UNIVERSIDADE FEDERAL DO PIAUÍ
CAMPUS SENADOR HELVÍDIO NUNES DE BARROS
CURSO BACHARELADO EM SISTEMAS DE INFORMAÇÃO

ANÁLISE EXPERIMENTAL ENTRE AS TECNOLOGIAS RELACIONAIS E NÃO RELACIONAIS, UTILIZANDO OS SGBDS MYSQL E APACHE CASSANDRA

MARIA VIVIANE DE ARAÚJO SOUSA

PICOS - PI
2016

MARIA VIVIANE DE ARAÚJO SOUSA

ANÁLISE EXPERIMENTAL ENTRE AS TECNOLOGIAS RELACIONAIS E NÃO RELACIONAIS, UTILIZANDO OS SGBDS MYSQL E APACHE CASSANDRA

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Sistemas de Informação do Campus Senador Helvídio Nunes de Barros (CSHNB) da Universidade Federal do Piauí (UFPI) como parte dos requisitos para obtenção do Grau de Bacharel em Sistemas de Informação, sob orientação do Professor Francisco das Chagas Imperes Filho.

PICOS - PI
2016

FICHA CATALOGRÁFICA
Serviço de Processamento Técnico da Universidade Federal do Piauí
Biblioteca José Albano de Macêdo

S725a Sousa, Maria Viviane de Araújo.

Análise experimental entre as tecnologias relacionais e não relacionais, utilizando os SGBDS MySQL e apache cassandra / Maria Viviane de Araújo Sousa. – 2016.

CD-ROM : il.; 4 ¾ pol. (56 f.)

Monografia(Bacharelado em Sistemas de Informação) – Universidade Federal do Piauí, Picos, 2016.

Orientador(A): Profº. Francisco das Chagas Imperes Filho

1. Análise experimental de Dados. 2. Banco de Dados Relacionais-Não Relacionais. 3. Apache Cassandra. I. Título.

CDD 005.3

ANÁLISE EXPERIMENTAL ENTRE AS TECNOLOGIAS RELACIONAIS E NÃO RELACIONAIS, UTILIZANDO OS SGBDS MYSQL E APACHE CASSANDRA

MARIA VIVIANE DE ARAÚJO SOUSA

Monografia Aprovado como exigência parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Data de Aprovação

Picos – PI, 18 de fevereiro de 2016

Francisco das Chagas Imperes Filho
Profª. Esp. Francisco das Chagas Imperes Filho
Orientador

Allan J. R. Gonçalves
Prof. Esp. Allan Jheyson Ramos Gonçalves
Membro

Patrícia Vieira da Silva Barros
Prof. Ma. Patricia Vieira da Silva Barros
Membro

Dedico este trabalho à Deus, pela sua força e proteção diária a mim concebida, e a minha família, em especial aos meus pais e irmãos, Vantuil Hildebrando, Valdeci, Valquécia e Vândeson por acreditarem sempre no meu potencial e apostarem em mim. O anseio em atender as vossas expectativas sempre me impulsionou.

AGRADECIMENTOS

Agradeço primeiramente a Deus, meu porto, fonte de fé e esperança. Sem ele era com se minha vida não tivesse prumo, regras, limites. É a minha fé e confiança que fazes querer ser uma pessoa melhor.

Ao meu orientador Francisco das Chagas Imperes Filho, que em muito me aconselhou, me dedicando tempo e conhecimento nessa empreitada.

A Universidade Federal do Piauí e aos que lhe compõem, em especial aos professores Patrícia Medyna, Alcilene Dalilia, Leonardo Pereira, Dennis Savio e Ivenilton Alexandre, de vocês absorvi mais que o conhecimento ministrado. Vocês me ensinaram a lidar com a vida.

Aos servidores da UFPI, Ivanildo, Giorge, Dona Puri, Verinha, Nice e Sr. Paulo, Acreditem! Eu tenho um vínculo afetivo com cada um de vocês e irei levar comigo tudo que vivemos nesses anos.

A minha irmã de alma, Lívia Maria Alencar Rocha, nossa semelhança sempre nos surpreendia e a cada situação nossa amizade se fortalecia. Tenho por você um sentimento enorme e sei que a reciproca confere.

Aos amigos verdadeiros que a vida acadêmica me presenteou, Kaio, Laronso, Gilberlon, Bell, Rai, Givanaldo, Willamys e Israel. Contarei para os meus filhos todas as coisas que vocês me fizeram passar.

Ao Janderson, quem me acompanhou, ouviu e me ajudou a ter discernimento para ultrapassar todas as barreiras, mesmo com a distância. Que Deus nos permita compartilhar mais que o mesmo plano e a mesma história.

Aos meus tios e primos, que sempre foram exemplo de perseverança e alegria. Primos, é a nossa união que me dá forças para seguir os passos dos nossos avós, e ser exemplo como eles foram.

Ao Otávio Santana, que respondeu prontamente a cada pergunta sobre o Cassandra. A humildade que conheci em você, demonstra que na Tecnologia tem sempre alguém disposto a ajudar.

A vocês citados, meu muito obrigado!

“É muito melhor aventurar-se em busca de conquistas grandiosas, mesmo expondo-se ao fracasso, do que alinhar-se com os pobres de espírito, que nem gozam muito nem sofrem muito, porque vivem numa penumbra cinzenta, onde não conhecem nem vitória nem derrota.”

(Theodore Roosevelt).

RESUMO

O armazenamento e manipulação de dados foram o foco desde a construção dos primeiros computadores, já que o desenvolvimento dessas máquinas deu-se devido a essas necessidades. Atualmente os Bancos de Dados são pontos cruciais em qualquer organização, pois neles são armazenadas as informações, bem mais precioso das empresas. Os Bancos de Dados são mecanismos que possibilitam o rápido acesso às informações e conduzem a uma correta tomada de decisões por parte dos administradores de uma instituição. Diante deste contexto, o trabalho ora produzido, realizou uma análise experimental de desempenho entre as tecnologias de armazenamento e manutenção de dados Relacional e Não Relacional, com ênfase nos Sistemas Gerenciadores de Banco de Dados (SGBD) MySQL e Apache Cassandra, respectivamente.

Palavras-chave: Análise experimental de dados; Banco de Dados Relacionais; Banco de Dados Não Relacionais.

ABSTRACT

The storage and handling of data have been the focus since the construction of the first computers, since the development of these machines took place due to these needs. Currently Data Bases are the heart of any organization, because the information in them, most precious of companies are stored. Data Bases are mechanisms that enable quick access to information and lead to a correct decision-making by administrators of an institution. Given this context, the work done now produced an experimental analysis of performance between storage technologies and maintaining relational and non-relational data, with emphasis on Data Base Management Systems (DBMS) and MySQL Apache Cassandra, respectively.

Keywords*: Analysis of experimental data; Relational Data Bases; There Relational Data Bases.*

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE GRÁFICOS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
ACID	Atomicidade, Consistência, Isolamento e Durabilidade
ANSI	*American National Standards Institute*
BASE	*Basically Available*, *Soft State* e *Eventual Consistency*
BD	*Data Base*
BDR	Banco de Dados Relacional
CAP	*Consistency*, *Available* e *Partation*
CEP	Código de Endereçamento Postal
CQL	*Cassandra Query Language*
DBA	Data Base Administration
DCL	*Data Control Language*
DDL	*Data Definition Language*
DML	*Data Manipulation Language*
DQL	*Data Query Language*
FK	*Foreign Key*
GNU-GPL	*General Public Licence*
IP	*Internet Protocol*
ISO	*International Standards Organization*
JSON	*JavaScript Object Nonation*
LP	Linguagem de Programação
NoSQL	*Not Only* SQL
ODBC	*Open Database Connectivity*
PDF	*Portable Document Format*
PHP	*PHP Hypertext Processor*
PK	*Primary Key*
SGBD	Sistema de Gerenciamento de Banco de Dados

SGBDR	Sistema de Gerenciamento de Banco de Dados Relacional
SQL	*Structured Query Language*
TCP	*Transmission Control Protocol*

SUMÁRIO

1 INTRODUÇÃO

A sociedade depara-se hoje, com uma quantidade exorbitante de leituras e gravações de dados na grande rede de computadores, a *Internet*. Os sistemas *web* apresentam-se no cenário mundial como os responsáveis pelo alto tráfego de informações, pela demanda de performance e disponibilidade cada vez melhores de acesso.

Segundo Ramakrishnan e Gehrke (2008), um *Data Base* ou Banco de Dados (BD), é uma coleção de dados que, tipicamente, descreve as atividades de uma ou mais organizações relacionadas. Com base nisso, as informações são tidas como um dos domínios mais importantes das organizações e devem ser gerenciadas com segurança e consistência. Aspectos esses assegurados pelos Sistemas de Gerenciamento de Bancos de Dados.

Os SGBDs relacionais perpetuaram-se por um longo período como unanimidade no que tange a forma de armazenamento de dados (HEUSER, 2009). Esse fato se deve às vantagens com relação a segurança de armazenamento, controle de concorrência, dentre outras.

Baseados na álgebra relacional, os bancos de dados relacionais oferecem a seus usuários verificação e garantias de integridade, controle de concorrência, recuperação de falhas, segurança e otimização de consultas. Todos esses benefícios fizeram com que essa tecnologia se mantivesse em destaque até os tempos atuais.

Entretanto considerando o grande volume de informações que as organizações vêm lidando, os SGBDs relacionais se apresentam de forma ineficientes em alguns casos, como por exemplo: redes sociais, grandes portais de informação, e a própria natureza dos novos sistemas executados na *web*. Por serem estruturadas, as tecnologias relacionais apresentam um gargalho no âmbito da distribuição de bases por vários servidores. Conciliar o modelo relacional com a demanda por escalabilidade, torna-se cada vez mais complexo, já que a medida que aumenta o número de dados em trafego na internet tende a cair o tempo de resposta de aplicações *web* (SADALAGE; FOWLER, 2013).

Buscando uma maior capacidade de escalabilidade e disponibilidade, desenvolvedores adotam uma forma de armazenamento relativamente nova e flexível, baseada em modelos distintos e direcionados aos vários tipos de aplicação. Essa forma de armazenamento não implementa a interface *Structured Query Language* (SQL) e é comumente conhecida como banco de dados não-relacionais ou *Not Only SQL* (NoSQL).

Devido a escalabilidade horizontal e os modelos de distribuição de dados a ela associada no *cluster* (conjunto de nós em rede) algumas propriedades asseguradas pelos SGBDs

relacionais não são possíveis nos não relacionais, causando questionamentos quanto a eficiência desse método de armazenamento, por alguns pesquisadores como, por exemplo: Tiware (2011), Pereira (2014), McCreary e Kelly (2014).

1.1 Objetivo

Estabelecer um comparativo entre as tecnologias relacionais e não relacionais, norteado por pesquisa bibliográfica e por testes realizados com o MySQL e o Apache Cassandra, ambos SGBDs, relacionais e não relacionais, respectivamente.

1.2 Metodologia do Trabalho

A abordagem utilizada para o desenvolvimento do trabalho foi a análise de material bibliográfico e de experimentos, empregados para demonstrar a proposta do trabalho. Para a realização dos experimentos práticos foi utilizada a base de endereços do Brasil, mantida pelos Correios e composta por 1.010.976 de registros distribuídos em quatro tabelas.

Os experimentos foram realizados em uma única máquina, sendo esta de propriedade do autor e descrita mais precisamente no decorrer do trabalho. Por meio dos experimentos, buscou-se manipular e controlar a variável tempo, demonstrando o custo de tempo gasto por cada tecnologia na execução de *scripts*. Operações de inserção, pesquisa e atualização foram as categorias de consultas utilizadas nos *scripts* de testes.

1.3 Organização do Trabalho

O presente trabalho está organizado em 6 capítulos. O capítulo 2, intitulado Referencial Teórico, apresenta o embasamento teórico para produção deste material. Nele são citados trabalhos e conceitos de diversos autores para auxiliar na compreensão da pesquisa proposta e, por consequência, subsidiar meios para efetivação deste esboço científico. No capítulo 3, nomeado Tecnologias de Gerenciamento Estudadas, serão apresentados os SGBDs utilizados para os experimentos, considerando características e pontos que os diferem. No capítulo 4, Banco de dados: Relacional ou Não Relacional?, são apresentados questionamentos sobre qual tecnologia utilizar e em qual situação utilizar. No capítulo 5 intitulado de Experimentos feitos com o Mysql e o Apache Cassandra, são descritos os *scripts* utilizados nas

instruções de inserção, pesquisa e atualização, assim como os resultados dos experimentos. Por fim, o capítulo 5, intitulado Considerações Finais, como o próprio nome sugere, contempla as considerações finais e indicações de trabalhos futuros. Além dos capítulos mencionados, a investigação contém seções para Referências Bibliográficas e Apêndices.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta o embasamento teórico dos temas relacionados à pesquisa e referência fontes de autores para os temas abordados que incluem definições de bancos de dados relacionais e não relacionais e, por fim, trabalhos referentes à temática desta apreciação.

2.1 Banco de Dados Relacional

A importância das informações como um ativo organizacional, leva a uma maior preocupação com os bancos de dados, ambientes onde os dados são armazenados. Elmasri e Navathe (2005) definem banco de dados como uma coleção de dados relacionados, onde estes são fatos que podem ser gravados e que têm um significado implícito. De modo geral, os bancos de dados guardam informações sobre um determinado escopo delimitado por um negócio, como exemplo, dados sobre transações, clientes e fornecedores que retratam a realidade do dia-a-dia de uma organização.

