UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC
CURSO DE CIÊNCIA DA COMPUTAÇÃO

ORLANDO VITALI WERNER

ESTUDO COMPARATIVO ENTRE BANCO DE DADOS RELACIONAIS, NOSQL E NEWSQL

CRICIÚMA
2014

ORLANDO VITALI WERNER

ESTUDO COMPARATIVO ENTRE BANCO DE DADOS RELACIONAIS, NOSQL E NEWSQL

Trabalho de Conclusão, apresentado para obtenção de grau de Bacharel no curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC.

Orientador: Prof. MSc. Gustavo Bisognin

CRICIÚMA

2014

ORLANDO VITALI WERNER

ESTUDO COMPARATIVO ENTRE BANCO DE DADOS RELACIONAIS, NOSQL E NEWSQL NA UTILIZAÇÃO POR APLICAÇÕES WEB

Trabalho de Conclusão de Curso aprovado pela Banca Examinadora para obtenção do Grau de Bacharel, no Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC, com Linha de Pesquisa em Banco de dados.

Criciúma, 27 de novembro de 2014

BANCA EXAMINADORA

Prof. Gustavo Bisognin - MSc – (UNESC) – Orientador

Prof. Gilberto Vieira da Silva –Esp- (UNESC)

Prof. Paracelso de Oliveira Caldas - MSc - (UNESC)

AGRADECIMENTOS

Agradeço aqui pela compreensão e oportunidade dada pelo meu Orientador Gustavo Bisognin que me guiou durante este ano. À Ema Software pelo apoio e incentivos e à minha família pela paciência e apoio durante este período de desenvolvimento deste projeto.

.

RESUMO

Bancos de dados são importantes ferramentas para armazenamento e gerenciamento de informação. Atualmente existem diversos modelos de bancos de dados disponíveis no mercado. Neste projeto estão descritos os modelos teóricos dos bancos de dados relacionais e NoSQL e uma breve apresentação dos bancos de dados NewSQL, que propõem um banco de dados com características de bancos relacionais e NoSQL ao mesmo tempo. O objetivo do projeto é formar uma base de conhecimento sobre os três modelos de bancos de dados, apresentando principalmente as características dos bancos NewSQL. O estudo compara as caraterísticas, segundo os desenvolvedores, de alguns bancos de dados dos três modelos e executa na pratica uma para avaliação funcional através de comparações entre os bancos relacionais e NewSQL utilizando o teste TPC-C e entre os bancos NoSQL e NewSQL utilizando o teste YCSB, para avaliar as características de cada modelo.

Palavras-chave: Banco de dados, NewSQL, NoSQL.

ABSTRACT

Databases are important tools for storing and managing information. Actually exists a lot of database models available on the market. This project described theoretical models of relational databases and NoSQL database and a brief presentation of the NewSQL database model, which propose a database with relational and NoSQL features. The objective of this project is form a knowledge base about the three databases models, mainly presenting the characteristics of NewSQL database model. The study compares the characteristics, according to the developers, of the three databases models and performs in practice a functional assessment through comparisons between relational databases and NewSQL using the TPC-C test and between NoSQL and NewSQL databases using the YCSB test, to assess the characteristics of each model.

Key words: Database, NoSQL, NewSQL.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACID	Atomicidade, Consistência, Isolamento, Durabilidade
BSON	Binary JSON
CAP	Consistency, Availability, Partition tolerance
JSON	JavaScript Object Notation
LDBC	Liberty DataBase Connectivity
OLTP	OnLine Transaction Processing
SGBD	Sistema de Gerenciamento de Banco de Dados
SQL	Structured Query Language
TPC	Transaction Processing Performance Council
TPC-C	TPC Benchmak C
TPS	Transações Por Segundo
YCSB	Yahoo! Cloud Serving Benchmark
XML	eXtended Markup Language

SUMÁRIO

1 INTRODUÇÃO

Por muitos anos as bases de dados do modelo relacional serviram muito bem para praticamente qualquer aplicação que necessitava de um banco de dados. Porém com a evolução das tecnologias na nuvem notou-se que o modelo tradicional de banco de dados já não estava suprindo a necessidade do mercado.

O mercado passou a ter necessidade de acessar enormes quantidades de dados e em uma velocidade muito maior que as que os bancos relacionais podiam oferecer. Para manter a integridade e a facilidade da linguagem de consulta estruturada, do inglês *Structured Query Language (*SQL), modelos relacionais exigem grande capacidade de processamento e de espaço em disco, se tornando uma tecnologia lenta para aplicações na nuvem (PAYANDEH, 2013).

Desta necessidade surgiram os bancos de dados conhecidos hoje por NoSQL. Bancos NoSQL são bases de dados não-relacionais que abriam mão das propriedades ACID e da interface SQL, para focarem em desempenho, alta escalabilidade e replicação de dados.

Estes bancos não foram desenvolvidos para substituir os bancos relacionais e sim para suprir as áreas onde o banco de dados relacional não é eficiente. Ao invés de trabalhar com dados relacionados, que requerem mais espaço e são processados por grandes servidores, bancos NoSQL trabalham de uma maneira onde todos os dados de uma informação estão agrupados em um registro e distribuídos de forma horizontal, com outros servidores, exigindo menos capacidade de processamento, desta forma ao invés dos grandes servidores, computadores e servidores de pequeno e médio porte conseguem processar os dados de maneira muito rápida.

Porém nos últimos anos os bancos de alta escalabilidade e desempenho começaram a encontrar alguns problemas. No inicio do crescimento da computação em nuvem, as principais aplicações a necessitarem de bancos neste modelo eram os sistemas de buscas, redes sociais, blogs, entre outros. Sistemas onde o acesso rápido é considerado muito mais importante que a total segurança dos dados. Mas nos tempos atuais o mercado ao mesmo em tempo que migra suas tecnologias para a nuvem, buscando velocidade e alta disponibilidade da informação, requer também segurança e facilidade de acesso a esta informação. Surgindo assim o conceito de

banco de dados NewSQL, um modelo de banco de dados que busca utilizar o melhor do modelo relacional tradicional e do NoSQL (STONEBRAKER, 2011).

Desta forma este estudo visa realizar uma analise comparativa entre os modelos através de uma aplicação web, levantando pontos fortes e fracos dos modelos comparados, principalmente em relação do modelo NewSQL.

1.1 OBJETIVOS

O objetivo deste projeto é analisar na teoria e na pratica os modelos de bancos de dados relacionais, NoSQL e NewSQL, aferindo as características, funcionamento e propósitos de cada modelo.

