colloquium exactarum

# ANÁLISE COMPARATIVA ENTRE O BANCO DE DADOS CASSANDRA (MODELO NOSQL) E O POSTGRESQL (MODELO RELACIONAL) EM DUAS DIFERENTES ORGANIZAÇÕES EMPRESARIAIS

## COMPARATIVE ANALYSIS BETWEEN THE CASSANDRA (NOSQL MODEL) AND THE POSTGRESQL (RELATIONAL MODEL) DATABASES IN TWO DIFFERENT BUSINESS ORGANIZATIONS

Ricardo Ribeiro Secco, Francisco Assis da Silva, Francisco Virginio Maracci, Mário Augusto Pazoti

UNOESTE - Universidade do Oeste Paulista, Faculdade de Informática – FIPP, Presidente Prudente – SP.
rrsecco@hotmail.com, {chico, francisco, mario}@unoeste.br

**RESUMO –** Atualmente, diferentes organizações utilizam a tecnologia como instrumento auxiliador para o trabalho, nas possíveis dificuldades que possam ocorrer na gestão de informações, tanto em setores empresariais como administrativos. A escolha de Bancos de Dados deve ser criteriosa para armazenar informações relevantes, pois os bancos devem apresentar alto poder de confiabilidade. Este trabalho faz uma análise comparativa entre o Banco de Dados Cassandra (Modelo NoSQL) e o Banco de Dados PostgreSQL (Modelo Relacional). Foram realizados testes com os Bancos de Dados Cassandra e PostgreSQL, para que possa ser verificada a atuação dos mesmos em duas organizações empresariais diferentes. As empresas X e Y utilizavam um mesmo Banco de Dados relacional para registrar produtos. Testes realizados com a implantação do Cassandra nessas empresas comprovaram sua alta escalabilidade e agilidade, bem como suas vantagens e desvantagens.
**Palavras-chave:** banco de dados; Cassandra; PostgreSQL; SGBD; SQL; NoSQL.

**ABSTRACT –** Nowadays, different organizations use technology as a helper tool for the job, the possible difficulties that may occur in the information management, both in business and administrative sectors. The Database choice should be careful to store relevant information, since the database must provide high power reliability. This work makes a comparative analysis between the Cassandra (NoSQL Model) and the PostgreSQL (Relational Model) databases. Tests were performed with Cassandra and PostgreSQL databases, so that the performance can be checked the same in two different business organizations. The X and Y companies use the same relational database to record products. Tests performed with the implantation of Cassandra in these companies proved their high scalability and agility as well as their advantages and disadvantages.
**Keywords:** database; Cassandra; PostgreSQL; DBMS; SQL; NoSQL.

Recebido em: 03/08/2015
Revisado em: 07/11/2016
Aprovado em: 10/11/2016

## 1. INTRODUÇÃO

Esse trabalho apresenta fundamentos, características e diferenças entre o banco de dados relacional PostgreSQL (POSTGRESQL, 2015) e banco de dados não-relacional Cassandra. Algumas comparações são tratadas entre os mesmos, para que se possa estabelecer uma compreensão da implantação desses dois bancos de dados.

É importante realçar que o modelo relacional baseado em SQL (*Structured Query Language*), prevalece há aproximadamente 30 anos. Para a escolha de um armazenamento de dados é unânime a escolha de um banco de dados relacional. Com o grande crescimento dos volumes de dados, observa-se que os bancos relacionais possuem limitações quando se trata de escalabilidade de um sistema.

Diante deste panorama surgem modelos alternativos, para sanar algumas necessidades, como um banco de dados, conhecido como NoSQL (termo usado para definir banco de dados não-relacional). O novo modelo de banco surgiu com a proposta de gerenciar grandes volumes de dados, objetivando um alto desempenho e disponibilidade. Neste são apresentadas as características de dois bancos de dados: PostgreSQL (modelo relacional) e Cassandra (modelo NoSQL).

De acordo com Toth (2011), o armazenamento de dados relacionais tornou-se arcaico para determinados serviços que precisam guardar grandes volumes de dados com demasiada rapidez e eficiência. Dessa forma, a solução para esses fins foi o surgimento dos bancos de dados NoSQL[1]. Nessa nova abordagem, os bancos de dados tornaram-se mais simples, porém mais reforçados. Os bancos de dados NoSQL possuem tolerância ao particionamento de dados, auxiliando na escalabilidade (TOTH, 2011).

Um exemplo a ser citado, foi o problema que o Facebook possuía com as buscas nas mensagens enviadas entre usuários em comum, o grande volume de dados, bem como, a característica de crescimento rápido. Como solução, surgiu o Cassandra (LAKASHMAN E MALIK, 2010), um sistema de armazenamento distribuído e altamente escalável para dados estruturados.

Cood (1985) estabeleceu um conjunto de 12 regras para a determinação de um Banco de Dados realmente relacional. Segundo esses preceitos, passa-se a ser discutida a fidelidade de SGBD ao modelo relacional.

[1] NoSQL, acrônimo de "Not Only SQL", é um termo genérico para uma classe definida de bancos de dados que diferem do modelo clássico relacional, rompendo uma longa história de Banco de Dados relacionais com propriedades transacionais (TOTH, 2011).

Machado e Abreu (2008) por sua vez, afirmam que, raros são os bancos de dados que se enquadram em mais do que 12 dessas regras. Regras que estão listadas a seguir:

1. Toda informação é apresentada num Banco de Dados relacional é apresentada a nível lógico por valores em tabelas;

2. Todo dado em um Banco de Dados relacional tem a garantia de ser logicamente acessível, recorrendo-se a uma combinação do nome da tabela, um valor de chave e o nome da coluna;

3. Tratamento sistemático de valores nulos (ausência de dados nulos);

4. O dicionário de dados (catálogo) relacional ativo é baseado no modelo relacional;

5. O SGBD relacional deve ter uma linguagem para definição, detalhamento e manipulação dos dados;

6. Tratamento das atualizações de visões dos dados;

7. Tratamento de alto nível para inserção, atualização e eliminação de dados;

8. Independência dos dados físicos (mudança na memória e no método de acesso);

9. Independência de dados lógicos (mudanças de qualquer tipo nas tabelas básicas, ex: divisão de uma tabela por linha ou coluna);

10. Independência das restrições de integridade;

11. Independência de distribuição;

12. Não subversão das regras de integridade ou restrições quando se utiliza uma linguagem de baixo nível.

Segundo Machado e Abreu (2008) o Modelo relacional deve apresentar as seguintes características:

- Uma tabela é acessível por qualquer campo (atributo) independente se esse é declarado como chave ou não;
- O relacionamento entre conjunto de dados (tabelas) não existe fisicamente, pois este relacionamento é apenas lógico e representado através das chaves estrangeiras;
- Utilização de linguagens auto-contidas e não procedurais;
- Os ambientes relacionais possuem um otimizador estratégico para escolher o melhor caminho para recuperação dos dados.

