, considerada uma questão desafiadora para a comunidade de pesquisa em banco de dados, pois são catalogados diversos modelos de dados NoSQL, o que torna complexo o estabelecimento de uma linguagem padrão (Corbellini et al., 2016; Burdakov et al., 2016; Abadi et al., 2016; Silva et al., 2016; Gomez et al., 2016; Adriana e Holanda, 2016a; Lee e Zheng, 2015; Yan, 2015; Bach e Werner, 2014; Atzeni et al., 2014; Porkorny, 2013; Kllapi et al., 2013; Nasholm, 2012).

Esses estudos demonstram que bancos de dados NoSQL implementam meios de consulta e recuperação de dados de forma particular, e que a proposta de uma linguagem de consulta comum, ou abordagem facilitadora para realização de consultas, é tema de pesquisa para estes modelos de banco de dados. As pesquisas reforçam a importância do tema realização de consultas em ambientes NoSQL, e como essa atividade é considerada complexa para os DBA em comparação com as bases de dados relacionais tradicionais.

De forma a ressaltar a importância desse tema, apresenta-se uma situação hipotética: Em um contexto de administração de banco de dados NoSQL, um DBA precisa executar comandos do tipo DDL (*Data Definition Language*). O DBA tem conhecimento de como realizar essa ação utilizando comandos da sintaxe da linguagem SQL, porém, como se trata de um banco de dados NoSQL, no qual ele não é especialista, ele tem como opções:

- Acessar diretamente o banco de dados NoSQL, e obter conhecimento sobre a sintaxe dos comandos a ser utilizada naquele banco de dados NoSQL específico; ou
- Conectar-se a uma solução que receberia os comandos informados usando a sintaxe da linguagem SQL. A solução faria a conversão dos comandos para a sintaxe da linguagem proprietária do banco de dados NoSQL, de forma transparente, sem que o DBA precise necessariamente conhecer detalhes dessa linguagem.

A questão de pesquisa no cenário apresentado deriva da escolha mais apropriada a ser realizada pelo DBA.

De forma a contribuir nesse campo de estudo, esta dissertação de mestrado apresenta o *middleware* NoSQL^2 para execução de tarefas de administração em SGBD NoSQL, utilizando uma sintaxe de comandos da sintaxe SQL, e que funciona para diferentes modelos NoSQL.

O NoSQL^2 procura trazer uma proposta de mecanismo facilitador, permitindo que os DBA se desvinculem das especificidades de cada NoSQL, podendo usar o padrão de comandos da linguagem SQL conhecido e amplamente utilizado pela comunidade de banco de dados, e também ser usado em bancos de dados de diferentes modelos NoSQL.

1.3 Objetivos

A presente dissertação possui como objetivo geral o desenvolvimento de um *middleware*, denominado NoSQL^2, que permita a execução de comandos (instruções) administrativos em banco de dados NoSQL utilizando a sintaxe de comandos SQL.

Os objetivos específicos da dissertação estão elencados a seguir:

- Desenvolver a proposta de um *middleware* para execução de tarefas de administração em banco de dados NoSQL;
- Propor, implementar e avaliar estratégia para execução do *middleware* em plataformas heterogêneas compostas por diferentes banco de dados NoSQL;
- Validar o *middleware* através da realização de testes: funcionalidade, extensibilidade e comparativo com outros bancos de dados.

1.4 Estrutura do Documento

Este documento está dividido nos capítulos apresentados a seguir:

- Capítulo 2 apresenta a fundamentação teórica necessária para o desenvolvimento da pesquisa. Os principais conceitos, características e desafios de bancos de dados NoSQL são descritos e discutidos, e são apresentados os trabalhos relacionados com a pesquisa;
- Capítulo 3 explica a solução NoSQL^2 para execução de tarefas de administração em banco de dados NoSQL, especificando a arquitetura da solução;
- Capítulo 4, expõe os resultados dos testes realizados no estudo de caso desenvolvido;
- Capítulo 5, apresenta as conclusões da pesquisa e os trabalhos futuros.

Capítulo 2

Referencial Teórico

Neste capítulo a fundamentação teórica para o desenvolvimento da pesquisa é apresentada, onde são abordados os principais conceitos de banco de dados NoSQL, incluindo detalhes sobre os seus modelos de dados NoSQL, suas formas de acesso aos dados e linguagens de consulta, sendo dividido em 7 seções: Seção 2.1 abordando aspectos da linguagem SQL; Seção 2.2 que aborda os conceitos de análise e síntese de compilação; Seção 2.3 que apresenta as tarefas de administração de banco de dados realizadas pelos DBA; Seção 2.4 conceitua os bancos de dados NoSQL, apresentando as características e modelos NoSQL; Seção 2.5 apresenta características de bases de dados NoSQL em relação aos mecanismos de consulta e sumariza as linguagens de consulta utilizadas pelos bancos de dados NoSQL; Seção 2.6 lista as propostas de formas/linguagens de consulta em banco de dados NoSQL definidas na literatura; e a Seção 2.7 apresenta 2 (duas) pesquisas-enquete realizadas com DBA da cidade de Brasília - Distrito Federal.

2.1 Linguagem de Consulta SQL

Elmasri e Navathe (2011) definem que uma linguagem de consulta é uma linguagem com sintaxe própria, e geralmente de fácil utilização, que permite ao usuário interagir diretamente com o software dos SGBD para realizar consultas em bancos de dados, sendo considerada um meio amigável para os usuários declararem as instruções de consultas e acesso aos dados.

Os SGBD relacionais organizam os dados em tabelas compostas por linhas e colunas, e possibilitam relacionamentos entre as tabelas. Desta forma, os dados são estruturalmente organizados e podem ser acessados via formas de acesso nativa do SGBD ou linguagens de consulta. Os SGBD relacionais definem um esquema de dados como uma descrição (textual ou gráfica) da estrutura de um banco de dados de acordo com um determinado modelo de dados. As consultas em um banco de dados geralmente recuperam dados das

tabelas, ou manipulam objetos do SGBD. Uma consulta é composta por uma sequência de caracteres contendo um significado coletivo, tais como palavras-chave, nomes de atributos, nomes de relação, operadores, operandos, constantes, literais, cadeias e símbolos de pontuação. Essa sequência de caracteres é denominada *token* (Aho et al., 1986).

A linguagem de consulta estruturada, denominada SQL, é uma das principais linguagens utilizadas para consultas, manipulação e recuperação de dados para os SGBD relacionais e fornece uma sintaxe declarativa de alto nível para que os usuários facilmente declarem o que eles desejam (Elmasri e Navathe, 2011; Silberschatz et al., 2010).

Em (Silva et al., 2016; Lawrence, 2014; Mullins, 2012; Silberschatz et al., 2010) aponta-se razões pelas quais a característica de suporte às consultas, tal qual SQL, é interessante para o SGBD:

- SQL permite consultas descritivas, onde os detalhes de execução e implementação da consulta são transparentes ao usuário;
- SQL facilita a portabilidade entre sistemas de banco de dados;
- Linguagem declarativa que especifica a lógica de um programa (o que precisa ser feito) em vez do fluxo de controle (como fazê-lo);
- Padrão *American National Standards Institute* (ANSI) e da *Internacional Standardization Organization* (ISO);
- SQL é amplamente conhecida pelos usuários e administradores de banco de dados;
- SQL possue suporte a integração com outros sistemas que usam SQL e JDBC/ODBC;
- SQL possue comandos que permitem aos administradores de banco de dados realizarem tarefas de administração de dados.

SQL contempla comandos de manipulação de dados (DML - *Data Manipulation Language*), comandos de definição de estrutura (DDL - *Data Definition Language*), comandos para controle de dados (DCL - *Data Control Language*) e comandos para transação de dados (DTL - *Data Transaction Language*). Estes comandos são úteis, por exemplo, no acesso ao dicionário de dados, durante a migração de um banco de dados para uma plataforma diferente, ou na definição de índices de forma a melhorar o desempenho de uma consulta, dentre outras situações. A Tabela 2.1 sumariza os tipos de comandos, sendo que os comandos DML mais comuns são *select, insert, update* e *delete*, usados para manipulação de dados, e os comandos DDL utilizados para criar e modificar esquemas, tabelas e restrições (Silberschatz et al., 2010).

Tabela 2.1: Tipos de Comandos SQL.

Tipo de Comandos	Descrição
Data Manipulation Language (DML)	Realiza inclusões, consultas, exclusões e alterações de dados. Os principais comandos são INSERT, SELECT, UPDATE e DELETE.
Data Definition Language (DDL)	Definição de objetos tais como tabelas, colunas, índices. Os principais comandos são CREATE, DROP e ALTER.
Data Control Language (DCL)	Concede e revoga permissões ao usuário para realizar ações sobre os objetos e dados. Os principais comandos são GRANT e REVOKE.
Data Transaction Language (DTL)	Gerencia transações ocorridas dentro do banco de dados. Os principais comandos são COMMIT e ROLLBACK.

(Gessert e Ritter, 2016) e (Elmasri e Navathe, 2011) ressaltam que os sistemas de banco de dados relacionais utilizam a linguagem de consulta SQL, devido a seu conjunto de recursos, tais como manipulação de objetos e gerenciamento de transações, e também, os SGBD relacionais geralmente fornecem uma interface interativa: *ad hoc* para executar comandos SQL o que facilita a realização de tarefas de administração em banco de dados pelos administradores de bancos de dados. Entre os principais sistemas gerenciadores de banco de dados que usam a linguagem SQL destacam-se: *Microsoft SQLServer, Oracle, MySQL, Sybase*, e *DB2 IBM*.

2.2 Tradução de Sintaxe

Uma revisão de conceitos pertinentes a Teoria de Compiladores faz-se necessária de forma a facilitar o entendimento das próximas seções que tratam da solução arquiteturial implementada nessa dissertação.

Os conceitos aqui apresentados são baseados no trabalho de Aho et al. (1986), onde as fases de compilação relacionadas a análise e síntese são descritas como segue. A análise é responsável por dividir o programa fonte nas partes constituintes e criar uma representação intermediária do mesmo. A síntese é responsável por construir o programa alvo desejado, a partir da representação intermediária.

Como ilustrado na Figura 2.1, durante a análise, são realizadas:

- Análise linear, léxica ou esquadrinhamento (*scanning*): um fluxo de caracteres de um programa fonte é lido da esquerda para a direita e agrupado em *tokens*;

- Análise sintática, gramatical ou hierárquica: agrupa os *tokens* do programa fonte em frases gramaticais usadas pelo compilador, a fim de sintetizar a saída;
- Análise semântica: verificações são realizadas a fim de se assegurar que os componentes de um programa fonte se combinam de forma significativa;
- Geração de Código: efetuada pelos dispositivos de tradução dirigida pela sintaxe para geração de um código intermediário.

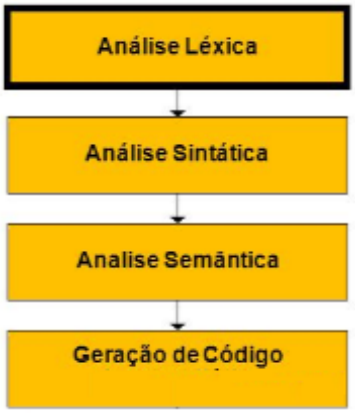


Figura 2.1: Fases de Compilação.

Geralmente, as construções léxicas não requerem recursão, enquanto as sintáticas frequentemente a exigem. O analisador léxico é a primeira fase de um compilador, tendo como tarefa principal ler os caracteres de entrada e produzir uma sequência de *token* que o *parser* utiliza para a análise sintática. Os analisadores léxicos são divididos em duas fases em cascata, a primeira chamada de "varredura"(*scanning*) e a segunda de "análise léxica". O *scanner* é responsável por realizar tarefas simples, enquanto o analisadore léxico realiza as tarefas mais complexas.

A fase de análise semântica verifica os erros semânticos no programa fonte e utiliza a estrutura hierárquica determinada pela fase de análise sintática, a fim de identificar os operadores e operandos das expressões e enunciados.

Após a execução das análises sintática e semântica, alguns compiladores geram uma representação intermediária explícita do programa fonte. A fase final do compilador é a geração do código alvo, consistindo normalmente de código de máquina relocável ou código de montagem, ou código traduzido. As fases da compilação são usualmente implementadas em uma única passagem, consistindo na leitura de um arquivo de entrada e da escrita de um arquivo de saída.

Conforme conceitura Aho et al. (1986), os dispositivos de tradução dirigida pela sintaxe produzem coleções de rotinas que percorrem uma árvore gramatical, gerando um código

intermediário. Definição dirigida pela sintaxe especifica a tradução de uma construção em termos dos atributos associados aos seus componentes sintáticos. A gramática e o conjunto de regras semânticas constituem a definição dirigida pela sintaxe. A tradução é um mapeamento de entrada e saída. Os geradores automáticos de código obtém uma coleção de regras que definem a tradução de cada operação da linguagem intermediária para a linguagem da máquina alvo.

Resumidamente, com os esquemas de definições dirigidas pela sintaxe e de tradução, sintaticamente analisa-se o fluxo de *tokens* de entrada, constroí-se a árvore gramatical, e, em seguida, ela é percorrida, avaliando as regras semânticas a cada nó. Casos especiais de definições dirigidas pela sintaxe podem ser implementados numa única passagem através da avaliação das regras semânticas durante a análise sintática, sem explicitamente construir uma árvore gramatical ou um grafo que exiba as dependências entre os atributos.

2.3 Tarefas de Administração de Dados

Elmasri e Navathe (2011) afirmam que os SGBD devem fornecer estruturas de dados e técnicas avançadas para melhorar as atividades de acesso aos dados. A administração das bases de dados contempla tarefas para manutenção do ambiente, sendo que o uso de softwares para administração faz parte da rotina administrativa de qualquer banco de dados, facilitando a detecção de problemas, análise de desempenho, geração de relatórios, e manipulação de dados e objetos. A administração de banco de dados é geralmente entendida como um conjunto de tarefas de gestão e manutenção dos SGBD, começando com o projeto de banco de dados e incluindo atividades de concessão de privilégios, realização de *backup* e recuperação em caso de perda de dados. Em um ambiente de banco de dados, as tarefas de gestão e manutenção dos bancos de dados são de responsabilidade dos DBA.

Os DBA tem a autoridade central para gerenciar o banco de dados, e executa tarefas de administração que envolvem atividades como as descritas em (Mullins, 2012):

(i) Projeto do banco de dados;

(ii) Criação do banco de dados;

(iii) Controle de acesso;

(iv) Concessão e revogação de privilégios;

(v) Migração de estrutura e dados entre bancos de dados;

(vi) Monitoramento do uso de recursos de software e hardware;

(vii) Melhoria de desempenho;

(viii) Auditoria do banco de dados;

(ix) Estratégia de *backup* e recuperação de dados.

Para executar as tarefas de administração em banco de dados, os DBA comumente utilizam meios de acesso disponibilizados pelo SGBD. Esses meios podem ser acionados por linhas de comandos, interfaces *ad hoc* ou softwares disponibilizados pelas base de dados, e caso existam, também são usados softwares auxiliares disponibilizados por terceiros. Uma linguagem de consulta facilita a execução das tarefas de administração em bancos de dados, reduzindo a complexidade na interação do usuário com o banco de dados. Os administradores de bancos de dados que utilizam linguagens de consulta na execução de tarefas de gerenciamento empregam comandos de mais alto nível, tais como comandos para concessão de privilégios, criação de tabelas e objetos. A ausência de uma linguagem de consulta, torna a atividade administrativa mais complexa, e exige um grau de especialização dos usuários para que possa ser executada apropriadamente (Gomez et al., 2016).

2.4 Banco de Dados NoSQL

A nomenclatura NoSQL indica que o dialeto da linguagem SQL não é suportado, o modelo de dados é não-relacional e carece de definições de esquema. Alguns autores consideram essa nomenclatura confusa, e sugerem o uso de outros termos, tais como, NOREL (Não relacionais) ou *NoJoin* (Não JOIN) (Sharma et al., 2017; Gomez et al., 2016; Lee e Zheng, 2015; Bach e Werner, 2014; Porkorny, 2013; Nasholm, 2012).

O volume de dados, estruturados e não estruturados, gerados atualmente, traz desafios para a área de gerenciamento e processamento de dados. Tauro et al. (2012) destacam um aumento de quase 25 vezes no volume de dados entre os anos de 2007 a 2010. Estatísticas do grupo *Gartner* destacam que 80% são dados não estruturados (Wei e Bo, 2010; Liu et al., 2011). Burdakov et al. (2016) ressaltam que o volume de dados diários gerados pelo *Twiter* é de 12 TB, enquanto do *Facebook* é de 500 TB.

Vários autores, entre eles, Gonzalez et al. (2016), Farias et al. (2016), Corbellini et al. (2016), Silva et al. (2016), Ringlstetter et al. (2016), Lee e Zheng (2015), Atzeni et al. (2014), Porkorny (2013), Kllapi et al. (2013) citam em seus trabalhos que as aplicações *Web 2.0* e *Big Data* tornam importante o gerenciamento de dados através do uso de tecnologias para armazenar e processar dados em massa. Esses mesmos autores comentam que os bancos de dados NoSQL são alternativas viáveis para essas aplicações, pois propiciam recursos eficientes para armazenamento de grandes volumes de dados estruturados e não-estruturados, fácil acesso, alta escalabilidade e disponibilidade, além de baixo custo.

O catálogo NoSQL presente na fonte de pesquisa *NoSQL-Database.org*[1] contém uma

[1]NoSQL: URL: http://nosql-database.org/

lista dos bancos de dados NoSQL, e o *DB-Engines Ranking*[2] apresenta índices de heterogeneidade dos SGBD por categoria de modelo de banco de dados (relacional e não-relacional).

Sharma et al. (2017), Gessert e Ritter (2016), Atzeni et al. (2014), Kllapi et al. (2013), Tauro et al. (2012), Indrawan (2012), Muhammad (2011) apresentam como características gerais dos NoSQL:

- Não fornece suporte a propriedades transacionais ou formas normais;
- Processamento distribuído: projetado para arquiteturas distribuídas em rede;
- Alta Disponibilidade: projetado para lidar com falhas de maneira eficiente;
- Alta escalabilidade e confiabilidade: distribuição e fragmentação de dados em diferentes servidores de rede (nós);
- Flexibilidade de esquema: usualmente existem modelos de dados NoSQL sem esquema, viabilizando processos de leitura e escrita mais rápidos;
- Suporte à replicação de dados: replicação de dados em diversos nós;
- Armazenamento de dados estruturados e não estruturados: projetados para tratar com diversos tipos de dados como texto, multimídia, mídia social, gráficos, imagens, áudio e vídeo;
- Acesso aos dados via API: não utilização de linguagem de uma consulta, como SQL;
- Menos aderência às propriedades ACID;
- Alta Performance: benefícios na melhoria de desempenho dos bancos de dados.

Os autores Sharma et al. (2017), Gessert e Ritter (2016), Lotfy et al. (2016), Bach e Werner (2014), Indrawan-Santiago (2012), Hecht e Jablonski (2011) classificam os bancos de dados NoSQL em quatro principais modelos: Chave-Valor, Orientado a Colunas, Orientado a Documentos, e Baseado em Grafos. No banco de dados chave-valor, os dados são armazenados como pares chave-valor, e são endereçados por uma chave. Os bancos de dados orientado a colunas definem a estrutura de valores como um conjunto pré-definido de colunas. No banco de dados orientado a documentos, os documentos são conjuntos de atributos e valores, e cada documento contém um identificador. Um banco de dados baseado em grafos utiliza o modelo de grafos para representar o esquema de dados.

[2]DB-engines Ranking: URL http://db-engines.com/en/ranking

A Figura 2.2 ilustra o *ranking* de popularidade dos sistemas de banco de dados, onde se destacam os NoSQL dos modelos Chave-Valor, Orientado a Documentos, Orientado a Colunas, e Baseado em Grafos. As informações foram obtidas, em novembro de 2016, através do site *DB-Engines Ranking*[2]. Cada modelo NoSQL é descrito como se segue.

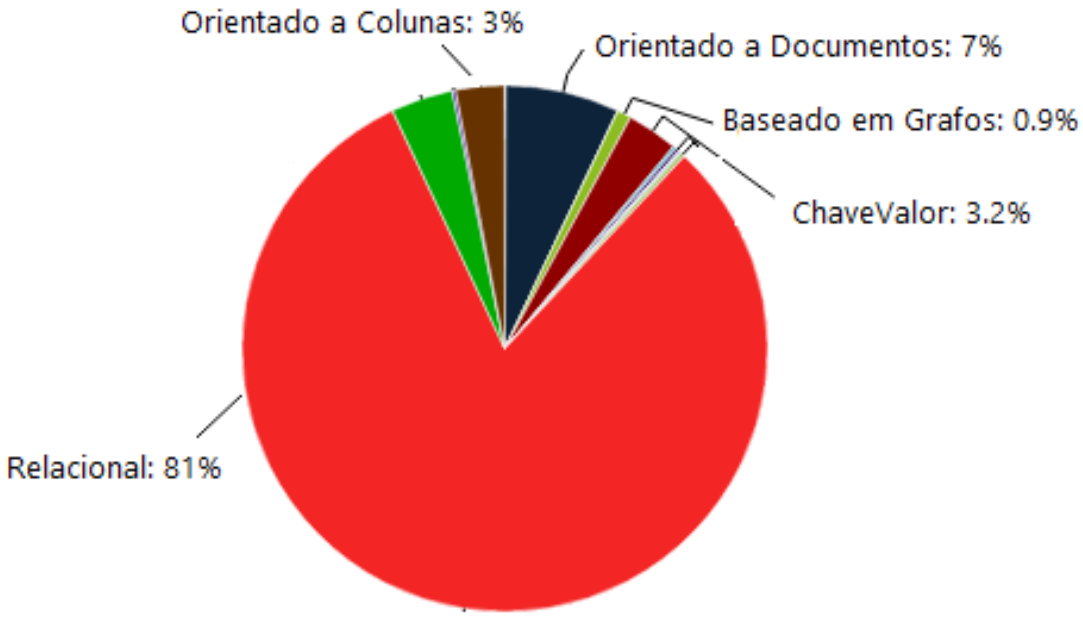


Figura 2.2: Índice de Popularidade dos SGBD por categoria (adaptado de *DB-engines Ranking* em novembro de 2016).

Modelo Chave-Valor

Nos SGBD NoSQL baseados em chave-valor, os dados são armazenados como pares chave-valor. Estes sistemas são semelhantes a um dicionário de palavras, onde os dados são endereçados por uma chave (palavra). Os valores são isolados e independentes, e o relacionamento é tratado pela lógica da aplicação (Gonzalez et al., 2016).

O banco de dados chave-valor é livre de esquema, sendo que a aplicação-cliente fica responsável por resolver problemas de incompatibilidade, caso exista. Como exemplo, a Figura 2.3 apresenta o banco de dados de uma biblioteca, onde a chave refere-se ao campo ISBN (*International Standard Book Number*) (Hakala, 2001) e o valor armazenado é o identificador ISBN de um livro.

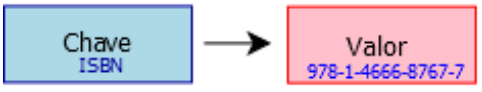


Figura 2.3: Modelo Chave-valor.

Este tipo de banco de dados é considerado útil para aplicações em que o processamento das transações é baseado em chaves e para aplicações que realizam constantes leituras nos dados, tal como o catálogo *on-the-fly*[3] (Indrawan-Santiago, 2012). O banco de dados chave-valor é uma solução adequada para aplicações simples que funcionam com um único tipo de objeto, e esses objetos são baseados em um único atributo (Catell, 2010).

Exemplos de banco de dados chave-valor são: Toquio Tyrant (Han et al., 2011), Berkeley DB (Olson et al., 1999), Voldemort (Sumbaly et al., 2012), Dynamo (DeCandia et al., 2007), Riak (Klophaus, 2010) e Redis (Carlson, 2013).

Modelo Orientado a Colunas

Os sistemas gerenciadores de banco de dados NoSQL orientado a colunas definem a estrutura de valores como um conjunto pré-definido de colunas (Figura 2.4). Esse modelo pode ser considerado como um esquema de base de dados, organizando os dados com base em uma distribuição de colunas (Gessert e Ritter, 2016; Schreiner, 2016).

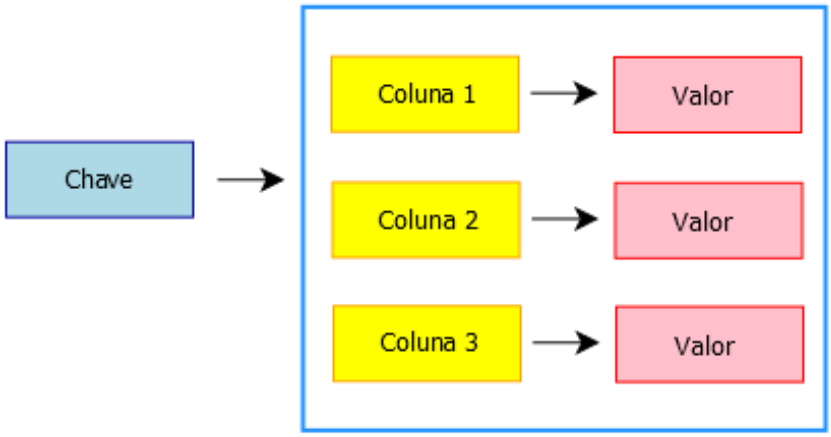


Figura 2.4: Modelo Orientado a Colunas.