Os pesquisadores acrescentam que um banco de dados é uma coleção lógica e coerente com algum significado inerente (ELMASRI; NAVATHE, 2005). Dessa forma, dados reunidos aleatoriamente não podem ser interpretados como um banco de dados. A Figura 1 representa um sistema computacional, tendo o banco de dados como um dos seus pontos fundamentais.

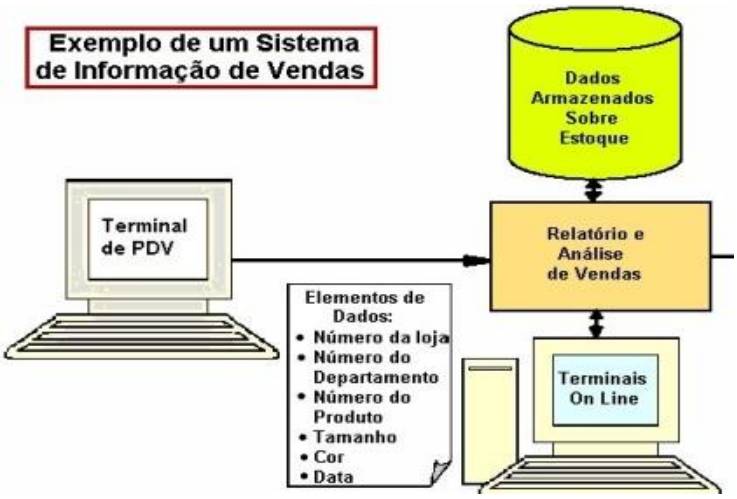

Figura 1- Esquema de um Sistema Computacional, destacando a base de dados.
Fonte: Heuser, 2009.

Um banco de dados pode ser gerado e mantido manualmente, assim como pode ser de qualquer tamanho e de complexidade variável, porém não é a forma mais adequada para gerir as informações de uma organização. Segundo Elmasri e Navathe (2005) um banco de dados computadorizado pode ser criado e mantido tanto por um grupo de aplicativos escritos especialmente para essa tarefa como por um SGBD. É importante mencionar que no armazenamento de dados por meio de gravação de arquivos, são escritos programas específicos, direcionados a realizar certos tipos de atividades sobre os dados, cada usuário do banco mantém seus dados separados e os programas que manipulam esses arquivos.

Os Sistemas Gerenciadores de Banco de Dados Relacionais (SGBDR), baseados no modelo relacional, surgiram como uma forma de administrar grandes volumes de dados das organizações de maneira segura e fácil. Aspectos estes difíceis de serem garantidos com o armazenamento em sistemas de arquivos, método usado anteriormente pelas empresas quando necessitavam armazenar dados sobre sua área de atuação.

2.1.1 Modelo Relacional

O modelo de Banco de Dados Relacional (BDR) foi introduzido em 1970 por *Ted Cood*, em um artigo que descrevia treze novas regras nas quais o novo modelo era baseado (ABRIL, 1974). Esse modelo tem como alicerce a álgebra relacional e possui uma linguagem padrão de manipulação de dados, a SQL.

O modelo relacional, utilizado pelos BDR, representa o banco de dados como uma coleção de relações, essas últimas se pensadas como uma tabela de valores, cada linha é formada por colunas e representa uma coleção de valores de dados relacionados (ELMASRI; NAVATHE, 2005). Dessa forma, cada tabela pode ser considerada como um conjunto de registros que descrevem um ente, onde uma coluna pode se referir a outra linha em uma tabela diferente, o que constitui os relacionamentos entre as tabelas explícitos pelos atributos chaves.

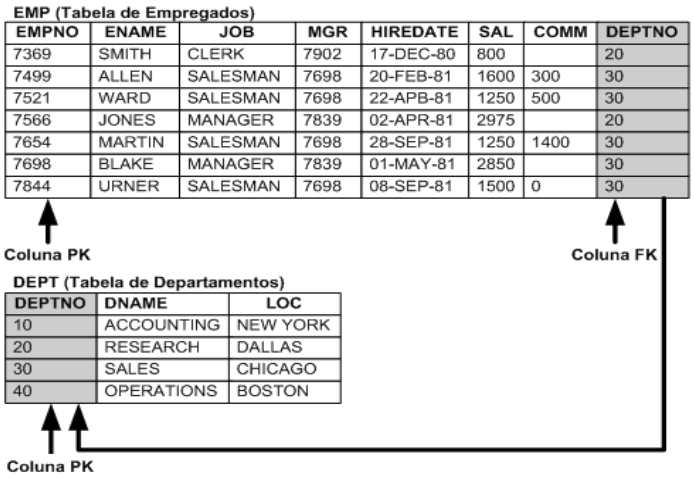

Figura 2- Representação de uma relação em um BDR mantido pelas chaves.
Fonte: <http://www.devmedia.com.br/arquitetura-de-um-relacionamento/2500796>. Acesso nov. 2015.

Uma linha ou tupla de uma tabela é construída por várias colunas e identificada por meio de um campo que não se repete, a chave primária ou *primary key* (PK). Ramakrishnan e Gehrke (2008) dissertam acerca das chaves dizendo que a PK em uma tabela será a chave estrangeira ou *foreign key* (FK) em outra com a qual se relacione. É por meio da FK que as tabelas estabelecem seus relacionamentos, sendo possível retornar dados de mais de uma tabela em uma única consulta no banco. Estas características são explicitadas na Figura 2.

A FK na relação de referência deve corresponder a PK da relação referenciada, tanto em valor quanto em tipo de dado. Para tanto, toda a estruturação do modelo relacional, seja ela de tipos de dados, relacionamentos ou restrições de chave, são armazenadas em *schemas* (esquemas) de banco de dados. Definido pelos autores como:

> Um *schema* é a descrição do banco de dados, definido no desenvolver do projeto de banco de dados e não se espera que seja alterado frequentemente, refere-se à organização dos dados e de como a base foi construída, podendo ser constituído de vários elementos, tabelas, sentenças de restrição, relacionamentos e outros, suportados pelo gerenciador (RAMAKRISHNAN; GEHRKE, 2008).

Os gerenciadores de bases relacionas, suportam e são baseados na criação de *schemas*, eles ajudam no detalhamento do banco de dados e contribuem na recuperação de falhas. Os *schemas* são estruturas criadas por comandos e podem restringir acesso a seus usuários.

2.1.2 *Structured Query Language*

A linguagem SQL foi projetada e implementada na *IBM Research* como uma interface para um sistema experimental de um BDR. Está evoluiu e decorrente da sua eficiência e

simplicidade chegou a ser padronizada pelo *American National Standards Institute* (ANSI) e *International Standards Organization* (ISO).

A SQL é uma linguagem de consulta estruturada de alto nível, usadas pelos BDR para definição, manipulação, controle e consulta dos dados em uma base (HEUSER, 2009). O padrão de consultas, é considerado um dos grandes pilares que alavancaram o sucesso dos SGBDRs, principalmente pela sua abrangência e eficiência na manipulação de dados. Formada por subconjuntos, a SQL possui algumas divisões, dissertadas por Macário e Baldo (2015) como:

- ✓ A Linguagem de Definição de Dados (DDL – *Data Definition Language*): suporta a criação, exclusão e modificação das definições das tabela e visões, criação e exclusão de índices e definição de restrições de integridade, abrange os comandos, *CREATE*, *ALTER* e *DROP*;
- ✓ A Linguem de Manipulação de Dados (DML – *Data Manipulation Language*): permite que usuários formulem consultas e insiram, excluam e modifiquem tuplas, suporta os comandos, *INSERT*, *UPDATE* e *DELETE*;
- ✓ A Linguagem de Controle de Dados (DCL - *Data Control Language*): permite que o DBA(*Administration Data BASE*), conceda ou remova privilégios dos usuários, abrange os comandos, *GRANT* e *REVOKE*;
- ✓ A Linguagem de Consulta de Dados (DQL - *Data Query Language*): permite que o usuário realize seleções de conjuntos de dados na base por meio de consultas, usando o comando *SELECT*.

Foram descritos apenas alguns dos comandos da linguagem, sendo estes a nível de tabela, tupla, controle de privilégios e consultas, respectivamente. No entanto, a SQL, é poderosa e completa, e possui outras especificações relevantes que de acordo com Ramakrishnan e Gehrke (2008), são citadas abaixo:

- ✓ Gatilhos e restrições de Integridade Avançada: são ações que são executadas pelo SGBD, sempre que alterações no banco de dados, satisfaçam a condições especificadas no gatilho (*triggers*), as restrições forçam a base a atender suas condições moldando o banco as necessidades do escopo;
- ✓ SQL Embutida e Dinâmica: permite que o código SQL seja chamado por meio de uma linguagem hospedeira;

- ✓ Execução Cliente-Servidor e Acesso a Banco de Dados Remoto: estes comandos controlam como um aplicativo cliente, pode se conectar a um servidor de banco de dados SQL, ou acessar dados de um banco de dados através de uma rede;
- ✓ Gerenciamento de Transações: diversos comandos permitem que um usuário controle explicitamente os aspectos da execução de uma transação;
- ✓ Recursos avançados: o padrão SQL:1999 inclui recursos a orientação a objetos, consultas recursivas, consultas de apoio a decisão, e também trata áreas emergências como mineração de dados, dados espaciais e gerenciamento de texto e de dados XML.

Neste contexto, a SQL é completa e, portanto, a linguagem mais amplamente usada pelas aplicações. Disponibiliza recursos para manipulação de dados em aplicações *desktop*, *web*, *mobile* e outros. Dentre tantas outras funcionalidades, a linguagem oferece um eficiente e seguro controle de transações que juntamente com os programas de gerenciamento de banco de dados asseguram o isolamento e integridade dos dados em transações.

2.1.3 Transações e Propriedades ACID

O termo transação, refere-se a uma execução de programa do usuário, vista pelo SGBD como uma série de operações de leituras e escritas (RAMAKRISHNAN; GEHRKE, 2008). Por exemplo, uma transferência de valores de uma conta para outra é uma transação consistindo de duas atualizações, uma para cada conta. A Figura 3 demonstra esse procedimento.

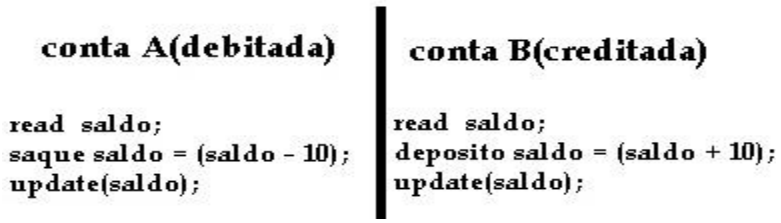

Figura 3 - Representação de uma operação de transferência bancária
Fonte: Autor.

Em sistemas multiusuários, o acesso ao banco de dados concorrentemente pode resultar em dados inconsistentes, isto é, levar o usuário a operar sobre dados incertos, acarretando em uma base inconsistente.

Dessa forma, o controle de transações é um aspecto bastante importante a considerar quando se pensa na performance e consistência de bases de dados implementadas num ambiente

de computação distribuída (MCCREARY; KELLY, 2014). Portanto, em ambientes distribuídos o custo em gerenciar transações é possivelmente alto, interferindo no tempo de resposta do usuário.

Por meio de comandos SQL e implementações de bloqueios dos gerenciadores, o banco se mantém consistente, mesmo com centenas de operações sendo realizadas paralelamente pelos usuários. Essas vantagens são implementadas pelos SGBDR, e asseguradas pelas propriedades ACID (acrônimo para Atomicidade, Consistência, Isolamento e Durabilidade), características de uma transação que devem ser conservadas de acordo com a necessidade do negócio, sendo indispensáveis em alguns casos e diminuto em outros.

Conforme Elmasri e Navathe (2005) as transações devem possuir algumas propriedades, chamadas propriedades ACID, e elas devem ser impostas pelo controle de concorrência e métodos de restauração do SGBD. As propriedades ACID são as seguintes:

- ✓ Atomicidade: uma transação é uma unidade atômica de processamento; ou ela será executada em sua totalidade ou não será de modo nenhum.
- ✓ Preservação de Consistência: uma transação será preservadora de consistência se sua execução completa fizer o banco de dados passar de um estado consistente para outro.
- ✓ Isolamento: uma transação deve ser executada como se estivesse isolada das demais. Isto é, a execução de uma transação não deve sofrer interferência de quaisquer outras transações concorrentes.
- ✓ Durabilidade ou Permanência: as mudanças aplicadas ao banco de dados por uma transação efetivada devem persistir no banco de dados. Essas mudanças não devem ser perdidas em razão de uma falha.

A segurança e a confiabilidade dos dados são fatores esperados pelas organizações que utilizam sistemas de informação em seus negócios. Desta forma, os SGBDs relacionais asseguram por meio da implementação das propriedades ACID, o controle de transações. Isso implica que, mesmo os usuários executando operações simultâneas sob os dados, esses se manterão consistentes, proporcionando as organizações confiabilidade em suas transações.

É importante ressaltar que a normalização aplicada nos BDR, é um caminho para que os SGBDs consigam assegurar as propriedades ACID e que a SQL tenha seus recursos potencialmente utilizados.

A necessidade de manter os dados de um banco consistente, quando estes são acessados ao mesmo tempo, é um dos aspectos que resultou no sucesso dos sistemas *web* e na

consolidação dos bancos relacionais. Sadalage e Fowler (2013), discorrem que o mecanismo transacional tem executado bem sua função de controlar a complexidade da concorrência.