Guimarães (2003) assegura que muitas foram as alterações na recomendação de alternativas ao uso do Modelo Relacional. Portanto, seus desenvolvedores iniciaram um modo alternativo para modelar as bases de dados. A estrutura utilizada anteriormente transformou-se em uma dificuldade a ser contornada, todos os recursos solucionáveis ofertados buscavam a eliminação, bem como, a minimização dessa estruturação.

O principal objetivo dos projetistas de Banco de Dados das grandes organizações foi

o desenvolvimento de uma estratégia de armazenamento livre de estruturas e regras anteriores. Dessa forma, as soluções encontradas assemelhavam-se aos sistemas de gerenciamento de arquivos. De modo que, as soluções afastavam-se de regras presentes no Modelo Relacional, entretanto, flexibilizavam os sistemas de Banco de Dados para as particularidades de cada instituição (GUIMARÃES, 2003).

Em 1998, o Banco de Dados Cassandra, denominado NoSQL, surgiu baseado em um Banco de Dados e na arquitetura relacional. Em seguida, surgiu a abreviação de *NotOnly* SQL (não apenas SQL). As soluções de NoSQL (NOSQL, 2015) não visavam a substituição do Modelo Relacional, somente em casos que necessitavam de maior flexibilidade na estruturação do banco.

Segundo Lakashman e Malik (2010), outras práticas de sistemas não relacionais surgiram em 2004 (BigTable: um Banco de Dados do Google com alta performance, promotor de escalabilidade e disponibilidade) e 2007 (Sistema Dynamo: um Banco de Dados não relacional, de alta disponibilidade, utilizado pelos Web Services da Amazon).

Projetos iniciados posteriormente também tiveram sua importância. Em 2008, desenvolvido pelo site Facebook o projeto do Banco de Dados Cassandra focava um sistema distribuído, com alta disponibilidade. (MUTHUKKARUPPAN, 2010).

Seguidamente, em 2009 o Banco de Dados MongoDB (MONGODB, 2015) direcionado a documentos, de alta performance e livre de esquemas, possuía características semelhantes ao CouchDB[2]. O CouchDB (ANDERSON; SLATER; LEHNARDT, 2009) foi desenvolvido com a meta de tornar-se o Banco de Dados de fato para desenvolvimento de aplicativos da Web. É um sistema de software livre de gerenciamento de Banco de Dados.

Em 2010, o Cassandra evidenciando sua importância, desbancou o MySQL como Banco de Dados do Twitter. Ao mesmo tempo, o desenvolvimento do Apache CouchDB, visto como um Banco de Dados livre e orientado a documentos, foi direcionado para suportar computação em larga escala. As soluções apresentadas oferecem por um lado, características em comum e por outro, particularidades que os distinguem.

Segundo Machado e Abreu (2008) com o uso da padronização do NoSQL, algumas vantagens são vistas como diretas:

- Independência de fabricante;
- Portabilidade entre computadores, desde um computador pessoal, de uma estação de trabalho e de outras corporações;
- Redução dos custos como treinamento, as aplicações movimentam-se

[2] O termo "Couch" é um acrônimo para "*Cluster Of Unreliable Commodity Hardware*", que reflete a meta do CouchDB de ser extremamente escalável, oferecendo alta disponibilidade e confiabilidade.

internamente na empresa sem reciclagem da equipe de desenvolvimento;

- Inglês estruturado de alto nível, formado por um conjunto simples de sentenças, capaz de oferecer entendimento;
- Consulta interativa, acesso rápido aos dados, as respostas a questões complexas surgem em segundos;
- Múltipla visão dos dados, o criador do Banco de Dados pode obter diferentes visões;
- Definição dinâmica dos dados, as estruturas armazenadas podem ser alteradas com flexibilidade.

Entretanto, Machado e Abreu (2008) apontam por outro lado, desvantagens elencadas sobre o NoSQL. Os autores expõem que, a padronização leva a uma inibição da criatividade, pois o desenvolvedor das aplicações passa a ficar preso a soluções uniformizadas, sem que possa sofrer alterações benéficas. Do mesmo modo que, está longe de ser uma linguagem relacional ideal.

Sobre as vantagens e desvantagens apresentadas no Banco de Dados NoSQL, Machado e Abreu (2008) afirmam que, mesmo enfrentando alguns problemas e críticas, este banco é capaz de auxiliar de forma intensa a vida dos usuários e analistas no trabalho de manipulação dos dados armazenados em um banco relacional.

Neste trabalho buscou-se mostrar o desempenho dos dois bancos de dados Cassandra (Modelo NoSQL) e o PostgreSQL (Modelo Relacional) em duas diferentes organizações empresariais (empresa X e Y, assim nomeadas), de modo que, a análise comparativa possa ser realizada nesses dois bancos. O Banco de Dados NoSQL (Cassandra) foi utilizado nas organizações, como meio de substituição do Banco de Dados relacional, como recurso tecnológico inovador com o intuito de favorecer o desempenho, ampliando suas necessidades de armazenamento das diferentes informações, com códigos desenvolvidos na linguagem de programação Java.

As demais seções deste trabalho estão organizadas da seguinte maneira: na Seção 2 são relatados os Conceitos Fundamentais dos dois bancos de dados tratados neste trabalho; na Seção 3 é detalhada a Metodologia proposta; na Seção 4 é apresentada a análise de desempenho conclusiva dos bancos PostgreSQL e Cassandra; por fim, a Seção 5 são apresentadas as considerações finais do trabalho.

## 2. CONCEITOS FUNDAMENTAIS

O PostgreSQL é um sistema de gerenciamento de Banco de Dados objeto-relacional (ORDBMS) (STONEBRAKER; HANSON; HONG, 1987). Segundo Simkovics

(1998) o PostgreSQL é um Banco de Dados de código aberto avançado, sua implementação iniciou-se em 1986 e passou por várias versões. Este sistema pode ser usado, modificado e distribuído por qualquer pessoa para fins privados, comerciais ou acadêmicos.

O PostgreSQL é capaz de suportar grande parte do padrão SQL, de modo a oferecer características modernas, como: consultas complexas, chaves estrangeiras, integridade transacional, controle de concorrência, entre outras. Além disso, o PostgreSQL pode ser programado pelo usuário de várias maneiras, por exemplo, adicionando tipos de dados, funções, operadores, funções agregadas, métodos de índice e linguagens (OLSON, 1993).