Nestes sistemas, cada linha de dados não necessita possuir o mesmo grau, ou seja, pode ter um número variável de colunas, e devido a esta característica, esse modelo tem dados espaçados. As colunas de uma tabela estão divididas sobre os nós usando o conceito de grupos de colunas. Grupos de colunas são uma maneira simples dos usuários indicarem quais colunas devem ser armazenadas juntas (Catell, 2010).

Este modelo é adequado para aplicações que tratam com grandes volumes de dados, de modo que o modelo de dados pode ser eficientemente particionado (Hecht e Jablonski, 2011). De acordo com Catell (2010), para os casos onde se deseja otimizar a leitura de dados estruturados, o modelo orientado a colunas é mais interessante, pois mantêm os

[3]Aplicações que ocorrem de forma dinâmica e não como o resultado de algo que é estaticamente predefinido.

dados de forma contígua por coluna. Os autores Kaur e Rani (2013) consideram esses bancos de dados apropriados para propósitos analíticos porque eles podem tratar com colunas específicas.

Exemplos de banco de dados orientado a colunas são BigTable (BigTable, 2008), HBase (Hbase, 2011) e Cassandra (Cassandra, 2010).

Modelo Orientado a Documentos

Nos SGBD NoSQL orientado a documentos, os documentos são conjuntos de atributos e valores, onde um atributo pode ser multivalorado. As chaves dentro dos documentos são únicas. Cada documento contém um identificador, que é único dentro do conjunto (Figura 2.5).

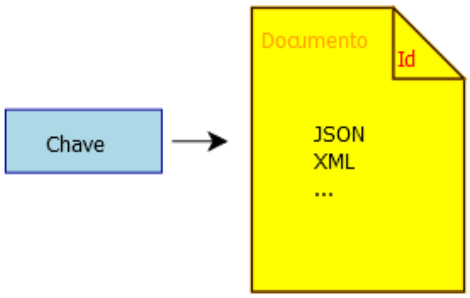


Figura 2.5: Modelo Orientado a Documentos.

Em geral, o banco de dados orientado a documentos não possui esquema: o documento não precisa ter uma estrutura comum. Esse modelo costuma armazenar os valores em uma estrutura como JSON (*JavaScript Object Notation*) ou XML (*Extensible Markup Language*). O formato JSON suporta os tipos de dados *list, map, date, boolean* e também números com diferentes precisões (Gessert and Ritter, 2016; Indrawan-Santiago, 2012; Porkorny, 2013). O formato XML também suporta diferentes tipos de dados.

Catell (2010) aponta que o modelo orientado a documentos geralmente suporta índices e diversos tipos de documentos (objetos) por banco de dados, além de documentos aninhados ou listas.

Hecht e Jablonski (2011) afirmam que o modelo orientado a documentos é de fácil manutenção e são portanto desejáveis para aplicações *web* que precisam executar consultas dinâmicas, tais como aplicações de análise em tempo real e *blogs*. Um exemplo de consulta dinâmica é quando se tem diferentes tipos de objetos, e é necessário pesquisar objetos com base em múltiplos campos.

Exemplos de banco de dados orientado a documentos são: MongoDB (Banker, 2011), CouchDB (Anderson et al., 2010). Riak (Klophaus, 2010) também é um exemplo de

banco de dados desse modelo, sendo descrito como um SGBD do modelo Chave-Valor e Orientado a Documento.

Modelo Baseado em Grafos

Os SGBD NoSQL baseado em grafos, utilizam o modelo de grafos para representar o esquema. A modelagem dos dados é um conjunto de vértices e arestas, expressa como G = (V,E), onde V é o conjunto de vértices (nós) e E é o conjunto de arestas. Cada aresta representa uma relação entre dois nós. Um conjunto de vértices conectados por meio de arestas definem um caminho no grafo (Hecht e Jablonski, 2011 ; Schreiner, 2016). O modelo baseado em grafos funciona com três abstrações: nós, as arestas entre nós e o par chave-valor anexado aos nós e relacionamentos (Corbellini et al., 2016; Tauro et al., 2012; Nasholm, 2012).

A Figura 2.6 apresenta um exemplo de um conjunto de nós e arestas de um banco de dados.

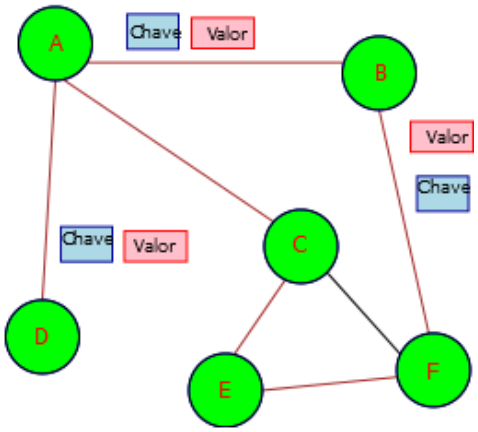


Figura 2.6: Modelo Baseado em Grafos.

Este modelo suporta a utilização de restrições de integridade de dados, garantindo assim as relações entre elementos de forma consistente. (Hecht e Jablonski, 2011) e (De Diana e Gerosa, 2010) comentam que os bancos de dados baseado em grafos são aplicáveis para bancos de dados com muitos relacionamentos entre os dados, e também quando informações sobre a topologia ou interconectividade dos dados é importante, ou tão importante como os dados em si.

Exemplos de bases de dados baseada em grafos são: Neo4J

Com relação a popularidade dessas bases de dados, o *DB-Engines Ranking*[2] apresenta graficamente o *ranking* de popularidade dos SGBD, onde se destacam SGBD NoSQL entre eles, o MongoDB na 4ª posição, Cassandra na 7ª posição, Redis na 10ª posição, e o Neo4J na 21ª posição (Figura 2.7).

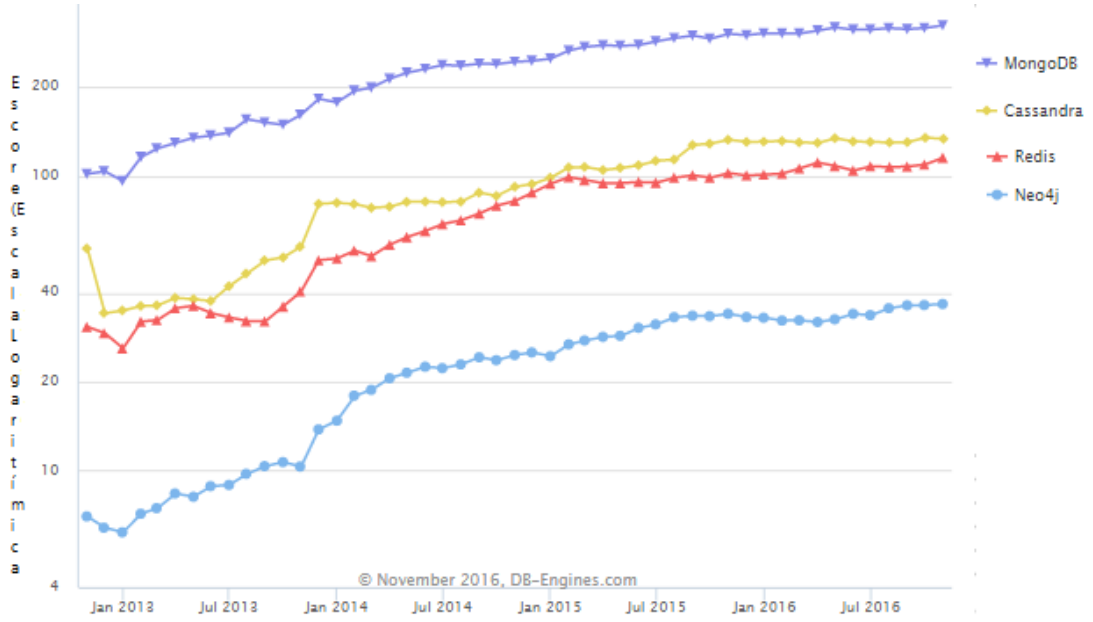


Figura 2.7: Índice de Popularidade SGBD (adaptado de *DB-engines Ranking* em novembro de 2016).

Esses índices de popularidade foram um incentivo para a utilização desses sistemas gerenciadores de banco de dados NoSQL nos testes dessa dissertação.

2.5 Consultas em Bancos de Dados NoSQL

Diferentemente do padrão da linguagem SQL adotado por vários SGBD relacionais, as linguagens de consulta existentes nos NoSQL exigem que os usuários tenham conhecimento sobre o conjunto de comandos disponíveis e a sintaxe própria da linguagem, de forma a serem capazes de usar seu mecanismo de consulta e recuperação de dados.

Cada sistema gerenciador de banco de dados NoSQL possui uma linguagem de consulta com sintaxe de comandos própria e não padronizada nem mesmo entre os modelos, forçando que o usuário tenha conhecimento da sintaxe de comandos adotada por aquela linguagem em questão. A ausência de uma linguagem de consulta padrão contribui para que a comunidade de usuários considere os bancos de dados NoSQL mais complicados no quesito de realização de consultas, além de exigir mais especialização dos administradores de bancos de dados em cada base NoSQL durante a execução de tarefas de administração. Os autores Abadi et al. (2016), Guo et al. (2016), Burdakov et al. (2016), Bach e Werner (2014) e Nasholm (2012) ressaltam que o estabelecimento de uma linguagem de consulta padronizada para os bancos de dados NoSQL, tal como SQL, é um importante desafio e objeto de pesquisa pela academia de banco de dados.

2.5.1 Características de Consultas em Bancos de Dados NoSQL

A heterogeneidade de mecanismos de consulta, recuperação de dados, e linguagens de consulta presentes nos modelos NoSQL exige dos usuários um nível de conhecimento específico em cada NoSQL, pois não existe padronização de linguagens de consulta nem mesmo entre os modelos NoSQL. As características dos principais bancos de dados NoSQL são analisadas na sequência.

Redis

Redis é um banco de dados chave-valor, flexível e sem estrutura definida, tendo como linguagem de desenvolvimento C (Atzeni et al., 2014). Hecht e Jablonski (2011) mostraram que Redis pode tratar com mais de 100.000 operações de leitura e escrita por segundo, e possui um conjunto nativo de comandos e operadores que podem ser usados para acessar, inserir e retornar os dados. O Código 2.1 mostra a inserção de 2 elementos (chave 0: `Capítulo` e chave -1: `Título`) na lista, denominada `minhaLista`. Nas linhas 1 e 3, são inseridos os elementos `Capítulo` e `Título` na lista `minhaLista`. Nas linhas 5 e 7, esses elementos são consultados pela chave 0 ou -1. O resultado é apresentado nas linhas 6 e 8.

Código 2.1: Redis

```
1 LPUSH minhaLista "Capitulo"
2 (Integer) 1
3 LPUSH minhaLista "Titulo"
4 (Integer) 2
5 LINDEX minhaLista 0
6 "Capitulo"
7 LINDEX minhaLista -1
8 "Titulo"
9 }
```

Uma chave *key* pode pertencer a uma *list* ou a um *set*, onde *list* representa conjuntos ordenados de *strings*, e *set* representam conjuntos desordenadas de *strings*. O conjunto de comandos do Redis contém os operadores nativos tais como: *insert, delete, lookup operations, list, lpush, lindex* e *set* (Gonzalez et al., 2016; Corbellini et al., 2016; Atzeni et al., 2014).

Tokyo Tyrant / Tokyo Cabinet

Tokyo Tyrant (TT) e Tokyo Cabinet (TC) são bases de dados chave-valor, escritas em C suportados por *Sourcefourge.net* e FAL Labs (Han et al., 2011). Tauro et al. (2012) esclarecem que TC são servidores de armazenamento de alto desempenho e TT é a biblioteca cliente para acesso remoto.

TT e TC podem lidar com cerca de 4 ou 5 milhões de operações de leitura e escrita por segundo, garantindo alto desempenho e utilizando mecanismos confiáveis para persistência dos dados. As consultas são realizadas usando a sintaxe Lucene (Hatcher and Gospodnetic, 2004), e suportam operações comuns de *get, set* e *update*. O Código 2.2 mostra a seleção de registros da lista `capitulo`. A linha 1 especifa o comando `GET` que retorna o valor de um registro a partir de uma chave. A linha 2 especifica o comando `GETLIST` que a partir do valor de uma chave retorna as chaves e valores dos registros correspondentes.

Código 2.2: TT e TC

```
1 capitulo: GET(chave)
2 capitulo: GETLIST(chave)
```

Cassandra

Cassandra (Cassandra, 2010) é um sistema gerenciador de banco de dados orientado a colunas, que foi inicialmente desenhado pela equipe de engenharia do *Facebook*, em 2008, com base no *Dynamo* e *BigTable* (Catell, 2010). Sharma et al. (2017) e Muhammad (2011) descrevem Cassandra como tendo alta escalabilidade, durabilidade e tolerância a falhas.

Cassandra tem uma linguagem de consulta denominada CQL (*Cassandra Query Language*), semelhante ao SQL, onde junções e subconsultas não são suportadas (Sharma et al., 2017).

O Código 2.3 apresenta uma consulta (linha 1) as colunas `capitulo_escritor` e `capitulo_nome` da tabela `Capítulo` (linha 2), onde a coluna `capitulo_ano` compreende o ano de 1988 (linha 3). Como pode ser observado no código 2.3, a sintaxe da linguagem CQL é semelhante a sintaxe SQL.

Código 2.3: Cassandra

```
1 SELECT capitulo_escritor, capitulo_nome
2 FROM Capitulo
3 where capitulo_ano='1988'
```

HBase

HBase (Hbase, 2011) é um SGBD do modelo Orientado a Colunas, *open source*, projetado pela Apache em 2007. HBase é modelado com base no *BigTable* (BigTable, 2008) e HDFS (*Hadoop Distributed File System*) ((Shvachko et al., 2010). Em relação ao acesso aos dados, HBase usa o *framework MapReduce* (Dean and Ghemawat, 2008) e permite o acesso via *java*, *Thrift API*[4], REST (Masse, 2011) e suporta JDBC/ODBC.

O Código 2.4 insere e lista dados na tabela `Capítulo`, onde as linhas 1 e 2 inserem valores os `Maria` e `João` na coluna `escritor` da tabela `Capítulo` e a linha 3 lista o conteúdo da tabela `Capítulo`.

Código 2.4: HBase

```
1 PUT  'Capitulo', 'escritor','Maria'
2 PUT  'Capitulo', 'escritor','JoÃ£o'
3 LIST 'Capitulo'
```

HyperTable

HyperTable (Khetrapal and Ganesh, 2006) é um banco de dados orientado a colunas, *opensource*, com estrutura semelhante ao HBase e *BigTable* (BigTable, 2008). Ele foi desenvolvido em C++ e tem como patrocinador a empresa Baidu (Banker, 2011). Em relação ao acesso aos dados, HyperTable possui uma interface cliente para programação (Catell, 2010).

HyperTable utiliza uma linguagem de consulta denominada HQL (*Hypertable Query Language*). HQL é uma linguagem declarativa semelhante ao SQL. O agrupamento lógico de tabelas é feito por *namespaces* que podem ser comparados com a hierarquia de pastas de sistemas de arquivos.

O Código 2.4 mostra uma consulta dos capítulos de livro onde o nome do escritor obedece ao critério definido como escritor igual a `Souza`. A linha 1 seleciona a coluna `capitulo_nome` da tabela `Capítulo` (linha 2) onde o `capitulo_escritor` é igual a `Souza` (linha 3). Como pode ser observado no código, a sintaxe da linguagem HQL é semelhante a sintaxe SQL.

Código 2.5: HyperTable

```
1 SELECT capitulo_nome
2 FROM Capitulo
3 where capitulo_escritor='Souza'
```

[4]Thrift API: URL: https://thrift.apache.org/

MongoDB

O MongoDB (Banker, 2011) é um banco de dados orientado a documentos, cuja linguagem de desenvolvimento é C++, e o fornecedor é 10gen[5].

MongoDB tem como características a escalabilidade horizontal e a alta disponibilidade (Kaur e Rani, 2013). A simplicidade e flexibilidade do modelo contribui para que os usuários tenham facilidade de uso do banco de dados. Tauro et al.. (2012) apontam que, devido às suas características, muitos projetos que tem expectativa de grande aumento no volume de dados usam MongoDB. Outra interessante comparação é que MongoDB provê acesso rápido aos dados.

MongoDB armazena dados complexos, tais como *arrays*, *timestamp*, *float*, *date*, dados binários e expressões regulares, em formatos binários denominado JSON e BSON (*Binary JSON*) (Moniruzzaman and Hossain, 2013).

Este banco de dados possui uma linguagem de consulta denominada MQL (*MongoDB Query Language*), que permite funções tais como nos bancos de dados relacionais, suporta índices e fornece operações adicionais tais como *count* e *distinct*, além de poder realizar consultas com base em um predicado, mas não suporta *joins* (Tauro et al., 2012). O Código 2.6 mostra a instrução SELECT escrita na sintaxe MQL, onde a linha 1 seleciona todos os escritores que tem mais de 4 livros escritos. Como pode ser observado no código, a sintaxe da linguagem MQL não é semelhante a sintaxe SQL.

Código 2.6: MongoDB

```
1 DB.COLLECTION.FIND({escritores},{quantidade: { $gt: 4 }})
```

CouchDB

CouchDB (Anderson et al., 2010) é um sistema gerenciador de banco de dados orientado a documentos criado em 2008. A linguagem de desenvolvimento é Erlang (Armstrong et al., 1993), e o fornecedor é o Apache. CouchDB é um banco de dados flexível, tolerante a falhas e usa o formato de dados JSON.

O banco de dados CouchDB usa um mecanismo denominado *view* escrito em *JavaScript* para realizar consultas, e essas podem ser distribuídas sobre os nós usando o *framework* MapReduce (Dean and Ghemawat, 2008). O acesso ao banco é realizado através de interface REST (Sharma et al., 2017).

De acordo com Catell (2010), CouchDB suporta linguagem de consulta procedural e possui funcionalidades básicas para administração do banco de dados. O Código 2.7 mostra uma consulta para retornar os capítulos de livro onde o escritor é `Souza`. A linha 1

[5]10gen: URL: https://www.mongodb.com/company

declara a função `Capitulo`, e na linha 2 a condição de `capítulo_escritor` igual a `Souza` é verificada. Caso a condição seja atendida o resultado deverá ser exibido (linha 3).

Código 2.7: CouchDB

```
1 FUNCTION(Capitulo) {
2         IF (Capitulo.capitulo_escritor == "Souza")
3                 EMIT(capitulo_id, Capitulo);
4 }
```

Riak

Riak (Klophaus, 2010) é um sistema distribuído fornecido pela empresa Basho[6], licenciado *Apache2 Open Source*, desde 2009. Riak é descrito como SGBD chave-valor e orientado a documento. A linguagem de desenvolvimento é Erlang (Armstrong et al., 1993), que foi criada pela Ericsson para suporte a aplicações distribuídas tolerantes a falhas executada em ambientes de tempo real (Catell, 2010).

Riak fornece uma interface de consulta amigável baseada em HTTP/JSON e usa o *framework MapReduce* (Dean and Ghemawat, 2008) e acesso via REST e operadores *get, put, post* para armazenamento e leitura. Riak usa a linguagem de consulta com a sintaxe Lucene (Hatcher and Gospodnetic, 2004) que é um produto de código aberto da Apache.

O Código 2.8 usa a função `SEARCH_KEY` para retornar os capítulos de livro no intervalo de uma determinada data. A linha 1 busca os nomes dos livros `capitulo_nome` cuja data de publicação do livro `capitulo_data` esteja entre 01 de janeiro a 01 de fevereiro de 2015.

Código 2.8: Riak

```
1 SEARCH_KEY(capitulo_nome, capitulo_data:[20150101 TO 20150201]
```

Riak é recomendado para aplicações que fazem muitas leituras e requisições de escrita paralelas, por exemplo, para aplicativos de jogos.

Neo4J

Neo4J (Vukotic et al., 2015) é um banco de dados baseado em grafos, de alto desempenho, e com suporte a replicação. Neo4j foi desenvolvido em *java* tendo como fornecedor *NeoTechnologies*. Esse banco de dados usa uma linguagem declarativa denominada Cypher (Holzschuher and Peinl, 2013), e as linguagens de *script* Gremlin (Holzschuher and Peinl, 2013) e JRuby (Nutter et al., 2011) para consultas de seus dados.

[6]Basho: URL:http://basho.com/

Cypher é uma linguagem declarativa para realização de consultas e atualizações em grafos. Ela se baseia na clareza da declaração da expressão para a recuperação do item a partir do grafo, em contraste com linguagens imperativas como *java* e linguagens de *script* como Gremlin e JRuby (Bach e Werner, 2014). A sintaxe da linguagem Cypher é semelhante a SQL e foi projetada para ser uma linguagem de consulta parecida com o vocabulário humano, de forma a facilitar o entendimento pelos usuários (Kaur e Rani, 2013). O Código 2.9 demonstra a expressividade da linguagem através da consulta aos capítulos de livro onde o nome do autor obedece ao critério definido. A cláusula `START` (linha 1) especifica o ponto de partida no grafo, a partir de onde a consulta é executada. A linha 2 especifica a condição do nó a ser alcançado `capitulo_escritor='Souza'` e a linha 3 retorna o resultado com os capítulos que atendem a condição especificada.

Código 2.9: Neo4J

```
1 START
2 NODE=Capitulo:Capitulo(capitulo_escritor='Souza')
3 RETURN Capitulo
```

AllegroGraph

AllegroGraph (Aasman, 2006) é um sistema gerenciador de banco de dados baseado em grafos fornecido por *Franz Inc*[7]. AllegroGraph utiliza uma linguagem de consulta denominada SPARQL (*SPARQL Protocol And RDF Query Language*) para arquivos RDF projetada pelo grupo W3C RDF. A sintaxe da linguagem SPARQL é semelhante ao SQL clássico, sendo considerada uma linguagem de consulta popular, declarativa e com uma sintaxe simples.

O Código 2.10 consulta os capítulos de livro onde o nome do escritor obedece ao critério definido. A linha 1 seleciona todos os escritores que atende o critério estabelecido na linha 2, que é o nome ser `Souza`.

Código 2.10: AllegroGraph

```
1 SELECT *
2 ?Capitulo FTI:MATCH('Souza')
```

[7]Franz: URL:http://franz.com/

2.5.2 Análise das Linguagens de Consulta NoSQL

As linguagens de consulta oferecem um alto nível de abstração, além de reduzir a complexidade na interação do usuário com o banco de dados e são úteis para os usuários e DBA, que precisam de acesso as estruturas de dados usando mecanismos amigáveis.

O capítulo *Query languages in NoSQL databases* do livro *Handbook of Research on Innovative Database Query Processing Techniques*, foi elaborado durante o desenvolvimento do Mestrado (Adriana e Holanda, 2015). Esse capítulo faz um mapeamento sobre os avanços na área, e desafios relacionados a formas e técnicas de consultas aos dados armazenados em bancos de dados NoSQL. Como resultado foi possível obter um embasamento teórico sobre o tema realização de consultas em banco de dados NoSQL, as características de consulta nas bases de dados NoSQL e a problemática sobre a ausência de uma linguagem declarativa de consulta nos NoSQL.

O capítulo do livro corroborou na elaboração da Tabela 2.2 que apresenta informações sobre alguns NoSQL com relação aos parâmetros: Modelo de Dados (MD), Linguagem de Consulta (LC) e Formas de Acesso. Observe que "—" indica que não foram encontradas informações sobre o parâmetro. No item Modelo de Dados (Coluna MD) tem-se, K representando o Modelo Chave-valor, C representando o Modelo Orientado a Colunas, D representando o Modelo Orientado a Documento e G representando o Modelo Baseado em Grafo.

Analisando a Tabela 2.2 é possível concluir que nos NoSQL: a linguagem de consulta, quando existe, procura adotar sintaxe semelhante à linguagem SQL tradicional, e a funcionalidade de consulta em bancos de dados NoSQL não está padronizada como nos bancos de dados relacionais, uma vez que cada banco de dados NoSQL tem a sua própria linguagem.

Junto com estes resultados, outras verificações podem ser extraídas:

- As linguagens de consulta NoSQL tem dialeto com poucos comandos;
- A funcionalidade de consulta em bancos de dados NoSQL não está consolidada como nos bancos de dados relacionais;
- Não fornecem mecanismos de consulta com interfaces amigáveis;
- Não há um padrão de linguagem de consulta entre os modelos NoSQL;
- Há limitações no uso de funções de agrupamento ou agregação para cada linguagem de consulta, como o uso de índices e *joins* para as linguagens CQL e HQL.

Os usuários de banco de dados NoSQL precisam conhecer os modelos de dados para identificar os tipos de consulta suportados pelos banco de dados, pois cada banco de dados NoSQL difere nas características de consulta suportadas (Yan, 2015).

Tabela 2.2: Consulta em Bancos de Dados NoSQL

NoSQL	**MD**	**LC**	**Formas de Acesso**
Redis	K	—	Utiliza API.
TT/TC	K	—	Utiliza a sintaxe Lucene (McCandless et al., 2010)
Cassandra	C	CQL	Linguagem de consulta CQL baseada na sintaxe SQL, com limitações para uso de índices e *joins*.
HBase	C	—	Utiliza API, REST e JDBC/ODBC.
HyperTable	C	HQL	Linguagem de consulta HQL baseada na sintaxe SQL, com limitações no uso de funções de agregação e *joins*.
MongoDB	D	MQL	Linguagem de consulta MQL, não baseada na sintaxe SQL.
CouchDB	D	—	Consultas em *JavaScript* e API.
Riak	K,D	—	Utiliza a sintaxe Lucene.
Neo4J	G	Cypher	Linguagem de consulta *Cypher* baseada na sintaxe SQL, com recursos para uso de funções de agregação.
Allegrograph	G	SPARQL	Linguagem de consulta SPARQL baseada na sintaxe SQL.