Os objetivos específicos deste estudo consistem em:

a) estudar sobre conceitos, definições e estrutura geral dos bancos de dados;

b) analisar a estrutura dos bancos dados relacionais e suas origens;

c) conceituar bancos de dados NoSQL e os motivos de seu surgimento;

d) conhecer os conceitos por traz dos bancos de dados NewSQL, e qual seus objetivos.

1.2 JUSTIFICATIVA

Desde o surgimento dos primeiros Sistemas Gerenciadores de Banco de Dados (SGBD), por volta dos anos 60 e 70, os modelos de armazenamento de dados se adaptaram para atender a demanda de mercado. Os modelos com maior destaque atualmente são, modelo relacional, modelo concebido em 1970 por Edgar Todd e aplicado até hoje em bancos como Oracle, Postgresql, DB2, etc. conhecido pela interface SQL e garantias de integridade de dados através das propriedades ACID, acrônimo para Atomicidade, Consistência, Isolamento e Durabilidade, o outro é o modelo NoSQL, modelo que abriu mão das principais características do modelo relacional em troca de alta escalabilidade e desempenho.

O modelo relacional surgiu para suprir a dificuldade das bases de dados baseadas nos modelos de rede e hierárquico de navegar pelos registros e relacionar dados. Este modelo dominou o mercado por muitos anos, até o crescimento da computação na nuvem.

Por volta dos anos 2000 a computação na nuvem começa a ganhar destaque através de redes sociais e sites de buscas. Estes sistemas requeriam acesso a grandes quantidades de dados em tempo real. Para bancos de dados relacionais trabalharem com aplicações em tempo real o custo dos servidores e das estruturas de redes são extremamente alto. Nisto surgiu o modelo NoSQL, um modelo de bases de dados, que abria mão de SQL e propriedades ACID, voltados para aplicações que exigem alto desempenho e escalabilidade. Estes bancos possuem facilidade de replicação de dados, desta forma podem ser processados por computadores e servidores de menor porte que os servidores de alto desempenho utilizados para bancos relacionais. Vale ressaltar que o NoSQL, não pretendeu substituir o modelo relacional e sim suprir uma área onde os bancos relacionais eram ineficazes.

Há alguns anos, Michael Stonebraker, publicou um artigo propondo um novo modelo, batizado de NewSQL pelo analista sênior do 451 Group Matt Aslett. Stonebraker propôs um modelo de banco de dados pudesse ser altamente escalável e de alto desempenho sem abrir mão das propriedades ACID e da interface SQL.

Este modelo busca atender áreas onde nem o modelo relacional, nem o NoSQL atendem de maneiras eficazes, como Análises de inteligência de negócios, do inglês Bussiness Intelligence (BI).

Analisar a estrutura, desempenho e os conceitos deste novo modelo, ajudará na escolha do modelo ideal de gerenciamento de banco de dados de acordo com a aplicação, trazendo uma análise mais profunda deste conceito.

2 BANCOS DE DADOS

Um banco de dados é um sistema para armazenamento de dados eletrônicos, que permite interação com usuários que podem executar diversas ações nele, comumente chamadas de operações, como consultas, inserção de registros, edição, exclusão, entre outros. Ou seja, um banco de dados é um sistema que visa armazenar quaisquer tipos de dados eletrônicos que tenham relevância dentro de um contexto, para ser utilizado por um sistema de aplicação. Desta forma um banco de dados pode ser utilizado tanto para um sistema pessoal, quanto para organizações e empresas de todos os portes e de todos os setores, variando de hospitais, onde os dados armazenados são de pacientes e doenças, por exemplo, a fábricas, que armazenas dados de produção, produtos acabados, etc., a outras áreas de atuação (DATE, 2004).

Banco de dados é um conjunto de dados, dispostos em uma determinada ordem, que relacionados possuem significado, atendendo requisitos que visam atender a necessidade de um usuário (OLIVEIRA, 2002).

2.1 SISTEMAS DE GERENCIAMENTO DE BANCO DE DADOS

Para realizar operações em um banco de dados é necessário utilizar um sistema próprio para a manipulação e gerenciamento do próprio banco conhecido como Sistema de Gerenciamento de Banco de Dados, ou SGBD. Alguns bancos de dados possuem a alternativa de armazená-los em arquivos e que seja escrito um programa para sua manipulação, porém os SGBD’s oferecem uma gama de vantagens quando comparado a coleções de arquivos:

a) independência de dados: em uma aplicação os dados são apresentados de forma tratada, ocultando detalhes que só são visíveis através do SGBD;
b) acesso eficiente aos dados: um SGBD é desenvolvido especificamente para acessar e manipular o banco de dados, desta forma ele possui técnicas especificas e eficazes para visualização e manipulação de dados;
c) integridade e segurança dos dados: através do SGBD é possível configurar níveis de acesso, determinando quais dados são visíveis e

quais podem ser manipulados para cada nível de usuário, além de respeitar as restrições do banco ao executar uma operação;

d) administração de dados: o SGBD oferece uma administração centralizada dos dados, desta forma os administradores, que conhecem a estrutura do banco e os dados armazenados de forma geral, diferente de usuários do sistema de aplicação, que tem geralmente conhecimento apenas dos dados que está habituado a utilizar, possuem mais facilidades para organizar a representação dos dados, realizar melhorias na estrutura, garantindo a eficiência do banco;
e) acesso concorrente e recuperação de falhas: o SGBD garantirá que as transações executadas por um usuário não afete as transações de outro usuário que esteja acessando o banco ao mesmo tempo e garante a recuperação dos dados em caso de falha;
f) tempo reduzido de desenvolvimento no aplicativo: o SGBD, por ser um sistema específico para a manipulação do banco de dados, possui uma serie de rotinas e funções para a manipulação de dados que são comuns a diversos aplicativos que acessam o banco, diminuindo o tempo de desenvolvimento de um aplicativo, pois evita que rotinas já implementadas no SGBD tenham que ser desenvolvidas novamente e depuradas (RAMAKRISHNAN; GEHRKE, 2008).

2.2 MODELOS DE DADOS

Um modelo de dados é uma coleção de conceitos que descreve os dados, suas relações e suas restrições de consistência. Esta coleção serve para apoiar a estrutura de um banco de dados oferecendo uma maneira de desenvolver o projeto físico e lógico do banco (SILBERSCHATZ, 2006).

O modelo de dados também pode ser descrito como uma coleção de construtores de alto nível para descrição de dados, onde este modelo oculta diversos detalhes de baixo nível de armazenamento e através do SGBD o usuário pode definir os dados a serem armazenados seguindo os termos do modelo escolhido. Atualmente as maiorias dos SGBDs seguem o modelo relacional (RAMAKRISHNAN; GEHRKE, 2008).