Por outro lado, o projeto Apache Cassandra mantém um Banco de Dados altamente escalável de segunda geração, aliando a arquitetura completamente distribuída do Dynamo da Amazon com o modelo de dados baseado em famílias de colunas do Bigtable da Google.

De Candia *et al*. (2005) assegura que o Dynamo da Amazon é um Banco de Dados rápido, flexível e confiável. Totalmente gerenciado, representa uma escolha adequada para aplicativos de mobilidade, tecnologia de anúncios, entre outros.

O Bigtable é um sistema distribuído de armazenamento de dados em larga escala, muito utilizado em projetos da Google, facilmente escalável. Chang *et al*. (2008) afirma que o modelo de uma tabela é um mapa esparso e distribuído, sendo organizado em três dimensões (linha, coluna e hora).

O Cassandra foi inicialmente criado pelo Facebook, tendo seu código fonte liberado para a Fundação Apache em 2008. Desde então é mantido por esta, através de desenvolvedores e contribuidores de diversas empresas.

O Cassandra veio para que o usuário tenha um tempo de resposta mais acelerado, evidentemente rápido em relação ao Banco de Dados relacional. Este trabalho contribui com uma visão específica sobre o desempenho em acessos simultâneos possível com a utilização do Banco de Dados Cassandra (Modelo NoSQL), com a apresentação de suas vantagens e desvantagens quando utilizados em empesas de pequeno e grande porte, comparando-o com os sistemas de armazenamento comumente utilizados, os Bancos de Dados Relacionais.

O banco de dados Cassandra é capaz de implementar o conceito de supercolunas, isto é, colunas que carregam subcolunas. Baseado no Google Bigtable, o Cassandra possui as seguintes nomenclaturas:

- *Keyspace*: agrupamento de famílias de colunas (similar a um Banco de Dados do modelo relacional);

• *Column Family*: agrupamento de colunas com ordenação fixada (similar a uma tabela);

• *Row Key* ou simplemente *Key*: chave que representa uma linha de colunas (similar a uma chave primária);

• *Column*: representação de um valor, contendo o tripleto: Nome (*Name*), Valor (*Value*) e *Timestamp*.

Existem pontos positivos no Apache Cassandra e muitas foram as vantagens adquiridas com sua criação. Atualmente, grandes empresas o utilizam, devido à quantidade acentuada de usuários online que o acessam simultaneamente.

As vantagens do Cassandra apresentam-se na alta escalabilidade e disponibilidade, na implementação da família de colunas NoSQL, no bom rendimento de gravação e leitura, na linguagem de consulta semelhante a SQL, na consistência ajustável, bem como, no esquema flexível (CHANG *et al*., 2008). O Cassandra armazena informações, baseado no modelo de dados de família de colunas (Figura 1), que apresenta um espaço ("keyspace 1") com duas famílias de colunas ("Column family 1" e "Column family 2"), cada coluna possui suas subdivisões com informações relevantes, dados que são armazenados com flexibilidade.

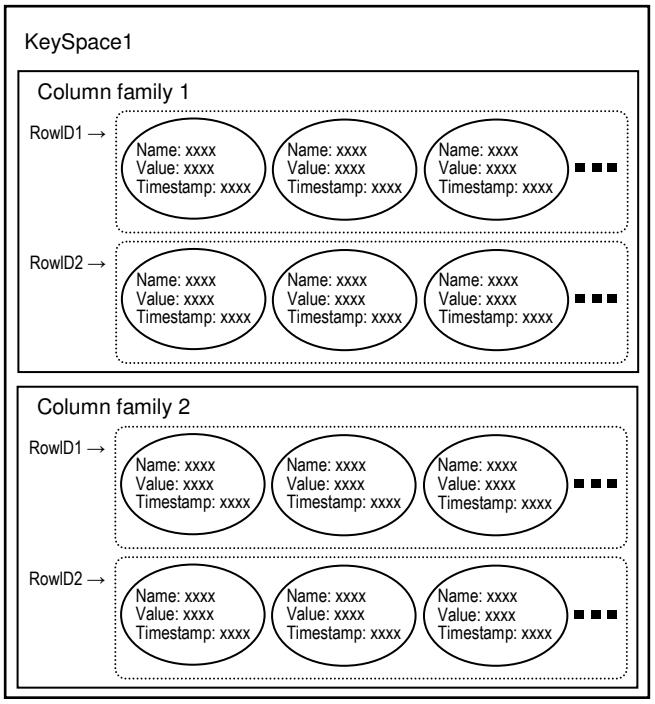



**Figura 1.** Modelo de dados do NoSQL Cassandra.
Fonte: (PERERA, 2012).

O modelo de dados do Cassandra possui colunas, linhas, famílias de colunas e *keyspaces*. Dessa forma, por exemplo, quando um usuário em uma rede social, realiza suas postagens, curte um documento postado por alguém, entre outras situações, o Cassandra apresenta uma maior velocidade em relação ao Banco de Dados relacional. A seguir, cada particularidade é descrita em detalhes.

• Coluna: a unidade mais básica do modelo de dados do Cassandra contém nome, valor e registro de data e hora.

• Linha: uma coleção de colunas rotuladas com um nome. O banco de dados Cassandra consiste em muitos "nós" de armazenamento e registra cada linha em um único "nó". Em cada linha, grava as colunas

classificadas por seus nomes. Com essa ordem de classificação, suporta consultas de fatias, nas quais, dada uma linha, os usuários podem recuperar um subconjunto de suas colunas que estejam em um intervalo de nomes de coluna.

- Família de colunas: uma coleção de linhas rotuladas com um nome. Uma família de colunas é como uma tabela no modelo relacional.
- *Keyspace*: um grupo de várias famílias de colunas juntas. É apenas um agrupamento lógico de famílias de colunas e fornece um escopo isolado para nomes.
- Por fim, as supercolunas residem em uma família de colunas que agrupa várias colunas em uma única chave.

Na possibilidade de se optar por uma estratégia NoSQL (Cassandra) ao invés de um Banco de Dados Relacional, questões básicas devem ser levadas em consideração, como: os critérios de escalonamento, a consistência de dados, bem como, a disponibilidade.