Outra observação importante diz respeito a maturidade de bancos de dados baseado em grafos visto que a Teoria de Grafos tem uma base teórica estabelecida, o que permite o uso de recursos de pesquisa fornecidos pelo próprio modelo de grafos. Em relação ao banco de dados chave-valor, devido à simplicidade do modelo, o mecanismo de consulta é aparentemente simples, pois se baseia apenas na identificação da chave. Tauro et al. (2012) apontam que uma estrutura simples, tal como o modelo chave-valor, implica em um desempenho mais elevado para retorno de dados, inclusive quando comparado aos bancos tradicionais, e devido a isso, Hecht e Jablonski (2011) consideram que uma linguagem de consulta seria desnecessária para esse modelo de banco de dados.

Usualmente, durante o projeto do banco de dados NoSQL, considera-se antecipadamente quais consultas poderão ser realizadas pela aplicação-cliente. Em outras palavras, o projeto do NoSQL define primeiramente como as aplicações-cliente irão utilizá-lo, e se será necessário consultar ou retornar os dados da base de dados NoSQL. Os usuários de aplicações-cliente que usam bancos de dados NoSQL devem se preocupar em estabelecer no código da aplicação os quesitos para lidar com restrições de integridade, concorrência e operações complexas como *joins*, e com a questão da otimização de consultas. Diferentemente, nos bancos de dados relacionais, a preocupação com essas questões não é levada a nível de usuário.

O fato de cada NoSQL, inclusive entre os modelos NoSQL, oferecer sua própria forma de acesso aos dados ou linguagem de consulta gera uma heterogeneidade e complexidade de uso dos bancos de dados NoSQL.

2.6 Trabalhos Relacionados

A possibilidade de criação de uma linguagem de consulta amigável, e o uso de interfaces interativas é objeto de estudos acadêmicos (Burdakov et al., 2016; Silva et al., 2016; Yan, 2015). Enquanto muitos sistemas NoSQL não suportam nativamente a linguagem SQL, emergem propostas de como permitir realização de consulta sobre estes sistemas usando a sintaxe SQL.

Durante o desenvolvimento dessa dissertação, foi realizado um levantamento de trabalhos ciêntificos que especificam como definir uma linguagem de consulta padrão ou uma interface de consulta amigável para bases NoSQL. Trata-se de um levantamento em repositórios, com a finalidade de obter referências e contribuições ciêntificas sobre o tema realização de consultas em bancos de dados NoSQL. Para a busca sobre o tema investigado utilizou-se formatos diversos, tais como, livros, sites, estudos, revistas, períodicos, revisões de literatura, produções ciêntificas, entre outros. As fontes utilizadas foram a base de periódicos nacionais e internacionais da Capes [8], portal de periódicos da UnB [9], biblioteca da UnB [10] e o Google Acadêmico[11], entre outras fontes de pesquisa reconhecidas pela comunidade ciêntifica.

A seguir são listados alguns desses trabalhos:

- (Costa et al., 2016) e (Lehmayr et al., 2014) apresentam abordagens que propõe a criação de *middlewares* que suportem o mapeamento de comandos DML SQL para alguns SGBD NoSQL;
- Em (Schreiner, 2016) e (Calil e Mello, 2011) uma proposta de linguagem de consulta, denominada SimpleSQL, é apresentada para o modelo de dados orientado a documentos - SimpleDB;
- (Abadi et al., 2016) sugerem a adoção de linguagens declarativas para o processamento de *Big Data*, e o desenvolvimento de um *middleware* de componentes reusáveis que suportem múltiplas linguagens, e também sugere o uso de interfaces multimodais que combinem visualização, consulta e navegação quando o usuário deseja realizar uma *query*;

[8]Capes: URL http://www.periodicos.capes.gov.br/
[9]UnB Periódicos: URL http://periodicos.unb.br/
[10]Biblioteca UnB: URL http://www.bce.unb.br/
[11]GoogleAcadêmico: URL https://scholar.google.com.br/

- (Gomez et al., 2016) destacam que a ausência de uma linguagem declarativa de consulta nos NoSQL aumenta a responsabilidade dos administradores de banco de dados e usuários durante a definição e estruturação de objetos e dados nas base de dados, impactando em uma maior complexidade na execução dessas atividades;

- (Costa et al., 2016) apresentam SlamData, um sistema que permite a execução de instruções SQL no MongoDB;

- (Silva et al., 2016) comentam projetos que envolvem o reconhecimento de uma linguagem declarativa e padronizada a ser usada nos NoSQL e projetos que habilitam a incorporação de recursos da linguagem SQL, como Presto Dourish et al. (1999);

- Em (Burdakov et al., 2016) visualiza a existência de implementações NoSQL aplicáveis em diversas áreas, destacando-se os mecanismos de consulta e recuperação de dados, atividades consideradas complexas para os utilizadores em comparação com as bases de dados relacionais tradicionais;

- (Costa et al., 2016) comentam sobre propostas que caminham na direção de SGBD relacionais projetados para a nuvem, a citar o *Relational Cloud* Curino et al. (2011) e o *ConPaaS* (Pierre and Stratan, 2012);

- (Liu et al., 2016) comentam sobre *JSON DataGuide* e a representação OSON (*Oracle binary JSON encoding*) como propostas de integração entre os bancos de dados relacionais e NoSQL que utilizam dados textuais no formato JSON, permitindo aos usuários manipularem os dados estando no ambiente relacional ou não;

- Em (Costa et al., 2016), uma camada de mapeamento de instruções SQL DML para o banco de dados chave-valor Voldemort, denominada VoldemortSQL, é apresentada. A finalidade da camada é prover o acesso transparente a dados relacionais mantidos na nuvem por sistemas de informação baseados em banco de dados relacionais. Desta forma, esses sistemas não necessitam alterar suas interfaces de acesso relacionais, continuando a manipular dados na nuvem através do padrão SQL;

- (Silva et al., 2016) destacam que a linguagem SQL é considerada uma linguagem útil, robusta e eficaz também para os sistemas de banco de dados distribuídos e escaláveis que processam *Big Data*, particularmente os que precisam processar grandes volumes de dados de forma rápida e diversificada, tais como os do movimento NoSQL;

- Em (Yan, 2015), os SGBD NoSQL são considerados alternativas para o processamento e armazenamento de dados em massa, porém cada banco de dados NoSQL possue um mecanismo ou linguagem de consulta proprietário, e que linguagens de

consulta são úteis para o DBA que necessita realizar acesso a dados e estruturas utilizando mecanismos amigáveis e eficientes;

- (Atzeni et al., 2014) uma interface comum, denominada SOS (*Save Our Systems*) é proposta para os bancos de dados Redis, HBase (Hbase, 2011) e MongoDB (Banker, 2011). A ferramenta permite inserir e consultar dados usando um conjunto de operações simples;

- Em (Atzeni et al., 2014) relatam a ausência de uma linguagem de consulta para os NoSQL, e o fato dos sistemas não relacionais possuirem muitos modelos de dados, os quais não podem ser submetidos a um conjunto de regras fixas como nos bancos de dados relacionais, e conclui que as ferramentas podem aliviar as consequências da heterogeneidade oferecidas pelos NoSQL;

- Em (Lawrence, 2014), o projeto *Unity* é apresentado com uma contribuição de linguagem de consulta a ser usada por sistemas relacionais e não relacionais, permitindo que as consultas sejam traduzidas automaticamente para o banco de dados. O projeto propõe generalizar a interface de consulta tanto para os bancos de dados relacionais como NoSQL;

- Em (Lehmayr et al., 2014), um *middleware* que mapeia um subconjunto restrito de comandos DML SQL e os traduz para bancos de dados Cassandra (Cassandra, 2010) e MongoDB (Banker, 2011) é apresentado. O *middleware* fornece uma interface comum onde instruções SQL e conectores são enviados para o banco de dados NoSQL;

- (Luo et al., 2013) consideram uma abordagem para armazenar os dados não estruturados em bancos de dados relacional, e reconstruir as instruções SQL para terem capacidade de processar dados não estruturados;

- (Porkorny, 2013) destaca que nos bancos de dados relacionais, a linguagem SQL é considerada valiosa porque suporta portabilidade e expressividade, e é amplamente utilizado pelos usuários, e que nos bancos de dados NoSQL a linguagem SQL não é utilizada;

- (Kllapi et al., 2013) informam que linguagens para acesso aos dados, como o mecanismo *MapReduce* (Dean and Ghemawat, 2008) podem ser utilizadas em banco de dados NoSQL para realização de consultas, contudo, os usuários consideram essas linguagens e mecanismos dificéis de usar, e com funções complicadas de serem expressas;

- (Atzeni et al., 2012) detalham o projeto SOS (*Save Our Systems*), como uma interface comum para comandos SQL nos bancos de dados Redis, HBase (Hbase, 2011) e MongoDB (Banker, 2011);

- O projeto UnQL (*Unstructured Query Language*) (Buneman et al., 2000) propõe uma linguagem de consulta comum para bancos NoSQL. A ideia era criar uma linguagem para bases orientada a documentos, como o CouchDB e MongoDB. A sintaxe da linguagem de consulta é semelhante ao SQL e inclue comandos CRUD (*create, read, update, delete*). A linguagem não permite alterar o esquema do banco de dados;

- A ferramenta *Toad for Cloud Databases* (Bach and Werner, 2014) fornece uma interface gráfica com comandos baseados em SQL de forma a simplificar a migração de dados entre bases SQL e NoSQL. O projeto é suportado por *Apache HBase, Amazon SimpleDB, Azure Table, MongoDB, Apache Cassandra, Hadoop* e os bancos de dados que usam *Open Database Connectivity (ODBC)*;

- Impala (Kornacker et al., 2015) é um mecanismo de consulta de código aberto que permite consultas SQL nas tabelas de HBase, e foi construído de forma a estender componentes-chave tal como *Hive*, sintaxe SQL, metadados e esquema.

As propostas listadas contribuem para enfatizar a necessidade de uma linguagem de consulta, semelhante a SQL, no contexto de bancos de dados NoSQL e tratam do desenvolvimento de um mecanismo, tal como interfaces e linguagens de consulta para facilitar a consulta aos dados em bancos de dados NoSQL.

A Tabela 2.3 apresenta um agrupamento por aplicabilidade em SGBD NoSQL das abordagens, onde se destaca que a coluna Implementação indica o tipo de implementação utilizado, tal como linguagem de consulta, *middleware*, interface ou aplicação. Quanto as demais colunas, destaca-se que a coluna Abordagem apresenta o nome da abordagem de pesquisa; a coluna Aplicável em indica em que tipo de SGBD ou modelo de banco de dados a abordagem é aplicável; e a coluna Tipo ressalta os tipos de comandos que podem ser utilizados na abordagem, a citar os tipos DML, DDL, DCL ou DTL.

Apesar dos projetos supracitados estarem relacionados à problemática de realização de consultas em banco de dados NoSQL, não foram identificados projetos que tratem o mapeamento de comandos DDL, ou comandos de gerenciamento e administração para SGBD NoSQL.

Diferentemente das propostas apresentadas na literatura, o NoSQL^2 tem o propósito de auxiliar na administração de bancos de dados NoSQL, com o intuito de desenvolver uma solução, que proporcione recursos para administração de bases de dados NoSQL.

Tabela 2.3: Tabela Comparativa das Abordagens

Abordagem	**Aplicável em**	**Tipo**	**Implementação**
SimpleSQL	Orientado a Documentos	DML	Linguagem de Consulta: SimpleSQL
SlamData	MongoDB	DML	Aplicação
Relational Cloud	*Amazon* RDS e Microsoft SQL *Azure*		Aplicação DBaaS
Voldemort	Chave-Valor	DML	Linguagem de Consulta: VoldemortSQL
SOS	Redis, HBase, MongoDB	DML	Interface
Unity	MongoDB e SGBD Relacionais	DML	Interface e Linguagem de Consulta: *Unity*
Interface Lehmayr	Cassandra, MongoDB	DML	Interface e *Middleware*
UnQL	CouchDB, MongoDB	DML	Linguagem de Consulta: UnQL
Toad for Cloud Databases	HBase, SimpleDB, MongoDB, Cassandra, Hadoop	DML	Interface
Impala	HBase	DML	Aplicação
Presto	Redis, Cassandra, MongoDB e SGBD Relacionais	DML	Aplicação

Isso possibilitará que os DBA se desvinculem das especificidades dos diferentes NoSQL e utilizem uma sintaxe de comandos padronizada, e semelhante a linguagem SQL.

Como contribuição da solução apresentada pelo NoSQL^2 pode-se citar a transparência e facilidade proporcionada para a execução das tarefas de administração em SGBD. É possível estender o uso do NoSQL^2 com variados SGBD NoSQL, uma vez que as primitivas de configuração do *middleware* possibilitam a adição de conectores NoSQL a solução. A adição de novos comandos SQL pode ocorrer de forma simples e fácil via código do *middleware*.

2.7 A Pesquisa-Enquete

Organizações, tais como instituições públicas, financeiras, e empresas de telecomunicações, possuem em seus quadros especialistas em banco de dados para executar uma gestão segura e confiável dos seus dados, de forma a garantir a disponibilidade sempre que os mesmos forem requeridos. Com o constante crescimento do volume de dados, a adoção das práticas e procedimentos é primordial para a realização das tarefas de administração de banco de dados.

Novas tecnologias, como em nuvem e virtualização permitem compartilhar recursos, mas também aumentam a complexidade de administração dos dados. As novas tecnologias

estão tornando o mundo cada vez mais conectado gerando assim um rápido crescimento no volume de informações (Adriana e Holanda, 2016b). A administração e gestão desta heterogênea e complexa área exige uma compreensão do ambiente de trabalho e das práticas e procedimentos adequados para serem utilizados. Existem muitas complexidades relacionados à gestão e administração de dados, e Aiken et al. (2011) destacam que o gerenciamento de dados ainda está evoluindo. Abiteboul et al. (2016) afirmam que "as necessidades de banco de dados estão mudando, impulsionadas pela internet e pelo aumento do volume de dados."

De forma a obter dados estatísticos sobre a comunidade de DBA, na capital federal, Brasília, com respeito ao gerenciamento e formas de consulta em banco de dados NoSQL, durante o período do mestrado foram desenvolvidas 2 (duas) pesquisas-enquete. As enquetes contemplam diferentes perguntas endereçadas para o público-alvo formado por um conjunto de DBA da cidade de Brasília, que desempenham suas tarefas de administração em instituições públicas e privadas. O parque computacional dessas instituições é formado por um ambiente heterogêneo de SGBD relacionais e não-relacionais, com características de disponibilidade e gerenciamento de dados seguro e confiável.

O principal objetivo das pesquisas foi investigar como bases de dados são administradas e identificar quais práticas e procedimentos são utilizadas pelos DBA. Outros objetivos foram coletar informações demográficas sobre os DBA, e investigar se eles estão familiarizados com as novas tecnologias da área.

Em relação a metodologia adotada, o trabalho de Holt et al. (2015) foi referência para a escolha das perguntas, pois contempla entrevistas realizadas com uma comunidade de banco de dados diversificada de forma a identificar práticas e procedimentos utilizadas no ciclo de vida do banco de dados.

Os respondentes são 44 (quarenta e quatro) DBA, a época empregados, e desempenhando suas funções administrativas em instituições de médio e grande porte. Eles foram entrevistados em dois períodos: entre 25 de maio e 20 de junho de 2015; e 13 de abril e 18 de abril de 2016.

O questionário é composto por 20 (vinte) questões de diferentes tipos, incluindo dicotômica, escolha múltipla e escala *Likert* (Albaum, 1997). As perguntas abordam temas como práticas e técnicas utilizadas, e as perspectivas dos respondentes sobre seu ambiente de trabalho, volumetria, e familiaridade com novas tecnologias da área de banco de dados. O levantamento abrangeu os tipos de organizações, servidores de rede, quantitativos de bancos, administração, gerenciamento de mudanças, e novas tecnologias dos bancos de dados. De forma a ilustrar, a seguir apresenta-se 2 perguntas e respectivas respostas presentes na enquete.

A pergunta "Você está familiarizado com o conceito de Banco de Dados NoSQL?"mede

o conhecimento dos respondentes sobre o conceito Banco de Dados NoSQL, onde as respostas mensuram o nível de conhecimento na escala de 1 (nenhum conhecimento) até 5 (muito conhecimento). A Figura 2.8 apresenta graficamente as respostas dos DBA a essa pergunta nos anos de 2015 e 2016.

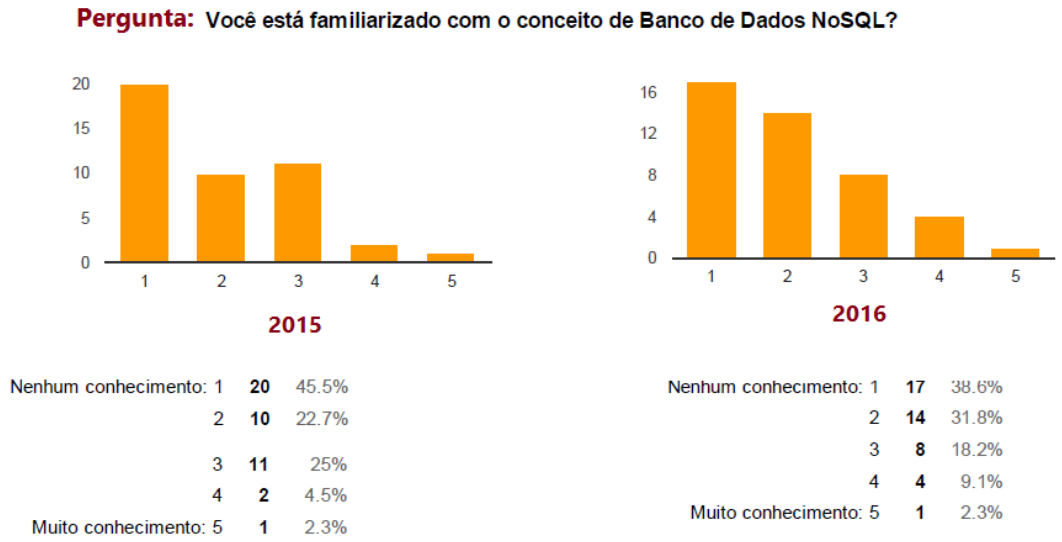


Figura 2.8: Conhecimento dos DBAs sobre o conceito Banco de Dados NoSQL.

Vale ressaltar um leve aumento de conhecimento sobre o assunto NoSQL, na comparação das respostas entre os anos de 2015 e 2016, o que evidenciar um maior interesse no tema pelos DBA.

"Como DBA você considera a existência de uma linguagem de consulta de alto nível (tal qual SQL padrão para os bancos de dados relacionais) fundamental para a administração de um banco de dados?"é uma pergunta dicotômica sobre a importância de uma linguagem de alto nível na administração de SGBD, tendo como opções as respostas sim ou não. A Figura 2.9 apresenta graficamente as respostas dos DBAs a essa pergunta nos anos de 2015 e 2016.

Destaca-se que a grande maioria dos DBA considera relevante a existência de uma linguagem de alto nível para a execução de suas tarefas de administração de banco de dados.

O conjunto completo de perguntas que compõe a enquete estão presentes no Apêndice A.

O instrumento de pesquisa utilizado foi *Google Forms*, e a enquete em formato digital está disponível em (PesquisaEnquete, 2015).

A divulgação da enquete ocorreu por meio das redes sociais como *e-mail, Twitter, Facebook* e *WhatsApp*. A decisão de divulgação foi tomada para coletar dados tão amplamente quanto possível na comunidade de banco de dados de Brasília.

Figura 2.9: Importância de uma linguagem de alto nível para os DBA.

Os Apêndices B e C ilustram graficamente o resultados percentuais da pesquisa, respectivamente entre os anos de 2015 e 2016. É importante destacar que em relação aos resultados, as 2 (duas) pesquisas-enquete desencadearam uma análise comparativa das respostas entre os anos 2015-2016, que resultou em 2 artigos científicos disponíveis em (Adriana e Holanda, 2016b) e (Adriana e Holanda, 2016a). De forma sucinta, os resultados destacam a pouca familiaridade dos DBA com novos conceitos em bancos de dados, tal qual NoSQL. É possível verificar também que as novas tecnologias em matéria de gestão de banco de dados são atualmente relevantes temas para a comunidade de banco de dados em Brasília, e destaca-se a importância da linguagem SQL, como forma de acesso aos bancos de dados.

As pesquisas-enquete, em sua versão completa, podem ser vistas em (Adriana e Holanda, 2016a) e (Adriana e Holanda, 2016b).

Capítulo 3

Arquitetura e Implementação

A presente dissertação propõe o NoSQL^2 para execução de comandos administrativos em banco de dados NoSQL utilizando uma sintaxe de comandos semelhante ao SQL, e que funciona para diferentes modelos NoSQL. O NoSQL^2 permite a execução de comandos básicos, usualmente necessários às tarefas de administração do SGBD pelo administrador de banco de dados, tais como criação e remoção de objetos e atribuição de privilégios.

Este capítulo está dividido em 4 seções: Seção 3.1 descreve os aspectos metodológicos utilizados para a definição arquiteturial; Seção 3.2 apresenta a arquitetura da solução, detalhando seus componentes e classes; Seção 3.3 descreve as características de extensibilidade da solução para diferentes bases de dados NoSQL; Seção 3.4 explícita os modelos NoSQL e comandos implementados no NoSQL^2.

3.1 Aspectos Metodológicos da Arquitetura NoSQL^2

Para a definição da solução arquiteturial, a metodologia é descrita como a estabelecida durante o projeto de Qualificação, onde se tem:

1. Primeira Fase: Estudo e Análise

   Entendimento do problema com o intuito de conhecer as tecnologias e soluções existente, com a definição dos requisitos necessários para a solução do problema.

2. Segunda Fase: Especificação

   Especificação da proposta do *middleware* que solucione o problema verificado na fase anterior, com os requisitos necessários para que ela atinja o seu objetivo.

3. Terceira Fase: Implementação

Definição das características técnicas para o desenvolvimento do *middleware*, ou seja, as ferramentas e os protocolos necessários para o seu desenvolvimento. O desenvolvimento de fato do *middleware* ocorre nesta fase.

4. Quarta Fase: Testes

   Testes de validação do *middleware* em diferentes bancos de dados NoSQL, para verificar se a proposta atende aos seus objetivos.

5. Quinta Fase: Avaliação e Correção

   Avaliação do *middleware* proposto, assim como as devidas correções e reavaliações da proposta para que os objetivos sejam cumpridos.

Essa metodologia visa estabelecer uma solução simples e objetiva para a resolução do problema definido. A solução arquiteturial do NoSQL^2 contempla:

- Definição de uma camada intermediária (*middleware*), de forma transparente, que receba comandos, preferencialmente comandos DDL SQL, e retorne o resultado da execução dos comandos;
- Configuração do contexto do SGBD NoSQL;
- Recursos funcionais para conversão dos comandos para a sintaxe NoSQL;
- O reuso e extensibilidade dos componentes da solução, com a possibilidade de incorporação de conexões com diversos SGBD NoSQL, e novos comandos na sintaxe SQL;
- A validação da arquitetura através de uma aplicação-cliente e SGBD NoSQL. A aplicação-cliente realiza uma troca de mensagens com o NoSQL^2, através do fornecimento de comandos na linguagem SQL, e recebimento dos resultados advindos do NoSQL^2;
- A escolha dos SGBD NoSQL pelo critério popularidade por modelo NoSQL;
- A escolha dos comandos DDL SQL implementados pelo critério comandos comumente utilizados.

A solução arquiteturial NoSQL^2, proposta nessa dissertação, é uma espécie de tradutor dirigido pela sintaxe, baseada nos estudos sobre mecanismos de tradução e compiladores, abordados, principalmente, no trabalho de Aho et al. (1986). Esse tradutor é uma combinação de um analisador sintático e de um gerador de código intermediário, que realiza a conversão de uma entrada para uma sáida traduzida.

3.2 Arquitetura NoSQL^2

O NoSQL^2 é um *middleware* para execução de comandos administrativos em banco de dados NoSQL, permitindo que os DBA se desvinculem das especificidades de cada NoSQL, podendo usar o padrão de comandos da linguagem SQL em diferentes bases de dados NoSQL.

A arquitetura do NoSQL^2 é uma camada intermediária entre uma aplicação-cliente e os bancos de dados NoSQL, e disponibiliza um conjunto de funções, tais como:

(i) Execução de instruções SQL (comandos);

(ii) Configuração do ambiente de execução da aplicação-cliente, possibilitando, por exemplo, a escolha do modelo e do banco de dados NoSQL;

(iii) Conexão com o banco de dados NoSQL; e

(iv) Retorno do resultado da execução dos comandos.