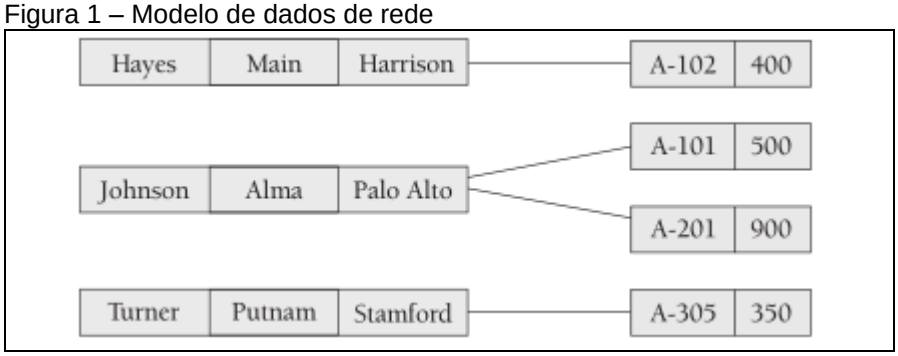
2.2.1 Modelo de rede e modelo hierárquico

Segundo Silberschatz (2006), os modelos de dados de rede o modelo de dados hierárquico eram os modelos que existiam antes do relacional e eram complicados de trabalhar.

2.2.1.1 Modelo de dados de redes

O modelo de dados de redes é uma coleção de registros que possuem vínculos entre si, onde um registro deste modelo consiste em uma coleção de atributos contendo apenas o valor dos dados e o vinculo representa a associação entre dois registros (SILBERSCHATZ, 2006).

Figura 1 – Modelo de dados de rede



Fonte: Silberchatz (2006).

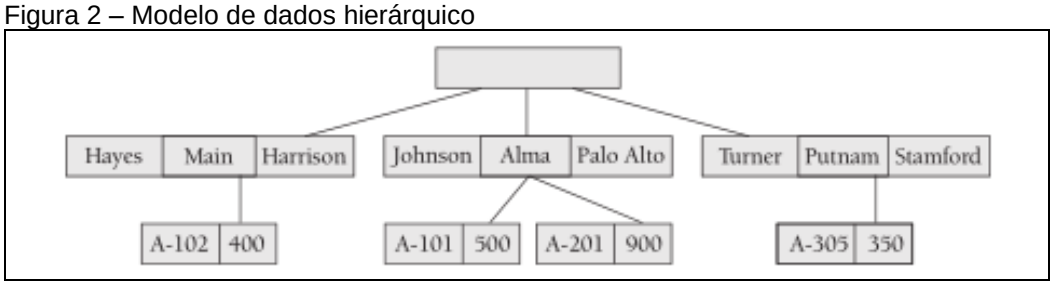
Tomando por exemplo a figura 1, que demonstra o relacionamento entre os registros de cliente e conta, onde cliente é composto por nome, rua e cidade e a conta por número da conta e saldo. A figura demonstra que o cliente Hayes, possui a conta A-102, Johnson possui as contas A-101 e A-201 enquanto Turner possui a conta A-305 (SILBERSCHATZ, 2006).

2.2.1.2 Modelo de dados hierárquico

O modelo de dados hierárquico é uma coleção de registros que possuem vínculos entre si, onde os registros e vínculos são semelhantes ao do modelo em rede, o registro é uma coleção de atributos que possuem apenas o valor dos dados e os vínculos representam a ligação entre dois registros. Porém a maneira em que os dados são representados é diferente do modelo de redes. Tomamos por exemplo o mesmo relacionamento citado no modelo de redes, um relacionamento entre

cliente e *conta*, com os mesmos atributos para cada tipo de registro, *cliente* consiste em nome, rua e cidade e a *conta* consiste em número da conta e saldo.

Figura 2 – Modelo de dados hierárquico



Fonte: Silberchatz (2006).

A figura 2 demonstra os mesmos relacionamentos da figura 1, porém desta vez em forma de árvore, onde o cliente Hayes, possui a conta A-102, Johnson possui as contas A-101 e A-201 enquanto Turner possui a conta A-305 e o primeiro nó da arvore, ou raiz da arvore, é um nó fictício. Um banco de dados hierárquico é uma coleção dessas arvores (SILBERSCHATZ, 2006).

2.2.2 Modelo relacional

O principal modelo de dados atual, o modelo relacional, utilizado em aplicações comerciais, por ser simples de implementar, diferente dos dois modelos anteriores, obteve destaque no mercado. Consiste basicamente na relação de linhas e colunas, de uma tabela. As colunas são conhecidas como ‘atributos’ e cada linha é uma ‘entidade’, uma relação de um conjunto de valores, A tabela é a coleção desses conjunto de entidades e atributos (SILBERSCHATZ, 2006).

Tabela 1 – A relação conta

número_conta	nome_agência	saldo
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

Fonte: Silberchatz (2006).

Considere a tabela da tabela 1. Nela estão descritas 3 colunas, ou 3 atributos: *número_conta, nome_agência e saldo*. Para cada um destes atributos existe um conjunto de valores, que é denominado domínio, por exemplo, todos os

nomes de agência compõem o domínio do atributo nome_agência. Cada linha da tabela é considerada uma relação e tabela em si é considerada uma coleção de relações (SILBERSCHATZ, 2006).

No modelo relacional, um conjunto de registros pode ser considerado uma relação. A descrição dos dados em nível de modelo e chamada de esquema ou schema, no esquema estão descritos os nomes de cada atributo, o tipo de cada atributo e seu próprio nome (RAMAKRISHNAN; GEHRKE, 2008).

O modelo relacional de dados de maneira informal descreve o banco através de tabelas, que respeitam restrições de integridade e possuem recursos para a manipulação das tabelas (DATE, 2004).

2.2.3 Modelo entidade/relacionamento

Uma entidade neste modelo pode ser um ‘objeto’ ou alguma ‘coisa’ que tenha relação com o mundo real e que seja distinguível. O modelo entidade/relacionamento ou modelo E-R, se baseia em uma coleção de entidades e suas relações, trabalhando em cima de uma percepção de mundo real (SILBERSCHATZ, 2006).

O modelo entidade relacionamento busca adotar uma maneira mais natural, mais próxima ao mundo real, para visualizar entidades e relacionamentos, através de técnicas de semânticas e diagramas (CHEN, 1976).