Sabe-se que, o escalonamento é essencial, pois a partir dele, os bancos NoSQL oferecem as vantagens fundamentais em relação aos relacionais, por terem sido criados para essa finalidade. Por outro lado, os sistemas relacionais apresentam uma estruturação menos flexível (LAKASHMAN E MALIK, 2010). Trata-se de uma técnica que surgiu da necessidade de aplicações com desempenho superior e alta escalabilidade, funcionalidades que os bancos de dados relacionais não eram capazes de alcançar com facilidade. Os bancos de dados ditos NoSQL possuem as seguintes características em comum, em geral eles são: não relacionais; distribuídos; horizontalmente escaláveis; possuem esquemas flexíveis; são replicáveis; possuem APIs[3] simples (MACHADO E ABREU, 2008).

## 3. METODOLOGIA

Para realizar a análise comparativa entre os Banco de Dados PostgreSQL (Modelo Relacional) e o Cassandra (Modelo NoSQL), foi utilizado uma aplicação para "Controle de Estoque", que integra vendas e compras de produtos, gerenciamento de fornecedores, além de controlar contas a receber e a pagar em duas empresas distintas de pequeno porte.

A aplicação foi desenvolvida na linguagem Java em 03 Camadas: Camada de Apresentação (Interação com o Usuário), Camada Lógica (Classes de Conexão com o Banco de Dados, utiliza o SQL – *Structured Query Language* – para PostgreSQL e CQL – *Cassandra Query Language* – para NoSQL Cassandra) e Camada de Negócio (Interação entre a Aplicação e o Banco de Dados). O MER (Modelo Entidade Relacionamento) do banco de dados utilizado por esse sistema é

[3] API (*Application Programming Interface*): é composta por uma série de funções acessíveis somente por programação, que permitem utilizar características do software menos evidentes ao utilizador tradicional.

mostrado na Figura 2. As duas empresas (X e Y) acessam o banco de Dados PostgreSQL implantado na empresa X, em que a empresa Y acessa esse banco remotamente.

Foi instalado em um único servidor, com um único core e um único disco hígido, o banco NoSQL Cassandra na empresa X juntamente com o banco PostgreSQL, e da mesma forma a empresa Y acessa esse banco remotamente. A partir desse cenário foram feitos os testes com execuções de instruções de CRUD (*Create*, *Retrieve* (ou *Read*), *Update* e *Delete*) para posteriormente analisar o desempenho dos bancos de dados PostgreSQL e NoSQL Cassandra.

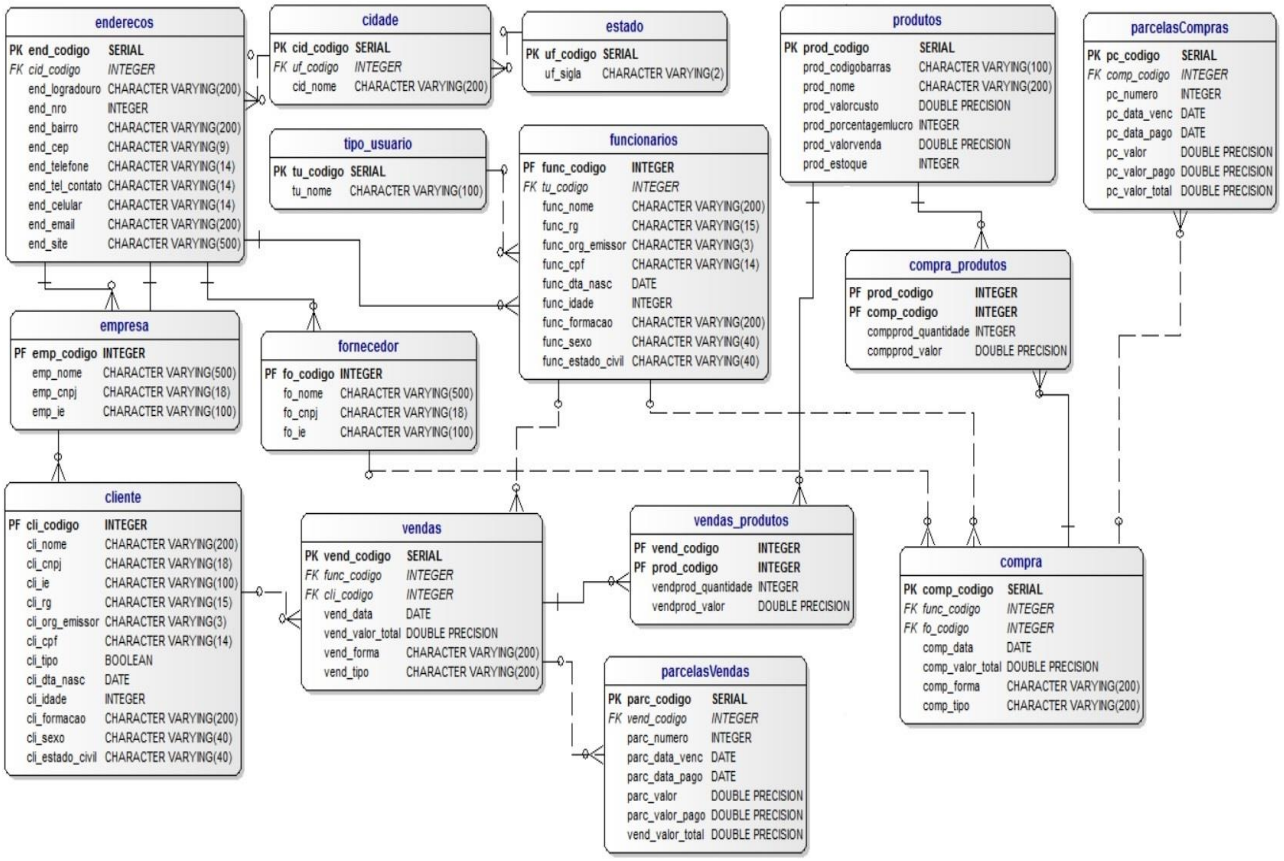



**Figura 2.** Modelo do Banco de Dados Relacional do Sistema “Controle de Estoque” presente nas empresas X e Y.

Para realizar a conexão entre os bancos de dados com a aplicação das empresas X e Y foram criadas duas classes de conexão, uma para o PostgreSQL e outra para o NoSQL Cassandra. Essas classes são responsáveis pela realização de todas as operações CRUD nos bancos.

No NoSQL Cassandra os códigos de comandos para inserção, alteração e exclusão são iguais ao do PostgreSQL, exceto na busca com condição. Na sequência é mostrada a diferença do comando de busca nos dois bancos. Primeiramente, é mostrado um exemplo de uma consulta ao banco

PostgreSQL fazendo seleção à tabela "cidade":

```
"SELECT * FROM cidade WHERE cid_codigo >= 1"
```

As linhas no NoSQL Cassandra são ordenados por um *hash* (algoritmo que mapeia dados de comprimento variável para dados de comprimento fixo), e portanto, a ordem das linhas não é importante. Usando CQL, pode-se selecionar uma linha, mesmo quando usando um particionador aleatório ou a função de TOKEN.