No entanto, Lóscio *et al.* (2011), afirma que com a utilização maciça dos sistemas *web*, problemas de disponibilidade surgiram. Assim, a fim de atender aos requisitos destas aplicações, novas soluções para gerenciamento de dados começaram a ser propostas, como, por exemplo, os bancos de dados NoSQL.

2.2 Banco de Dados Não Relacional

Com o advento da *web 2.0* e a mudança de como os usuários passaram a vê-la, o número de aplicações rodando na rede aumentaram em um ritmo acelerado, ocasionando problemas como indisponibilidade de informações e perda de performance. Visando atender a demanda por disponibilidade de acesso às informações, os SGBDs não relacionais surgem como nova tecnologia e alternativa flexível, que apresenta suporte para lidar com o crescimento da *Internet.*

Os Bancos de Dados não Relacionais ganharam destaque em 2009, quando *Johan Oskarsson*, organizou um evento baseado em bancos de dados *open source* distribuídos (TIWARI, 2011). O termo já havia sido mencionado antes, no intuito de definir um novo padrão de armazenamento de dados que não fosse baseado na álgebra relacional e nem implementasse uma interface SQL. As tecnologias não relacionais surgiram como proposta de solução para o manejo de grandes volumes de dados na *web*, e possuem alguns outros pontos que lhes diferenciam do método de armazenamento dominante no mercado.

Sadalage e Fowler (2013) destacam as seguintes características, como principais dos bancos de dados NoSQL:

- ✓ Não utilizam o modelo relacional;
- ✓ Tem boa execução em *clusters*;
- ✓ Seu código é aberto (open source);
- ✓ São criados para propriedades na *web* do século XXI;
- ✓ Não tem esquema;
- ✓ Não possuem uma linguagem de consulta padrão.

Tiwari (2011) relata, que manter as propriedades ACIDs de uma transação em um ambiente de *clusters* demanda tempo, custo evitado nas atuais aplicações da *Internet*, onde a busca por melhor performance é cada vez mais essencial.

Partindo desse ponto, Sadalage e Fowler (2013) afirmam que a disponibilidade rápida aos usuários, advinda do uso de BDs NoSQL em aplicações *web*, decorre principalmente do fato destes, não utilizarem *schemas*, serem executados em *clusters* e principalmente em optarem por propriedades mais uteis à aplicação em detrimento a consistência tradicional.

Nesse contexto, os SGBDs NoSQL apresentam características comuns entre si, ao mesmo tempo em que se diferenciam quanto a modelagem e armazenamento dos dados. Os dados podem ser semiestruturados ou não estruturados, diferenciando a classificação dos modelos e das ferramentas NoSQL.

Para Tiware (2011) os produtos NoSQL são organizados nos seguintes modelos de dados: chave-valor, documento, famílias de colunas e grafos. Desses, os três primeiros compartilham uma característica comum em seus modelos de dados, a orientação a agregados. A Figura 4 exibe a classificação dos modelos de dados NoSQL e exemplos de produtos que se encaixam em cada uma das categorias mencionadas.

Modelo de dados	Exemplos de bancos de dados
Chave-valor ("Bancos de dados de chave-valor", p. 123)	BerkeleyDB
	LevelDB
	Memcached
	Project Voldemort
	Redis
	Riak
Documentos ("Bancos de dados de documentos", p. 133)	CouchDB
	MongoDB
	OrientDB
	RavenDB
	Terrastore
Famílias de colunas ("Armazenamentos em famílias de colunas", p. 147)	Amazon SimpleDB
	Cassandra
	HBase
	Hypertable
Grafos ("Bancos de dados de grafos", p. 161)	FlockDB
	HyperGraphDB
	Infinite Graph
	Neo4J
	OrientDB

Figura 4- Modelos de Dados NoSQL e suas respectivas ferramentas
Fonte: Sadalage e Fowler, 2013.

Sadalage e Fowler (2013) dizem que os bancos de dados de grafos, consistem em um estilo NoSQL que utiliza o modelo de distribuição semelhante aos bancos relacionais e não orientado a agregado, sendo projetados, portanto, para não executarem em um cluster. Os itens seguintes detalham os modelos de dados que são orientados a agregado.

2.2.1 Modelo Agregado

O modelo relacional recebe as informações e as armazena em tuplas (linhas). De forma limitada a tupla captura um conjunto de valores, de modo que não é possível aninhar uma dentro de outra.

A orientação agregada utilizada pela maioria dos bancos de dados NoSQL, emprega uma abordagem diferente. Para Sadalage e Fowler (2013) um agregado é um conjunto de objetos relacionados que deseja-se tratar como uma unidade. Nesse modelo pelo qual o gerenciador organiza seus dados, um agregado é pensado como um registro complexo que permita que listas e outras estruturas de dados sejam aninhadas dentro dele. Não possui o conceito de chave estrangeira e na maioria dos casos, os relacionamentos são estabelecidos na camada de aplicação. O pseudocódigo em PHP, descrito na Figura 5 exibe essa característica.

```php
<?php
$usuario = array(
    'id' => '01',
    'nome' => 'Jonas',
    'email' => 'jonascesar@gmail.com',
    'senha' => '123456',
    'mesagens' => array(
        array('id' => '01', 'texto' => 'mensagem 01'),
        array('id' => '02', 'texto' => 'mensagem 02')
     )
);
?>
```

Figura 5- Representação de um agregado
Fonte: Lóscio *et al*, 2011.

Assim como no modelo relacional, na orientação a agregados não existe uma única forma de modelagem. Os limites dos agregados são definidos de acordo às necessidades da aplicação, e de como os dados serão manipulados. Para tanto, as soluções NoSQL, orientadas

a agregação, possuem modelos distintos sendo mais utilizados os modelos chave-valor, documento e família de colunas.

2.2.1.1 Chave Valor

Este modelo é considerado simples e permite a visualização do banco de dados como uma grande tabela *hash* (LÓSCIO, 2011). Nessa estrutura o banco de dados é composto por um conjunto de chaves, as quais estão associadas um único valor.

Considerado de fácil implementação, permite que os dados sejam rapidamente acessados pela chave, principalmente em sistemas que possuem alta escalabilidade, contribuindo também para aumentar a disponibilidade de acesso aos dados.

Os sistemas de armazenamento do tipo chave-valor não possuem nenhuma linguagem destinada à realização de *queries* (consultas). As operações disponíveis para manipulação de dados são bem simples, como o *get( )* e o *set( )*, que permitem retornar e capturar valores, respectivamente (PEREIRA, 2014). A desvantagem deste modelo é que ele não permite a recuperação de objetos por meio de consultas mais complexas.

Apesar de simples, o modelo chave valor, não permite obter uma percepção total das informações e conhecimento contidos nos dados armazenados. Sadalage e Fowler (2013) dizem que o depósito de chave-valor não se preocupa com o que é armazenado na parte do valor do par da chave-valor, ele pode ser um texto, uma lista, um *JavaScript Object Notation* (JSON) e assim por diante. JSON é um formato leve de informações/dados trocado entre sistemas.

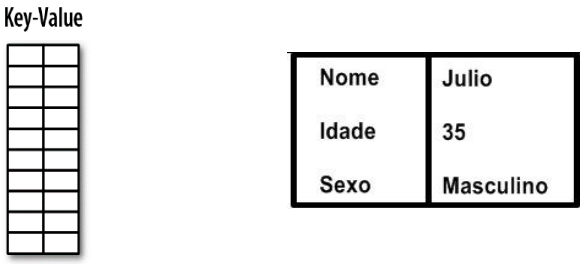

Figura 6- Modelo Chave Valor
Fonte: Pereira, 2014.

A Figura 6 demonstra graficamente o modelo chave-valor, onde a chave representa um campo como nome e idade, enquanto que o valor representa a instância para o campo correspondente.

Tiware (2011), diz que todos os armazenamentos de chave-valor podem ser consultados pela chave, se for preciso consultar algum atributo da coluna de valores não é possível utilizar o banco de dados. Para tanto, a aplicação deverá ler o valor para descobrir se ele satisfaz às condições.

2.2.1.2 Documentos

Como o próprio nome diz, este modelo armazena coleções de documentos. Um documento, em geral, é um objeto com um identificador único e um conjunto de campos, que podem ser *strings*, listas ou documentos aninhados.

Os bancos de dados de documentos armazenam documentos na parte valor do armazenamento chave-valor, em que o valor pode ser examinado (PEREIRA,2014). Diferentemente dos bancos de dados orientados a chave-valor, o documento armazenado admite pesquisa e pode ser retornado campos específicos do documento. Os documentos armazenados são normalmente de tipos bastante comuns como XML, JSON ou PDF (NAYAK, 2013).

No modelo orientado a documentos tem-se um conjunto de documentos, e em cada documento tem-se um conjunto de campos (chaves) que não aceitam ser marcados como vazios ou nulo, possuindo sempre um valor associado ao campo. Outra característica importante é que este modelo não depende de um esquema rígido, ou seja, não exige uma estrutura fixa como ocorre nos bancos relacionais (LOSCIO, 2011). Assim, é possível que ocorra uma atualização na estrutura do documento, com a adição de novos campos, por exemplo, sem causar problemas ao banco de dados. A representação gráfica do modelo orientado a documentos é demonstrada na Figura 7.

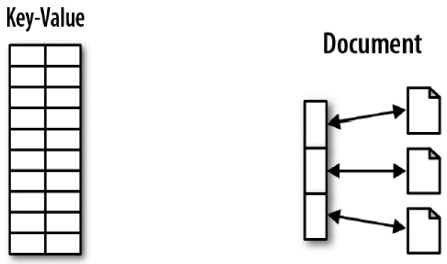

Figura 7 - Modelo orientado a documento
Fonte: Pereira, 2014.

Documentos são agrupados em coleções (KAUR; RANI, 2013). Se for realizado uma comparação com as bases de dados relacionais, pode-se considerar que as tabelas do modelo

relacional, são o equivalente às coleções do modelo de armazenamento de documentos, e os registos de cada tabela são equivalentes aos documentos de cada coleção. Estas bases de dados são bastante flexíveis em contraposto ao modelo relacional. Isto porque documentos diferentes de uma mesma coleção podem ter diferentes números de campos ou propriedades, assim como criar coleções que são armazenadas dentro de uma coleção (MCCREARY; KELLY, 2014).

2.2.1.3 Famílias de Colunas

Este modelo é um pouco mais complexo que o modelo chave-valor. Ele permite que sejam armazenados dados com chaves mapeadas para valores, onde estes são agrupados em múltiplas famílias de colunas (SADALAGE; FOWLER, 2013).

A melhor forma de pensar nesse banco de dados, seja em um agregado de dois níveis. Assim como em um chave-valor, a primeira chave é descrita como um identificador de linha, capturando o agregado de interesse, este por sua vez pode ser composto por famílias de colunas.

Família de colunas é um conjunto de linhas que tenham muitas colunas associadas fazendo uso de uma chave de linha, os dados de uma família de colunas são geralmente acessados juntos. Nesse modelo além de acessarem a linha como um todo, as operações também permitem a seleção de uma coluna em particular.

Os sistemas de bases de dados que recorrem a este modelo definem a estrutura do valor (no par chave-valor) como um conjunto predefinido de colunas (INDRAWAN, 2012). De um modo pragmático, pode-se definir o modelo de dados destes sistemas como sendo uma tabela, cujas linhas podem conter um número arbitrário de colunas. As chaves de cada linha (ou par chave-valor) fornecem um meio de indexação natural (ROBINSON, 2013).

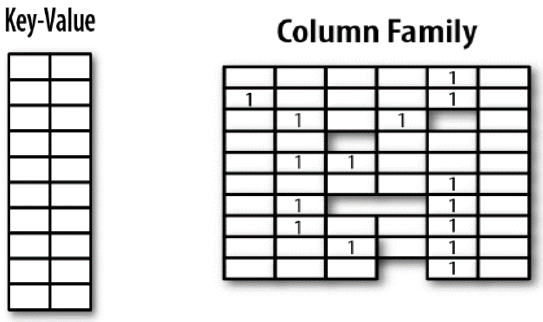

Figura 8 - Modelo Família de Colunas
Fonte: Pereira, 2014.

Devido ao seu formato tabular, sistemas baseados no modelo família de colunas apresentam algumas semelhanças visuais com as bases de dados relacionais. A grande diferença entre os dois está na maneira como ambos gerem os valores nulos. Em contraposto ao que ocorre nos SGBDRs, em sistemas NoSQL de armazenamento família de colunas irão ser armazenados pares chaves-valor numa linha, apenas se esses dados existirem e forem necessários (HECHT; JABLONSKI, 2011). A Figura 8 sintetiza o esquema de armazenamento do modelo família de colunas.

Outro contraponto, está no fato das linhas que formam uma família de coluna não precisam obrigatoriamente ter o mesmo número de colunas ou as mesmas colunas e tipos de dados, isto é, em um conjunto de linhas cada uma pode ter quantidade de colunas, e colunas diferentes das demais.

Dentro dessa noção de agregado, há algumas diferenças. O modelo de dados chave-valor trata o agregado como um nodo opaco, o que significa que somente será possível fazer uma pesquisa por chave para o agregado como um todo, não sendo possível executar uma consulta nem recuperar apenas uma parte do agregado.

O modelo de documentos, torna o agregado transparente ao banco, permitindo que sejam executadas consultas e recuperações parciais. Entretanto, pelo fato do documento não possuir um esquema, o banco não pode atuar muito na estrutura desse documento.

Quanto a organização dos agregados no modelo famílias de colunas, esses são divididos em famílias de colunas, permitindo ao banco de dados trata-las como unidades de dados dentro do agregado da linha.