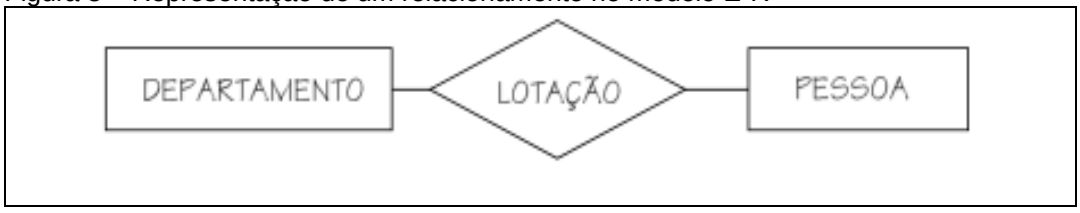
O modelo ER é descrito por meio de diagramas, na sua forma mais básica é composta por:

a) entidades, que são representadas por retângulos;

b) relacionamentos, quesão representados por losangos.

Conforme a figura 3, onde existem as entidades DEPARTAMENTO e PESSOA e relação LOTAÇÃO que faz uma associação entre os dois objetos (HEUSER, 1998).

Figura 3 – Representação de um relacionamento no modelo E-R



Fonte: Silberchatz (2006).

2.2.4 Modelo de dados orientado a objetos

O modelo de dados orientado a objetos é visto como um modelo que estende as funcionalidades de um banco no modelo E-R, adicionando encapsulamento, métodos e identidades de objeto. São sistemas de bancos de dados que podem trabalhar de forma direta com linguagens orientadas a objetos usando o sistema nativo da linguagem (SILBERSCHATZ, 2006).

Desenvolver um banco de dados orientado a objetos representa uma forma de integrar diretamente linguagens de programação complexas com persistência, coordenação e características de dados (HOROWITZ, 1991).

3 BANCOS DE DADOS RELACIONAIS

Bancos de dados relacionais são bancos de dados desenvolvidos com base no modelo relacional onde utilizam uma coleção de tabelas, que são compostas por atributos e entidades, onde atributos são as colunas e as entidades as linhas da tabela, relacionadas entre si e geralmente utilizam linguagem SQL (SILBERSCHATZ, 2006).

3.1 ESTRUTURA BÁSICA

Cada tabela de um banco relacional deve possuir um nome único, diferente de todas as outras tabelas do mesmo banco, assim como os atributos de cada tabela. Tomando por exemplo a tabela 2, que representa uma tabela de um banco relacional composta por quatro atributos: *id_cliente, nome_cliente, rua_cliente* e *cidade_cliente,* e para cada um destes atributos existe um domínio, que consiste no seu conjunto de valores, por exemplo, para o atributo *nome_cliente* o seu domínio ou conjunto de valores são todos os nomes de clientes. Desta forma um registro ou tupla que é representado pela linha da tabela consiste em um conjunto de quatro elementos, sendo cada elemento pertencente a um domínio.

Tabela 2 – Tabela de clientes

id_cliente	nome_cliente	rua_cliente	cidade_cliente
192-83-745	Johnson	12 Alma St.	Palo Alto
677-89-9011	Hayes	3 Main St.	Harrison
182-73-6091	Turner	123 Putnam Ave.	Stamford
321-12-3123	Jones	100 Main St.	Harrison
336-66-9999	Lindsey	175 Park Ave.	Pittsfield
019-28-3746	Smith	72 North St.	Rye

Fonte: Silberchatz (2006).

Analisando a figura 4, que demonstra um exemplo de banco de dados relacional, nela está descrita três tabelas que demonstram a relação cliente-conta. Na primeira tabela, Tabela de clientes, os atributos são: *id_cliente, nome_cliente, rua_cliente* e *cidade_cliente.* Na segunda tabela, Tabela de contas são: *número_conta* e *saldo.* E na ultima tabela, Tabela de depositante, que é onde está feita a relação entre cliente e conta os atributos são: *id_cliente* e *número_conta.* Através da tabela de depositantes é que está determinado de qual cliente é cada

conta, por exemplo, as contas A-101 e A-201 pertencem ao cliente Johnson que possui o id_cliente 192-83-745 e a conta A-102 pertence ao cliente Hayes, id_cliente 019-28-3746.

Figura 4 – Exemplo de um banco relacional

id_cliente	nome_cliente	rua_cliente	cidade_cliente
192-83-745	Johnson	12 Alma St.	Palo Alto
677-89-9011	Hayes	3 Main St.	Harrison
182-73-6091	Turner	123 Putnam Ave.	Stamford
321-12-3123	Jones	100 Main St.	Harrison
336-66-9999	Lindsey	175 Park Ave.	Pittsfield
019-28-3746	Smith	72 North St.	Rye

(a) Tabela *cliente*

número_conta	saldo
A-101	500
A-215	700
A-102	400
A-305	350
A-201	900
A-217	750
A-222	700

(b) Tabela *contas*

id_cliente	número_conta
192-83-745	A-101
192-83-745	A-201
019-28-3746	A-215
677-89-9011	A-102
182-73-6091	A-305
321-66-9999	A-217
019-28-3746	A-222

(c) Tabela *depositante*

Fonte: Silberchatz (2006).

Para poder distinguir um registro de outro, é necessário de que um ou mais valores de um registro, que são chamados de *chave* possam identificar todo o registro, ou seja, em uma relação não pode existir registros iguais. Por exemplo o atributo *id_cliente* da tabela cliente da figura 4 pode diferenciar todo o registro de um cliente dos demais clientes, neste caso *id_cliente* pode ser considerado uma chave, já o atributo *nome_cliente* não pode ser utilizado como chave, pois pode existir diversos cliente com o mesmo nome (SILBERSCHATZ, 2006).

3.2 ACID

Um dos principais fatores de um sistema de banco de dados relacional são as propriedades ACID que estão descritas no centro do banco de dados. ACID é um acrónimo para Atomicidade, Consistência, Isolamento e Durabilidade (SILBERSCHATZ, 2006).

Atomicidade é a propriedade que deve garantir que uma operação só será aplicada em definitivo no banco se todas as transações dessa operação obtiverem sucesso (HAEDER; REUTER, 1983).

Consistência é a propriedade que deve garantir que cada transação aplicada no banco de dados irá preservar a estrutura do banco de dados. Esta condição é essencial para o funcionamento da quarta propriedade, Durabilidade (HAEDER; REUTER, 1983).

Isolamento é a propriedade que garante que quando há mais de uma transação sendo executadas ao mesmo tempo, as transações são invisíveis umas às outras, uma transação não pode interferir em outra (HAEDER, REUTER, 1983).

Durabilidade, depois que uma transação executada no banco de dados é aplicada com sucesso, esta propriedade garante que os dados do banco estarão armazenados no banco independente de mau funcionamento do banco, falhas de outras transações, entre outros (HAEDER; REUTER, 1983).