A função de TOKEN torna possível percorrer estes resultados particionados não ordenados; a função pode ser usada para expressar uma relação condicional sobre uma coluna de chave de partição. Neste caso, a consulta retorna linhas com base na chave de partição, em vez de sobre o valor. Segue um exemplo de consulta no NoSQL Cassandra:

```
"SELECT * FROM cidade WHERE TOKEN(cid_codigo) >= TOKEN(1)"
```

O banco de dados PostgreSQL possui um sistema gerenciador denominado de PgAdmin (software para administração do banco de dados PostgreSQL) e está disponível para Windows, GNU/Linux e FreeBSD. Por sua vez, o NoSQL Cassandra possui um sistema gerenciador denominado de DataStax OpsCenter (solução de gerenciamento e monitoramento visual baseado na Web para o NoSQL Cassandra) (DATASTAX, 2015). É importante ressaltar que para a comprovação dos testes apresentados neste trabalho, não foi usado o ambiente gráfico para o NoSQL Cassandra, foi usado o Terminal Cassandra CQL Shell para o Windows (CQLSH).

Para que a aplicação "Controle de Estoque" possa armazenar os registros no banco de dados das empresas X e Y, foram realizados os procedimentos para criação do banco de dados no ambiente PgAdmin e para a criação do banco de dados no NoSQL Cassandra, foi usado o Terminal Cassandra CQL Shell.

A diferença do código utilizado para a criação dos bancos de dados PostgreSQL e NoSQL Cassandra está representada na Tabela 1.

**Tabela 1.** Códigos comparativos para criação dos bancos de dados.

| Códigos para Criação do Banco de Dados no PostgreSQL. | Códigos para Criação do Banco de Dados no NoSQL Cassandra |
|---|---|
| **CREATE DATABASE** BancoPostgre; | **CREATE KEYSPACE** BancoCassandra **WITH** replication = { 'class' : 'SimpleStrategy', 'replication_factor' : 1 }; |

As diferenças no processo de criação das tabelas nos dois bancos de dados são apresentadas na Tabela 2, no exemplo, foi criada a tabela de "cidade".

**Tabela 2.** Códigos demostrando a diferença para a criação de tabela nos bancos de dados.

| Códigos de Criação de Tabela no Banco de Dados PostgreSQL | Códigos de Criação de Tabela no Banco de Dados NoSQL Cassandra |
|---|---|
| **CREATE TABLE** cidade (cid_codigo **SERIAL NOT NULL**, uf_codigo **INTEGER**, cid_nome **CHARACTER** VARYING(200), **CONSTRAINT** PK_cidade **PRIMARY KEY** (cid_codigo)); | **CREATE TABLE** cidade (cid_codigo **INT**, uf_codigo **INT**, cid_nome **TEXT**, **PRIMARY KEY** (cid_codigo)); |

Foi necessário migrar os dados do banco de dados PostgreSQL para o NoSQL Cassandra antes de fazer uso da aplicação "Controle de Estoque", opção disponível na própria aplicação desenvolvida.

O NoSQL Cassandra oferece várias soluções para a migração de outros bancos de dados, o mesmo possui o comando "COPY", comando esse para realizar a importação e exportação de arquivos.

Quando existe a realização da transferência dos registros é preciso exportar para um arquivo com extensão "CVS", em que os dados dos registros são separados por vírgulas, a fim de serem identificados tanto para importação quanto para exportação.

## 4. ANÁLISE DE DESEMPENHO CONCLUSIVA DOS BANCOS POSTGRESQL E NOSQL CASSANDRA

Antes de analisar as empresas X e Y, apresentar as mudanças realizadas, torna-se imprescindível atentar-se a questões como critérios de escalonamento, consistência e disponibilidade de dados, entre o PostgreSQL e NoSQL Cassandra.

Quando se fala de escalabilidade o NoSQL Cassandra oferece uma grande vantagem em relação ao banco de dados PostgreSQL, pois este não tem tanta flexibilidade, quanto se trata em um número acentuado de acesso de usuários.

Quando se trata em distribuir o banco de dados em várias máquinas, particionando os dados, o banco PostgreSQL deixa a desejar, pois obedece os critérios de normalização, com o NoSQL Cassandra acontece ao contrário.

Existem diferenças entre os mesmos, enquanto o PostgreSQL prioriza a concentração dos dados em um servidor, o NoSQL Cassandra tem como objetivo a distribuição dos dados em diversos servidores tornando mais eficiente o seu gerenciamento na questão de inserção, alteração de dados e busca. Nesse caso, se houver, por exemplo, uma falha em qualquer aplicação em rede não existirá problema, porque outras aplicações continuarão trabalhando sem perdas.

Outro aspecto importante entre os bancos de dados é o controle de concorrência, o PostgreSQL utiliza bloqueios para que um registro seja acessado somente por um usuário, o NoSQL Cassandra faz uso de outros recursos que permitem maior grau de concorrência.

O NoSQL Cassandra apresenta como regra a criação de várias cópias dos dados, não precisando bloquear os itens presentes nos dados. Dessa forma, na substituição do banco de dados PostgreSQL para o NoSQL Cassandra é necessário conhecer que a arquitetura fica vulnerável à perda de consistência, porém, ganha-se em flexibilidade, disponibilidade e desempenho. A Tabela 3 faz uma comparação, segundo Kokay (2015), sobre Escalonamento, Consistência e Disponibilidade.

**Tabela 3.** Análise comparativa entre o Banco de Dados PostgreSQL e NoSQL Cassandra.
Fonte: (KOKAY, 2015).

| | Banco de Dados Relacional (PostgreSQL) | Banco de Dados NoSQL Cassandra |
|---|---|---|
| Escalonamento | É importante lembrar que é possível o escalonamento em um modelo relacional. No entanto, é muito complexo. Possui uma natureza estruturada, a inserção dinâmica e transparente de novos nós na tabela não é realizada naturalmente. | Não possui um esquema pré-definido fazendo com que este tipo de modelo seja flexível, favorecendo a inserção transparente de outros elementos. |
| Consistência | Neste quesito, o modelo de dados se mostra forte. As suas regras de consistência são extremamente rigorosas na consistência das informações. | É realizada eventualmente no modelo: tem apenas a garantia que se não houver nenhuma atualização nos dados, todos os acessos aos itens devolverão o último valor que foi atualizado. |
| Disponibilidade | Por não conseguir trabalhar de forma eficiente com a distribuição de dados, o modelo relacional não suporta uma demanda alta de informações. | Possui um alto nível de distribuição de dados, permitindo fazer com que um enorme fluxo de solicitações aos dados seja atendido com a vantagem do sistema ficar indisponível o menor tempo possível. |

Na análise realizada nas duas empresas X e Y com os banco de dados PostgreSQL e NoSQL Cassandra, instalados na empresa X, estes apresentaram diferenças notáveis.