Tendo em vista os fatos apresentados, percebe-se que um modelo não deve ser considerado melhor que o outro. Cada um possui particularidades e aplicações específicas que melhor se encaixam em uma determinada situação ou necessidade. É importante conhecer o problema e uma possível solução para que a escolha do modelo adequado seja efetivamente implementado.

É importante observar que a modelagem por meio de agregados busca por meio dele, um acesso mais fácil aos dados, e com isso uma resposta mais rápida os usuários. Por isso existe uma variedade de produtos NoSQL que lidam de formas diferentes, para que possam atender às necessidades das diversas aplicações atuais. A maneira desestruturada como os dados são armazenados não incomoda o movimento NoSQL, já que este tem seu foco principal concentrado na disponibilidade.

Como já mencionado os bancos orientados a agregados possuem uma boa execução em um *cluster*, tendo o processamento distribuído entre os nós desse conjunto. Para essa

distribuição é preciso ser levado em consideração a arquitetura de rede e modelos de distribuição.

2.2.2 Modelo de Distribuição

O que mais chama atenção nos SGBDs NoSQL, é a sua capacidade de executar bancos de dados em um grande *clusters.* A medida que o volume de dados aumenta, torna-se mais difícil e caro a prática da escalabilidade vertical. Para esse impasse, uma boa opção seria a escalabilidade horizontal, prática que consiste na execução do banco de dados em um conjunto de nós, distribuídos pela rede.

Em termos gerais, Sadalage e Fowler (2013) relatam que há dois caminhos a serem seguidos na distribuição dos dados no *cluster*, a replicação e fragmentação, técnicas ortogonais que possibilita o uso de uma ou de ambas.

2.2.2.1 Fragmentação

A fragmentação ou *sharding*, como especifica a Figura 9, consiste em distribuir os dados sobre os nós do *cluster*, colocando dados diferentes em nós também diferentes, cada um executando suas próprias leituras e gravações (TIWARI, 2011). Para um cenário como esse, precisa-se garantir que os dados acessados em conjunto sejam aglutinados no mesmo nó, garantindo o melhor acesso aos dados. É nesse ponto onde a orientação a agregado é útil, estes devem ser projetados para que combinem os dados que, normalmente, são acessados em conjunto.

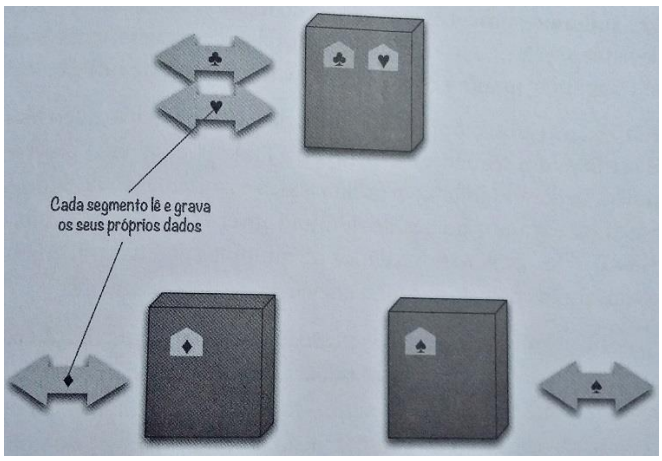

Figura 9- A fragmentação coloca dados diferentes em nodos separados, cada um executando suas próprias leituras e gravações. Fonte: Sadalage e Fowler, 2013.

Quando o assunto é organização dos dados, vários fatores podem ser considerados para ajudar a melhorar o desempenho. Se a maioria dos acessos a determinado agregado é em uma determinada localização física, deve-se armazenar essa porção em um servidor próximo ao acesso.

A respeito da recuperação de falhas, Lóscio *et al* (2011) relata que na falha de um nó os dados estariam indisponíveis aos seus usuários. Nesse contexto, fazendo uso do conceito de agregados a falha estaria afetando apenas os usuários desse fragmento; entretanto não é bom para uma base ter parte de seus dados inacessíveis. Provavelmente na prática, a utilização sozinha da fragmentação diminui a resiliência (capacidade de reparação em caso de falhas).

2.2.2.2 Replicação

A replicação consiste na cópia dos dados pelos vários nós do cluster (TIWARI, 2011). Dessa forma, os dados obtidos em um nó serão os mesmos copiados aos demais da rede. A replicação pode ser: mestre-escravo (*master-slave*) ou ponto a ponto (P2P).

Com a distribuição mestre-escravo, um nodo é designado como mestre, o qual é a fonte oficial dos dados e geralmente fica responsável por processar quaisquer atualizações nesses dados, sendo os outros denominados de escravos (SADALAGE; FOWLER, 2013). Por meio de um processo de sincronia os escravos são atualizados pelo mestre. A distribuição mestre-escravo pode ser visualizada na Figura 10.

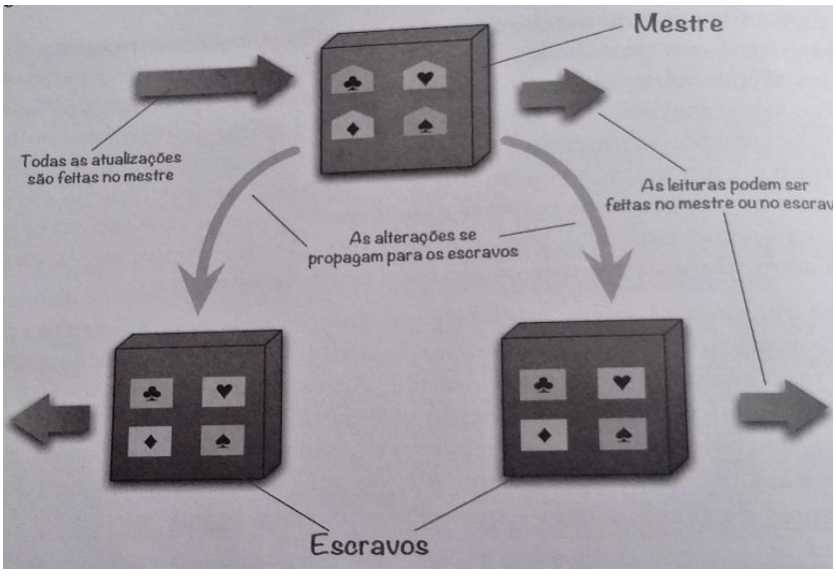

Figura 10- Os dados são replicados do mestre para os escravos.
Fonte: Sadalage e Fowler, 2013.

De maneira lógica a distribuição por meio dessa técnica é mais útil para a escalabilidade quando há um conjunto de dados com muitas leituras, já que estas não são dependentes do mestre e podem ser facilmente roteadas aos escravos. No entanto, quanto as operações de escrita em *clusters* são muito grandes, a replicação das atualizações nos escravos pode ser custosa.

Acerca da resiliência, se o mestre vir a falhar as operações de leitura não sofrerão, já as de escritas ficaram impossibilitadas até que o mestre seja restaurado ou um escravo seja designado mestre. Pereira (2014) diz que ter escravos com dados replicados acelera a recuperação do *cluster* após uma falha, já que facilmente um escravo pode ser elegido mestre.

Assim como a gravação, a inconsistência dos dados é outra desvantagem da replicação mestre-escravo. As réplicas dos dados, espalhadas pela rede, possibilitam que clientes diferentes, acessem escravos e leiam diferentes dados neles, em um determinado momento onde as alterações ainda não tenham sido propagadas em todos os escravos.

Quanto ao gargalho das operações de escrita na replicação mestre-escravo, este é solucionado com a replicação ponto a ponto, técnica em que não há um nó mestre. Todas as réplicas têm peso igual, todas podem receber gravações e a perda de algumas delas não impede o acesso ao armazenamento de dados (SADALAGE; FOWLER,2013).

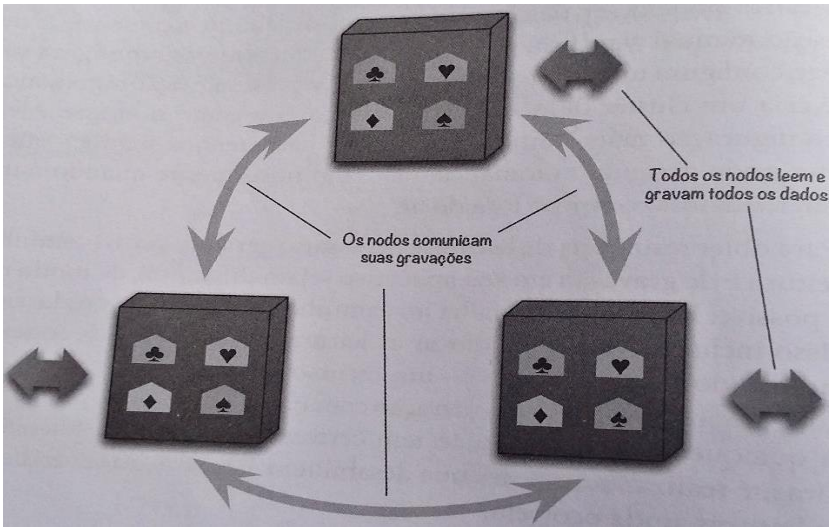

Figura 11- Todos os nós fazem leitura e gravação.
Fonte: Sadalage e Fowler, 2013.

A replicação P2P (Figura 11) soluciona o problema com as gravações, entretanto a inconsistência persiste. Quando pode-se gravar em dois lugares diferentes, corre o risco de que

duas pessoas tentem atualizar o mesmo registro ao mesmo tempo, o que gera um conflito de gravação (TIWARI, 2011).

Para resolver o impasse citado, pode-se garantir que as operações de gravação sejam sempre monitoradas de modo a impedir que estás ocorram sobre os mesmos dados ao mesmo tempo. No entanto, em contrapartida a essa solução tem-se o custo do tráfego de rede, que pode ocasionar na perda de performance da operação.

Como os produtos NoSQL não implementam bloqueios, lidar com a replicação dos dados é algo que deve ser controlado pela camada da aplicação ou relaxado a consistência, termo que admite um banco inconsistente por determinado tempo.

As técnicas de fragmentação e replicação combinadas, podem resultar em dezenas ou centenas de nós em um clusters com dados fragmentado por eles. Esta abordagem pode ser exemplificada no contexto da Figura 12.

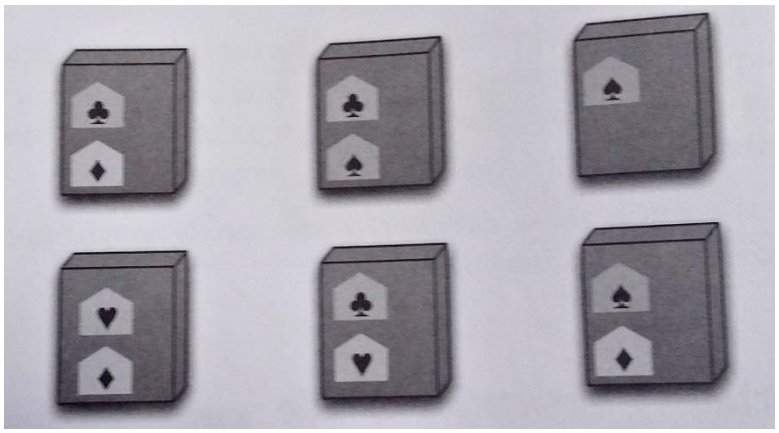

Figura 12- Utilizando a replicação P2P com a fragmentação.
Fonte: Sadalage e Fowler, 2013.

Os métodos de distribuição, favorecem um benefício em detrimento de outro. Em recorrência disso, a técnica mais eficiente é a que melhor se emparelhe às necessidades da aplicação. Portanto, um modelo não deve ser rotulado como falho, caso priorize a disponibilidade em detrimento a consistência, ou vice-versa.

2.2.3 Teorema CAP e Propriedades BASE

No movimento NoSQL, é comum referir-se ao teorema CAP como o motivo pelo qual a consistência poda ser relaxada. A afirmação base do teorema CAP é dado por três propriedades, como fala Tiwari (2011):

- ✓ Consistência (*Consistency*): alude a atomicidade e isolamento. De uma forma simples, isso implica que todos os processos executados de forma concorrente visualizem a mesma versão dos dados;
- ✓ Disponibilidade (*Availability*): significa que o sistema está disponível quando é solicitado. Segundo o Teorema CAP, o fato da resposta não ser imediata pode representar um problema;

- ✓ Tolerância a Partição (*Partition Tolerance*): Este ponto implica que o sistema seja capaz de funcionar corretamente, mesmo no caso de falha por parte de algum dos componentes.

O Teorema CAP indica que qualquer sistema que suporte bases de dados distribuídas, em um dado momento, só atenderá a duas das três propriedades citadas. Portanto, em algum momento o banco de dados terá que priorizar uma propriedade em detrimento da outra, de acordo com os requisitos da aplicação.

Os bancos de dados NoSQL tendem a implementar a disponibilidade e tolerância a partição, em detrimento da consistência, configurando um quadro de nenhuma consistência ou uma eventual consistência. Uma eventual consistência significa que, após a atualização de um atributo, "eventualmente" todos os nós do sistema irão sincronizar as informações (TIWARI, 2011). Definição que introduz o conceito *Basically Available*, *Soft State*, *Eventual Consistency* (BASE).

Ao contrário dos sistemas ACID que focam na consistência, os sistemas BASE dão ênfase na disponibilidade. As propriedades BASE estão intrinsecamente associadas aos sistemas NoSQL, porque o principal objetivo destas propriedades é garantir que os novos dados que chegam a todo momento sejam armazenados imediatamente, mesmo que isso implique no risco do sistema ficar dessincronizado por um breve período de tempo (MCCREARY; KELLY, 2014).