3.4 STRUCTURED QUERY LANGUAGE

Structured Query Language ou apenas SQL, é uma coleção de comandos que servem para manipular um sistema de banco de dados. Os comandos servem tanto para manipular a estrutura do banco, alterando, criando e removendo tabelas e regras de relacionamento ou para manipulação de dados, onde é possível inserir atualizar e remover, além de consultar (OLIVEIRA, 2002).

SQL pode ser considerado uma “sublinguagem” de programação já que não é uma linguagem autônoma, isto é, não é possível escrever aplicações em SQL. Devem-se escrever aplicações utilizando outras linguagens de programação, como C++, Java, Pascal, entre outras e inserir dentro das aplicações escritas comandos de SQL para manipular os dados do banco de dados (OLIVEIRA, 2002).

A linguagem SQL foi desenvolvida na década de 70 pela IBM. Originalmente era chamada de Sequel e fazia parte do projeto R, projeto que visava desenvolver um banco de dados relacional, e posteriormente a nomenclatura passou a ser SQL. Atualmente está é a linguagem padrão para bancos de dados relacionais (SILBERSCHATZ, 2006).

4 BANCOS DE DADOS NOSQL

O termo “NoSQL” surgiu através de Carlos Strozzi, que utilizou o termo para nomear um SGBD, que ele havia desenvolvido, que evitava o uso de SQL como linguagem mesmo sendo um banco relacional. Só recentemente o termo passou a ser interpretado como “Not Only SQL”, porém se relaciona a sistemas de bancos de dados não relacionais (ZOLLMANN, 2012).

A maioria dos provedores de bancos de dados NoSQL concordam que o termo ‘NoSQL’ não é claro, porém chama atenção. A maioria concorda que ‘Not Only’ não significa negar SQL, mas sim compensar as limitações técnicas da maioria dos bancos de dados relacionais (BURD, 2011).

Os bancos de dados relacionais surgiram e evoluíram em épocas diferentes e estavam sujeitos a limitações tecnológicas da época, e se tornaram uma ferramenta de qualidade para as aplicações típicas daquela época, porém atualmente o modelo de banco relacional apresenta limitações. Desta forma o NoSQL que surgiu em outro ambiente, operando de maneira diferente do modelo relacional, proporciona soluções mais adequadas para problemas de armazenamento atuais (BURD, 2011).

Nos últimos anos aplicações distribuídas voltadas para a web são cada vez mais populares. Especialmente serviços utilizados mundialmente como Facebook, Google ou Amazon necessitam armazenar e processar grandes quantidades de dados. Estes dados não tem como serem manuseados por um sistema de apenas um nó, por isso soluções distribuídas de armazenamento se tornaram necessárias. Existem limitações para escalonar um SGBD relacional comum, por isso diversos bancos não relacionais estão evoluindo para superar esta necessidade, bancos como Dynamo, desenvolvido pela Amazon ou Google’s BigTable são exemplos de bancos NoSQL que buscam atender esta demanda (ZOLLMANN, 2012).

Uma das vantagens dos bancos NoSQL é, diferente de bancos relacionais, a capacidade de manusear dados não estruturados como arquivos de texto, e-mails, dados multimídias como áudios e vídeos e dados de redes sociais de forma eficiente (LEAVITT, 2010).

Bancos de dados NoSQL também podem ser considerados simples de trabalhar para programadores que não familiarizados com SQL. Além de que alguns

dos bancos NoSQL foram desenvolvidos para trabalhar em ambiente distribuído, isto é, um único banco pode ser executado em varias máquinas de menor potencia e menor custo ao invés de executa-lo em um único servidor de alto custo e alto desempenho. Além de que esta distribuição permite um melhor desempenho, o que é essencial para aplicações com grande volume de dados (LEAVITT, 2010).

4.1 TIPOS DE BANCOS DE DADOS NOSQL

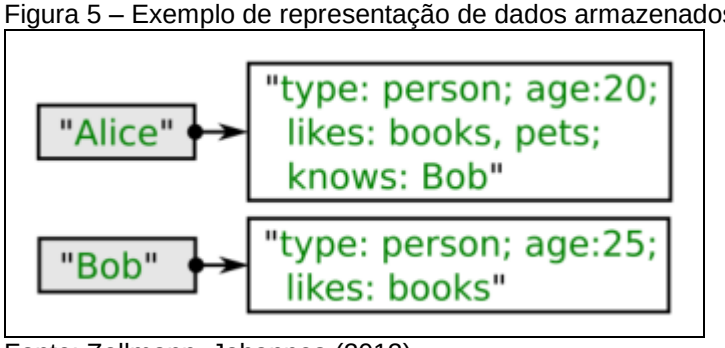
Segundo Leavitt (2010) existem três tipos populares de bancos de dados NoSQL: Bancos de dados chave-valor, Bancos de dados orientados a colunas e Bancos de dados orientados a documentos.

4.1.1 Banco de dados chave-valor

Sistemas de armazenamento baseados em chave-valor, de maneira básica, são conjuntos de vetores associados, que são formados por chaves e valores, onde cada chave deve ser única fornecendo uma informação não ambígua de valores (ZOLLMANN, 2012).

Sendo assim, o armazenamento por chave-valor é um sistema que armazena valores indexados por chaves que podem ser recuperadas, podendo conter dados estruturados ou não (LEAVITT, 2010).

Figura 5 – Exemplo de representação de dados armazenados em um banco chave-valor.



Fonte: Zollmann, Johannes (2012).

A figura 5 mostra um exemplo de dados armazenados em um banco chave-valor. A chave, neste caso “Alice” e “Bob”, são os objetos mais simples, enquanto os valores podem possuir dados mais complexos (ZOLLMANN, 2012).

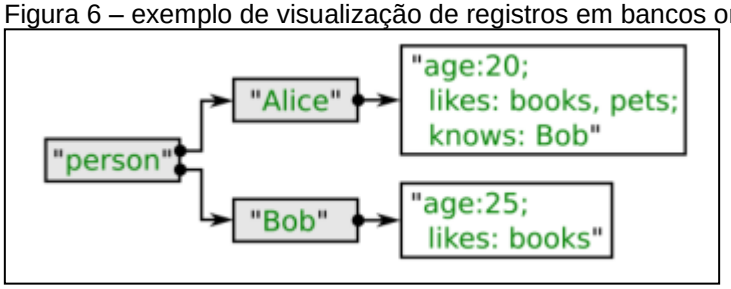
Este modelo de armazenamento permite que o banco de dados trabalhe de maneira eficiente e muito rápida, porém possui a possibilidade de montar consultar complexas limitadas (ZOLLMAN, 2012).