Na comparação referente a inserção, alteração e busca de registros, foi constatado algumas diferenças relevantes, ao inserir registros no banco NoSQL Cassandra, o mesmo não segue a ordem cronológica igual ao PostgreSQL, ou seja, não armazena os dados na ordem em que foram inseridos, não existe um controle de sequência realizado pelo banco, segue em via de regras o tempo que engloba hora, minuto, segundo e milésimo de segundo, e pelo tamanho em bytes do registro inserido. Nas questões de alteração e busca de registros, os dois

utilizam a cláusula de condição “WHERE” para identificar qual registro deve ser alterado.

Ambos possuem chave primária, aliás, a única diferença é que o NoSQL Cassandra precisa acrescentar na cláusula “WHERE” o comando “TOKEN”, colocando logo após, e entre parênteses a chave primária e qual registro será alterado. Por exemplo: “TOKEN (PRIMARY KEY)”.

Outro teste realizado no banco PostgreSQL e no banco NoSQL Cassandra, foi o de desempenho, para a verificação da comparação do tempo de processamento nas operações de inserção, alteração, exclusão e seleção. Para esses testes, foi escolhida uma tabela com três colunas existente nos dois bancos de dados. A primeira coluna representa a chave primária do tipo inteiro, a segunda coluna do tipo texto (*character varying* no banco PostgreSQL) e a terceira do tipo inteiro.

A Tabela 4 representa os testes de comparações de desempenho entre os banco de dados PostreSQL e NoSQL Cassandra. Teste esse realizado com as operações básica para inserção, alteração, exclusão e tempos de processamentos. O teste descreve qual o ambiente foi usado para executar os comandos, descreve a quantidade de linhas processadas e o tempo de processamento para cada comando e o tempo total, lembrando que o tempo é representado em milissegundos, e os resultados é fornecido pelo ambiente usado para cada banco de dados.

**Tabela 4.** Análise comparativa realizada entre o banco de dados PostgreSQL e o banco de dados NoSQL Cassandra em relação a quantidade de linhas processadas e ao tempo de processamento para comandos de Inserção, Alteração e Exclusão.

| | |
|---|---|
| **SQL no Banco de Dados PostgreSQL** | **Teste de Desempenho - Inserção** |
| | Foi realizado o teste no ambiente PgAdmin III, com 5564 linhas processadas, e o tempo de resposta foi de 161 milissegundos, calculado pelo ambiente, em que todas as linha foram executadas de uma única vez. Exemplo do comando usado:<br>**INSERT INTO** tabela (coluna1, coluna2, coluna3) **VALUES** (valor1, valor2, valor3); |
| | **Teste de Desempenho - Alteração** |
| | Foi realizado o teste no ambiente PgAdmin III, na alteração da coluna do tipo "*character varying*", com tamanho 200. Nesse teste foram processadas 5564 linhas, e o tempo de resposta foi de 209 milissegundos, tempo calculado pelo ambiente, em que foram executadas todas as linha de uma vez. Exemplo do comando usado:<br>**UPDATE** tabela **SET** coluna1 = valor1, coluna2 = valor2, coluna3 = valor3 **WHERE** coluna1 **=** valor1; |
| | **Teste de Desempenho - Exclusão** |
| | Foi realizado o teste no ambiente PgAdmin III, para excluir registros comparados pela chave primária, também foram processadas 5564 linhas, e o tempo de resposta foi de 209 milissegundos, tempo calculado pelo ambiente, com todas as linha executadas de uma vez. Exemplo do comando usado:<br>**DELETE FROM** tabela **WHERE** coluna **=** valor; |
| | **Teste de Desempenho - Processamento** |
| | Foi realizado o teste no ambiente PgAdmin III, para consultar registros do banco de dados. Para esse teste foi feita uma seleção de todos os campos, com a condição feita pela chave primária da tabela. O Resultado do teste processado foi realizado com 5564 linhas, e o tempo de resposta foi de 140 milissegundos. Lembrando que por se tratar de um banco relacional, foi preciso realizar a comparação das tabelas envolvidas. O tempo foi calculado pelo ambiente executando todas as linhas de uma vez. Exemplo do comando usado:<br>**SELECT** tabela.coluna1, tabela.coluna2, tabela.coluna3 **FROM** tabela **WHERE** tabela.coluna1 **>** valor1; |
| **CQL no Banco de Dados NoSQL Cassandra** | **Teste de Desempenho - Inserção** |
| | Foi realizado o teste no Terminal Cassandra CQL Shell, com 5564 linhas processadas, e o tempo de resposta foi de 21,12 milissegundos, tempo calculado pelo terminal, com todas as linhas executadas de uma única vez. Exemplo do comando usado:<br>**INSERT INTO** tabela (coluna1, coluna2, coluna3) **VALUES** (valor1, valor2, valor3); |
| | **Teste de Desempenho - Alteração** |
| | Foi realizado o teste no Terminal Cassandra CQL Shell, na alteração da coluna do tipo "text", nesse teste foi processado, 5564 linhas, e o tempo de resposta foi de 148 milissegundos, tempo calculado pelo terminal, foram executado todas as linha de uma vez. Exemplo do comando usado:<br>**UPDATE** tabela **SET** coluna2 = valor2, coluna3 = valor3 **WHERE TOKEN**(coluna1) **= TOKEN**(valor1); |
| | **Teste de Desempenho - Exclusão** |
| | Foi realizado o teste no Terminal Cassandra CQL Shell, para excluir os registros comparados pela chave primária. Foram processadas 5564 linhas, e o tempo de resposta foi de 62 milissegundos, tempo calculado pelo terminal, com todas as linhas executadas de uma vez. Exemplo do comando usado:<br>**DELETE FROM** tabela **WHERE** coluna **=** valor; |
| | **Teste de Desempenho - Processamento** |
| | Foi realizado o teste no Terminal Cassandra CQL Shell, para buscar registros do banco de dados, para esse teste foi feita uma seleção de todos os campos, com condição pela chave primária da tabela. O Resultado do teste processado foi realizado com 5564 linhas, e o tempo de resposta foi de 56 milissegundos. O tempo foi calculado pelo terminal, com todas as linhas executadas de uma vez. Exemplo do comando usado:<br>**SELECT** coluna1, coluna2, coluna3 **FROM** tabela **WHERE TOKEN**(coluna1) **>** valor1; |
| **Soma dos tempos de Resposta no PostgreSQL** | 719 milissegundos. |
| **Soma dos tempos de Resposta no NoSQL Cassandra** | 287,12 milissegundos. |

A análise feitas neste artigo comprovam que o NoSQL Cassandra tem um tempo de resposta mais rápido do que o PostgreSQL, quando se trata das operações

mais utilizadas no banco de dados, mesmo que os testes foram feitos em ambiente não distribuído.