Os sistemas BASE são normalmente mais rápidos e apresentam uma estrutura mais simples; não há a necessidade de escrever código relativo a bloqueios e desbloqueios de recursos. Bernardino e Abramova (2013) definem a sigla BASE como:

- ✓ Basicamente Disponível - significa que o sistema garante a disponibilidade dos dados de acordo com as propriedades do Teorema CAP*;*

- ✓ Estado Flexível - significa que os dados acedidos podem estar incorretos ou imprecisos durante um variável período de tempo, e que esses dados podem mudar ao mesmo tempo que são utilizados;
- ✓ Eventual Consistência - significa que o sistema irá eventualmente tornar-se consistente, as edições sobre os dados serão propagadas com o tempo.

Portanto, com o objetivo de fornecer melhor desempenho e alta escalabilidade, os produtos NoSQL (em contraste à política de controle de transação do tipo ACID dos SGBDRs) utilizam a abordagem denominada BASE. Esta abordagem envolve a eventual propagação de atualizações e a não garantia de consistência nas leituras. Dessa forma, o sistema deve ser projetado para tolerar inconsistências temporárias a fim de priorizar a disponibilidade, principal objetivo das implementações que priorizam o conceito BASE.

3 TECNOLOGIAS DE GERENCIAMENTO DE DADOS ESTUDADAS

A proposta do trabalho é realizar testes comparativos para atestar as diferenças entre os métodos de armazenamento e propor um embasamento para a escolha acertada no uso da aplicação.

Este capítulo apresenta as tecnologias utilizadas para realização dos testes. Foram selecionados dois gerenciadores de bases de dados, o MySQL como exemplo de um SGBD relacional e o Apache Cassandra como SGBD não relacional, além da utilização de ferramentas que auxiliam no uso das referidas tecnologias.

3.1 MySQL

Desenvolvido em 1995, o "MySQL é um SGBD relacional que faz uso da linguagem padrão SQL, e é largamente utilizado em aplicações para a *internet*. É o mais popular entre os bancos de dados com código-fonte aberto" (GONZAGA; BIRCKAN, 2015).

Atualmente propriedade da *Oracle*, o *My*SQL é considerado um dos BDR mais rápidos, sendo amplamente usado em aplicações *web.* Disponibiliza funcionalidades para os mais variados fins de aplicação, tendo um bom desempenho de execução do ponto de vista de armazenamento.

São citadas abaixo, algumas características do gerenciador de dados, MILANI (2007):

- ✓ Portabilidade;
- ✓ Compatibilidades;
- ✓ Excelente desempenho e estabilidade;
- ✓ Pouco exigente quanto a recursos de hardware;
- ✓ Suporta controle transacional;
- ✓ Facilidade de manuseio;
- ✓ Replicação facilmente configurável;
- ✓ Suporte a *triggers* (gatilhos).
- ✓ Suporte a múltiplos processadores;
- ✓ Um sofisticado sistema de senhas criptografadas flexível e seguro;
- ✓ Suporta até 32 índices por tabela;
- ✓ Código fonte escrito em C e C++ e testado com uma variedade de diferentes compiladores;

- ✓ As tabelas criadas podem ter tamanho de até 4 GB;
- ✓ Banco de dados de código aberto e gratuito, amparado pela licença GNU-GPL (*General Public Licence*);
- ✓ Suporte as API´s das linguagens: PHP, Perl, C,C++, *Java*, *Pynthon*, e outras;
- ✓ Suporte à ODBC, você pode facilmente conectar o Access a um banco de dados do MySQL;
- ✓ O Cliente conecta no MySQL através de conexões TCP/IP;
- ✓ Capacidade para manipular bancos com até 50 milhões de registros;
- ✓ Multi-plataforma, portanto, suporta diferentes plataformas: *Win32*, *Linux*, *FreeBSD*, *Unix* e outros.

Acerca dos tipos de dados suportados pelo SGBDR, estes se dividem categoricamente em três, possuindo subdivisões: numéricos, data e cadeia de caracteres. Os numéricos compreendem tipos que armazenam até 8 *bytes*, que vão de inteiros a flutuantes; os do tipo data suportam representações de tempo até milissegundos com limite de 4 *bytes* de armazenamento; os do tipo cadeia de caracteres armazenam campos que vão de 0 caractere até mais de *4 bytes*.

O SGBDR MySQL continua a ter uma elevada taxa de adoção por vários setores da indústria e é conhecida por sua confiabilidade, facilidade de uso e performance. Na versão *My*SQL *5.0* as características são susceptíveis de impulsionar o SGBD a taxas ainda maiores de aceitação e podendo abrir portas para suporte a aplicações e ferramentas diferenciadas (YUHANNA, 2015).

Contudo, os conhecedores e desenvolvedores do SGBD admitem que o MySQL não é arquitetado para enfrentar a nova onda de grandes aplicações de dados desenvolvidas atualmente (DATASTAX, 2015). Ele apresenta suporte para distribuição e manuseio de grandes bases, no entanto as aplicações *big data* crescem rapidamente e não se conhece ao certo seus limites.

Para o manejo do MySQL 5 [1] nos testes, fez-se uso de uma ferramenta de edição de consultas, o SQLYog[2]. De fácil manipulação, o SQLYog é leve e dispõe de uma interface intuitiva, auxiliando no gerenciamento de bases em conjunto ao MySQL.

[1] Disponível: https://www.mysql.com/downloads/
[2] Disponível: https://www.webyog.com/product/downloads/

3.2 Apache Cassandra

O Apache Cassandra é um sistema de armazenamento NoSQL desenvolvido, na LP *Java*, pela equipe do *Facebook* em 2008 e mantido atualmente pela fundação *Apache*. Este SGBD adota o tipo de armazenamento família de colunas e apresenta uma estrutura parecida com a de um sistema relacional tradicional (BERNARDINO; ABRAMOVA, 2013). O Quadro 1 demonstra pontos em comum entre os SGBDs.

Quadro 1- Equivalência entre os SGBDRs e o Cassandra. Fonte: Autor.

SGBDR	Cassandra
Banco de Dados	Keyspaces
Tabela	Familia de Colunas
Linha	Linha
Coluna (a mesma para todas as linhas)	Coluna (podem ser diferentes por linha)

O sistema Cassandra foi projetado para gerir grandes volumes de dados, alcançando simultaneamente uma grande disponibilidade e poder de escalabilidade, sem nenhum ponto de falha (ANIELLO *et al*., 2013). O SGBD supracitado, considera as falhas de *hardware* como normais e não como exceções. Assim o sistema está preparado para, automaticamente, compensar a perda de um nó sem que isto tenha qualquer significado para o utilizador final.

Em decorrência do modelo de distribuição P2P usado pelo Cassandra, ele dispõe de alta velocidade de escrita, sem que isso afete a eficiência das leituras de dados (BAGADE *et al*., 2012). Como é uma base de dados distribuída, o sistema replica os dados para manter a latência de pesquisa baixa. Desse modo, este sistema também é caracterizado pelo seu modelo, que relaxa as regras de consistência baseado em quóruns, tendo em algum momento versões distintas do mesmo dado.

O Cassandra dispõe de uma linguagem de manipulação de dados, a *Cassandra Query Language* (CQL). A CQL, em muito se assemelha a linguagem de consulta dos SGBDRs, provendo recursos para a definição e manipulação de dados (APACHE, 2015). Essa semelhança é argumentada pelos mantenedores como uma estratégia para adesão de novos clientes.

Estando em sua terceira versão, a CQL3 modificou o método principal de interação com o sistema, o *thrift* (linguagem de definição de interfaces e um protocolo binário), deixando

mais simples e em desuso alguns conceitos do modelo família de colunas, considerados pelos mestres do SGBD Cassandra como complexo e de difícil acesso aos dados.

Um conceito que a CQL3 deixou para trás é a possibilidade de inserir colunas dentro de outras colunas, criando *super* colunas. Dessa forma, *super* colunas não são suportadas em CQL3, aproximando cada vez mais a visão do usuário para a manipulação com a SQL (DATASTAX, 2012).

Outro conceito restringido pelo CQL3, é o de família de colunas dinâmicas, onde colunas podem ser criadas, sem que se conheça o nome que tomarão, possuindo linhas sem um padrão de colunas.

É preciso considerar que todo este dinamismo é restringido pela linguagem CQL3. No entanto, é possível através da aplicação ou de uma camada de manipulação, que os dados sejam manuseados diretamente com o Cassandra, sem que para isso validações no CQL3 sejam feitas.

Quanto aos tipos de dados, além dos de uso comum e já conhecidos pelos desenvolvedores de bases relacionais como, por exemplo, os numéricos, de data e cadeias de caracteres, o Cassandra dispõe de um tipo de dado específico para sistemas distribuídos, o *uuid.* A sua utilização tem por objetivo permitir que sistemas distribuídos consigam identificar unicamente informação, sem ser necessário uma significante coordenação central. O uso de coleções no Cassandra também é comum, pois estas possibilitam que sejam armazenadas listas no valor do par chave-valor. As coleções podem ser: *List, Set* e *Map*.

As coleções devem ser usadas quando se desejar armazenar uma pequena quantidade de dados, pois os valores de itens em coleções estão limitados a 64 kb (*Kbyte*). Outras limitações também se aplicam. Coleções funcionam bem para armazenar dados, tais como os números de telefone de um usuário e rótulos aplicados a um e-mail. Se os dados a armazenar têm ilimitado potencial de crescimento, tal como todas as mensagens enviadas por um utilizador ou eventos registados por um sensor, não se recomenda utilizar coleções.

Com base nas principais características do Apache Cassandra, foram realizados testes com o SGBD não relacional para demonstrar suas potencialidades com maior clareza. Para viabilização dos testes foi utilizada uma distribuição do Cassandra, o *DataStax Community*[3].

O *DataStax Community* é uma versão simplificada do *Cassandra,* com utilização livre de custos para fins não comerciais, disponibilizado pela empresa *DataStax.* Para a manipulação

[3] Disponível em: http://www.datastax.com/cassandra

do SGBD foi usado a ferramenta de edição *DevCenter*[4] e o *OpsCenter*[5] que é uma solução de monitoramento do Cassandra auxiliando na configuração de nós.

As tecnologias utilizadas nos testes e confecção de *Script*s de execução são descritas abaixo no Quadro 2:

Quadro 2- Principais Ferramentas Utilizadas. Fonte: Autor

Ferramenta	Descrição	Modelo
MySQL 5	SGBDR mais usado pelas aplicações, de modo geral.	Relacional
SQLyog	Editor de consultas SQL para o MySQL	Relacional
DataStax Communit Cassandra 2.1.11	Distribuição do SGBD Apache Cassandra para Windows	Não Relacional
DevCenter 1.4.1	Ferramenta de consulta para DataStax Cassandra	Não Relacional
OpsCenter 1.4.1	Ferramenta para controle e manipulação de *clusters*	Não Relacional

[4] Disponível em: http://www.datastax.com/devcenter
[5] Disponível em: http://www.datastax.com/opscenter

4 BANCO DE DADOS: RELACIONAL OU NÃO RELACIONAL?

O Capítulo 4 aborda questionamentos sobre a utilização, principais funcionalidades, pontos positivos e negativos referentes aos modelos foco do presente estudo.

4.1 Relacional ou Não Relacional?

Decidir por um SGBD no desenvolvimento de um projeto é tarefa tão importante quanto a escolha da LP a utilizar ou definição do escopo acerca do que vai ser concebido. Uma alternativa seria dizer que a escolha do SGBD deve casar com a LP e esses devem possuir recursos que venham a favorecer o desenvolvido de um produto para uma determinada área de negócio.

Apesar da interligação entre os mais variados ramos de negócio, cada aplicação é destinada a um público-alvo e seu sucesso depende da satisfação desse. Para atender as expectativas do usuário, a escolha do método de armazenamento deve contemplar as necessidades da aplicação, e consequentemente de quem irá usá-la, considerando os aspectos inerentes a cada campo de atuação.

Nessa conjuntura, os SGBDRs e os SGBDs não relacionais apresentam direcionamentos diferentes, cada um sendo solução indicada a determinado tipo de aplicação. Vista as tecnologias, são muitos os aspectos que as diferem, no entanto alguns podem contribuir para a escolha adequada do uso no gerenciamento dos dados.

Fatores como a consistência, escalabilidade e disponibilidade de dados podem ser fatores determinantes na decisão pela tecnologia a ser empregada. Nesse contexto, o Quadro 4 apresenta as principais diferenças e o impacto da presença ou ausência de cada uma das tecnologias:

Quadro 3 - Principais impactos da presença ou ausência dos SGBDRs e NoSQL. Fonte: Autor.

	Conceito	**Relacional**	**NoSQL**
Escalabilidade	É a capacidade de expansão do poder de processamento de um sistema, pode ser vertical (aumento da capacidade do servidor) ou horizontal (divisão do processamento em várias maquinas).	É possível, mas complexo e ineficiente. Devido a natureza estruturada do modelo, a adição de forma dinâmica e transparente de novos nós no *grid* não é realizada de modo natural como acontece com os agregados.	Uma das principais vantagens dessa tecnologia. Por não possuir nenhum tipo de esquema e ser baseado em agregados permite eficientemente a escalabilidade. Por não possuir um esquema pré-definido, os modelos possuem maior flexibilidade o que favorece a inclusão transparente de outros elementos.