4.1.2 Bancos de dados orientados a colunas

Ao contrario dos bancos relacionais onde as tabelas são totalmente estruturadas em linhas e colunas com campos dimensionados de maneira uniforme para cada registro, bancos de dados orientados a colunas possuem uma coluna expansível de dados relacionados (LEAVITT, 2010).

Este modelo de banco de dados também pode ser visto como um banco chave-valor, com chaves de duas dimensões. O armazenamento é feito por uma chave de colunas e chaves de linhas e dependendo da implementação pode haver mais dimensões. A figura 6 é um exemplo de visualização de registros em bancos orientados a colunas (ZOLLMANN, 2012).

Figura 6 – exemplo de visualização de registros em bancos orientados a colunas.



Fonte: Zollmann, Johannes (2012).

4.1.3 Bancos de dados orientados a documentos

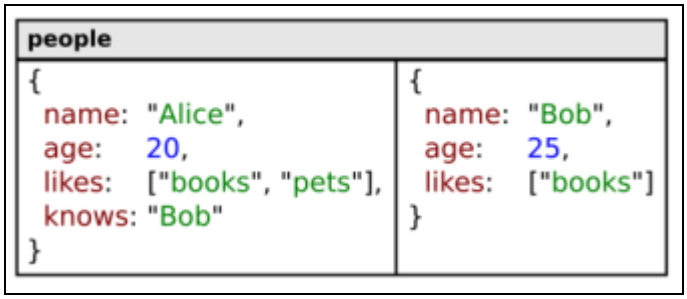
O termo documento neste tipo de banco refere-se a algum tipo de documento de dados estruturados como JSON, BSON ou XML. Enquanto o formato de armazenamento é fixo pelo tipo de documento, a estrutura é flexível, sendo possível armazenar diversos dados diferentes em um mesmo documento, contanto que eles sejam do mesmo tipo (ZOLLMANN, 2012).

Este tipo de banco armazena e organiza os seus dados em coleções de documentos ao invés de tabelas estruturadas com tamanhos iguais para cada registro. Nestes bancos um usuário pode inserir quantos campos quiser e com qualquer tamanho no documento. A figura 7 mostra a representação de dados

armazenados em um banco orientados a documentos no formato JSON (LEAVITT, 2010).

Este tipo de estrutura de dados pode ser interpretado pelo SGBD, permitindo consultas mais refinadas e operações sobre as propriedades dos documentos, ao contrario dos bancos chave-valor que geralmente não são interpretados por seus SGBDs (ZOLLMANN, 2012).

Figura 7 – Exemplo de representação de dados armazenados em um banco orientados a documentos no formato JSON.



Fonte: Zollmann, Johannes (2012).

4.2 CONSISTÊNCIA EVENTUAL

Uma das principais diferenças entre o modelo relacional e o NoSQL é a maneira como consistência é tratada. Sistemas NoSQL precisam abrir mão das propriedades ACID para permitir escalabilidade horizontal. A necessidade de abrir mão das propriedades ACID para permitir a escalabilidade, ocorre por conta do Teorema CAP, que foi proposto por Eric Brewer no ano 2000, que se baseia em três propriedades para sistemas distribuídos (ZOLLMANN, 2012).

a) consistência (*Consistency*): para ter consistência em sistemas distribuídos necessita-se que todas as solicitações e resultados de operações sejam vistos por todos os nós;
b) disponibilidade (*Availability*): implica de que cada nó que receber uma solicitação de uma aplicação cliente deve responder a esta solicitação;
c) tolerância à partição (*Partition Tolerance*): está propriedade descreve a necessidade de o sistema ser tolerante a falhas, o sistema tem que se manter funcionando mesmo quando a falha de comunicação entre os seus nós (GILBERT; LYNCH, 2002).

Segundo Brewer (2000) um sistema distribuído pode ter apenas duas destas três propriedades, sendo obrigado a abrir mão de uma delas.

5 BANCOS DE DADOS NEWSQL

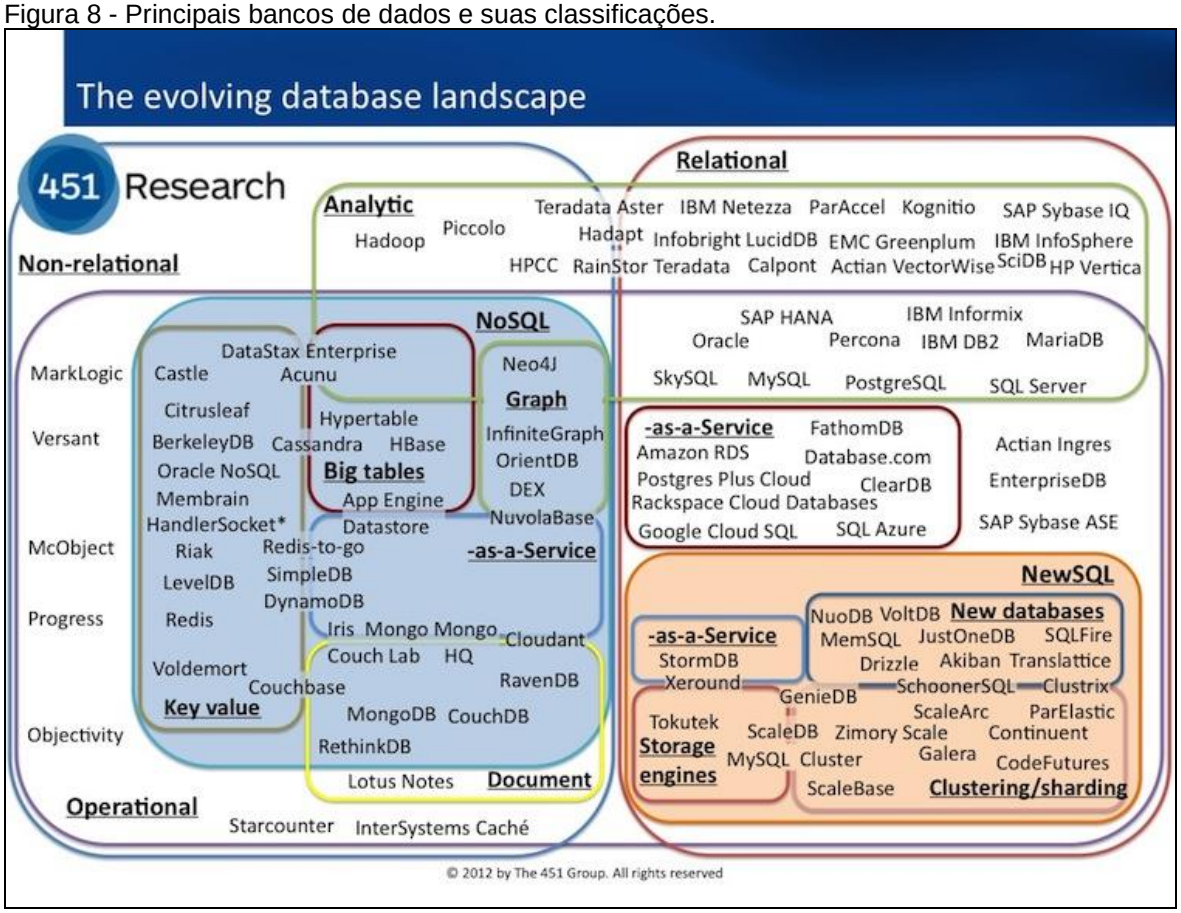
Atualmente estão surgindo sistemas de gerenciamento de banco de dados que tendem manter a interface SQL e oferecer desempenho e escalabilidade, preservando também noção tradicional de ACID para transações. Para distinguir estas soluções das soluções já conhecidas está classe de bancos de dados está sendo chamada de NewSQL. Estes sistemas devem ser capazes de prover alto rendimento, igual a bancos NoSQL, porém sem a necessidade garantir a consistência através de programação em nível de aplicativo, ou seja o SGBD deve garantir a consistência dos dados, além de oferecer suporte a SQL (STONEBRAKER).