A eficácia do banco de dados NoSQL Cassandra fica comprovada, pois sua capacidade de armazenamento, bem como, a agilidade no oferecimento das informações é gigantesca muito mais ascendente que no banco de dados relacional.

Além disso, a funcionalidade por apresentar todas as informações em uma única tabela também é uma vantagem, mas ocorre o oposto em transações diversas, quando a existências de diferentes tabelas devem ser envolvidas nos registros. O NoSQL Cassandra sempre armazena dados de forma que as colunas sejam classificadas de acordo com seus nomes. Isso facilita a procura de dados em uma coluna usando consultas de fatias, mas é mais difícil procurar dados em uma linha, a menos que o usuário use um particionador que preserve a ordem, exemplo: comando “TOKEN”.

Outra diferença fundamental é que o NoSQL Cassandra não suporta junções de tabelas, agregações e a maioria dos métodos do SQL, pois não suporta chaves estrangeiras, portanto, não pode gerenciar a consistência de dados para o usuário. Por isso, o aplicativo “Controle de Estoque” teve que lidar com a consistência de dados da chave primária, pois, as chaves não podem ser alteradas. Deve-se ter cuidado com as chaves, devem ser exclusivas em seu escopo, caso a mesma chave seja usada duas vezes, os dados serão sobrescritos, portanto recomenda-se usar chaves compostas ou incluir na chave um valor aleatório ou registro de data e hora, isso serve para evitar que a mesma coluna tenha o mesmo nome e a mesma classificação, o NoSQL Cassandra inclui no registro a data e hora após o nome da coluna.

O NoSQL Cassandra não suporta operações atômicas e sim operações idempotentes. Se uma operação falhar, é possível tentar novamente, isso é um mecanismo para recuperação de falhas temporárias, ou seja, o NoSQL Cassandra suporta operações em lote, mas elas também não têm garantia de atomicidade.

Sua eficácia se dá pelo tempo de resposta extremamente rápido, pois o mesmo usa um algoritmo *hash* para calcular o endereço de chaves de cada item de dados armazenado nele (por exemplo, nome da coluna, ID de linha), e não pelo modo como suas informações são armazenadas.

No NoSQL Cassandra não existe a mesma consistência do Banco PostgreSQL. É rápido nas buscas, porém a manipulação dos dados é confusa.

Outra eficácia importante é que a arquitetura resultante é altamente escalável, sendo possível desenvolver um *cluster* do NoSQL Cassandra com vários nós, capaz de lidar com TeraBytes a PetaBytes de dados. A leveza e a capacidade de armazenagem

extensa tornam o NoSQL Cassandra eficaz em organizações amplas, porém nas empresas X e Y torna-se inviável, pois são empresas de pequeno porte.

Segundo Perera (2012) as vantagens do Banco de Dados podem ser elencadas facilmente. É fundamental para um desenvolvedor observar pontos negativos. Na sequência, na Tabela 5, são apresentadas as vantagens e desvantagens do Apache NoSQL Cassandra em sua utilização.

**Tabela 5.** Vantagens e desvantagens do Apache NoSQL em sua utilização.
Fonte: (PERERA, 2012)

| Vantagens do Apache NoSQL Cassandra | Desvantagens do Apache NoSQL Cassandra |
|---|---|
| Esquema flexível e sistema distribuído. Alta escalabilidade e disponibilidade, sem um ponto único de falha. | Não suporta transações ACID (Propriedades fundamentais nos processos transacionais). |
| Implementação da família de colunas NoSQL. Os nomes das colunas podem incluir dados. As linhas das colunas podem ter muitas colunas. | Não suporta chaves estrangeiras. Portanto, não pode gerenciar a consistência dos dados para o usuário. Assim, os usuários não podem alterar a chaves. |
| Rendimento de gravação muito alto e bom rendimento de leitura. Alternativa flexível. | Não suporta operações atômicas, suporta operações idempotentes e deixa o sistema no mesmo estado. |
| Armazena dados, as colunas são classificadas de acordo com seus nomes, facilitando a procura dos dados em uma coluna. | Dificuldade na procura de dados em uma linha, com exceção se o usuário utilizar um particionador que preserve a ordem. |
| Muitos nós de armazenamento e distribui os dados entre eles. | Designa cada item de dados ao nó que é responsável por armazenar o item de dados. |
| Esquemas ajustados para as consultas exigidas pelo aplicativo. | Não suporta JOINs (busca de dados em várias tabelas) e a maioria dos métodos de procura de SQL. |
| As supercolunas do Cassandra podem ser úteis ao modelar dados em mais de um nível, pois inclui mais um nível na hierarquia. | As supercolunas não fornecem eficiência adicional e não suportam índices secundários. |

Pereira-Neto (2012) elenca as vantagens do banco relacional PostgreSQL. Entretanto, adverte que as desvantagens são em número menor que as apresentadas no Apache NoSQL Cassandra. As vantagens e desvantagens do banco de dados PostgreSQL, segundo o autor, são apresentadas na Tabela 6.

**Tabela 5.** Vantagens e desvantagens do Apache NoSQL em sua utilização.
Fonte: (PEREIRA-NETO, 2012).

| Vantagens do Banco de dados PostgresSQL | Desvantagens do Banco de dados PostgreSQL |
|---|---|
| Oferece estabilidade, segurança, confiabilidade para uso. Possui alta performance, é de fácil administração e com gratuidade comercial, particular ou acadêmico. | Os programas para gerenciamento e auxílio são pagos, e os programas disponíveis gratuitos são considerados péssimos. |
| Suporta completas transações ACID (Propriedades fundamentais nos processos transacionais). É robusto e possui muitas funções. O conceito de transações está ligado à segurança das operações. | Os resultados das buscas são considerados lentos quando comparados a outros modelos de bancos. |
| Possui interface com diversos ambientes e linguagens de programação, como C, C++, MS Visual Basic, Perl e Java. | Muitas são as regras de relacionamento do Postgre, denotando em ponto negativo, acarreta lentidão nas informações. |

## 5. CONSIDERAÇÕES FINAIS

No desenvolvimento de qualquer aplicação para computadores que faça uso de um banco para armazenamento de dados, a questão primordial é a escolha de um sistema gerenciador de Banco de Dados.