Consistência	Está relacionada a confiabilidade dos dados, a precisão. Dados consistentes são íntegros e seguros.	Um dos pontos mais forte do modelo relacional. Por meio dos bloqueios transacionais a consistência é mantida, no entanto como consequência, em sistemas com um grande número de usuários o uso de bloqueios pode acarretar em um maior tempo de resposta ao usuário.	Realizada de modo eventual, ou seja, sem muito rigor. Por não implementar as propriedades ACID de uma transação, os gerenciadores NoSQL controlam as operações por meio de uma camada da aplicação, balanceando a consistência por meio de quóruns como forma de possuir alguma consistência ou nenhuma.
Disponibilidade	Está relacionada ao fato de um sistema ou recurso estar sempre ativo e disponível em tempo de excelência.	Devido à dificuldade de lidar com a distribuição dos dados, esse modelo pode não suportar um número de requisições muito grande. Em uma requisição online feita ao banco muitas verificações, junções, bloqueios são empregados, os efeitos desse uso aumentam o tempo de resposta ao usuário.	Outro fator fundamental do sucesso desse modelo. O alto grau de distribuição dos dados propicia que um maior número de solicitações sejam atendidas, outro fator que influencia na velocidade dessas respostas é o uso de agregados, estrutura que viabiliza o armazenamento de características de mais de um ente, não fazendo uso de junções entre tabelas como no modelo relacional.

Com base nas informações acima, os SGBDs NoSQL têm a disponibilidade como argumento na adoção à tecnologia. Se a aplicação demanda por *receive* rápidas ao usuário e a eventual consistência não venha a implicar de maneira desastrosa no negócio, a melhor opção talvez seja o uso de um dos modelos de dados NoSQL.

Contudo, se há necessidade de uma consistência rigorosa, em que a manipulação de dados imprecisos possa acarretar em custos muito altos, como transações bancárias, a integridade dos dados deve ser preservada em detrimento a disponibilidade, sendo mais confiável optar pelo uso de SGBDRs.

5 EXPERIMENTOS FEITOS COM O MYSQL E O APACHE CASSANDRA

Neste capitulo, serão apresentados os experimentos comparativos entre os modelos estudados. Nele pode-se atestar pontos comuns e adversos entre o modelo relacional e o não relacional, incluindo as linguagens de manipulação utilizadas por cada tecnologia.

5.1 Testes

Nesta seção, serão apresentados testes comparativos usando a SQL e a CQL, ambas linguagens de manipulação de dados, sendo uma de uso relacional e outra de uso não relacional, respectivamente.

Os experimentos objetivam demonstram, dentre outros aspectos, a similaridade entre as sintaxes, desempenho de determinado tipo de instrução e as diferenças existentes entre um modelo e outro. É importante ressaltar que mesmo que estejamos falando em desempenho de SGBDs que trabalham com um grande número de dados voltados para a *web,* os experimentos foram realizados em ambiente *Windows* com apenas uma máquina. O Quadro 3 representa a configuração do equipamento utilizado para a realização dos experimentos.

Quadro 4 - Configuração da Máquina. Fonte: Autor

COMPONENTE	CONFIGURAÇÃO
Processador	Core i3
RAM	DDR3 de 4 Gb
Disco Rígido	500 Gb
Sistema Operação	*Windows 7 Ultimate*

Foi utilizado para os experimentos a base de dados de CEPs (Código de Endereçamento Postal) do país, mantida pela Empresa Brasileira de Correios e Telégrafos (Correios). A base de dados é composta por quatro tabelas: cepbr_estado, cepbr_cidade, cepbr_bairro e cepbr_endereço; as quais se relacionam. A Figura 13 exibe o diagrama das tabelas usadas em ambos os modelos.

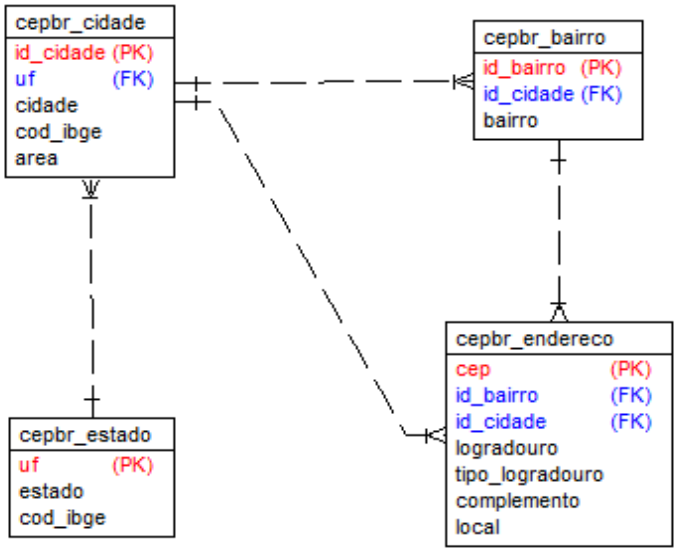

Figura 13 - Diagrama das Tabelas Usadas nos Experimentos.
Fonte: Autor.

É importante ressaltar que as tabelas representadas no diagrama estão modeladas no padrão relacional, e este foi expresso de acordo aos relacionamentos evidenciados no *script* SQL, disponibilizado pelos correios. A estrutura supracitada foi a mesma usada nos testes com o *Cassandra*. Contudo, tanto as tabelas quanto os campos foram adaptados ao padrão *CQL*, que não suporta relacionamentos por meio de chaves estrangeiras.

Para a criação de um banco de dados as sintaxes são bem parecidas, visto que ambas as linguagens utilizam o comando de definição de dados CREATE. No entanto, como o *Cassandra* é um banco naturalmente distribuído, deve ser definido a estratégia de replicação a utilizar, que pode ser: "*NetworkTopologyStrategy*", utilizada quando pretende-se ter um sistema a funcionar sobre diferentes conjuntos de nós, podendo ou não estar fisicamente em locais distintos. A estratégia "*SimpleStrategy*", é utilizada quando pretende-se utilizar um único *cluster*, usualmente presentes na mesma localização física.

Script 1 - Criação de um banco de dados no Cassandra:

```
CREATE KEYSPACE IF NOT EXISTS cepbr WITH replication = {'class': 'SimpleStrategy',
'replication_factor' : 1};
```

O Segundo parâmetro define o fator de replicação do nó e é obrigatório ser informado. Quando o *keyspace* faz uso da classe *NetworkTopologyStrategy*, deve ser informado o fator de replicação para cada *cluster*.

Script 2- Criação de uma tabela no Cassandra:

```
CREATE TABLE cepbr.cepbr_endereco(cep varchar, logradouro varchar, tipo_logradouro varchar, complemento varchar, local varchar, id_cidade int, id_bairro int,  PRIMARY KEY (cep));
```

No *Script 2* a tabela ou família de coluna endereço é criada. Um ponto importante a ser mencionado são os campos "id_cidade" e "id_bairro" que fazem alusão às famílias de colunas cidade e bairro. A modelagem no *Apache Cassandra* é feita desta forma, mesmo não existindo o conceito de chave estrangeira, a qual estabelece fisicamente a ligação entre os entes. Na lógica o relacionamento existe e deve ser manipulado pela camada da aplicação, por isso a necessidade dos campos citados. Como o conceito de FK não existe nesse modelo, assim também sucede com as restrições de chave, que no modelo relacional impede que um registro seja cadastrado caso ainda não possua ainda a chave estrangeira cadastrada como primaria em outra tabela.

Script 3 - Inserção na base de dados, no Cassandra:

```
Insert into cepbr_endereco (cep, logradouro, tipo_logradouro, complemento, local, id_cidade, id_bairro)     values      ('01001000','DA      SÉ','PRAÇA','-      LADO ÍMPAR','',9668,29400);
```

As inserções na *CQL* procedem exatamente como no uso da SQL, conforme descrita pelo *Script 3* . Ressalva-se que estão sendo demonstrados principalmente os s*cripts* CQL, por está não ser disseminada como a SQL, oportunizando assim, a promoção do conhecimento da linguagem. Quando as instruções apresentarem diferenças significativas ambos os *scripts* serão demonstrados.

Script 4 - Junção de tabelas no MySQL:

```
select ce.logradouro as NOME_DA_RUA, b.bairro as BAIRRO, c.cidade as CIDADE, c.area as EXTENSAO_TERRITORIAL_DA_CIDADE, c.uf as ESTADO from cepbr_endereco  as ce inner join cepbr_bairro as b inner join cepbr_cidade as c inner join cepbr_estado as u on ce.id_bairro=b.id_bairro and b.id_cidade=c.id_cidade and c.id_cidade=ce.id_cidade and u.uf=c.uf and c.cidade='Picos' and b.bairro= 'JUNCO' and ce.tipo_logradouro='RUA';
```

A consulta mostrada no *Script 4* agrupa dados de quatro tabelas. Essa junção é viabilizada por meio das chaves estrangeiras que evidenciam o relacionamento entre elas. Em decorrência desse fato esse tipo de consulta não é suportado pelos SGBDs NoSQL. Os *joins*, como são conhecidos esse tipo de consulta, são muito usados na geração de relatórios que envolvem várias tabelas, no *Script 4* é retornado todos os ceps das ruas do bairro Junco, da cidade de Picos no Piauí.

Script 5 - Consultas usadas para devolver o mesmo resultado do *Script 4* anterior, no Cassandra:

```
select id_cidade from cepbr_cidade  where cidade='PICOS' and uf='PI';
select * from cepbr_bairro where bairro='JUNCO'and id_cidade=5665;
select * from cepbr_endereco where cep >= '64607630' and cep <='64607840' and
tipo_logradouro ='RUA';
```

O *Script 5*, retorna praticamente os mesmos dados do *Script 4*, no entanto para alcançar esse resultado é preciso fazer três instruções *selects*, uma por vez, já que o *Cassandra*, não admite consultas aninhadas. Apenas a última instrução *select* possivelmente alcançaria o objetivo do usuário, no entanto para conhecer os dados resultantes faz-se necessário executar primeiramente as duas instruções que a antecedem.

Uma característica importante do Cassandra e de outros bancos NoSQL, é o uso de coleções. Elas representam de maneira simplificada o conceito de agregado, já mencionado.

Script 6 – Criação de tipos de lista, no Cassandra:

```
CREATE TYPE contato_representante(nome text, phone text, email list<text>);CREATE
TYPE IF NOT EXISTS pontos_turistico (nome text, log float, lat float,
contato_responsavel set<frozen<contato_responsavel>>);
```

No *Script 6* tem-se a instrução de criação de tipos de coleções. Nele percebe-se que listas podem conter listas, todas aninhadas dentro de um único campo. O exemplo, mostra a criação de dois tipos de dados cada um contendo seus campos e listas. Supondo que se decida armazenar na tabela cidade os pontos turísticos de cada uma. No modelo relacional criaríamos uma tabela “pontos_turisticos” que conteria uma chave estrangeira vinda de cidade. Contudo, como os SGBDs NoSQL não são adeptos a essa modelagem, por isso desnormalizados, e possuem a opção de usar listas, “pontos_turisticos” passa a ser um campo da tabela cidade, que por sua vez armazena outra coleção, os dados do “contato_responsavel” pelo ponto turístico.

Script 7 – Adicionar coleção como campo da tabela, no Cassandra:

```
ALTER TABLE cepbr_cidade ADD ponto_turistico frozen<ponto_turistico>;
```

Script 8 – Atualizar a coleção de dados, no Cassandra:

```
UPDATE cepbr_cidade SET ponto_turistico = {nome: 'Cristo Redentor', lat: 25º55S,
log: 34º53W, contato_responsavel: {{ nome: 'Prefeitura do RJ', phone: '21999247204',
email: { 'preferj@gmail.com','rjprefeitura@hotmail.com' }} }} WHERE cidade = 'Rio
De Janeiro';
```

O *Script 8* atualizado um campo novo que acaba de ser adicionado a família de colunas. Esse formato de armazenamento não é disponibilizado pelos bancos de dados relacionais e pode ser acessado em porções menores, conforme o *Script 9*.

Script 9 – Seleção de todos os pontos turísticos do Rio de Janeiro, no Cassandra:

```
SELECT pontos_turisticos.nome, pontos_turisticos.lat, pontos_turisticos.log, contado_responsavel.phone FROM cepbr_cidade WHERE uf = 'RJ';
```

A CQL busca reproduzir as funcionalidades da SQL, visando ganhar adeptos a linguagem. Comandos como *drop*, *delete*, *alter*, *truncate*, *trigers*, *count*, *indexes*, são utilizados sem mudanças radicas. Contudo funções como *between* e *like* não são suportadas pela CQL.

Script 10- Atualizando registro da tabela ‘”cepbr_cidade”, no MySQL:

```
update cepbr_cidade as c set c.cidade='PICOS CIDADE DO FRIO' and c.cod_ibge='1234234' and c.area = 800.715 where id_cidade = 5665;
```

5.2 Resultados

Nesta seção serão apresentados os resultados dos testes de desempenho entre o SGBD MySQL e o SGBD Apache Cassandra. Estes testes têm como objetivo demonstrar performance (velocidade) de resposta de três consultas criadas anteriormente: instruções de inserção (*Script 3*), pesquisa (*Script 4* e *5*) e atualização (*Script 8* e *10*). Cada *Script* foi executado quatro vezes, possibilitando a obtenção de uma média. É importante ressaltar que a cada execução a máquina foi reiniciada com o intuito de limpar o *buffer* da memória.

Os gráficos expostos a seguir, demonstram em milissegundos o tempo gasto na execução de cada instrução pelos SGBDs avaliados.