Em 2011 a empresa de analise de mercado 451 Group publicou um relatório onde o analista Matt Aslett mencionou pela primeira vez o termo ‘NewSQL’, que segundo Aslett (2011) são bancos de dados que prometem garantir escalabilidade e flexibilidade assim como os bancos NoSQL ao mesmo tempo em que dão suporte a consultas SQL e/ou garantia à ACID.

Assim como o termo NoSQL, o termo NewSQL também não deve ser levado ao pé da letra, o termo refere-se ao modelo de banco não ao SQL especificamente (ASLETT, 2011).

Para obter melhor compreensão sobre sistemas de bancos de dados NewSQL, será descrito um pouco da estrutura de alguns SGBDs já existente no mercado. O NuoDB, desenvolvido pela NuoDB, Inc. e o VoltDB desenvolvido pela empresa homônima, fundada por Micheal Stonebraker e o MySQL Cluster, versão NewSQL de um dos bancos mais conhecidos do mundo. Estes e outros bancos NewSQL podem ser vistos na figura 8 mostra um retrato da classificação dos principais bancos de dados do mercado atualmente segundo Aslett.

Figura 8 - Principais bancos de dados e suas classificações.



Fonte: The 451 Group.

5.1 NUODB

O NuoDB é um banco de dados distribuído desenvolvido para implementação de aplicações globais com suporte a transações ACID, linguagem SQL e lógica relacional (NUODB, 2013)

Algumas características do NuoDB além de suporte a linguagem SQL:

a) transações ACID: na arquitetura do NuoDB a durabilidade fica separada dos processos de transações, as propriedades de atomicidade, consistência e isolação dos dados são gerenciadas em memoria pelas transações em instancias independentes, enquanto uma arquitetura de armazenamento garante a durabilidade dos dados após a execução das transações;

b) *scale-out* performance: uma tecnologia onde a cada novo nó adicionado, o desempenho geral do banco é melhorado. Esta tecnologia também garante a elasticidade e escalabilidade do banco de

dados;

c) geo-distribuição: um único banco em locais fisicamente diferentes.

O NuoDB roda em cima de uma arquitetura de três camadas: administrativa, transacional e armazenamento. Está divisão é responsável por permitir o escalonamento horizontal do banco de dados e ao mesmo tempo garantir as propriedades ACID do SGBD. Tradicionalmente um banco relacional possui grande dependência do disco rígido (*shared-disk)* e esta estrutura torna os bancos relacionais comuns difíceis de serem escalonados horizontalmente, com as camadas separadas isto é possível. Enquanto a atomicidade, consistência e isolamento são tratados na camada transacional, a durabilidade é tratada na camada de armazenamento, desta forma como cada camada é independente, falhas de outras camadas não afetam os processos de outras. Cada instancia destas camadas são considerados ‘pessoas’ ou *peers* e se comunicam entre si para garantir a segurança e a replicação dos dados automaticamente, os *peers* da camada de transação são chamados de *Transactions Engines (TE)* e os da camada de armazenamento *Storage Manager (SM)* e não existe um controlador comum, cada *peers* é independente no banco. Como a camada de transação é totalmente em memoria, o disco rígido do servidor só será requisitado quando for consultada uma informação que não está em um dos nós de transação ou em operações de escrita. No NuoDB não existe particionamento de dados, cada SM possui uma ‘copia’ de todo o banco de dados e a cada alteração o SGBD automaticamente sincroniza a alteração nos outros SM.

5.2 VOLTDB

O VoltDB é um SGBD de memória principal de alta disponibilidade e alto desempenho com suporte a transações tradicionais, isto é, transações ACID e suporte a SQL (STONEBRAKER, 2013).

O VoltDB também possui acesso a SQL e transações garantidas por ACID, abaixo, algumas características:

a) alta disponibilidade: existem diversas partições replicadas em nós diferentes para caso em que um nó pare de funcionar, o sistema continue funcionando normalmente sem perda de dados;

b) operações em memoria principal: todos os dados são gravados em

formato de memoria principal, desta forma não há necessidade de buffers de dados e códigos;

c) particionamento automático: o Voltdb utiliza uma arquitetura shared-nothing, isto é, cada nó é considerado independente, para garantir o paralelismo do banco de dados, distribuindo automaticamente os as atividades entre os nós disponíveis.

O VoltDB, diferente do NuoDB não utiliza particionamento de dados, roda sob uma arquitetura que fragmenta de forma horizontal seus dados entre todos os computadores do conglomerado, mesmo com dados particionados entre diversos servidores o desempenho é garantido devido ao fato de que o VoltDB armazena todo o banco de dados em memoria principal, só utilizando o disco para garantir a durabilidade dos dados. Além de que os dados são gerenciados por um controlador global, que determina a ordem das execuções das transações.