A Figura 3.1 apresenta a arquitetura geral do NoSQL^2 tendo o funcionamento dos componentes descrito conforme segue:

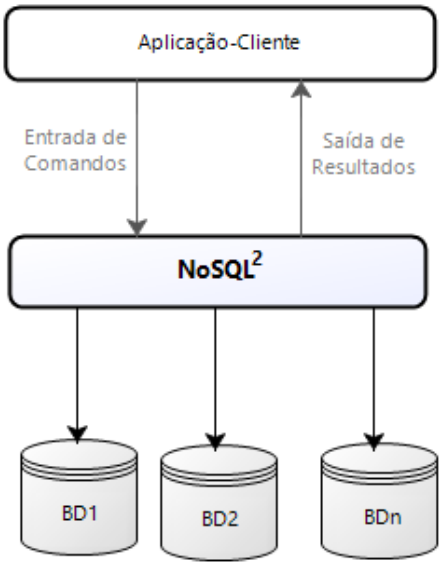


Figura 3.1: Arquitetura Geral NoSQL^2.

- `Aplicação-Cliente:` representa uma entidade externa, tal como uma interface gráfica ou linha de comandos, que troca mensagens com o NoSQL^2 para envio dos comandos via `Entrada de Comandos` e apresentação dos resultados ao usuário via `Saída de Resultados`. A aplicação-cliente deve possibilitar a entrada e apresentação do resultado dos comandos tornando transparente para o usuário a existência

da camada intermediária, e liberando o usuário de ter que conhecer detalhes sobre as fontes de dados NoSQL e interagir com cada uma delas individualmente;

- `{BD1, BD2 .... BDn}` representam o conjunto das bases de dados NoSQL disponíveis para o NoSQL^2. O NoSQL^2 possui característica de não intrusão nos bancos de dados NoSQL, ou seja, os componentes da solução encaminham ao banco de dados NoSQL comandos sintaticamente corretos de forma que o banco entenda e execute as atividades de processamento dos comandos e retorne o resultado para o componente responsável pela exibição dos resultados.

A arquitetura abstrata NoSQL^2 é constituída pelos componentes: `Interface`, `Varredura da Consulta`, `Tradução de Comandos`, `Conectores` e o conjunto `{NoSQL1, NoSQL2 .... NoSQLn}`. A Figura 3.2 ilustra a estrutura interna da arquitetura NoSQL^2 apresentando seus componentes e a forma de comunicação entre os mesmos:

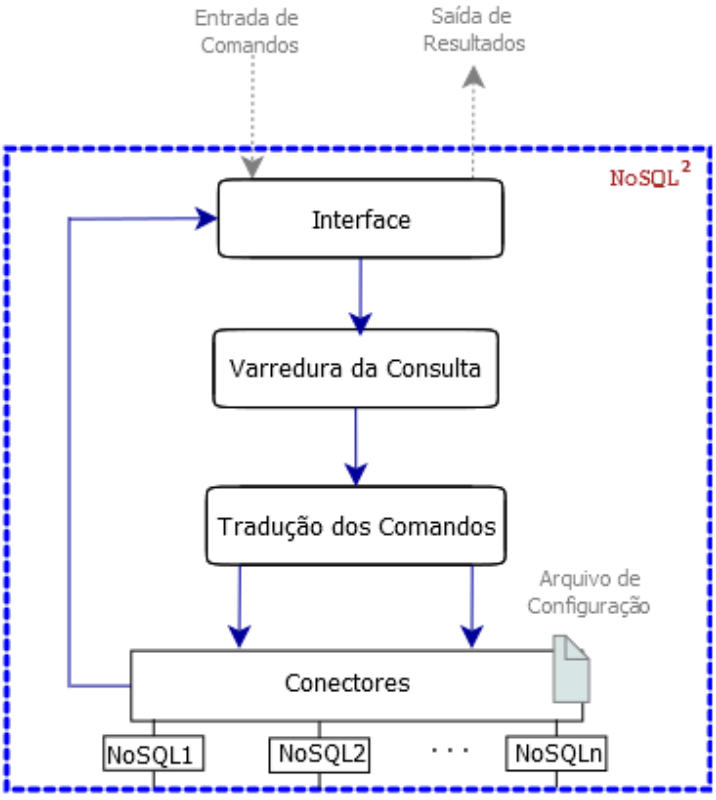


Figura 3.2: Arquitetura Abstrata NoSQL^2.

- `Interface`: responsável por receber os comandos via canal de comunicação `Entrada de Comandos` da `Aplicação-Cliente`, e devolver o resultado da execução dos comandos via canal de comunicação `Saída de Resultados` da `Aplicação-Cliente`.

- `Varredura da Consulta`: responsável por efetuar a análise sintática da consulta, verificando se a mesma está formulada de acordo com as regras de sintaxe da linguagem SQL, e identificando os *tokens* que compõe a consulta.

- `Tradução dos Comandos`: responsável pelo mapeamento DE-PARA do comando (instrução SQL) para a correspondente sintaxe NoSQL. Para o mapeamento das instrução SQL são propostos métodos DE-PARA que realizam a tradução do comando para a sintaxe própria banco de dados NoSQL, de forma transparente para o usuário, ou DBA, que não precisa conhecer as especificidades da linguagem ou forma de acesso aos dados do banco de dados NoSQL.

- `Conectores`: responsável por estabelecer a conexão com os conectores NoSQL, e também auxiliar o componente `Tradução dos Comandos` durante o mapeamento DE-PARA;

- `Arquivo de Configuração`: arquivo textual, integrado ao componente `Conectores`, que contém os parâmetros de conexão com os bancos de dados NoSQL, entre eles, nome da base de dados, servidor de banco de dados, porta de acesso, e modelo NoSQL. Esse componente é inicializado com valores *default* desses parâmetros, que são informados pelo usuário durante a edição do arquivo de configuração. O arquivo pode ser alterado por um editor textual;

- {NoSQL1, NoSQL2, ..., NoSQLn} : *java Drivers* necessários à conexão com as bases de dados NoSQL.

Elsmari e Navathe (2011) afirmam que uma consulta expressa em uma linguagem de consulta de alto nível, como SQL, precisa primeiro ser lida, analisada e validada. O NoSQL^2 segue essas diretrizes, tendo o seu funcionamento descrito conforme segue:

O NoSQL^2 recebe a instrução SQL (comando) via o componente `Interface`. A fase `Varredura da Consulta` é acionada para verificação e validação da instrução SQL. Caso a varredura apresente erro o processo é finalizado. Caso contrário, ou seja, a varredura ocorra com sucesso, a fase `Tradução de Comandos` de mapeamento da consulta para a sintaxe do banco de dados NoSQL é acionada. A execução da consulta (já traduzida) ocorre através dos componentes `Conectores` e `{NoSQL1, NoSQL2 .... NoSQLn}` configurado no componente `Arquivo de Configuração`. O componente `Interface` retorna o resultado da consulta para o componente externo `Aplicação-Cliente`.

Destaca-se que o componente `Tradução de Comandos` realiza um mapeamento, denominado DE-PARA, onde o comando (instrução SQL) é convertido para um comando traduzido para a correspondente sintaxe NoSQL associada ao banco de dados NoSQL determinado pelo componente `Arquivo de Configuração`. Um detalhamento das ações

executadas pelo componente `Tradução de Comandos` mostra que a realização do mapeamento DE-PARA ocorre em associação com os componentes `Conectores` e `Arquivo de Configuração`:

Ação 1 : Análise do comando: Identificação do tipo de instrução, podendo tratar-se de uma instrução do tipo DDL, DCL, DTL ou DML (Vide Página 8 e Tabela 2.1).
Componente envolvido: `Tradução de Comandos`;

Ação 2 : Identificação da base de dados NoSQL. Nesse ponto, ressalta-se a importância das informações contidas no `Arquivo de Configuração`, pois o NoSQL^2 necessita das informações sobre o modelo NoSQL e a base de dados NoSQL no qual o comando será executado, de forma a efetuar a conversão adequada do comando para a sintaxe correspondente à base de dados NoSQL escolhida.
Componentes envolvidos: `Conectores` e `Arquivo de Configuração`;

Ação 3 : Mapeamento DE-PARA: Conversão do comando para a sintaxe correspondente ao comando na base de dados NoSQL, gerando assim o comando traduzido.
Componentes envolvidos: `Tradução de Comandos` e `Conectores`.

O componente `Arquivo de Configuração` tem um papel importante na arquitetura NoSQL^2, pois detém informações sobre a configuração do contexto NoSQL, no qual o comando deverá ser executado. Portanto, o comando (instrução SQL) é encaminhado para execução na base de dados pertencente ao conjunto `{BD1, BD2 .... BDn}` que foi determinada previamente no componente `Arquivo de Configuração`.

A Figura 3.3 ilustra o funcionamento dos componentes do NoSQL^2, e para um melhor entendimento considera-se:

. C: comando em SQL;

. ARQ: arquivo de configuração do componente `Arquivo de Configuração`;

. NBD: Banco de dados NoSQL definido em ARQ;

. C': comando traduzido para a sintaxe da linguagem do NoSQL escolhido em NBD;

. R: resultado da execução do comando.

O processo de funcionamento do NoSQL^2 pode ser detalhado como:

A1 : Início da Execução;

A2 : `Interface` recebe o comando C;

A3 : `Varredura da Consulta` verifica as regras de sintaxe e identifica os *tokens* do comando C; Se as regras são válidas, inicia-se o processo de tradução do comando, caso contrário R é retornado para `Interface` informando o erro encontrado;

A4 : `Tradução dos Comandos` inicia a fase da tradução do comando para a correspondente sintaxe no NBD;

A5 : `Tradução dos Comandos` aciona o componente `Conectores` para identificar o conector correspondente ao banco de dados NoSQL;

A6 : `Conectores` identifica o conector referente ao NBD através da leitura de ARQ;

A7 : `Tradução dos Comandos` converte o comando C em C';

A8 : `Tradução dos Comandos` aciona o componente `Conectores` para encaminhamento do comando C' ao NBD;

A9 : `Conectores` dispara o comando C' para execução em NBD;

A10 : NBD executa o comando C' e retorna R para `Conectores`;

A11 : `Conectores` retorna R para `Interface`;

A12 : Fim da Execução.

3.2.1 Diagrama de Classes do NoSQL^2

A Figura 3.4 apresenta o diagrama de classes do NoSQL^2 especificado através da linguagem UML (*Unified Modeling Language*) (Rumbaugh et al., 2004), tendo as classes `Interface`, `Varredura`, `Tradutor` e `Conectores` como o conjunto de classes núcleo do NoSQL^2, ou seja, são necessárias para o funcionamento da arquitetura. Cada uma dessas classes é descrita a seguir:

O conjunto de Classes Núcleo é composto pelas classes:

- A classe `Interface` é responsável pela comunicação entre o *middleware* e a aplicação-cliente, recebendo o comando SQL, e retornando o resultado da execução da consulta;
- A classe `Varredura` verifica os *tokens* (símbolos) que compõe a consulta, e realiza uma análise sintática para verificar se o comando está escrito de acordo com as regras da linguagem. Se a consulta é validada, ela é encaminhada para a etapa

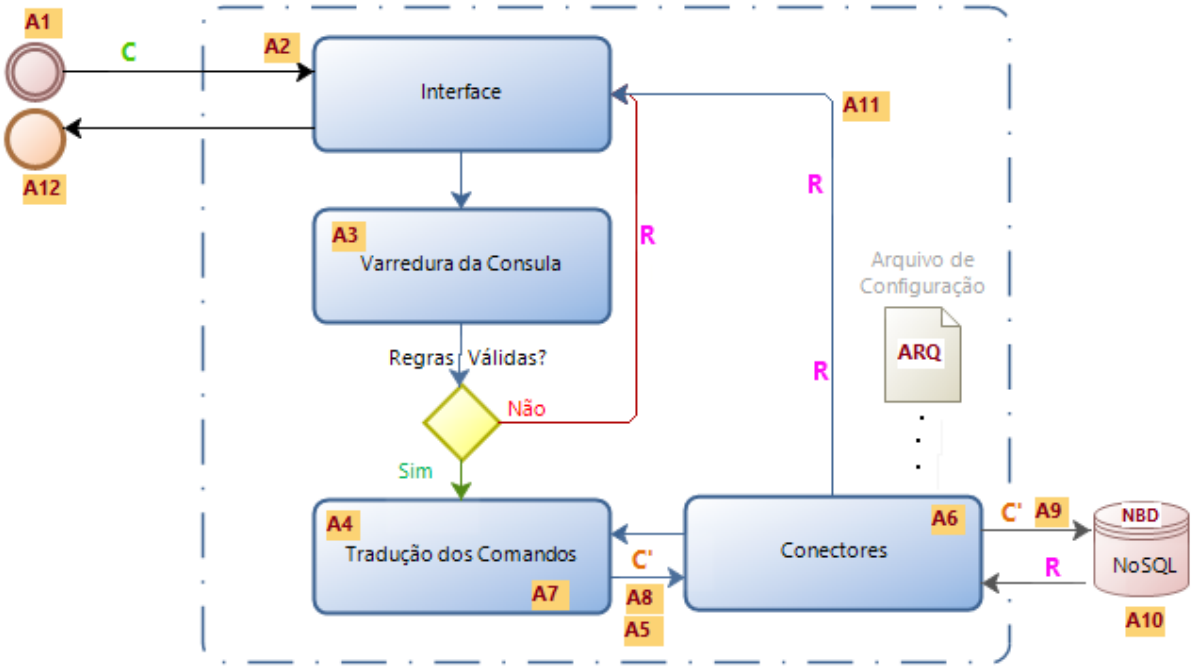


Figura 3.3: Funcionamento dos Componentes do NoSQL[2].

`Tradução dos Comandos`; Se a consulta não é validada, o processo é finalizado com uma mensagem informativa.

- A classe `Tradutor` identifica qual o tipo do comando (DDL, DML, DTL ou DCL), e em seguida, realiza o mapeamento DE-PARA do comando para o correspondente método interno da classe `ConectorNoSQL`. Ao final, o método correspondente do conjunto de subclasses conectores `{NoSQL1, NoSQL2 .... NoSQLn}` é executado;

- A classe `Conectores` obtém os dados sobre qual conector NoSQL utilizar pela leitura dos atributos e métodos públicos disponibilizados pela classe `ConexaoConfiguracao`;

A classe `ConexaoConfiguracao` é responsável pela configuração dos parâmetros de conexão necessários ao funcionamento da solução. Ela possue um conjunto de métodos, e atributos necessários para configuração do componente `Arquivo de Configuração`;

O conjunto de Classes de Conectores é composto pelas classes:

- A superclasse `ConectorNoSQL` é uma superclasse abstrata que possui as assinaturas dos métodos disponíveis no NoSQL[2];

- O conjunto de subclasses `{NoSQL1, NoSQL2 .... NoSQLn}` implementa os métodos para execução no banco de dados NoSQL, sendo responsável por estabelecer a conexão com o banco de dados NoSQL, e disparar o comando para o banco de dados NoSQL definido no `Arquivo de Configuração`.

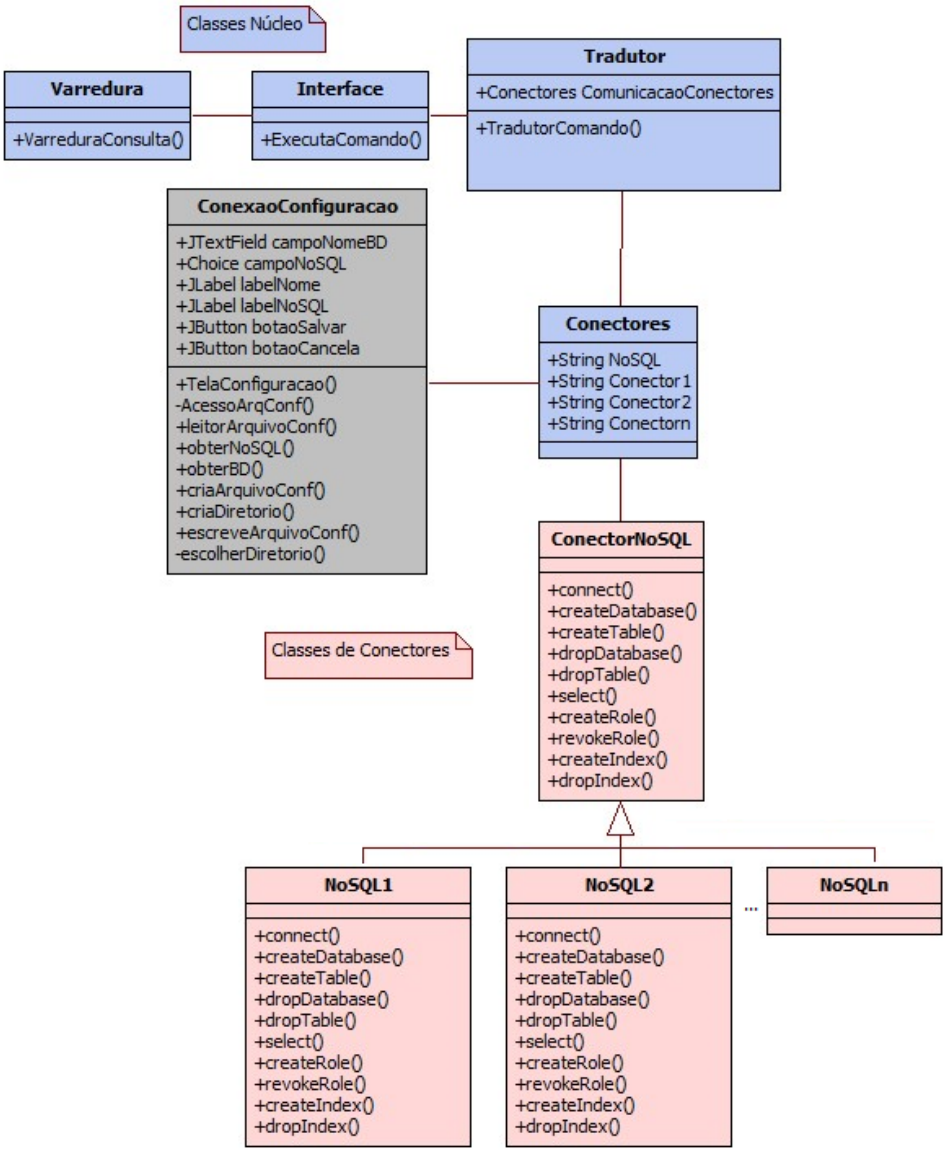


Figura 3.4: Diagrama de Classes.

3.3 Extensibilidade do NoSQL^2

A estrutura de classes do NoSQL^2 permite a incorporação/adição de diferentes conectores extensores NoSQL, e métodos, possibilitando a utilização de variados modelos de dados NoSQL, e novos comandos SQL, o que proporciona características de extensibilidade e reusabildade à solução. Por definição, uma classe abstrata é uma superclasse, que não possui instâncias, usada para representar entidades e conceitos abstratos e que funciona como modelo para a implementação genérica dos métodos das subclasses. Classes abstratas fornecem a assinatura dos seus métodos e as subclasses que os herdam, podem implementá-los de forma particular, de acordo com as necessidades da solução, permi-

tindo o reuso de código e também a extensibilidade sem alterar as especificações (Elmasri e Navathe, 2011; Silberschatz et al., 2010).

A superclasse `ConectorNoSQL` é extensível, ou seja, é possível adicionar um novo conector para implementar uma nova conexão a um SGBD NoSQL, desde que atendendo as definições de sobreposição dos métodos necessários para a nova subclasse. A subclasse adicionada deve sobrescrever os métodos gerais já definidos na superclasse `ConectorNoSQL` para que se tornem específicos para ela, usando os mecanismos de herança e polimorfismo. Assim, a superclasse `ConectorNoSQL` é uma superclasse abstrata que possui apenas assinaturas dos métodos. Cada uma das subclasses {NoSQL1, NoSQL2, ..., NoSQLn} completa os métodos da superclasse `ConectorNoSQL`, adicionando um comportamento específico a eles.

Conforme apresentado na Figura 3.4, os conectores são informados como atributos `{Conector1, Conector2, ...,ConectorN}` da classe `Conectores`, sendo que a adição de um novo conector implica na adição de um novo atributo `{ConectorN}` nesta classe.

De forma a ilustrar, considere-se que um novo conector deva ser incorporado a arquitetura NoSQL^2. Dado o novo conector NovoConector, tem-se que :

Definição 1. *Um java Driver, denominado JD, é um driver necessário ao estabelecimento da conexão do conector com o sistema gerenciador de banco de dados NoSQL.*

Definição 2. *Um CN é um conjunto de conectores NoSQL, onde CN = {NoSQL1, NoSQL2, ..., NoSQLn}, tal que NoSQLn é uma subclasse de* `ConectorNoSQL`.

Definição 3. NovoConector *pode ser adicionado a arquitetura NoSQL^2, se e somente se, existe um JD correspondente. Se* NovoConector *é adicionado a arquitetura NoSQL^2, então* NovoConector $\in CN$.

A adição do conector NovoConector está condicionada a existência do *java Driver* correspondente. Se o *java Driver* existe, procede-se a adição da subclasse NovoConector ao conjunto de subclasses de `ConectorNoSQL`. Com a adição do novo conector, a nova subclasse NovoConector deve implementar os métodos gerais definidos por `ConectorNoSQL`. Também, os atributos da classe `Conectores` devem ser atualizados com a informação de conexão ao novo conector NovoConector.

A arquitetura NoSQL^2 permite a adição de novos comandos SQL, através da definição do novo comando na classe `Tradutor`, a assinatura do método relativo ao comando na classe `ConectorNoSQL` e a implementação do método na subclasse NoSQLn. De forma a possibilitar que o novo comando seja usado nos conectores já existentes no NoSQL^2, é necessário implementar o método associado ao novo comando nas subclasses {NoSQL1, NoSQL2, ..., NoSQLn-1}.

De forma a ilustrar, considere-se que um novo comando deva ser incorporado a arquitetura NoSQL^2. Dado que C representa o comando CREATE USER <USUARIO>, tem-se que :

Definição 4. *Um conjunto de comandos implementados, denominado CCI, é um conjunto de comandos, onde CCI = {C1, C2, ..., Cn}, tal que um comando Cn possui a assinatura do método associado na classe* `ConectorNoSQL` *e a implementação do método na classe* NoSQLn.

Definição 5. *Um conjunto de métodos válidos, denominado CMV, é um conjunto de métodos, onde CMV = {M1, M2, ..., Mn}, tal que, Mn é definido na classe* `Tradutor`.

Definição 6. C *é um comando válido na arquitetura NoSQL^2, se e somente se,* C *tem um correspondente método* M*, e* $C \in CCI$, $M \in CMV$.

A adição do novo comando C na arquitetura de classes NoSQL^2 está condicionada a definição do comando C na classe `Tradutor`, a assinatura do método M correspondente ao comando C na classe `ConectorNoSQL` e a implementação do método na subclasse NoSQLn. As subclasses {NoSQL1, NoSQL2, ..., NoSQLn-1} devem implementar internamente o método M para possibilitar a execução na correspondente base de dados NoSQL.

3.4 Implementação do NoSQL^2

A linguagem de implementação tem um papel importante no desenvolvimento da arquitetura proposta, visto que a solução pretende ser de fácil utilização e com requisitos de extensibilidade de seus componentes. Com relação as tecnologias usadas, o NoSQL^2 foi desenvolvido utilizando a linguagem *java* via ferramenta *Eclipse java EE IDE for Web Developers Version Luna Service Release 2.* A escolha da linguagem foi realizada levando em consideração as características da mesma, tais como ser concisa e simples, favorecer a extensibilidade, sendo robusta e portável.

3.4.1 Os Modelos NoSQL Implementados

(Ringlstetter e Corbellini, 2016) e *DB-Engines Ranking*[2] destacam os índices de popularidade crescente dos bancos de dados NoSQL, sendo que especificamente MongoDB encontra-se na 4ª posição, Cassandra na 7ª posição, Redis na 10ª posição, e o Neo4J na 21ª posição.

Esses índices de popularidade foram um incentivo para o estudo de caso utilizando os 4 principais modelos NoSQL apresentados na Seção 2.4, e a implementação do conector

de um sistema gerenciador de banco de dados NoSQL: MongoDB do modelo Orientado a Documentos, Cassandra do modelo Orientado a Colunas, Redis do modelo Chave-Valor, e Neo4J do modelo Baseado em Grafos:

- Cassandra tem uma linguagem de consulta chamada CQL (*Cassandra Query Language*) com sintaxe semelhante ao SQL e disponibiliza uma console de comandos denominada *cqlsh* que permite a inserção de comandos básicos, tais como para criação de tabelas, inserção e consulta de dados. Também é possível utilizar uma ferramenta gráfica disponível no *DataStax DevCenter*[12]. O conjunto de comandos CQL incluem desde comandos do tipo DDL como create table e create index, comandos DML como select e update, além de comandos DCL como grant e revoke;

- Redis é um SGBD representante do modelo chave-valor, onde os dados são acessados através de uma chave. Redis não possui uma linguagem de consulta, e a inserção e consulta aos dados ocorre através da utilização de comandos e operadores nativos do SGBD (Gonzalez et al., 2016; Corbellini et al., 2016; Atzeni et al., 2014);

- MongoDB possui uma linguagem de consulta baseada em documentos denominada MQL (*MongoDB query language*) que permite funções, tais como nos bancos de dados relacionais e disponibiliza uma console de comandos para manipulação de objetos e dados. A sintaxe de comandos da linguagem MQL é proprietária, ou seja, funciona unicamente em seu banco de dados e possui uma sintaxe própria, não semelhante a SQL (Gomez et al., 2016);

- Neo4J é um representante do modelo orientado a grafos, onde os dados podem ser acessados via API *java*, ou usando as linguagens para grafos como SPARQL ou Gremlin (Corbellini et al., 2016).