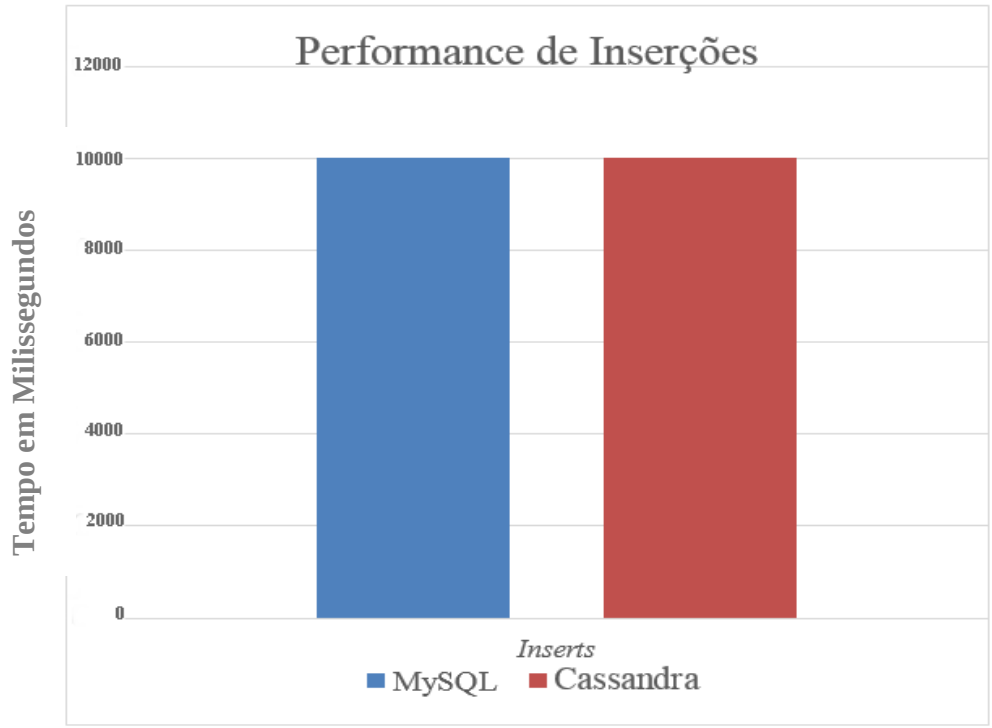

Gráfico 1 - Performance de Inserções no MySQL e Cassandra. Fonte: Autor.

No Gráfico 1, são apresentados os resultados das inserções na tabela "cepbr_endereco". O *Script 3* executado, possui 954.277 registros e apresentou resultados de performance semelhantes em ambos os SGBDs. Com um custo de tempo aproximado em 10. 000 milissegundos, a conclusão a recolher de tais números, deve-se principalmente ao fato do SGBD Apache Cassandra está configurado para fazer uso da *SimpleStrategy*, estratégia onde o processamento não ocorre de forma distribuída.

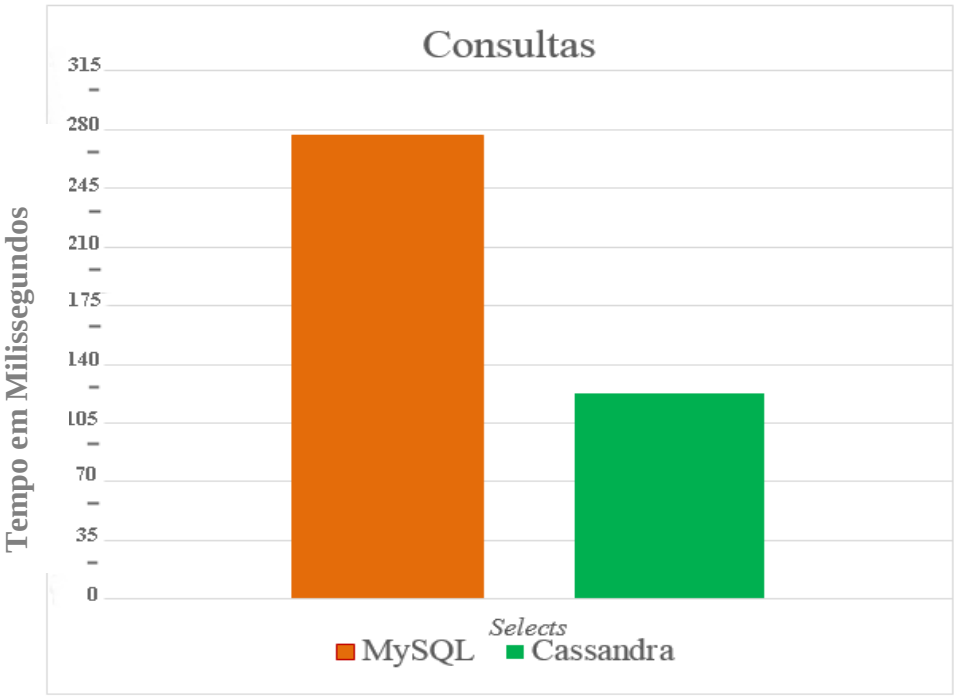

Gráfico 2 - Performance de consultas no MySQL e Cassandra. Fonte: Autor.

O Gráfico 2, representa o resultado em milissegundos de instruções *selects* executada sobre a tabela de "cepbr_cidade" que contem 10.716 linhas. As consultas executadas, apesar de ambas serem do tipo *select*, possuem verificações diferentes. No *Script 4* (Mysql) temos uma seleção com *join* que requer uma serie de comparações entre as tabelas, enquanto no *Script 5* (Cassandra) as seleções ocorrem de maneira bem mais simples. Apesar da camada de aplicação precisar realizar mais de uma instrução *select* para buscar o que precisa, estas requerem menos custo. Portanto, como pode ser observado no Gráfico 2 o tempo de resposta apresentado pelo Apache Cassandra foi inferior ao seu concorrente MySQL, demonstrando desta forma, melhor desempenho no quesito consultas de dados.

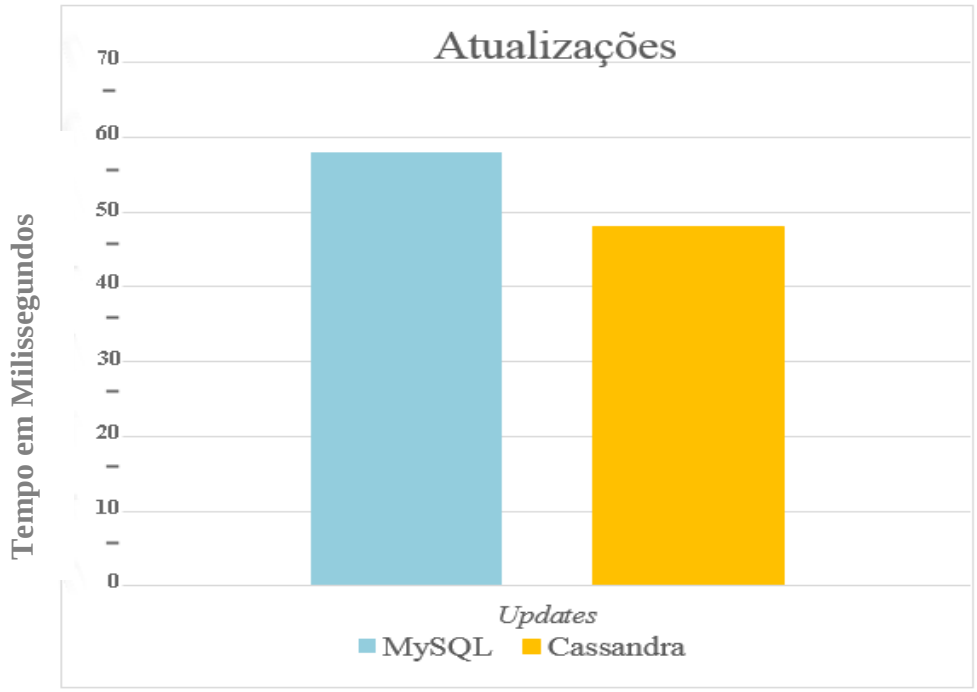

Gráfico 3 - Performance de atualizações no MySQL e Cassandra. Fonte: Autor.

O Gráfico 3, esboça o resultado obtido por atualizações feitas na tabela "cepbr_cidade". Com relação ao desempenho, o Cassandra obteve menor custo de tempo comparado ao MySQL, ou seja, melhor desempenho. Apesar do *Script 8* (Cassandra) atualizar uma lista, com vários campos o tempo gasto para tal operação foi menor que o tempo gasto pelo *Script 10* (MySQL). Apresentando um desempenho inferior ao SGBD Cassandra, o MySQL consumiu o aproximado a 60 milissegundos para a execução do *Script 10* enquanto seu concorrente apresentou o aproximado a 50 milissegundos.

Sabe-se que os *Script*s executados não representam de fato a capacidade de cada tecnologia. Como já foi mencionado, cada aplicação requer uma modelagem especifica, nos bancos de dados relacionais e especialmente tratando-se de bancos de dados não relacionais. Por meio dos experimentos, pretende-se contribuir para a disseminação e conhecimento dos bancos de dados NoSQL.

6 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Um banco de dados eficiente e modelado em consonância as necessidades da aplicação e do usuário é capaz de agregar inúmeros benefícios para uma organização. Para uma escolha acertada no armazenamento dos dados cabe conhecer além das tecnologias disponíveis no mercado, é preciso identificar a que melhor se adequa as necessidades e a concepção do negócio.

O trabalho descrito foi desenvolvido com o propósito de apresentar uma análise comparativa entre o modelo de banco de dados relacional e o não relacional, com enfoque nos SGBDs MySQL e Apache Cassandra. Foram desenvolvidos *script*s de inserção, pesquisa e atualização de dados objetivando conhecer o custo de desempenho de cada instrução nos SGBDs supracitados.

Para os experimentos, foi utilizada a base de endereços do Brasil, constituída de quatro tabelas que totalizam 1.010.976 registros. É muito importante que os resultados obtidos sejam encarados apenas num contexto acadêmico. Com outras regras e objetivos a alcançar, os resultados poderiam ser bastante distintos dos obtidos.

As bases de dados NoSQL foram desenhadas com o intuito de escalar horizontalmente. Remover ou não usar os fatores inerentes ao armazenamento distribuído está a retirar-se dos sistemas NoSQL o seu habitat natural e, por consequência, a desperdiçar a sua verdadeira utilidade. Neste trabalho os sistemas foram simplesmente utilizados numa máquina local, sem distribuição de dados por outros nós.

Nessa conjuntura, como trabalho futuros, sugerimos realizar experimentos em um *clustes* com mais de um nó e com uma quantidade maior de registros, buscando chegar cada vez mais próximo da realidade de grandes empresas que lidam com o conceito *big data*. Propomos ainda um estudo aprofundado acerca da CQL, avaliando cada ponto que difere da SQL.

REFERÊNCIAS

ABRIL, J Data semantics. **Data Base Management.** Amsterdam: North-Holland, 1974.

ANIELLO, L.; BONOMI, S.; BRENO, M. & BALDONI, R. **Assessing data availability of Cassandra in the presence of non-accurate membership**. Proceedings of the 2nd International Workshop on Dependability Issues in Cloud Computing. Braga, Portugal, 2013.

APACHE. ***Cassandra Query Language (CQL) v3.1.7***. Disponível em: <https://cassandra.apache.org/doc/cql3/CQL.html >. Acessado em 11 nov. 2015.

BAGADE, P.; CHANDRA, A. & DHENDE, A. B. **Designing performance monitoring tool for NoSQL Cassandra distributed database**. Education and e-Learning Innovations (ICEELI), 2012 International Conference. Sousse, Tunísia, 2012.

BALDO, S. M.; MACÁRIO, C. G. N. **O modelo relacional**. Disponível em <http://www.ic.unicamp.br/~geovane/mo410-091/Ch03-RM-Resumo>. Acessado em nov. 2015.

BERNARDINO, J; ABRAMOVA, V. **NoSQL Databases**: MongoDB vs Cassandra. C3S2E '13 - International C* Conference on Computer Science and Software Engineering. Porto, Portugal, 2013.

DATASTAX. **Why Migrate from MySQL to Cassandra**. White Paper By Datastax CORPORATION, Jun 2012.

DATASTAX. **CQL3 for Cassandra 2.x**. Disponível em: <http://www.datastax.com/documentation/cql/3.1/cql/cql_intro_c.html>. Acessado 10 nov. 2015.

DEVMEDIA. Banco de Dados Relacionais. Disponível em: <http://www.devmedia.com.br/arquitetura-de-um-relacionamento/2500796>. Acessado em 30 nov. 2015.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de Banco de Dados**. 4a ed. São Paulo: Pearson Addison Wesley, 2005.

GONZAGA, F. S; BIRCKAN, G. **PHP e banco de dados MySQL**. Disponível em <http://tudodownloads.uol.com.br/desenvolvedor/flavio_s_gonzaga_e_guilherme_birckan/1948-curso_de_php_e_mysql.html. Acessado 10 de jan. 2015

HECHT, R.; JABLONSKI, S. **NoSQL evaluation**: A use case oriented survey. Proceedings of the 2011 International Conference on Cloud and Service Computing. Hong Kong, China, 2011.

HEUSER, C. A. **Projeto de Banco de Dados**. 6a ed. Rio Grande do Sul: Sagra, 2009.

INDRAWAN-S, M. **Database Research**: Are We at a Crossroad? Reflection on NoSQL. Proceedings of the 15th International Conference on Network-Based Information Systems, 2012 Melbourne, Austrália. IEEE Computer Society, 2012.