Diante da análise comparativa realizada entre os bancos Apache NoSQL Cassandra e o PostgreSQL, nota-se que o banco de dados relacional (PostgreSQL) é um destacado pelo código aberto e por ser um dos mais avançados em termos de recursos. Por possuir um Sistema Gerenciador de Base de Dados Relacional (SGBDR), sua administração é fácil e dinâmica, além de ser gratuito. Entretanto, sua vagarosidade na busca das informações, devido as muitas regras de relacionamento entre as tabelas promovem um tempo menos acelerado na aquisição dos resultados.

Observa-se também que para a administração e armazenamento dos dados, o SGBD é a alternativa mais eficiente e compatível com as tecnologias utilizadas no mercado. Além disso, todos os recursos disponíveis proporcionam ao programador e ao administrador de Banco de Dados a realização das tarefas e o atendimento das perspectivas mais específicas. Em relação às empresas X e Y, dos testes realizados, por serem empresas pequenas, o banco relacional atende as expectativas de modo coerente, com detalhes imprescindíveis.

O armazenamento realizado no banco de dados NoSQL (Cassandra) é uma alternativa flexível e escalável, por ser uma implementação de família de colunas capaz de suportar o modelo de dados de uma única tabela (Big Table). Utiliza ainda, aspectos de arquitetura introduzidos por Amazon Dynamo. Possui disponibilidade, rendimento de gravação alto, bom rendimento de leitura, suporte para procura por índices secundários, consistência ajustável e suporte para replicação, além de esquema flexível. Entretanto, pontos negativos que foram encontrados nesta análise, os muitos nós de armazenamento, demonstram que todas as linhas estão armazenadas em um único nó. Em cada linha, o NoSQL Cassandra sempre armazena as colunas classificadas por seus nomes. Usando essa ordem de classificação, o banco suporta consultas de fatia, nas quais, dada uma linha, os usuários podem recuperar um subconjunto de suas colunas que estejam em um dado intervalo de nomes de coluna. Desse modo, qualquer consulta que venha a ser realizada tanto pela empresa X, quanto pela empresa Y, buscará inicialmente uma fatia com o intervalo que retornará todas as colunas cujos nomes estão unidos a ela. A procura de dados é difícil em uma linha, a menos que o usuário use um particionador que possibilite a preservação da ordem.

Para a administração e armazenamento dos dados de formas alternativas, o Apache Cassandra é considerado uma boa escolha para empresas de grande porte. As empresas X e Y são consideradas o contrário, o que inviabiliza a utilização no Cassandra para armazenamento de seus dados, haja vista que, os detalhes são essenciais nela, seus produtos devem ser interligados nas buscas.

## REFERÊNCIAS


ANDERSON, J.C.; SLATER, N.; LEHNARDT, J. **CouchDB: The Definitive Guide**. 1ª edição: O'Reilly Media, 2009.

CHANG, F. et al. Bigtable: A distributed storage system for Structured data. **ACM Transactions on Computer Systems (TOCS)**, v. 26, n. 2, 2008. https://doi.org/10.1145/1365815.1365816

COOD, E.F. **Is your DBMS Really Relational?** EUA: Computer-World, 1985.

**DataStax.** 2015. Disponível em: <http:// http://www.datastax.com>. Acesso em: 02 maio 2015.

GUIMARÃES, C.C. **Fundamentos de banco de dados: modelagem, projeto e linguagem SQL.** Campinas: Unicamp, 2003.

KOKAY, M.C. Banco de dados NoSQL: um novo paradigm. **SQL Magazine,** 2015. Disponível em: <http://www.devmedia.com.br/websys.5/ webreader.asp?cat=2&artigo=4773&revista= sqlmagazine_102#a-4773>. Acesso em: 10 abr. 2015.

LAKASHMAN, A.; MALIK, P. Cassandra – a decentralized structured storage system. **ACM SIGOPS Operating Systems Review,** v. 44, n. 2, p. 35-40, 2010. https://doi.org/10.1145/1773912.1773922

MACHADO, F.N.R.; ABREU, M.P. **Projeto de banco de dados.** 15. ed. São Paulo: Érica, 2008.

MUTHUKKARUPPAN, K. **The underlying technology of messages**. 2010. Disponível em: <http://www.facebook.com/ notes/facebookengineering/the-underlying-technology-ofmessages/454991608919>. Acesso em: 10 fev. 2015.

**NoSQL Relational Database Management System: Home Page. Strozzi.it.** 2015. Disponível em: <http://www.strozzi.it/cgi-bin/CSA/tw7/I/ en_US/nosql/Home Page>. Acesso em: 13 abr. 2015.

**Official MongoDB Project Website.** 2015. Disponível em: <http://www.mongodb.org/ display/DOCS/Home>. Acesso em: 13 abr. 2015.

OLSON, N. **Partial indexing in POSTGRES: research project**. University of California, UCB Engin T7.49.1993 O676, 1993.

PERERA, S. **Considerações sobre o Banco de Dados Apache Cassandra. Quais são as vantagens e desvantagens desse banco de dados NoSQL?** 2012. Disponível em: <http://www.ibm.com/developerworks/br/library/os-apache-cassandra/>. Acesso em: 13 fev. 2015.

PEREIRA-NETO, A. **PostgreSQL:** técnicas avançadas versoes open source 7.x e 8.x. São Paulo: Érica, 2012.

**PostgreSQL 8.3.16 Documentation.** 2015. Disponível em: <http://www.postgresql.org/ docs/8.3/static/index.html>. Acesso em: 30 abr. 2015.

SIMKOVICS, S. **Enhancement of the ANSI SQL Implementation of PostgreSQL.** Vienna: Department of Information Systems, Vienna University of Technology, November 29, 1998.

STONEBRAKER, M.; HANSON E.; HONG, C. H. The design of the POSTGRES rules system. **Proc. IEEE Conference on Data Engineering**, feb. 1987.

TOTH, R.M. **AbordagemNoSQL** – uma real alternativa. 2011. Disponível em: <http://www2.sor.ufscar.br/~verdi/topicosCloud/nosql_artigo.pdf>. Acesso em: 16 maio 2015.