3.4.2 Os Comandos Implementados

A linguagem SQL possui comandos para realização de tarefas de administração de banco de dados, a citar: o comando CREATE para criação de tabelas e bases de dados; o comando ALTER para alteração da estrutura das tabelas e bases de dados; e o comando DROP para remoção de tabelas e bases de dados (Mullins, 2012).

A escolha dos comandos implementados no NoSQL[2] justifica-se no fato que comandos de definição de dados são comumente utilizados pelos DBA durante a realização de tarefas de gerenciamento e administração das bases de dados. Dessa forma, a implementação utiliza um conjunto de comandos para criação e remoção de objetos, além de comandos

[12]DataStax: URL https://docs.datastax.com/en/cql/3.1/cql/cql_intro_c.html

para seleção de dados de tabelas, de forma a demontrar que o NoSQL^2 também funciona com comandos DML.

As Tabelas 3.1, 3.2 , 3.3, 3.4 e 3.5 apresentam comandos em SQL disponíveis no NoSQL^2 e a sintaxe de mapeamento desses comandos nos bancos de dados Cassandra, MongoDB, Redis e Neo4J. A lista de comandos é composta por: Create Database, Drop Database, Create Table, Drop Table, Select, Create Role, Revoke Role, Create Index e Drop Index.

Tabela 3.1: Comandos Create Database e Drop Database no NoSQL^2.

NoSQL	**Create Database**	**Drop Database**
Cassandra	create keyspace	drop keyspace
MongoDB	getDB + getCollection	dropDatabase
Redis	select	-
Neo4J	create	match + delete

Tabela 3.2: Comandos Create Table e Drop Table no NoSQL^2.

NoSQL	**Create Table**	**Drop Table**
Cassandra	create table	drop Keyspace
MongoDB	getCollection	drop
Redis	set	del
Neo4J	create	match + delete

Tabela 3.3: Comando Select no NoSQL^2.

NoSQL	**Select**
Cassandra	select
MongoDB	table.find
Redis	hget
Neo4J	match

Tabela 3.4: Comandos Create Role e Revoke Role no NoSQL^2.

NoSQL	**Create Role**	**Revoke Role**
Cassandra	create role	revoke
MongoDB	createRole	revokeRolesFromUser
Redis	-	-
Neo4J	-	-

Tabela 3.5: Comandos Create Index e Drop Index no NoSQL^2.

NoSQL	Create Index	Drop Index
Cassandra	create custom index	drop index
MongoDB	db.collection.createIndex	db.collection.dropIndex
Redis	-	-
Neo4J	create index on	drop index on

É importante destacar, que o mapeamento das instruções SQL é dependente da aplicabilidade dessas instruções no sistema gerenciador de banco de dados NoSQL, visto que existem particularidades no conjunto de comandos nativos dos NoSQL, onde o mapeamento da instrução SQL para sua corresponde sintaxe nativa NoSQL não é possível. A citar, o comando Create Role não é suportado no Neo4J, e os comandos Create Role, Revoke Role, Create Index e Drop Index não tem implementação no Redis. Assim, os comandos Create Role, Revoke Role, Create Index e Drop Index para a base de dados Redis não foram passíveis de implementação no NoSQL^2, o mesmo ocorrendo com os comandos Create Role e Revoke Role para o Neo4J. Outra particularidade do Redis diz respeito a criação de bases de dados, pois por padrão esse sistema gerenciador de banco de dados habilita 16 bancos que podem ser usados ou escalonados separadamente, portanto a criação de uma nova base de dados implica na seleção de um dos bancos disponíveis no Redis. O MongoDB tem como uma de suas características que para a execução efetiva do comando de criação de tabelas (create table) faz-se necessário a inserção de no mínimo 1 registro na tabela sendo criada.

Capítulo 4

Experimentos e Resultados

Nesta dissertação, foi proposto o *middleware*, denominado NoSQL^2, com o objetivo de executar um conjunto de comandos administrativos usando a sintaxe SQL em bancos de dados NoSQL.

Este capítulo apresenta a análise dos resultados da avaliação do NoSQL^2, sendo composto pela Seção 4.1 que explícita as tecnologias utilizadas para a construção do NoSQL^2, o protótipo e os experimentos implementados para validação das funcionalidades; e a Seção 4.2 que apresenta os resultados obtidos através da realização de provas de conceito.

4.1 O Experimento

No escopo dessa dissertação, as provas de conceito foram realizadas utilizando a implementação descrita a seguir, para validação das funcionalidades, extensibilidade e execução do NoSQL^2.

O experimento apresentado na Figura 4.1 foi implementado para avaliar as funcionalidades do *middleware* NoSQL^2, e é composto por um protótipo da Aplicação-Cliente, do NoSQL^2 e de bases de dados NoSQL, tendo como pré-requisito o `Arquivo de Configuração`, que contém as informações sobre o NoSQL a ser utilizado.

Protótipo `Aplicação-Cliente`

O protótipo `Aplicação-Cliente` é uma aplicação *java* com uma interface gráfica, que permite ao usuário digitar o comando (instrução SQL), ou acionar através de botões e um conjunto de *menus* disponíveis, o comando desejado (Figura 4.2). A `Aplicação-Cliente` comunica-se com o NoSQL^2 através de troca mensagens para envio dos comandos via `Entrada de Comandos` e apresentação dos resultados ao usuário via `Saída de Resultados`.

A interface gráfica é constituída por um conjunto de *menus*: `Comandos`, `Opções` e `Sair`; uma janela `Entrada de Comandos`; botões de comandos, entre eles `Executar` e `Limpa`

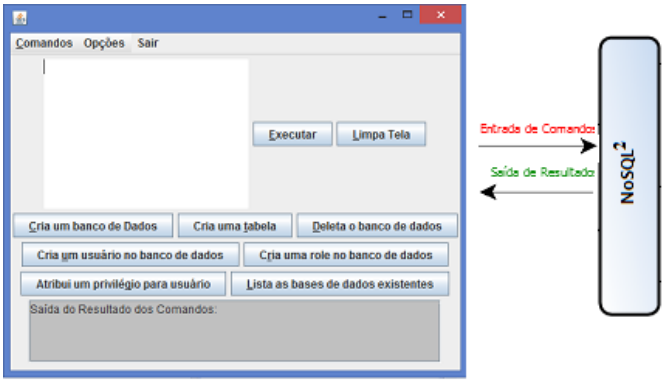


Figura 4.1: Protótipo e o NoSQL^2.

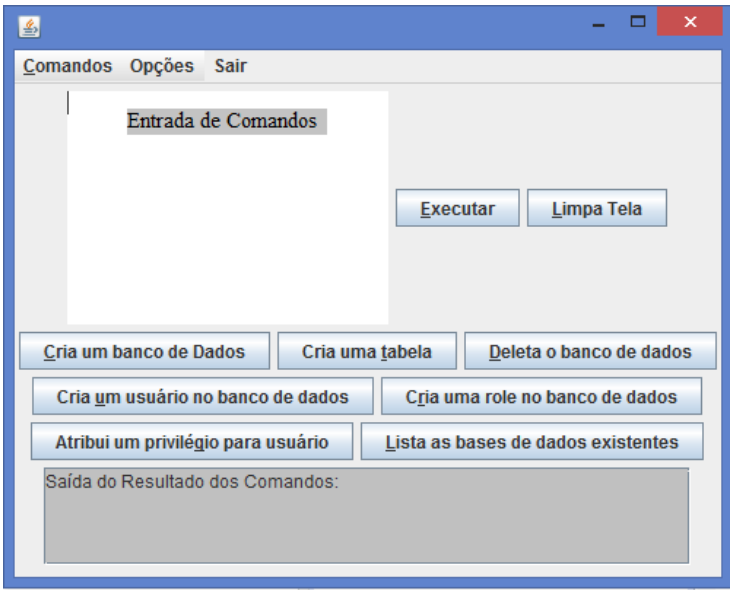


Figura 4.2: Protótipo Aplicação-Cliente.

`Tela`; e uma janela `Saída do Resultado dos Comandos`. Os parâmetros do arquivo de configuração podem ser configurados via o menu `Opções` da interface gráfica, onde devem ser informado os dados da conexão, tais como o modelo e o nome da base NoSQL.

A instrução SQL pode ser inserida na janela Entrada de Comandos, via o menu Comandos ou ainda através dos botões disponíveis na tela gráfica. A janela `Saída do Resultado dos Comandos` exibe o *status* e o resultado da execução do comando, o menu `Sair` permite que o usuário finalize a aplicação, e o botão `Limpa Tela` limpa todos os dados da janela `Entrada de Comandos`.

A `Aplicação-Cliente` foi desenvolvida para permitir a entrada e apresentação do resultado dos comandos, tornando transparente para o usuário a existência da camada intermediária NoSQL^2.

Ambiente Computacional dos Testes

O ambiente computacional dos testes é composto por um conjunto de equipamentos e os sistemas gerenciadores de banco de dados NoSQL necessários à configuração do ambiente e ao armazenamento de dados.

Os testes foram executados em um ambiente computacional heterogêneo, ou seja, utilizando tecnologias diferentes, tais como máquina física e virtual, sistema operacional *Windows* e Linux, acesso remoto usando ferramentas como o VPN *Fortinet 5.2.3*[13] e *TeamViewer 11.0*[14], os sistemas gerenciadores de banco de dados NoSQL: Cassandra, Redis, Neo4J, MongoDB e CouchDB e com o SGBD Relacional SQLServer, de forma a comprovar a funcionalidade da solução também em ambientes relacionais.

A configuração dos equipamentos utilizados no ambientes de testes é descrita conforme segue:

- 1 Notebook Dell Inspiron 14 com processador Intel(R) Core(TM) i5-5200U CPU 2.20GHz memória 8 GB Sistema Operacional 64bits Windows 8.1 *Single Language*;
- 1 Notebook HP Mini 210-1000 com processador Intel(R) Atom(TM) CPU 1.66GHz memória 2 GB Sistema Operacional 32bits Windows 7 Stater ;
- 2 Máquinas virtuais com 1 processador memória de 2GB *HardDisk* (SCSI) 20GB Sistema Operacional *RedHat Enterprise Linux* 6 64-bits;
- 1 Instância do Banco de Dados MongoDB;
- 1 Instância do Banco de Dados Cassandra;
- 1 Instância do Banco de Dados Neo4J;
- 1 Instância do Banco de Dados Redis;
- 1 Instância do Banco de Dados CouchDB;
- 1 Instância do Banco de Dados SQLServer.

A Figura 4.3 ilustra o ambiente computacional dos testes utilizado para simular o funcionamento do protótipo, sendo que nesse experimento, para avaliação da proposta, utilizou-se os bancos de dados NoSQL: Cassandra versão 2.2 do modelo Orientado a Coluna, MongoDB versão 3.0 e CouchDB Versão 2.0 do modelo Orientado a Documentos, Redis versão 2.8 do modelo Chave-valor e Neo4J versão 1.9 do modelo Baseado em Grafos, além do SGBD relacional SQLServer Versão 2016.

[13]Fortinet: http://www.fortinet.com
[14]TeamViewer: http://www.teamviewer.com/

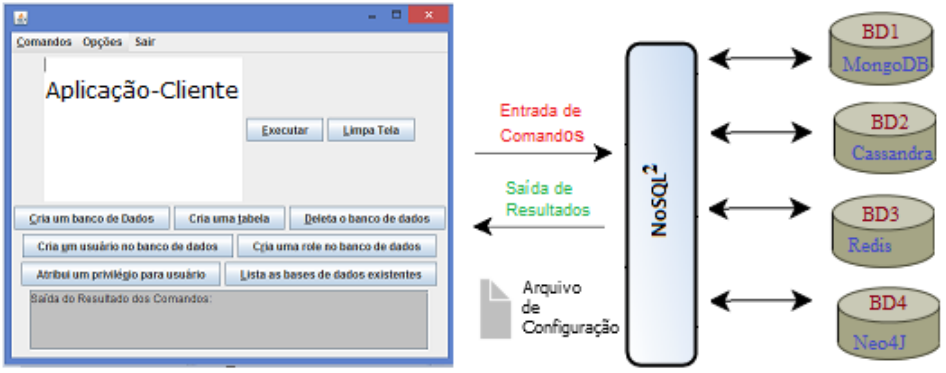


Figura 4.3: Ambiente de Testes.

Em relação as instâncias de banco de dados NoSQL utilizados nos testes, é importante destacar que:

- `BD1`: Servidor físico que contém a configuração do sistema gerenciador de banco de dados MongoDB, e porta de acesso *default*: 27017;
- `BD2`: Servidor virtual que contém a configuração do sistema gerenciador de banco de dados Cassandra, e porta de acesso *default*: 9042;
- `BD3`: Servidor físico que contém a configuração do sistema gerenciador de banco de dados Neo4J, e porta de acesso *default*: 6379;
- `BD4`: Servidor virtual que contém a configuração do sistema gerenciador de banco de dados Redis, e porta de acesso *default*: 7474.

4.2 Resultados Obtidos

Os resultados obtidos são descritos a seguir, e vale observar que visam:

(i) Validar as funcionalidades do NoSQL^2, através de realização de testes de funcionalidade dos comandos implementados na solução;

(ii) Verificar a extensibilidade dos componentes da solução NoSQL^2;

(iii) Comparar a solução NoSQL^2 com outros SGBD, e a aplicabilidade em abordagens relacionais.

4.2.1 Teste das funcionalidades do NoSQL^2

Os testes de funcionalidade executados nessa etapa visam verificar a corretude e integração dos componentes do NoSQL^2 através da validação do código, de forma a garantir que o usuário consiga executar os comandos corretamente.

Os estudos de casos foram realizados através da execução do experimento apresentado na Seção 3.4, composto por um protótipo da Aplicação-Cliente e do NoSQL^2, e contemplam as instruções SQL DDL: create database, drop database, create table, drop table, a instrução SQL DML: select *, e os SGBD MongoDB, Cassandra, Redis, Neo4J, CouchDB e SQLServer. A análise dos resultados é descrita pela apresentação dos *logs* de saída dos resultados obtidos quando da execução de comandos SQL no experimento, e para um melhor entendimento considera-se que:

C1: `CREATE DATABASE mestrado`

C2: `CREATE TABLE monografia (titulo VARCHAR(50), faculdade VARCHAR(50))`

C3: `DROP DATABASE mestrado`

C4: `SELECT * FROM monografia`

A Saída de Resultado 4.1 mostra o *log* com o resultado da execução do comando C1 usando o experimento e o SGBD MongoDB, onde as linhas de resultados são descritas conforme segue:

Saída de Resultado 4.1: Execução do comando C1

```
1 Comando (Instrução SQL): CREATE DATABASE <mestrado>
2 Processo de Varredura da Consulta!
3 Processo de Tradução do Comando!
4 Processo: Comando Traduzido NoSQL: this.conectar().getDB(mestrado);
5 Processo: Conectores!
6 Processo: MONGODB --> CREATE DATABASE --> OK!
7 Tempo Execução: 629 milissegundos
```

. Linha 1: Comando (Instrução SQL) a ser executado;

. Linhas 2: Processo de Varredura da consulta C1;

. Linha 3: Processo de Tradução do comando C1;

. Linha 4: Comando traduzido para a sintaxe do MongoDB;

. Linha 5: Processo Conectores para conexão com o MongoDB;

. Linhas 6: Execução do comando C1 traduzido no MongoDB;

. Linha 7: Tempo de execução.

A Figura 4.4 exibe a listagem dos bancos de dados no servidor físico `BD1`, e como pode ser observado entre os bancos de dados do MongoDB (`show dbs`) encontra-se o banco de dados `mestrado`.

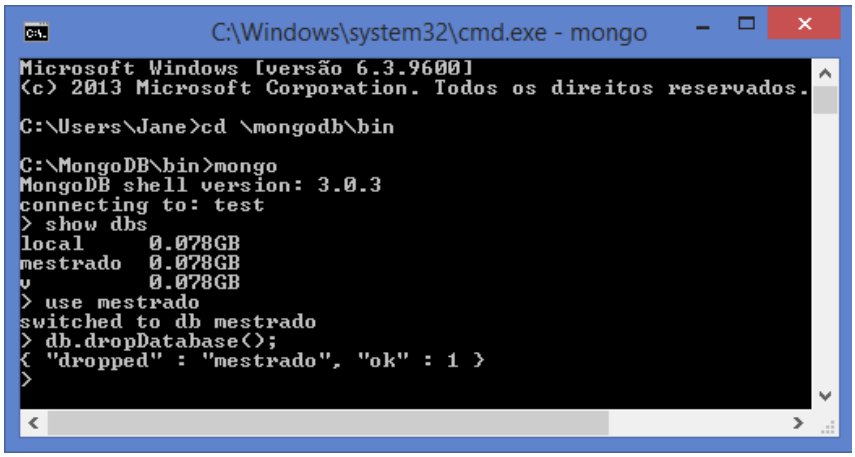


Figura 4.4: Listagem dos bancos de dados no servidor físico `BD1`.

A verificaçao é efetuada através do acesso via console windows (cmd.exe), onde o comando `mongo` efetua a conexão com a base de dados *default*: `test`; `show dbs` exibe a listagem das bases de dados disponíveis; `use mestrado` efetua a conexão com a base de dados `mestrado`; e o comando `db.dropDatabase()` elimina a base de dados do SGBD MongoDB.

A Saída de Resultado 4.2 mostra o *log* com o resultado da execução do comando C2 usando o experimento e o SGBD Cassandra, onde as linhas de resultados são descritas conforme segue:

Saída de Resultado 4.2: Execução do comando C2

```
1 Comando (Instrução SQL): CREATE TABLE monografia (titulo VARCHAR(50), faculdade VARCHAR
      (50))
2 Processo de Varredura da Consulta!
3 Processo de Tradução do Comando!
4 Processo: Comando Traduzido NoSQL: CREATE TABLE monografia titulo text, faculdade text;
5 Processo: Conectores!
6 Processo: Cassandra --> CREATE TABLE--> OK!
7 Tempo Execução: 1785 milissegundos
```

. Linha 1: Comando (Instrução SQL) a ser executado;

. Linhas 2: Processo de Varredura da consulta C2;

. Linha 3: Processo de Tradução do comando C2;

. Linha 4: Comando traduzido para a sintaxe do Cassandra;

. Linhas 5: Processo Conectores para conexão com o Cassandra;

. Linhas 6: Execução do comando C2 traduzido no Cassandra;

. Linha 7: Tempo de execução.

A Figura 4.5 exibe a listagem dos bancos de dados no servidor virtual BD2, e como pode ser observado, a estrutura da tabela `monografia` com os campos `titulo` e `faculdade` criada pelo comando C2 está presente na instância `mestrado`.

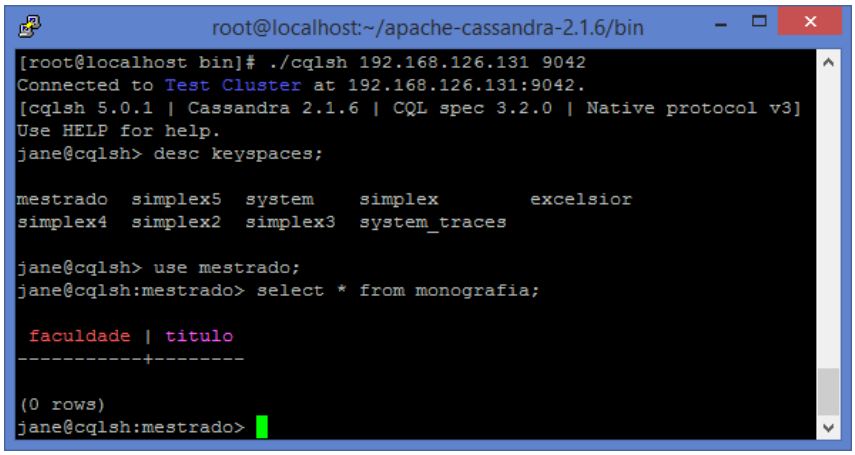


Figura 4.5: Listagem dos bancos de dados no servidor físico BD2.

A verificação é efetuada através do acesso via console *linux* através da ferramenta putty.exe, onde o comando `./cqlsh 192.168.126.131 9042` efetua a conexão com a base de dados *default*: `excelsior`; `desc kesypaces` exibe a listagem das bases de dados disponíveis; `use mestrado` efetua a conexão com a base de dados `mestrado`; e o comando `select * from monografia` exibe a estrutura e os dados da tabela `monografia`.

A Saída de Resultado 4.3 mostra o *log* com o resultado da execução do comando C3 usando o experimento e o SGBD Neo4J, onde as linhas de resultados são descritas conforme segue:

Saída de Resultado 4.3: Execução do comando C3

```
1 Comando (Instrução SQL): DROP DATABASE <teste>
2 Processo de Varredura da Consulta!
3 Processo de Tradução do Comando!
4 Processo: Comando Traduzido NoSQL: MATCH (e:teste) DELETE e
5 Processo: Conectores!
6 Processo: Neo4J: DROP DATABASE --> OK!
7 Tempo Execução: 1272 milissegundos
```

. Linha 1: Comando (Instrução SQL) a ser executado;

. Linhas 2: Processo de Varredura da consulta C3;

. Linha 3: Processo de Tradução do comando C3;

. Linha 4: Comando traduzido para a sintaxe do Neo4J;

. Linhas 5: Processo Conectores para conexão com o Neo4J;

. Linhas 6: Execução do comando C3 traduzido no Neo4J;

. Linha 7: Tempo de execução.

A Figura 4.6 ilustra, via interface gráfica, informações sobre os nós existentes no SGBD Neo4J, e como pode ser observado na Parte 1 da figura, o conjunto de nós é composto por {Employee, mestrado, Pessoa}, e na Parte 2 da figura, o conjunto de nós é formado apenas por {Employee, Pessoa}.

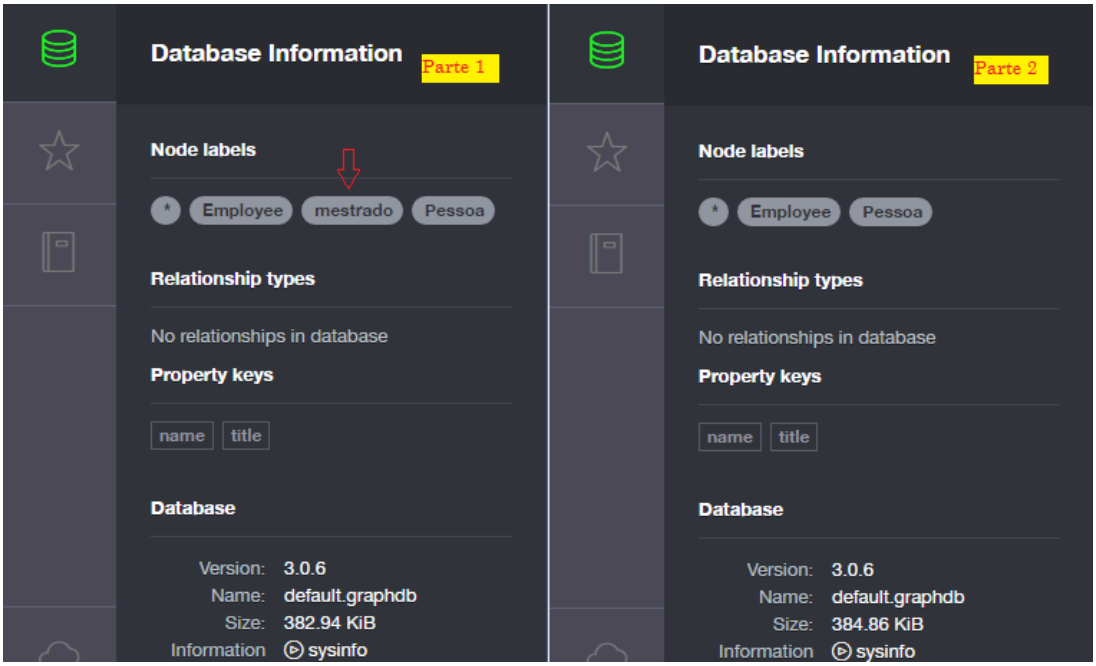


Figura 4.6: Listagem dos bancos de dados no servidor físico BD3.

A verificação é efetuada via interface gráfica do Neo4J no servidor virtual BD3:7474/browser/, onde a porta de acesso é 7474. A Parte 1 da Figura 4.6 elenca um conjunto de nós composto por {Employee, mestrado, Pessoa}, e após a execução do comando C3, conforme pode ser visto na Parte 2 da figura, o conjunto de nós é formado por {Employee, Pessoa}, pois o nó mestrado foi removido do conjunto.

O NoSQL[2] implementa o comando DML: Select, que é comando comumente utilizado pelos usuários e DBA durante o acesso aos dados e ao dicionário de dados dos SGBD. A Saída de Resultado 4.4 mostra o resultado da execução do comando C4 no SGBD Redis, onde as linhas de resultados são descritas conforme segue:

Saída de Resultado 4.4: Execução do comando C4

```
1 Comando (Instrução SQL): select * from monografia
2 Processo de Varredura da Consulta!
3 Processo de Tradução do Comando!
4 Processo: Comando Traduzido NoSQL: jedis.hgetAll(monografia)
5 Iniciando Processo: Conectores!
6 Processo: Redis -> SELECT * -> OK!
7 Tempo Execução: 106 milissegundos
```

. Linha 1: Comando (Instrução SQL) a ser executado;

. Linhas 2: Processo de Varredura da consulta C4;

. Linha 3: Processo de Tradução do comando C4;

. Linha 4: Comando traduzido para a sintaxe do Redis;

. Linhas 5: Processo Conectores para conexão com o Redis;

. Linhas 17: Execução do comando C3 traduzido no Redis;

. Linha 18: Tempo de execução.