KAUR, K.; RANI, R. **Modeling and querying data in NoSQL databases**. Big Data, IEEE International Conference, California, Estados Unidos da América. IEEE Computer Society, 2013.

LÓSCIO, B. F.; OLIVEIRA, H. R.; PONTES, J. C. S. **NoSQL no desenvolvimento de Aplicações Web colaborativas**. VIII Simpósio Brasileiro de Sistemas Colaborativos, 2011.

MCCREARY, D.; KELLY, A. **Making Sense of NoSQL:** A Guide for Managers and the Rest of Us: 1ª Edição. Shelter Island, Nova York, Estados Unidos da América, Manning Publications, 2014.

MILANI, A. **MySQL**: Guia do Programador. Novatec, 2007. Disponível em: < http://www.novateceditora.com.br/livros/mysqlcompleto/capitulo8575221035.pdf > Acesso em: 06 mai. 2016.

NAYAK, A.; PORIYA, A.; POOJARY, D. **Type of NOSQL Databases and its Comparison with Relational Databases**. *International Journal of Applied Information Systems*, 2013.

PEREIRA, D. J. P. **Armazéns de Dados NoSQL**. Instituto Politécnico do Porto, 2014.

RAMAKRISHNAN, R.; GEHRKE, J. **Sistemas de Gerenciamento de Banco de Dados**. 3a ed. São Paulo: McGraw-Hill, 2008.

ROBINSON, I.; WEBBER, J.; EIFREM, E . **Graph Databases***: 1ª Edição.* Sebastopol, California, Estados Unidos da América, 2013.

SADALAGE, J.P.; FOWLER, M. **NoSQL:** Um Guia Conciso para o Mundo Emergente da Persistência Poliglota, Essencial. Novatec Editora, São Paulo, 2013.

TIWARI, S. **Professional NoSQL**: 1ª Edição. Indianapolis, Indiana, Estados Unidos da América, 2011.

YUHANNA, N. **MySQL**. Disponível em: < https://tecnoesis.wordpress.com/tag/noel-yuhanna/mysql>. Acessado 20 nov. de 2015.

APÊNDICES

APÊNDICE A – Ferramenta *OpsCenter*

Essa ferramenta foi usada para a criação e gerenciamento do *cluster*, por meio dela um *clusters* é adicionado facilmente a outro, basta possuir o endereço deste na rede. A figura 14 demonstra o nó configurado e ativo, usado nos testes.

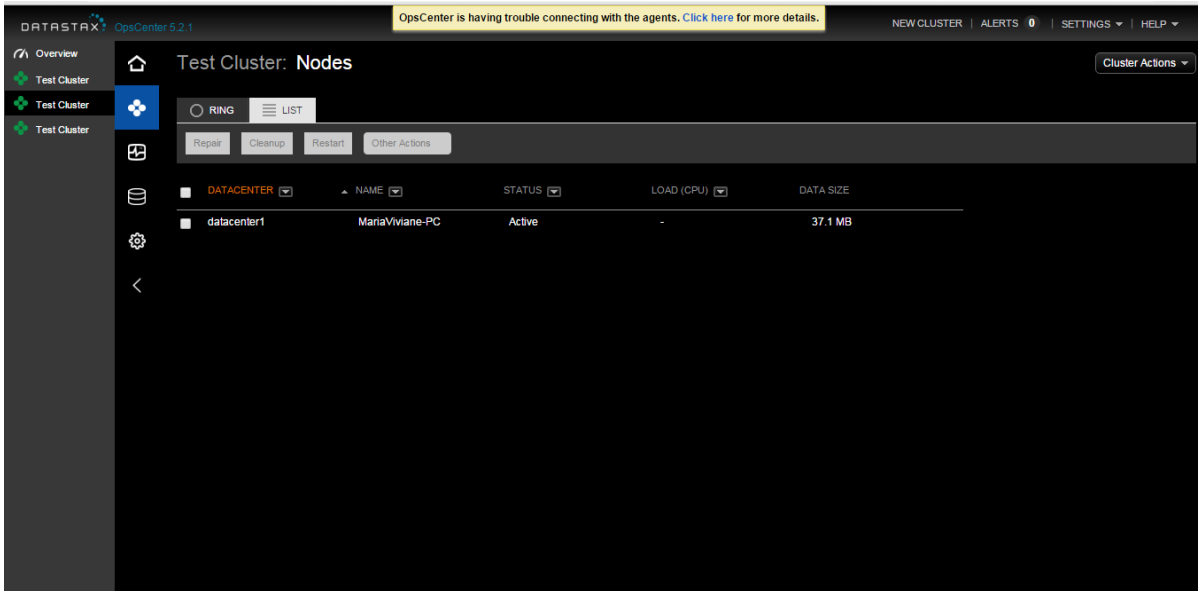

Figura 14 - Nó utilizado para os testes, configurado e ativo no *cluster*

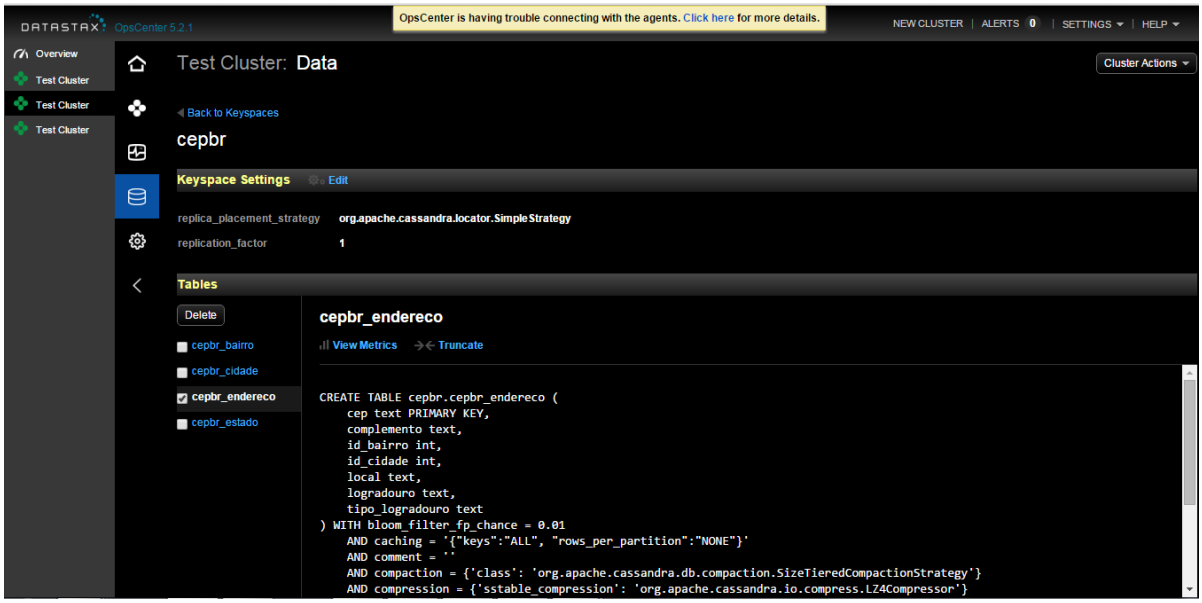

Figura 15 - Base de Ceps, armazenada no nó do *cluster*

APÊNDICE B – Ferramenta *DevCenter* e *SQLYog*

As ferramentas *DevCenter* e *SQLYog*, foram usadas na edição de *querys* e criação das bases de dados. Sendo uma de uso relacional e outra de uso não relacional, ambas as ferramentas apresentam semelhanças quanto a manipulação das bases, demonstradas nas Figuras 16 e 17.

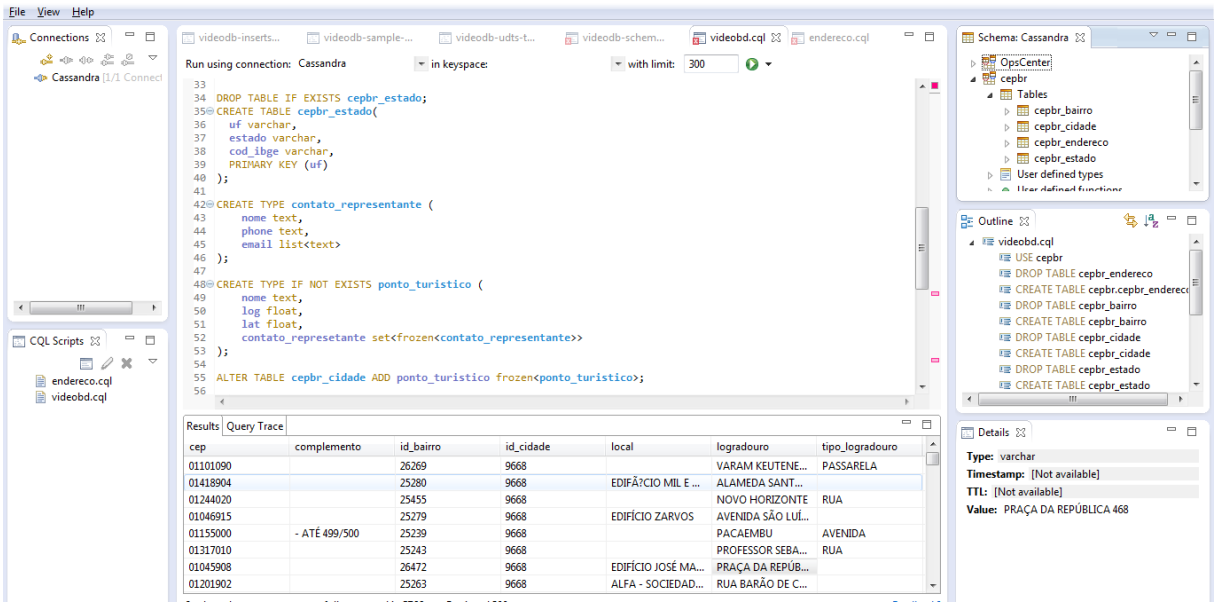

Figura 16 - Ferramenta para Edição de consultas do Cassandra Datastax - DevCenter

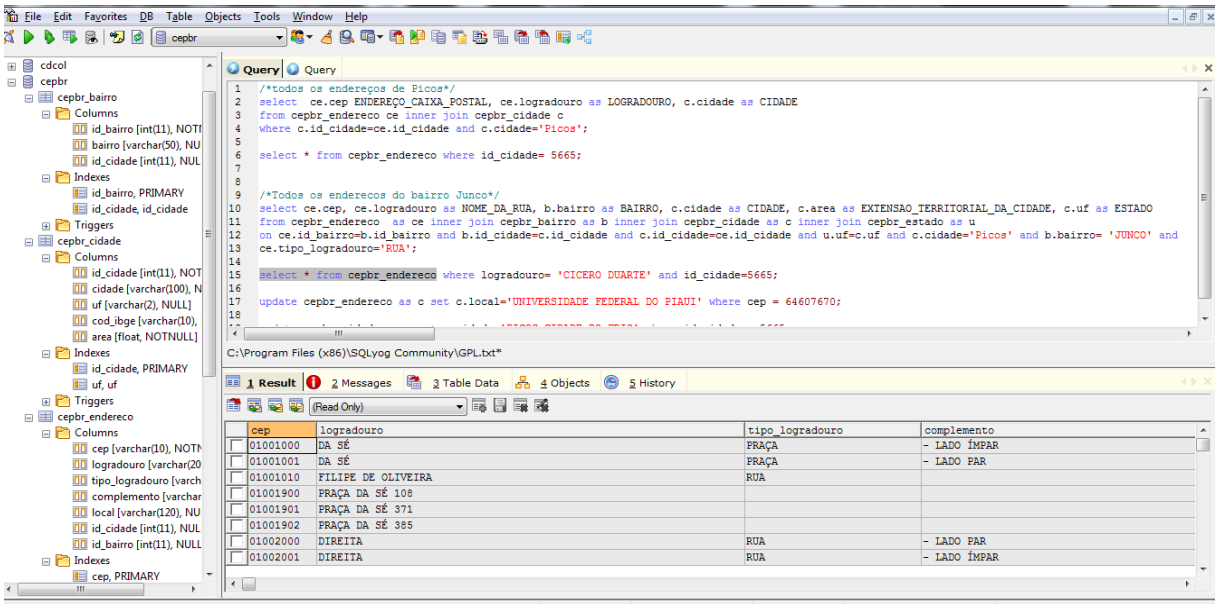

Figura 17 - Ferramenta para edição de consultas do MySQL – SQLYog

TERMO DE AUTORIZAÇÃO PARA PUBLICAÇÃO DIGITAL NA BIBLIOTECA "JOSÉ ALBANO DE MACEDO"

Identificação do Tipo de Documento

( ) Tese

( ) Dissertação

(x) Monografia

( ) Artigo

Eu, MARIA VIVIANE DE ARAÚJO SOUSA,

autorizo com base na Lei Federal nº 9.610 de 19 de Fevereiro de 1998 e na Lei nº 10.973 de 02 de dezembro de 2004, a biblioteca da Universidade Federal do Piauí a divulgar, gratuitamente, sem ressarcimento de direitos autorais, o texto integral da publicação ANÁLISE EXPERIMENTAL ENTRE AS TECNOLOGIAS RELACIONAIS E NÃO RELACIONAIS UTILIZANDO O MYSQL E CASSANDRIA de minha autoria, em formato PDF, para fins de leitura e/ou impressão, pela internet a título de divulgação da produção científica gerada pela Universidade.

Picos-PI 19 de Fevereiro de 2016.

Maria Viviane de Araújo Sousa
Assinatura

Maria Viviane de Araújo Sousa
Assinatura