Como pode ser observado na Figura 4.7, a consulta a tabela `monografia` apresenta como resultado os valores `nosql2` e `UnB` para as colunas `titulo` e `faculdade`, respectivamente.

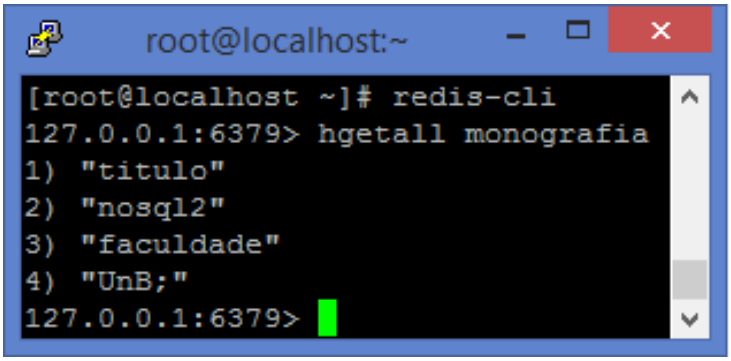


Figura 4.7: Select * no Redis via servidor físico BD4.

A verificação é efetuada através da console da ferramenta putty.exe para acesso ao servidor *linux*: BD4, onde o comando `redis-cli` efetua a conexão com o SGBD Redis:

`127.0.0.1` e porta de acesso: `6379`; e `hgetall monografia` apresenta os dados da tabela `monografia`.

Os testes de funcionalidade demonstram que a solução NoSQL^2 persiste adequadamente os dados, tendo o funcionamento esperado, não se havendo detectado erros de difícil resolução ou problemas na configuração do experimento.

4.2.2 Testes de Extensibilidade

Os testes de extensibilidade realizados nessa etapa visam validar os aspectos relacionados as características de extensibilidade dos componentes da solução NoSQL^2. A estrutura de classes do NoSQL^2 permite a incorporação/adição de diferentes conectores extensores NoSQL, possibilitando a utilização de variados modelos e bases de dados NoSQL.

O experimento apresentado na Subseção 4.1 permite realizar conexões com os bancos de dados NoSQL: MongoDB, Redis, Neo4J e Cassandra. De forma, a validarmos a característica de extensibilidade da solução com relação a adição de novos SGBD NoSQL, o estudo de caso aqui apresentado realiza a incorporação do conector para acesso ao banco de dados CouchDB Versão 2.0.0, e consequentemente a utilização de mais um representante do modelo orientado a documentos na arquitetura do NoSQL^2.

Conforme apresentado na Seção 3.3, a incorporação de novos conectores NoSQL implica na importação de uma classe *java* correspondente ao *driver* de acesso do conector. Assim, para o CouchDB, faz-se necessário a adição do *Java Driver*: couchdb4j-0.1.2.jar disponível na página do fabricante *Apache* CouchDB (Anderson et al., 2010). A Tabela 4.1 apresenta a listagem dos *drivers Java* correspondente aos sistemas gerenciadores de banco de dados NoSQL usados no experimento.

Tabela 4.1: Tabela *Java Driver* e NoSQL

SGBD NoSQL	***Java Driver***
Cassandra	com.datastax.driver.core.*
MongoDB	com.mongodb.*
Redis	jedis.jar
Neo4J	org.neo4j.driver.v1.*
CouchDB	couchdb4j-0.1.2

O Código 4.5 mostra a implementação dos métodos conectar() e createDatabase() para a classe *java* CouchDB, onde as linhas de comandos são descritas conforme segue:

Código 4.5: Conexão no CouchDB

```
1 public class CouchDB {
```

```
2          public void conectar() {
3              Session session = new Session("localhost", 5984);
4          }
5          public void createDatabase(String nomeDatabase) {
6              Session session = new Session("localhost", 5984);
7              session.createDatabase(nomeDatabase);
8          }
9  }
```

. Linha 1: Declaração da classe principal do CouchDB;

. Linhas 2 a 4: Implementação dos comandos para conexão com o CouchDB;

. Linha 5 a 8: Implementação do método createDatabase().

A Figura 4.8 exibe a listagem dos bancos de dados no servidor físico 127.0.0.1, e como pode ser observado entre os bancos de dados disponíveis na console gráfica do CouchDB (`Databases`) encontra-se o banco de dados `mestrado`.

A verificação é efetuada via interface gráfica do CouchDB disponível em http://127.0.0.1:5984/utils/, porta de acesso: 5984 e servidor físico: `127.0.0.1`, onde a janela Databases ilustra a base de dados mestrado criada após a execução do comando C1.

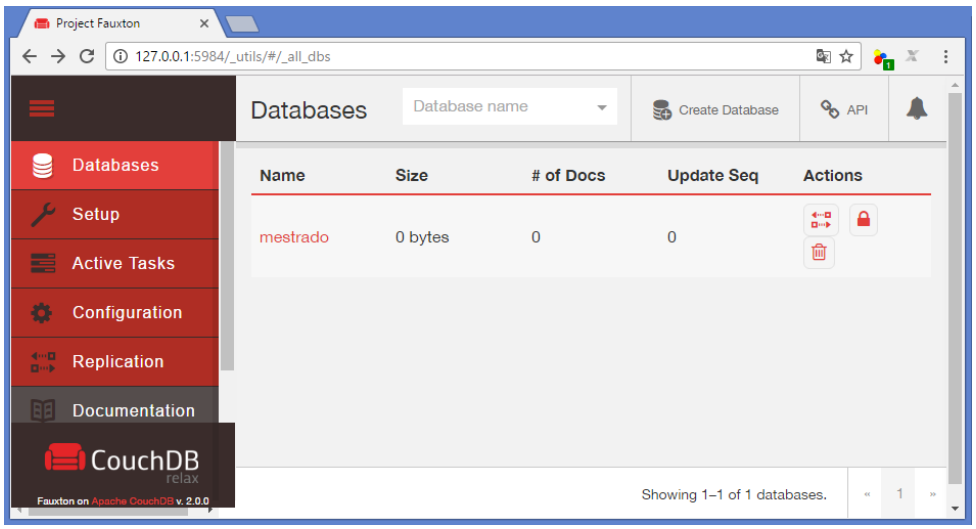


Figura 4.8: Criação do banco de dados Mestrado no CouchDB.

A Saída de Resultado 4.6 mostra o *log* com o resultado da execução do comando C1 usando o experimento e o SGBD CouchDB, onde as linhas de resultados são descritas conforme segue:

Saída de Resultado 4.6: Execução do comando C1

```
1 Comando (Instrução SQL): CREATE DATABASE <mestrado>
2 Processo de Varredura da Consulta!
3 Processo de Tradução do Comando!
4 Processo: Comando NoSQL: dbSession.createDatabase(mestrado);
5 Processo: Conectores!
6 Processo: CouchDB --> CREATE DATABASE --> OK!
7 Tempo Execução: 30 milissegundos
```

. Linha 1: Comando (Instrução SQL) a ser executado;

. Linhas 2: Processo de Varredura da consulta C1;

. Linha 3: Processo de Tradução do comando C1;

. Linha 4: Comando traduzido para a sintaxe do CouchDB ;

. Linhas 5: Processo Conectores para conexão com o CouchDB;

. Linhas 6: Execução do comando C1 traduzido no CouchDB;

. Linha 7: Tempo de execução.

A arquitetura NoSQL[2] permite o aprimoramento das classes através da adição de novos comandos, e para validar essa característica de extensibilidade, o estudo de caso aqui apresentado realiza a incorporação do comando `TRUNCATE`, que remove todos os dados de uma tabela sem eliminar a estrutura da tabela da base de dados.

O Código 4.7 apresenta o exemplo da implementação do método `truncateTable`, relativo ao comando `TRUNCATE`, na classe *java* associada ao conector para o Cassandra, onde as linhas de comandos são descritas conforme segue:

Código 4.7: Implementação do método `truncateTable`

```
1 public void truncateTable(String nomeTabela){
2           this.connect("192.168.126.131", 9042);
3           this.getSession().execute("USE mestrado");
4           this.getSession().execute("TRUNCATE " + nomeTabela + "
              ;");
5  }
```

. Linha 1: Declaração do método `truncateTable`;

. Linha 3: Conexão com a base de dados Cassandra no servidor virtual BD2;

. Linha 3: Conexão com a base de dados `mestrado`; e

. Linha 4: Sintaxe do comando `Truncate()` para o Cassandra.

A Saída de Resultado 4.8 apresenta o *log* com o resultado da execução do comando `TRUNCATE` usando o experimento e o SGBD Cassandra, onde as linhas de resultados são descritas conforme segue:

Saída de Resultado 4.8: Execução do comando `TRUNCATE`

```
1 Comando (Instrução SQL): TRUNCATE TABLE monografia
2 Processo de Varredura da Consulta!
3 Processo de Tradução do Comando!
4 Processo: Comando Traduzido NoSQL: TRUNCATE TABLE monografia
5 Processo: Conectores!
6 Processo: Cassandra --> TRUNCATE TABLE --> OK
7 Tempo Execução: 1267 milissegundos
```

. Linha 1: Comando (Instrução SQL) a ser executado;

. Linhas 2: Processo de Varredura da consulta;

. Linha 3: Processo de Tradução do comando;

. Linha 4: Comando traduzido para a sintaxe do Cassandra;

. Linhas 5: Processo Conectores para conexão com o Cassandra;

. Linhas 6: Execução do comando C1 traduzido no Cassandra;

. Linha 7: Tempo de execução.

A Figura 4.9 ilustra a console *linux* via ferramenta putty.exe, onde o comando `./cqlsh 192.168.126.131 9042` efetua a conexão com a base de dados *default* no Cassandra; `use mestrado` efetua a conexão com a base de dados `mestrado`; o comando `insert into monografia (titulo, faculdade) values ('NoSQL2','UnB')` insere dados dentro da tabela `monografia`; o comando `select * from monografia` seleciona os registros da tabela NoSQL^2 `monografia`. O resultado do comando `TRUNCATE` após a execução pelo experimento, é verificado através de consulta a tabela `monografia` via comando `select * from monografia`, que conforme demonstra a Figura 4.9 apresenta uma tabela sem dados, comprovando assim que a execução do comando no NoSQL^2 removeu todos os registros da tabela `monografia` mantendo a estrutura da tabela na base de dados `mestrado`.

O comando `TRUNCATE` pode ser implementando nas demais classes de sistemas gerenciadores de banco de dados NoSQL incorporadas no NoSQL^2, sendo necessário a definição da implementação do método nessas classes.

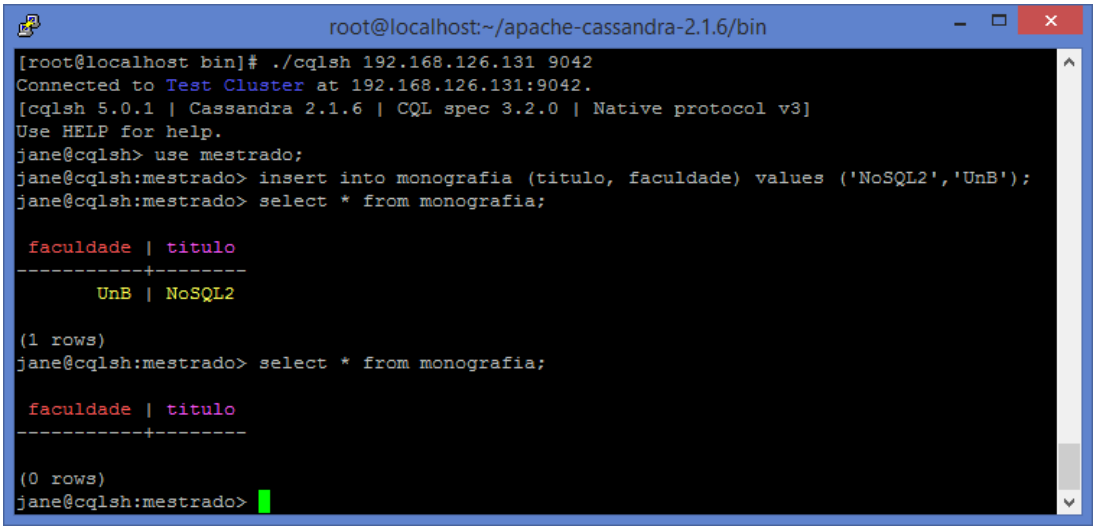


Figura 4.9: Truncate da tabela `monografia` no Cassandra via servidor virtual `BD2`.

É importante salientar, que o mapeamento das instruções SQL para a sintaxe nativa do NoSQL, esbarra em particularidades de cada SGBD, pois comandos usualmente conhecidos na linguagem SQL, podem não ser aplicáveis ao SGBD, afetando assim a extensibilidade da solução com relação a adição de novos comandos. Portanto, previamente a adição de novos comandos, estes devem ser mapeados para verificar se sua implementação é condizente com o conjunto de comandos e operadores permitidos para o SGBD NoSQL.

Os testes de extensibilidade demonstram que a solução NoSQL^2 possui características de extensibilidade de componentes permitindo:

- Adicionar diferentes conectores extensores NoSQL possibilitando a utilização de variados modelos e bases de dados NoSQL;
- Adicionar novos comandos a solução, ampliando o conjunto de comandos a serem utilizados.

4.2.3 Comparação com outros SGBD

A comparação com outros SGBD nessa dissertação, visa verificar se o NoSQL^2 é aplicável a abordagens de SGBD relacionais, e assim aumentando a aplicabilidade da solução englobando também SGBD relacionais.

O estudo de caso apresentado realiza a incorporação do conector para acesso ao banco de dados SQLServer, e consequentemente a utilização de um representante do modelo relacional na arquitetura do NoSQL^2.

Usando a mesma ideia exposta na Seção 3.3, a incorporação de um conector implica na importação de uma classe *java* correspondente ao *driver* de acesso do conector. Assim,

para o SQLServer, faz-se necessário a adição do *Java Driver*: java.sql.* disponível na página do fabricante Microsoft[15]. A plataforma para o estudo de caso é um notebook Dell Inspiron 14 com processador Intel Core memória 8 GB Sistema Operacional 64bits Windows 8.1 com o SGBD SQLServer Express versão 2016.

O Código 4.9 mostra o código que implementa os métodos conectar() e createDatabase na classe SQLServer.

Saída de Resultado 4.9: Conexão no SQLServer

```
1  public class SQLServer {
2           Connection con;
3          public void conectar() {
4             try
5                 {
6                   Class.forName("com.microsoft.jdbc.sqlserver.
                       SQLServerDriver");
7                 }
8                 catch(java.lang.ClassNotFoundException e)
9                 {
10                        System.err.print("Erro de conexÃ&o: ");
11                        System.err.println(e.getMessage());
12                }
13
14         public void createDatabase(String nomeDatabase) {
15              String sql = "CREATE DATABASE " + nomeDatabase;
16
17              Statement statement = con.createStatement();
18              statement.executeUpdate(sql);
19              statement.close();
20            }
21 }
```

As linhas de comandos são descritas conforme segue:

. Linha 1: Declaração da classe principal do SQLServer;

. Linha 2: Declaração da variável pública para conexão Con;

. Linhas 3 a 12: Implementação do método conectar() para conexão com o SGBD SQLServer;

[15]Microsoft: https://www.microsoft.com/

. Linhas 14 a 21: Implementação do método createDatabase() para criação de uma base de dados no SGBD SQLServer;

O estudo de caso realizado demonstra que o NoSQL^2 pode ser usado para realização de tarefas de administração em bancos de dados relacionais seguindo os mesmos critérios para incorporação de SGBD NoSQL, ou seja, adicionar o *java driver* correspondente ao SGBD, definir o mapeamento e assinatura dos comandos nas classes `Tradutor` e `Conectores` e a implementar o codigo do método na classe correspondente ao SGBD. Com essa abordagem, considera-se que o NoSQL^2 consiste em uma alternativa viável de uso para os administradores de banco de dados no gerenciamento de SGBD relacionais e não-relacionais.

Capítulo 5

Conclusões

Os SGBD NoSQL diferem dos bancos de dados relacionais, que têm a linguagem SQL como um padrão, não apenas no seu modelo de dados, mas também no fornecimento de mecanismos de consulta e recuperação de dados. Os bancos de dados NoSQL possuem características particulares em relação a recuperação de dados ou formas de consulta: de uma maneira geral, os comandos são semelhantes à sintaxe tradicional do SQL, porém não há um padrão da linguagem entre as famílias NoSQL e, o nível de conhecimento exigido dos usuários para escrever simples consultas é mais avançado do que nos SGBD relacionais.

Os bancos de dados NoSQL diferem no conjunto de mecanismos de consulta suportados, e quando uma linguagem de consulta é estabelecida para o NoSQL, ela possui uma sintaxe de comandos própria e não padronizada nem mesmo entre os modelos. Isso é um complicador, principalmente para o DBA que precisa ter recursos para execução de tarefas administração e recuperação de dados, onde no caso dos SGBD relacionais, esse apoio é fornecido pela linguagem SQL, uma linguagem de consulta simples e familiar à comunidade de banco de dados.

A capacidade de fornecer linguagens de consulta eficientes e fáceis de usar é um assunto importante para o estudo acadêmico sobre NoSQL, e é considerado um desafio uma vez que existem vários SGBD NoSQL.

Neste contexto, o NoSQL^2, apresentado nesta dissertação, propõe um middleware para a execução de comandos de administração em bancos de dados NoSQL, usando a sintaxe de comandos da linguagem SQL, libertando assim, os usuários da preocupação com as especificidades de cada banco de dados NoSQL. O NoSQL^2 é uma camada intermediária entre uma aplicação-cliente e o sistema gerenciador de banco de dados NoSQL, permitindo que o usuário/DBA utilize instruções SQL para realização de consultas e gerenciamento da base de dados. A existência da camada intermediária é transparente para o usuário, sendo que ela realiza o mapeamento da instrução fornecida pelo usuário para a correspondente

sintaxe particular do SGBD NoSQL.

A característica de extensibilidade do NoSQL^2 contribui para a expansão do funcionamento da solução, visando abranger novos sistemas gerenciadores de banco de dados NoSQL e também novos comandos, que podem ser adicionados de maneira simples, tornando assim a solução interessante para ambientes com vários SGBD, e que requerem administração diversificada. Também a portabilidade da solução pode ser considerada uma característica de flexibilidade permitindo que a solução possa ser executada em ambientes heterogêneos.

O NoSQL^2 apresenta-se como uma solução flexível e extensível, permitindo que os DBA realizem tarefas de gerenciamento em ambientes com múltiplos SGBD NoSQL.

Em relação ao alcance dos objetivos específicos da dissertação conclui-se que:

- O NoSQL^2 é um *middleware* desenvolvido para execução de tarefas de administração em banco de dados NoSQL;
- O NoSQL^2 pode ser utilizado em plataformas heterogêneas compostas por diferentes banco de dados NoSQL, e também relacionais;
- Os testes de funcionalidade, extensibilidade e comparativo demonstraram que o NoSQL^2 é uma proposta válida, flexível e extensível para utilização em diversos SGBD.

Como trabalhos futuros, visualiza-se a incorporação de um conjunto amplo de comandos DDL para tarefas de administração de banco de dados, realização de rotinas de análise de desempenho e *backup*, e a incorporação de novos conectores NoSQL. A validação de estratégia de execução ou plano de execução e otimização da consulta de bases de dados NoSQL são candidatos a formulação de otimização e melhoria da solução.

5.1 Trabalhos Publicados

Durante os estudos e desenvolvimento da dissertação de mestrado, de forma a obter experiência e entender a relevância do tema junto a comunidade acadêmica, participamos da elaboração de 4 artigos publicados em conferências e revistas, a citar:

. Artigo NoSQL^2: Administrando Banco de Dados NoSQL com a Linguagem SQL, publicado na 13ª *Applied Computing Conference* (IADS 2016), em outubro de 2016, que aborda uma das sessões discutidas nessa dissertação: a arquitetura da camada intermediária NoSQL^2 (Adriana e Holanda, 2016c);

. Artigo *Brasilia's Database Community Profile*, publicado em 2016, na 11ª *Conferencia Ibérica de Sistemas y Tecnologías de Información (CISTI)* (Adriana e Holanda, 2016b) trata da adoção de práticas e procedimentos para a gestão de banco de dados, investigando técnicas e ferramentas utilizadas pelos administradores de banco de dados na cidade de Brasilia no ano de 2015. O estudo destaca características e particularidades em bancos de dados NoSQL e os resultados demonstram que as novas tecnologias em matéria de gestão de banco de dados são atualmente relevantes temas para a comunidade de banco de dados;

. Artigo *Brasilia's Database Administrators*, publicado em 2016, no *Journal of Information Systems Engineering Management* (Adriana e Holanda, 2016a) complementa o assunto sobre a adoção de práticas e procedimentos administrativos para a gestão de banco de dados, e as técnicas e ferramentas utilizadas pelos administradores de banco de dados na cidade de Brasilia entre os anos de 2015 e 2016;

. Livro *Handbook of Research on Innovative Database Query Processing Techniques* destaca em seu capítulo *Query languages in NoSQL databases* (Adriana e Holanda, 2015) que a ausência de uma linguagem declarativa de consulta nos NoSQL aumenta a responsabilidade dos desenvolvedores e usuários responsáveis pela estruturação de objetos, estruturas e dados na base de dados, e o impacto é uma maior complexidade na execução dessas atividades.

Referências

Aasman, J. (2006). Allegrograph: RDF triple database. *Franz Incorporated, Oakland*. 23

Abadi, D., Agrawal, R., Ailamaki, A., and Balazinska, M. (2016). The Beckman Report on Database Research. *Communications of the ACM*, pages 92–99. 1, 3, 17, 26

Abiteboul, S., Agrawal, R., Bernstein, P., Carey, M.and Ceri, S., Croft, B., and Naughton, J. (2016). Lowell database research self-assessment, published as microsoft technical report msr-tr. *In Proceedings of Journal of Information Systems Engineering Management, Lectito BV, Netherlands*, 48(3):5. 31

Adriana, J. and Holanda, M. (2015). *Chapter of Handbook of Research on Innovative Database Query Processing Techniques: Query languages in NoSQL databases*, volume 1. IGI Globall, Hershey PA, USA. 24, 66

Adriana, J. and Holanda, M. (2016a). Brasilia's database administrators. *In Proceedings of Journal of Information Systems Engineering Management, Lectito BV, Netherlands*, 1(3):149–157. 3, 33, 66

Adriana, J. and Holanda, M. (2016b). Brasilia's database community profile. *In Proceedings of 11a Conferência Iberica de Sistemas e Tecnologias da Informação (CISTI), Gran Canaria, España*, 1:1290–1294. 31, 33, 66

Adriana, J. and Holanda, M. (2016c). Nosql2: Administering nosql databases with sql. *In Proceedings of 13a International Conference Applied Computing (IADS), Mannheim, Germany*, 1:225–229. 65

Aho, A. V., Sethi, R., and Ullman, J. D. (1986). *Compilers, Principles, Techniques*. Addison wesley. 7, 8, 9, 35

Aiken, P., Gillenson, M., Zhang, X., and Rafner, D. (2011). *Data management and data administration: assessing 25 years of practice*, volume 22. J. Database Manag. 31

Albaum, G. (1997). The likert scale revisited: an alternative version. Disponível em http://repository.ust.hk/ir/Record/1783.1-17657. Acessado em: 03 de dezembro de 2016. 31

Anderson, J. C., Lehnardt, J., and Slater, N. (2010). *CouchDB: the definitive guide*. "O'Reilly Media, Inc.". 15, 21, 57

Armstrong, J., Virding, R., Wikström, C., and Williams, M. (1993). Concurrent programming in erlang. 21, 22

Bach, M. and Werner, A. (2014). Standardization of nosql database languages. *10th IEEE Beyond databases, architectures, and structure (BDAS)*, pages 50–60. 2, 3, 11, 17, 23, 29

Banker, K. (2011). *MongoDB in action*. Manning Publications Co. 15, 20, 21, 28, 29

Bilidas, D., Horrocks, I., Kllapi, H., and Zheleznyakov, D. (2013). Distributed query processing on the cloud the optique point of view. *OWL Experiences and Directions Workshop (OWLED), Montpellier, France*, pages 1–7. 3, 11, 12, 28

Bugiotti, F., Atzeni, P., and Rossi, L. (2012). Sos (save our systems): A uniform programming interface for non-relational systems. *In Proceedings of the 15th International Conference on Extending Database Technology (EDBT), ACM, Berlin, Germany*, pages 582–585. 29

Bugiotti, F., Atzeni, P., and Rossi, L. (2014). Uniform access to nosql systems. *Information Systems Elsevier*, page 117–133. 3, 11, 12, 18, 28, 45

Buneman, P., Fernandez, M., and Suciu, D. (2000). Unql: a query language and algebra for semistructured data based on structural recursion. *The VLDB Journal—The International Journal on Very Large Data Bases*, 9(1):76–110. 29

Burdakov, A., Grigorev, U., Ploutenko, A., and Ttsviashchenko, E. (2016). Estimation models for nosql database consistency characteristics. *In Proceedings of the 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processings*, pages 35–42. 2, 3, 11, 17, 26, 27

Calil, A. and Mello, R. d. S. (2011). Simplesql a relational layer for simpledb. *16th East European conference on Advances in Databases and Information Systems Springer (ADBIS), Poznan, Poland*, pages 99–110. 26

Carlson, J. L. (2013). *Redis in Action*. Manning Publications Co. 14

Catell, R. (2010). Scalable sql and nosql data store. *Newsletter ACM SIGMOD*, pages 12–27. 14, 15, 19, 20, 21, 22

Chang, F., Dean, J., and Ghemawat, S. (2008). Bigtable: A distributed storage system for structured data. Disponível em http://dl.acm.org/citation.cfm?id=1773922. Acessado em: 02 de dezembro de 2016. 15, 20

Corbellini, A., Mateos, C., Zunino, A., Godoy, D., and Schiaffino, S. (2016). Persisting big-data: The nosql landscape. *Information Systems*, 63:1–23. 1, 2, 3, 11, 16, 18, 45

Costa, A. V., Vilian, P., and Mello, R. d. S. (2016). Layer for the mapping management of sql dml instructions to the key-value nosql database voldemort. *In Proceedings of XII Brazilian Symposium on Information Systems, Florianopolis, SC*, pages 224–231. 26, 27

Curino, C., Jones, E. P., Popa, R. A., Malviya, N., Wu, E., Madden, S., Balakrishnan, H., and Zeldovich, N. (2011). Relational cloud: A database-as-a-service for the cloud. 27

De Diana, M. and Gerosa, M. A. (2010). Um estudo comparativo de bancos não-relacionais para armazenamento de dados na web 2.0. *In Proceeding of the WTDBD IX Workshop de Teses e Dissertações em Banco de Dados, Belo Horizonte, Brasil.* 16

Dean, J. and Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113. 20, 21, 22, 28

DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., and Vogels, W. (2007). Dynamo: amazon's highly available key-value store. *ACM SIGOPS Operating Systems Review*, 41(6):205–220. 14

Dourish, P., Edwards, W. K., LaMarca, A., and Salisbury, M. (1999). Presto: an experimental architecture for fluid interactive document spaces. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 6(2):133–161. 27

Elmasri, R. and Navathe, Shamkanr, S. P. B. (2011). *Sistemas De Banco De Dados.* Pearson AddisonWesley, São Paulo, Brasil. 1, 6, 7, 8, 10, 38, 43

Enquete (2015). Enquete administradores banco de dados. Disponível em https://docs.google.com/forms/d/1nlpgcWVjW4oWnkH96JdTe15ZXqoaHZ2no-Jp2ZWH3_s/viewform?usp=send_form. Acessado em: 03 de dezembro de 2016. 32

Farias, V. A. E., Sousa, F. R. C., Maia, J. G. R., Gomes, J. P. P., and Machado, J. C. (2016). Machine learning approach for cloud nosql databases performance modeling. *In Proceedings of the 16th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, Networking and Parallel/Distributed Computing (SNPD), Federal University of Ceara*, pages 617–620. 11

George, L. (2011). *HBase: The Definitive Guide: Random Access to Your Planet-Size Data*, volume 1. ORelly. 15, 20, 28, 29

Gessert, F. and Ritter, N. (2016). Scalable data management: Nosql data stores in research and practice. *In Proceedings of the IEEE 32nd International Conference on Data Engineering (ICDE), University of Hamburg*, pages 1420–1423. 8, 12, 14, 15

Gomez, P., Casallas, R., and Roncancio, C. (2016). Data schema does matter, even in nosql systems! *In Proceedings of the IEEE Tenth International Conference on Research Challenges in Information Science (RCIS)*, pages 1–6. 3, 11, 27, 45

Gonzalez-Aparicio, M. T., Younas, M., Tuya, J., and Casado, R. (2016). A new model for testing crud operations. *In Proceedings of the IEEE 30th International Conference on Advanced Information Networking and Applications*, pages 79–86. 11, 13, 18, 45

Guo, M., Qian, K., and Yang, L. (2016). Hands-on labs for learning mobile and nosql database security. *IEEE 40th Annual Computer Software and Applications Conference*, pages 606–607. 17

Hakala, J. (2001). Using international standard book numbers as uniform resource names. 13

Han, J., Haihong, E., Le, G., and Du, J. (2011). Survey on nosql database. In *Pervasive computing and applications (ICPCA), 2011 6th international conference on*, pages 363–366. IEEE. 14, 19

Hatcher, E. and Gospodnetic, O. (2004). Lucene in action. 19, 22

Hecht, R. and Jablonski, S. (2011). Nosql evaluation a use case oriented survey. *15th International Conference on Cloud and Service Computing, IEEE CSC*, pages 336–341. 14, 15, 16, 18, 25

Holt, V., Ramage, M., Kear, K., and Heap, N. (2015). The usage of best practices and procedures in the database community. *Journal Information Systems, Faculty of Mathematics, Computing and Technology,The Open University, UK*, pages 163–181. 31

Holzschuher, F. and Peinl, R. (2013). Performance of graph query languages: comparison of cypher, gremlin and native access in neo4j. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, pages 195–204. ACM. 22

Indrawan-Santiago, M. (2012). Database research: Are we at a crossroad? reflection on nosql. *15th International Conference on Network-Based Information Systems, IEEE NBIS*, pages 45–51. 12, 14, 15

Kaur, K. and Rani, R. (2013). Modeling and querying data in nosql databases. *In Proceedings of the IEEE International Conference on Big Data*, pages 1–7. 15, 21, 23

Khetrapal, A. and Ganesh, V. (2006). Hbase and hypertable for large scale distributed storage systems. *Dept. of Computer Science, Purdue University*, pages 22–28. 20

Klophaus, R. (2010). Riak core: building distributed applications without shared state. In *ACM SIGPLAN Commercial Users of Functional Programming*, page 14. ACM. 14, 15, 22

Kornacker, M., Behm, A., Bittorf, V., Bobrovytsky, T., Ching, C., Choi, A., Erickson, J., Grund, M., Hecht, D., Jacobs, M., et al. (2015). Impala: A modern, open-source sql engine for hadoop. In *CIDR*. 29

Lakshman, A. and Malik, P. (2010). Cassandra: a decentralized structured storage system. Disponível em http://dl.acm.org/citation.cfm?id=1773922. Acessado em: 02 de dezembro de 2016. 15, 19, 28

Lawrence, R. (2014). Integration and virtualization of relational sql and nosql systems including mysql and mongod. *International Conference on Computational Science and Computational Intelligence, IEEE CSCI*, pages 285–290. 2, 7, 28

Lee, C.-H. and Zheng, Y.-L. (2015). Automatic sql-to-nosql schema transformation over the mysql and hbase databases. *In Proceedings of the International Conference on Consumer Electronics Taiwan (ICCE-TW)*, pages 426–427. 2, 3, 11

Lehmayr, P., Rith, J., and Meyer-Wegener, K. (2014). Speaking in tongues: Sql access to nosql systems. *29th Annual ACM Symposium on Applied Computing, ACM SAC, Gyeongju, Republic of Korea*, pages 855–857. 26, 28

Liu, X., Lang, B., Yu, W., Luo, J., and Huang, L. (2011). Audr an advanced unstructured data repository. *In Proceedings 2011 6th International Conference on Pervasive Computing and Applications, ICPCA, State Key Laboratory of Software Development Environment, Beihang University, China*, pages 462–469. 11

Liu, Z. H., Hammerschmidt, B., McMahon, D., Liu, Y., and Chang, H. J. (2016). Closing the functional and performance gap between sql and nosql. *In Proceedings of the International Conference on Management of Data SIGMOD, Redwood Shores, CA 94065, USA*, pages 227–238. 27

Masse, M. (2011). *REST API design rulebook*. "O'Reilly Media, Inc.". 20

McCandless, M., Hatcher, E., and Otis, G. (2010). *Lucene in action*. Manning, second edition. 25

Moniruzzaman, A. and Hossain, S. A. (2013). Nosql database: New era of databases for big data analytics-classification, characteristics and comparison. *arXiv preprint arXiv:1307.0191*. 21

Muhammad, Y. (2011). *Evaluation and Implementation of Distributed NoSQL Database for MMO Gaming Environment*. PhD thesis, Uppsala Universitet, Uppsala, Sweden. 12, 19

Mullins, C. (2012). *Database Administration The Complete Guide to DBA Practices*. Addison-Wesley Professional. 7, 10, 45

Nasholm, P. (2012). *Extracting data from nosql databases a step towards interactive visual analysis of nosql data*. PhD thesis, Chalmers University of Technology. 2, 3, 11, 16, 17

Nutter, C. O., Enebo, T., Sieger, N., Bini, O., and Dees, I. (2011). *Using JRuby: Bringing Ruby to Java*. Pragmatic Bookshelf. 22

Olson, M. A., Bostic, K., and Seltzer, M. I. (1999). Berkeley db. In *USENIX Annual Technical Conference, FREENIX Track*, pages 183–191. 14

Padhy, R. P., Patra, M. R., and Satapathy, S. C. (2011). Rdbms to nosql: Reviewing some next-generation non-relational databases. *International Journal of Advanced Engineering Science and Technologies, IJAEST*, pages 15–30. 1

Pierre, G. and Stratan, C. (2012). Conpaas: a platform for hosting elastic cloud applications. *IEEE Internet Computing*, 16(5):88–92. 27

Porkorny, J. (2013). Nosql databases: A step to database scalability in web enviromnent. *In Proceedings of the 13th International Conference on Information Integration and Web based Applications and Services, ACM, Faculty of Mathematics and Physics, Charles University, Praha, Czech Republic*, pages 69–82. 3, 11, 15, 28

Ringlstetter, A., Scherzinger, S., and Bissyande, T. F. (2016). Data model evolution using object-nosql mappers:folklore or state-of-the-art?, austin, texas. *In Proceedings of the 2nd International Workshop on BIG Data Software Engineering, BIGDSE*, pages 33–36. 11, 44

Rumbaugh, J., Jacobson, I., and Booch, G. (2004). *Unified Modeling Language Reference Manual, The*. Pearson Higher Education. 40

Schreiner, G. A. (2016). *Sql to key nosql: uma camada para mapeamento de esquemas relacionais e de operacoes sql para bancos de dados nosql baseados em chaves de acesso*. PhD thesis, Universidade Federal de Santa Catarina, Florianopolis, Brasil. 14, 16, 26

Sharma, S., Shandilya, R., Patnaik, S., and Mahapatra, A. (2017). Leading nosql models for handling big data: a brief review. *In Proceedings of the International Journal of Business Information Systems (IJBIS)*, 22(1):1–25. 11, 12, 19, 21

Shvachko, K., Kuang, H., Radia, S., and Chansler, R. (2010). The hadoop distributed file system. In *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*, pages 1–10. IEEE. 20

Silberschatz, A., Korth, e. F., and Sudarshan, S. (2010). *Database System Concepts*. McGraw-Hill. 1, 7, 43

Silva, Y. N., Almeida, I., and Queiroz, M. (2016). Sql: From traditional databases to big data. *In Proceedings of 47th ACM Technical Symposium on Computing Science Education, SIGCSE, Memphis, United States*, pages 413–418. 2, 3, 7, 11, 26, 27

Sumbaly, R., Kreps, J., Gao, L., Feinberg, A., Soman, C., and Shah, S. (2012). Serving large-scale batch computed data with project voldemort. In *Proceedings of the 10th USENIX conference on File and Storage Technologies*, pages 18–18. USENIX Association. 14

Tauro, C., Aravindh, S., and Shreeharsha, A. (2012). Comparative study of the new generation, agile, scalable, high performance nosql databases. *International Journal of Computer Applications*, pages 975–888. 11, 12, 16, 19, 21, 25

Vukotic, A., Watt, N., Abedrabbo, T., Fox, D., and Partner, J. (2015). *Neo4j in Action*. Manning. 22

Wei, L. and Bo, L. (2010). A tetrahedral data model for unstructured data management. *Science China Information Sciences*, pages 1497–1510. 11

Yan, L. (2015). *Handbook of Research on Innovative Database Query Processing Techniques*. IGI Global, Hershey PA, USA. 3, 24, 26, 27

Yi, L., Naihu, W., Chunjiang, H., Jiang, Z., and Jin, H. (2013). Managing large scale unstructured data with rdbms. *In Proceedings of the 11th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC), Jinan, China*, pages 613–620. 28

Apêndice A

Pesquisa-Enquete

Edit this form

Programa de Pós-Graduação em Informática - Departamento de Ciência da Computação - UnB

Enquete Administradores Banco de Dados

Qual a sua experiência como DBA?

- ○ Menos de 1 ano
- ○ Menos de 2 anos
- ○ Entre 2 anos e 5 anos
- ○ Entre 5 e 10 anos
- ○ Acima de 11 anos

Você desenvolveu sua experiência como DBA trabalhando em que tipos de organizações/instituições?

- ○ Empresa Privada Nacional
- ○ Empresa Privada Multinacional
- ○ Órgão Público
- ○ Outras

Você tem experiência na administração de quais tipos/modelos de banco de dados?

- ☐ Relacional
- ☐ NoSQL (chave-valor, família de colunas, grafo, etc)
- ☐ Em Rede
- ☐ Outros

Qual o quantitativo de funcionários da organização/instituição que você trabalha atualmente?

- ○ Menos de 100
- ○ Entre 100 e 300
- ○ Entre 301 e 500
- ○ Acima de 501
- ○ Não sei dizer

Qual a plataforma de banco de dados utilizada na organização/instituição que você trabalha atualmente é:

- ☐ Infraestrutura própria (servidores físicos, virtuais, etc)
- ☐ Infraestrutura em Nuvem (Cloud Platforma)
- ☐ Outras

Qual é a aproximadamente o quantitativo de bancos de dados existentes na organização/instituição que você trabalha?

- ◯ Menos de 10
- ◯ Entre 11 e 50
- ◯ Entre 51 e 100
- ◯ Acima de 101
- ◯ Não sei dizer

Quais são os modelos de bancos de dados existentes na organização/instituição que você trabalha?

- ☐ Tradicionais RDBMS (Ex. Oracle, SQLSERVER, etc)
- ☐ NoSQL (Ex.: MongoDB, Cassandra, etc)
- ☐ Orientado a objetos
- ☐ NewSQL
- ☐ Outros

Quantos administradores de bancos de dados atuam na organização/instituição que você trabalha?

- ◯ Menos de 2
- ◯ Entre 2 e 5
- ◯ Entre 6 e 10
- ◯ Entre 11 e 20
- ◯ Acima de 21

Qual é aproximadamente o tamanho do maior banco de dados da organização/instituição que você trabalha?

- ◯ Menos de 100GB
- ◯ Entre 101 e 500GB
- ◯ Entre 501GB e 1TB
- ◯ Entre 1TB e 20TB
- ◯ Acima de 20TB

Qual é aproximadamente o quantitativo de mudanças estruturais (execução de comandos DDL, alteração de esquema, objetos, etc) você realiza durante uma semana, na organização/instituição em que você trabalha?

- ◯ Menos de 2
- ◯ Entre 1 e 10
- ◯ Entre 11 e 20
- ◯ Entre 21 e 40
- ◯ Acima de 40

Qual(is) são as maiores preocupações da sua organização/instituição em termos dos bancos de dados que ela possui?

- ☐ Requisitos de Segurança
- ☐ Espaço de armazenamento
- ☐ Disponibilidade
- ☐ Armazenamento em nuvem
- ☐ Desempenho
- ☐ Backup

- [ ] Mecanismos ágeis para recuperação de desastres
- [ ] Arquitetura

No desenvolvimento de suas atividades como DBA, com que frequência você utiliza a linguagem SQL como uma ferramenta de administração do banco de dados?

	1	2	3	4	5	
Raramente	○	○	○	○	○	Muito Frequentemente

Como você aperfeiçoa seu conhecimento e capacitação na área de banco de dados e novas tecnologias associadas ao assunto?

- [ ] Através de cursos e treinamentos disponibilizados na organização/instituição em que trabalho
- [ ] Através de cursos e treinamentos que realizo por conta própria
- [ ] Através de auto-estudo
- [ ] Participando de conferências
- [ ] Lendo artigos e pesquisando em fóruns
- [ ] Não me aperfeiçoo

Você tem conhecimentos e prática em relação a BigData?

	1	2	3	4	5	
Pouco	○	○	○	○	○	Muito Conhecimento

Você está familiarizado com o conceito de Banco de Dados NoSQL?

	1	2	3	4	5	
Nenhum conhecimento	○	○	○	○	○	Muito conhecimento

Você como DBA incentiva a instituição/organização em conhecer novos modelos de bancos de dados, tais como NoSQL?

	1	2	3	4	5	
Muito Pouco	○	○	○	○	○	Insistentemente

Você considera que a existência de diferentes SGBDs e modelos de banco de dados, alinhada as peculiaridades de cada um, exige um alto nível de especialização dos DBAs, visto que torna a administração complexa e com necessidade de capacitação/estudos constantes devido as diferenças na arquitetura e comandos a serem empregados?

	1	2	3	4	5	
Baixo Nível de Especialização	○	○	○	○	○	Alto Nível de Especialização

No seu entendimento, a existência de uma ferramenta ou interface que possibilite de forma fácil, a administração de diferentes modelos de bancos de dados tais como relacionais e NoSQL é um assunto encorajador para os DBAs?

▼

Como DBA você considera a existência de uma linguagem de consulta de alto nível (tal qual SQL padrão para os bancos de dados relacionais) fundamental para a administração de um banco de dados?

▼

Você apoiaria o desenvolvimento de uma camada intermediária (middleware) que possibilitasse aos DBAs utilizar comandos DDL da linguagem SQL padrão para diferentes bancos de dados?

▼

Submit

Powered by

This content is neither created nor endorsed by Google.

Report Abuse - Terms of Service - Additional Terms

Apêndice B

Resultado da Pesquisa-Enquete no Ano de 2015

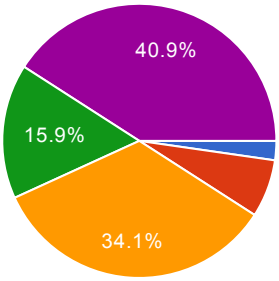
janeadriana88@gmail.com
Edit this form

44 responses

View all responses Publish analytics

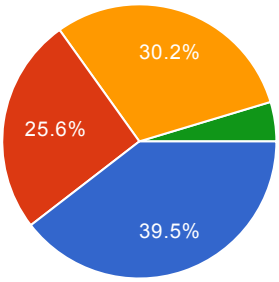
Summary

Qual a sua experiência como DBA?



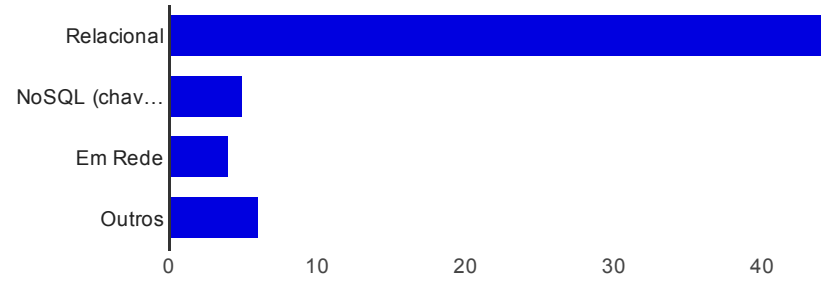
Menos de 1 ano	**1**	2.3%
Menos de 2 anos	**3**	6.8%
Entre 2 anos e 5 anos	**15**	34.1%
Entre 5 e 10 anos	**7**	15.9%
Acima de 11 anos	**18**	40.9%

Você desenvolveu sua experiência como DBA trabalhando em que tipos de organizações/instituições?



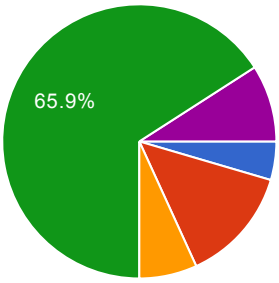
Empresa Privada Nacional	**17**	38.6%
Empresa Privada Multinacional	**11**	25%
Órgão Público	**13**	29.5%
Outras	**2**	4.5%

Você tem experiência na administração de quais tipos/modelos de banco de dados?



NoSQL (chave-valor, família de colunas, grafo, etc)	**5**	11.4%
Em Rede	**4**	9.1%
Outros	**6**	13.6%

Qual o quantitativo de funcionários da organização/instituição que você trabalha atualmente?

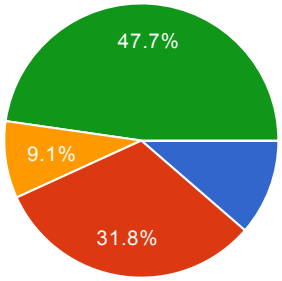


Menos de 100	**2**	4.5%
Entre 100 e 300	**6**	13.6%
Entre 301 e 500	**3**	6.8%
Acima de 501	**29**	65.9%
Não sei dizer	**4**	9.1%

Qual a plataforma de banco de dados utilizada na organização/instituição que você trabalha atualmente é:

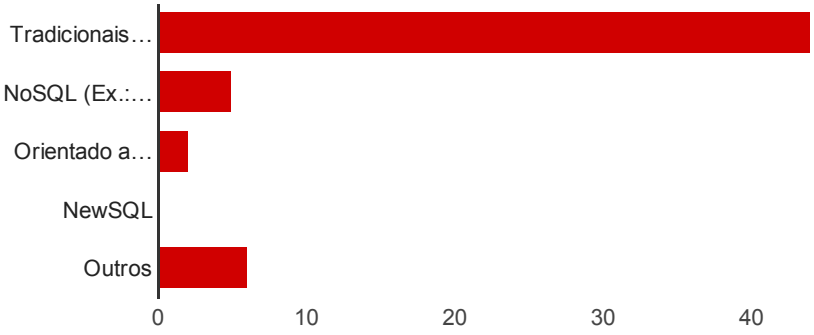
Infraestrutura própria (servidores físicos, virtuais, etc)	**43**	97.7%
Infraestrutura em Nuvem (Cloud Platforma)	**7**	15.9%
Outras	**0**	0%

Qual é a aproximadamente o quantitativo de bancos de dados existentes na organização/instituição que você trabalha?



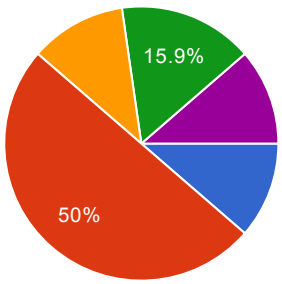
Menos de 10	**5**	11.4%
Entre 11 e 50	**14**	31.8%
Entre 51 e 100	**4**	9.1%
Acima de 101	**21**	47.7%
Não sei dizer	**0**	0%

Quais são os modelos de bancos de dados existentes na organização/instituição que você trabalha?



Tradicionais RDBMS (Ex. Oracle, SQLSERVER, etc)	**44**	100%
NoSQL (Ex.: MongoDB, Cassandra, etc)	**5**	11.4%
Orientado a objetos	**2**	4.5%
NewSQL	**0**	0%
Outros	**6**	13.6%

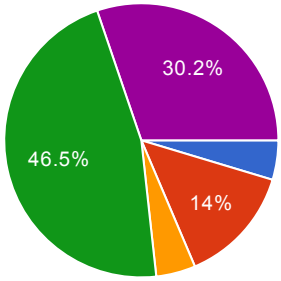
Quantos administradores de bancos de dados atuam na organização/instituição que você trabalha?



Menos de 2	**5**	11.4%
Entre 2 e 5	**22**	50%
Entre 6 e 10	**5**	11.4%

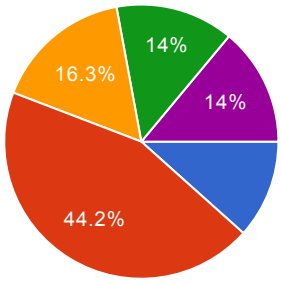
Entre 11 e 20	**7**	15.9%
Acima de 21	**5**	11.4%

Qual é aproximadamente o tamanho do maior banco de dados da organização/instituição que você trabalha?



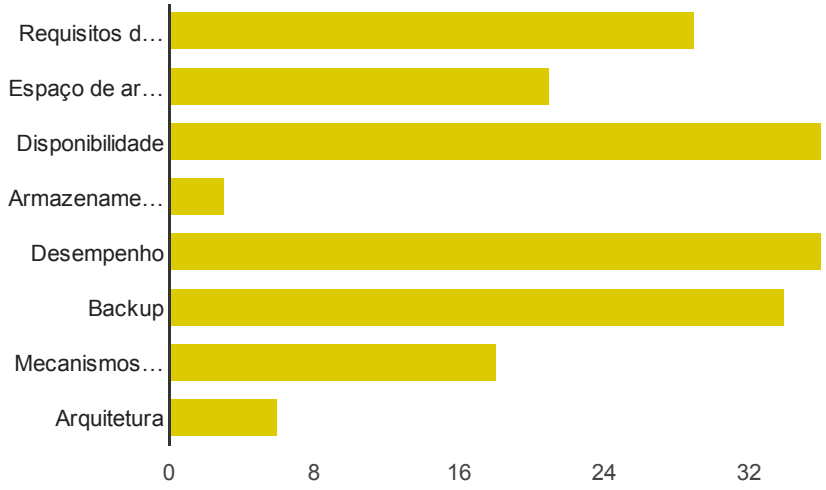
Menos de 100GB	**2**	4.5%
Entre 101 e 500GB	**6**	13.6%
Entre 501GB e 1TB	**2**	4.5%
Entre 1TB e 20TB	**20**	45.5%
Acima de 20TB	**13**	29.5%

Qual é aproximadamente o quantitativo de mudanças estruturais (execução de comandos DDL, alteração de esquema, objetos, etc) você realiza durante uma semana, na organização/instituição em que você trabalha?



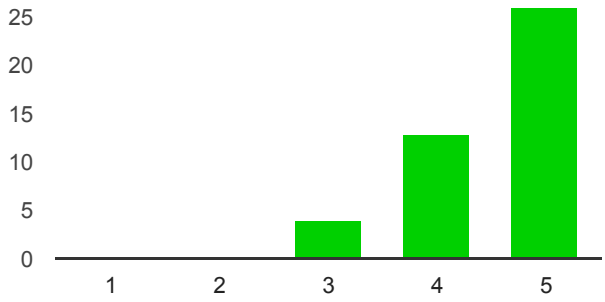
Menos de 2	**5**	11.4%
Entre 1 e 10	**19**	43.2%
Entre 11 e 20	**7**	15.9%
Entre 21 e 40	**6**	13.6%
Acima de 40	**6**	13.6%

Qual(is) são as maiores preocupações da sua organização/instituição em termos dos bancos de dados que ela possui?



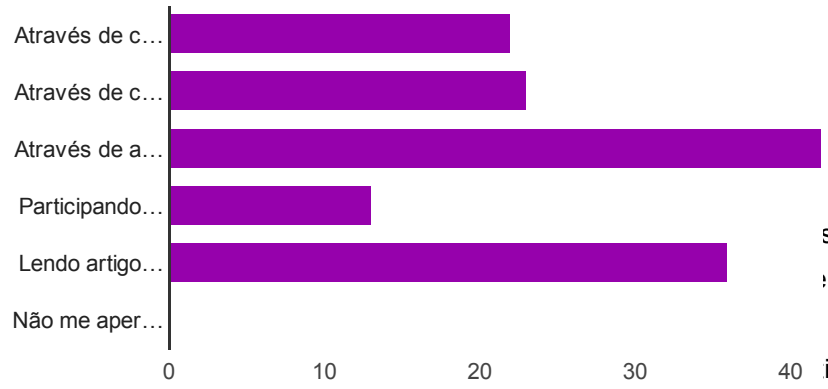
Desempenho	**36**	81.8%
Backup	**34**	77.3%
Mecanismos ágeis para recuperação de desastres	**18**	40.9%
Arquitetura	**6**	13.6%

No desenvolvimento de suas atividades como DBA, com que frequência você utiliza a linguagem SQL como uma ferramenta de administração do banco de dados?



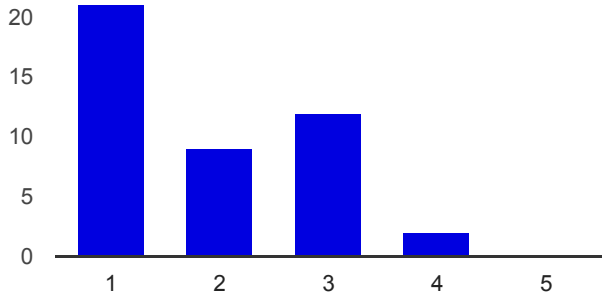
Raramente:	1	**0**	0%
	2	**0**	0%
	3	**4**	9.3%
	4	**13**	30.2%
Muito Frequentemente:	5	**26**	60.5%

Como você aperfeiçoa seu conhecimento e capacitação na área de banco de dados e novas tecnologias associadas ao assunto?



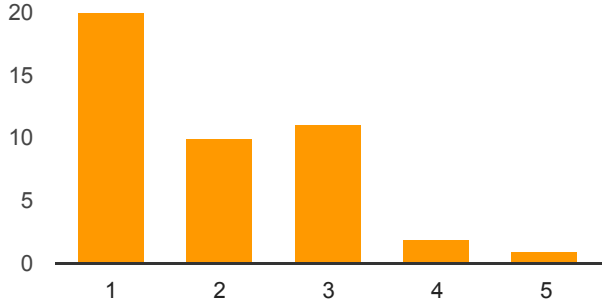
stituição em que trabalho	**22**	50%
realizo por conta própria	**23**	52.3%
Através de auto-estudo	**42**	95.5%
icipando de conferências	**13**	29.5%
Lendo artigos e pesquisando em fóruns	**36**	81.8%
Não me aperfeiçoo	**0**	0%

Você tem conhecimentos e prática em relação a BigData?



Pouco:	1	**21**	47.7%
	2	**9**	20.5%
	3	**12**	27.3%
	4	**2**	4.5%
Muito Conhecimento:	5	**0**	0%

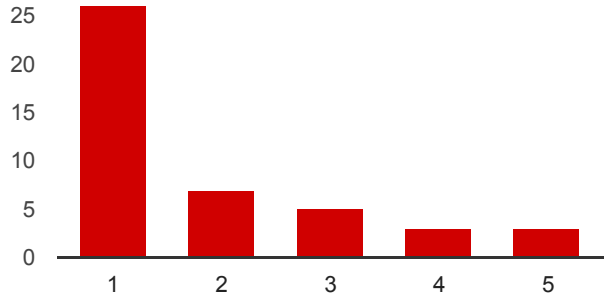
Você está familiarizado com o conceito de Banco de Dados NoSQL?



Nenhum conhecimento:	1	**20**	45.5%
	2	**10**	22.7%

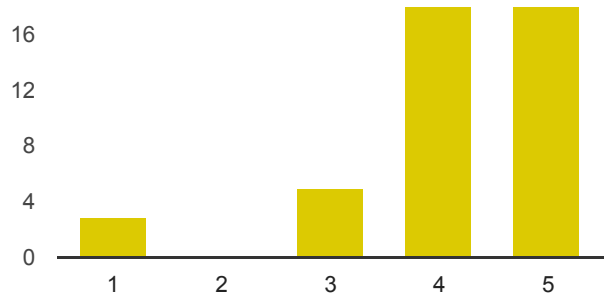
	3	**11**	25%
	4	**2**	4.5%
Muito conhecimento:	5	**1**	2.3%

Você como DBA incentiva a instituição/organização em conhecer novos modelos de bancos de dados, tais como NoSQL?



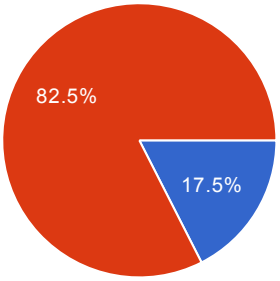
Muito Pouco:	1	**26**	59.1%
	2	**7**	15.9%
	3	**5**	11.4%
	4	**3**	6.8%
Insistentemente:	5	**3**	6.8%

Você considera que a existência de diferentes SGBDs e modelos de banco de dados, alinhada as peculiaridades de cada um, exige um alto nível de especialização dos DBAs, visto que torna a administração complexa e com necessidade de capacitação/estudos constantes devido as diferenças na arquitetura e comandos a serem empregados?



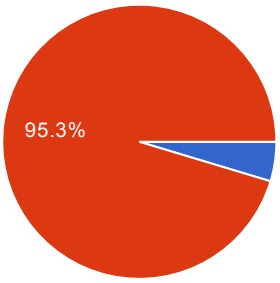
Baixo Nível de Especialização:	1	**3**	6.8%
	2	**0**	0%
	3	**5**	11.4%
	4	**18**	40.9%
Alto Nível de Especialização:	5	**18**	40.9%

No seu entendimento, a existência de uma ferramenta ou interface que possibilite de forma fácil, a administração de diferentes modelos de bancos de dados tais como relacionais e NoSQL é um assunto encorajador para os DBAs?



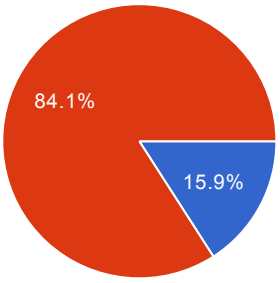
Não	**7**	17.5%
Sim	**33**	82.5%

Como DBA você considera a existência de uma linguagem de consulta de alto nível (tal qual SQL padrão para os bancos de dados relacionais) fundamental para a administração de um banco de dados?



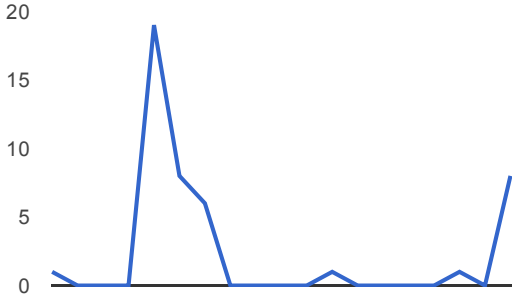
Não	**2**	4.7%
Sim	**41**	95.3%

Você apoiaria o desenvolvimento de uma camada intermediária (middleware) que possibilitasse aos DBAs utilizar comandos DDL da linguagem SQL padrão para diferentes bancos de dados?



Não	**7**	15.9%
Sim	**37**	84.1%

Number of daily responses



Apêndice C

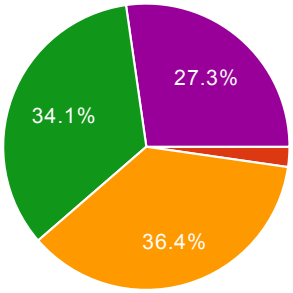
Resultado da Pesquisa-Enquete no Ano de 2016

44 responses

View all responses Publish analytics

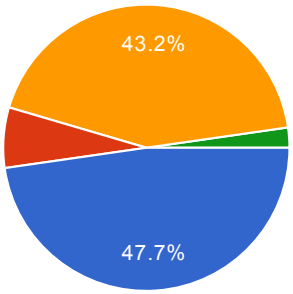
Summary

Qual a sua experiência como DBA?



Menos de 1 ano	**0**	0%
Menos de 2 anos	**1**	2.3%
Entre 2 anos e 5 anos	**16**	36.4%
Entre 5 e 10 anos	**15**	34.1%
Acima de 11 anos	**12**	27.3%

Você desenvolveu sua experiência como DBA trabalhando em que tipos de organizações/instituições?

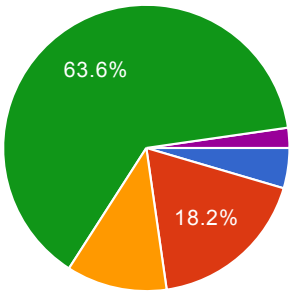


Empresa Privada Nacional	**21**	47.7%
Empresa Privada Multinacional	**3**	6.8%
Órgão Público	**19**	43.2%
Outras	**1**	2.3%

Você tem experiência na administração de quais tipos/modelos de banco de dados?

Outros	**3**	6.8%

Qual o quantitativo de funcionários da organização/instituição que você trabalha atualmente?

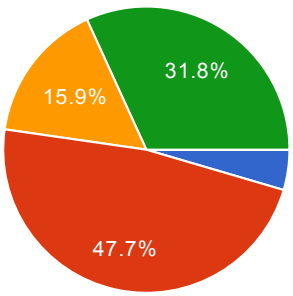


Menos de 100	**2**	4.5%
Entre 100 e 300	**8**	18.2%
Entre 301 e 500	**5**	11.4%
Acima de 501	**28**	63.6%
Não sei dizer	**1**	2.3%

Qual a plataforma de banco de dados utilizada na organização/instituição que você trabalha atualmente é:

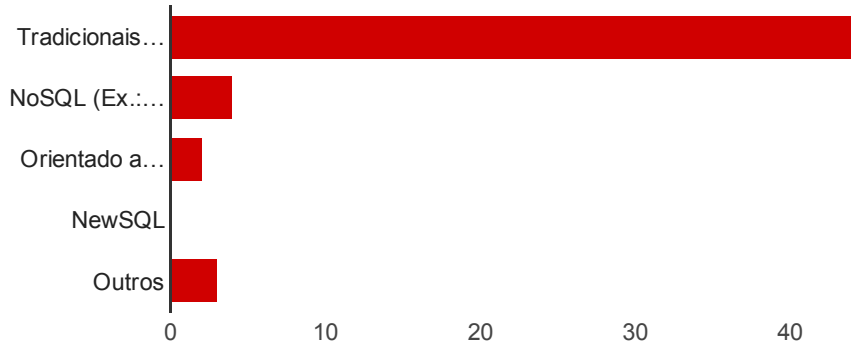
Infraestrutura própria (servidores físicos, virtuais, etc)	**44**	100%
Infraestrutura em Nuvem (Cloud Platforma)	**8**	18.2%
Outras	**1**	2.3%

Qual é a aproximadamente o quantitativo de bancos de dados existentes na organização/instituição que você trabalha?



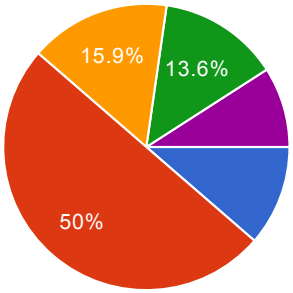
Menos de 10	**2**	4.5%
Entre 11 e 50	**21**	47.7%
Entre 51 e 100	**7**	15.9%
Acima de 101	**14**	31.8%
Não sei dizer	**0**	0%

Quais são os modelos de bancos de dados existentes na organização/instituição que você trabalha?



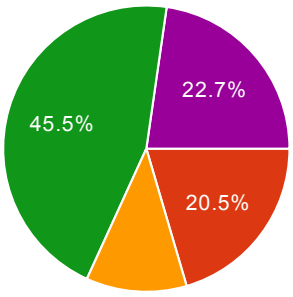
Tradicionais RDBMS (Ex. Oracle, SQLSERVER, etc)	**44**	100%
NoSQL (Ex.: MongoDB, Cassandra, etc)	**4**	9.1%
Orientado a objetos	**2**	4.5%
NewSQL	**0**	0%
Outros	**3**	6.8%

Quantos administradores de bancos de dados atuam na organização/instituição que você trabalha?



Menos de 2	**5**	11.4%
Entre 2 e 5	**22**	50%
Entre 6 e 10	**7**	15.9%
Entre 11 e 20	**6**	13.6%
Acima de 21	**4**	9.1%

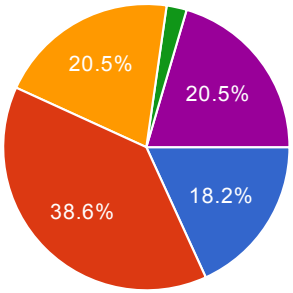
Qual é aproximadamente o tamanho do maior banco de dados da organização/instituição que você trabalha?



Menos de 100GB	**0**	0%
Entre 101 e 500GB	**9**	20.5%
Entre 501GB e 1TB	**5**	11.4%
Entre 1TB e 20TB	**20**	45.5%
Acima de 20TB	**10**	22.7%

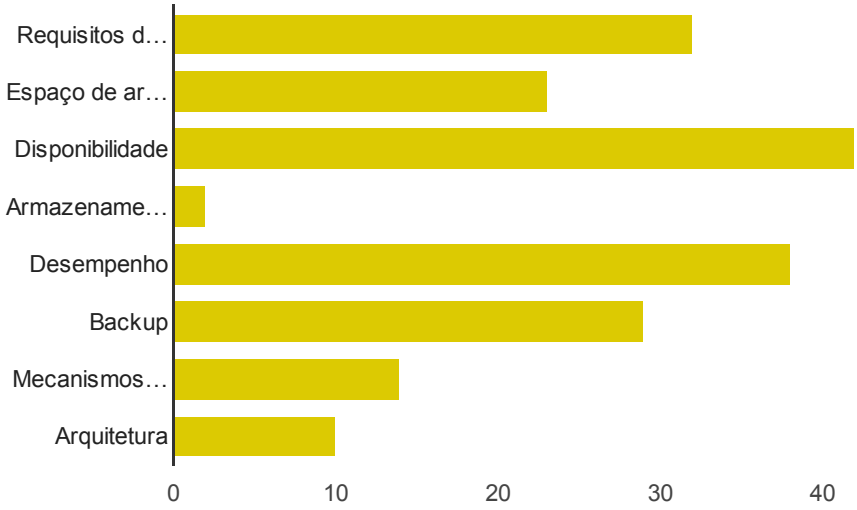
Qual é aproximadamente o quantitativo de mudanças estruturais (execução de comandos DDL, alteração de esquema, objetos, etc) você realiza durante uma semana, na organização/instituição em que você trabalha?

Menos de 2	**8**	18.2%



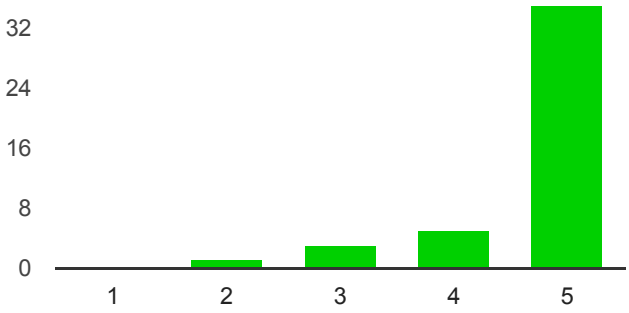
Entre 1 e 10	**17**	38.6%
Entre 11 e 20	**9**	20.5%
Entre 21 e 40	**1**	2.3%
Acima de 40	**9**	20.5%

Qual(is) são as maiores preocupações da sua organização/instituição em termos dos bancos de dados que ela possui?



Requisitos de Segurança	**32**	72.7%
Espaço de armazenamento	**23**	52.3%
Disponibilidade	**42**	95.5%
Armazenamento em nuvem	**2**	4.5%
Desempenho	**38**	86.4%
Backup	**29**	65.9%
Mecanismos ágeis para recuperação de desastres	**14**	31.8%
Arquitetura	**10**	22.7%

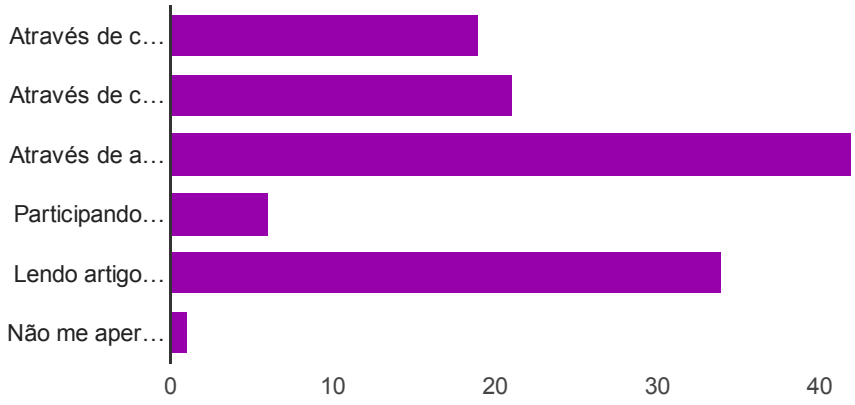
No desenvolvimento de suas atividades como DBA, com que frequência você utiliza a linguagem SQL como uma ferramenta de administração do banco de dados?



Raramente: 1	**0**	0%
2	**1**	2.3%

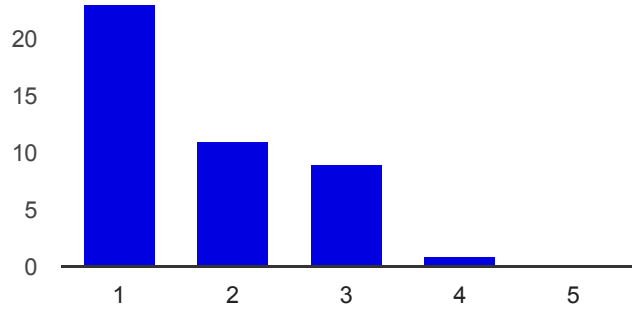
	3	**3**	6.8%
	4	**5**	11.4%
Muito Frequentemente:	5	**35**	79.5%

Como você aperfeiçoa seu conhecimento e capacitação na área de banco de dados e novas tecnologias associadas ao assunto?



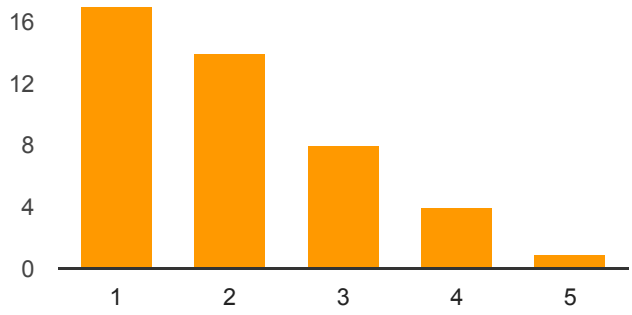
Através de cursos e treinamentos disponibilizados na organização/instituição em que trabalho	**19**	43.2%
Através de cursos e treinamentos que realizo por conta própria	**21**	47.7%
Através de auto-estudo	**42**	95.5%
Participando de conferências	**6**	13.6%
Lendo artigos e pesquisando em fóruns	**34**	77.3%
Não me aperfeiçoo	**1**	2.3%

Você tem conhecimentos e prática em relação a BigData?



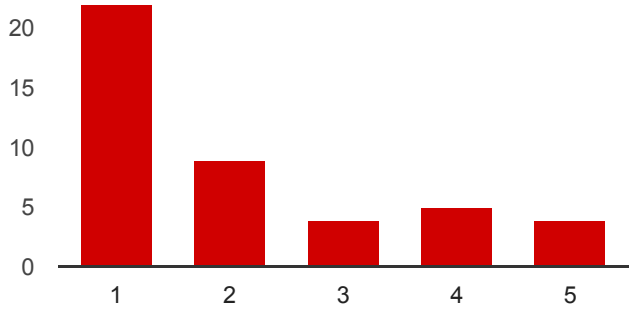
Pouco:	1	**23**	52.3%
	2	**11**	25%
	3	**9**	20.5%
	4	**1**	2.3%
Muito Conhecimento:	5	**0**	0%

Você está familiarizado com o conceito de Banco de Dados NoSQL?



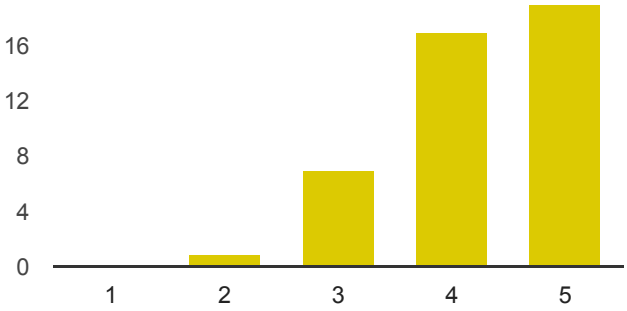
Nenhum conhecimento:	1	**17**	38.6%
	2	**14**	31.8%
	3	**8**	18.2%
	4	**4**	9.1%
Muito conhecimento:	5	**1**	2.3%

Você como DBA incentiva a instituição/organização em conhecer novos modelos de bancos de dados, tais como NoSQL?

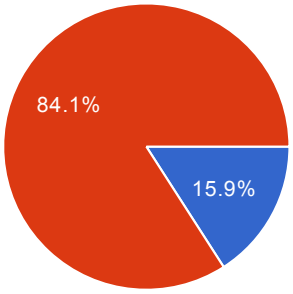


Muito Pouco:	1	**22**	50%
	2	**9**	20.5%
	3	**4**	9.1%
	4	**5**	11.4%
Insistentemente:	5	**4**	9.1%

Você considera que a existência de diferentes SGBDs e modelos de banco de dados, alinhada as peculiaridades de cada um, exige um alto nível de especialização dos DBAs, visto que torna a administração complexa e com necessidade de capacitação/estudos constantes devido as diferenças na arquitetura e comandos a serem empregados?

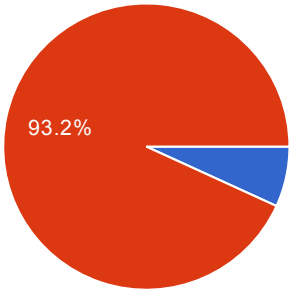


No seu entendimento, a existência de uma ferramenta ou interface que possibilite de forma fácil, a administração de diferentes modelos de bancos de dados tais como relacionais e NoSQL é um assunto encorajador para os DBAs?



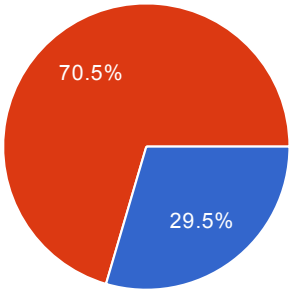
Não	**7**	15.9%
Sim	**37**	84.1%

Como DBA você considera a existência de uma linguagem de consulta de alto nível (tal qual SQL padrão para os bancos de dados relacionais) fundamental para a administração de um banco de dados?



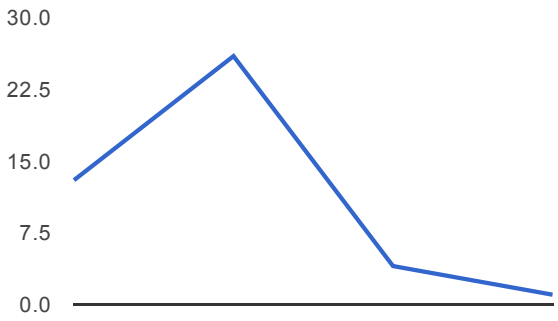
Não	**3**	6.8%
Sim	**41**	93.2%

Você apoiaria o desenvolvimento de uma camada intermediária (middleware) que possibilitasse aos DBAs utilizar comandos DDL da linguagem SQL padrão para diferentes bancos de dados?



Não	**13**	29.5%
Sim	**31**	70.5%

Number of daily responses


30.0
22.5
15.0
7.5
0.0