5.3 MYSQL CLUSTER

O MySQL Cluster é considerado a versão de alto desempenho e disponibilidade do MySQL, que é talvez o banco de dados *open source* mais popular do mundo. Ao contrario do MySQL tradicional que utiliza os motores InnoDB e MyISAM, o MySQL Cluster utiliza o motor NDBCluster que é um acrônimo de *Network DataBase Cluster.* Esta *engine* foi desenvolvida para ter suporte a ACID, interfaces SQL e NoSQL escalabilidade elástica e distribuição geográfica. Alguns dos pontos altos do MySQL são:

a) baixa latência e resposta em tempo real;
b) particionamento horizontal automático entre os nós do cluster;
c) suporte a interfaces NoSQL além de SQL;
d) arquitetura *shared—nothing* com múltiplos mestres;
e) *cross-shard JOINS*, possibilidade de realizar *JOINS* entre tabelas de nós diferentes, entre outras.

O MySQL Cluster foi desenvolvido para operações com alto volume de transações em tempo real, comercio eletrônico, jogos *online* e diversas outras aplicações voltadas para computação nas nuvens e móvel, onde há a necessidade de alto desempenho facilidade de expansão do banco e baixo custo (ORACLE, 2013).

6 TRABALHOS CORRELATOS

Bancos de dados NewSQL ainda são relativamente novos se comparados aos modelos NoSQL e ao relacional, porém se mostraram uma solução bastante atraente para os olhos de diversos desenvolvedores por propor permitir atender áreas de negocio onde nem o modelo relacional, nem o NoSQL são efetivos. Existem diversas pesquisas e analises sendo realizadas a respeito deste modelo de banco de dados não somente em comparações com os modelos já estabelecidos, mas também pesquisa apenas sobre o modelo e suas capacidades. A seguir serão descritos alguns trabalhos que envolvem bancos NewSQL e outras tecnologias as quais o modelo aparece como solução.

6.1 DATA MANAGEMENT IN CLOUD ENVIRONMENTS: NOSQL AND NEWSQL DATA STORES

Este artigo foi escrito por Kataringa Grolinger, Wilson A. Higashino, Abhinav Tiwari e Mirian AM Captrez em 2013 e publicado no *Journal of Cloud Computing*. O trabalho de Grolinger et al é focado na computação na nuvem, especificamente em soluções de armazenamento de dados em ambientes na nuvem. O trabalho avalia soluções NoSQL e NewSQL com três objetivos: Prover uma perspectiva de uso das tecnologias, servir de orientação para profissionais e pesquisadores na escolha do sistema de armazenamento apropriado e identificar desafios e oportunidades das tecnologias. Para isso, compararam das principais soluções o modelo de dados, consultas, escalabilidade, e recursos de segurança, capacidade de escalonar solicitações de leituras e escritas, escalonamento de banco, replicação, consistência e controle de concorrência. Além de casos de usos e cenários que utilizam bancos NewSQL e NoSQL.

Para realizar esta pesquisa os bancos foram selecionados de acordo com seus tipos, chave-valor, orientado a documentos ou a colunas. Forma considerados os bancos mais populares de acordo com o DB-Engine Ranking. Foram levantandos também os conceitos de computação na nuvem e do teorema CAP além das características dos bancos em analise.

Os pesquisadores concluíram que com a recente emersão da computação na nuvem como um paradigma de computacional que pode ser usado para atender a crescente necessidade de armazenamento de maneira continua e atender o processamento de aplicações atuais. As comparações dos modelos de armazenamentos de dados e os casos de uso irão servir como um guia para profissionais que buscam a melhor escolha para suas necessidades. A pesquisa também identificou alguns pontos onde as tecnologias NoSQL e NewSQL necessitam de melhorias, como a dificuldades de encontrar documentação dos modelos, comparações escassas e imaturidades das soluções.

6.2 CRITICAL ANALYSIS OF DATABASE MANAGEMENT USING NEWSQL

Este artigo foi publicado na revista *International Journal of Computer Science and Mobile Computing – IJVSMC* em 2014 foi escrito por Rakesh Kumar, Neha Gupta, Hitesh Maharwal, Shilpi Charu e Kusum Yadav. O trabalho de pesquisa trata sobre a introdução dos bancos NewSQL, os conceitos por traz do modelo, Liberty DataBase Conection (LDBC) e funcionamentos dos bancos NewSQL. O objetivo da pesquisa é mostrar a importância dos sistemas de gerenciamentos de dados NewSQL e prover quais as melhores soluções para aplicações tanto comerciais quanto de código aberto.

A pesquisa focou no funcionamento do LDBC, que é um drive de conexão JDBC que prove conexão entre uma aplicação Java e diversos bancos de dados sem a necessidade de alteração de código e na estrutura básica de nove exemplos de bancos NewSQL analisando as principais características dos bancos de dados NewSQL.

Os pesquisadores concluíram que bancos NewSQL são uma nova geração de sistemas de gerenciamento de informação e que serão para negócios que pretendem migrar soluções para nuvem ou desenvolver novas aplicações de processamento de transações em tempo real (Online Transaction Processing – OLTP) sendo considerada uma alternativa aos bancos NoSQL.

6.3 NOSQL DATABASE: NEW ERA OF DATABASES FOR BIG DATA ANALYTICS - CLASSIFICATION, CHARACTERISTICS AND COMPARISON

Este trabalho científico foi escrito por A B M Moniruzzaman e Prof. Dr. Syed Akhter Hossain em 2013 para o *International Journal of Database Theory and Application.* O objetivo do artigo é prover classificação, características e avaliações de bancos NoSQL em aplicações Big Data. Oferecendo uma visão independente das vantagens e desvantagens de várias soluções NoSQL no processamento de grandes volumes de dados.

A pesquisa analisa os conceitos dos bancos NoSQL e NewSQL, tipos dos bancos e teorema CAP e realiza uma analise comparativa entre alguns bancos NoSQL através de uma tabela, cruzando modelos de banco, orientado a documento, orientado a colunas, chave-valor e grafos, com atributos de integridade, indexação, distribuição, plataforma e outras características e um levantamento da adoção dos bancos NoSQL no mercado.

Os pesquisadores concluíram que os bancos NoSQL surgiram para liderar o desenvolvimento de aplicações de *big data* e analise de negócios, aplicações que já haviam chegado ao limite dos bancos relacionais tradicionais. Concluindo que bancos NoSQL possuem uma grande área para se desenvolver e proveram uma analise comparativa entre bancos a fim de facilitar na escolha por um banco no desenvolvimento de uma aplicação que lidará com grande volume de dados.

6.4 A STUDY OF NOSQL AND NEWSQL DATABASES FOR DATA AGGREGATION ON BIG DATA

Este trabalho foi desenvolvido como projeto de mestrado de Ananda Sentraya Perumal Murugan para o Royal Institute of Technology de Estocolmo, Suiça. Este trabalho foi desenvolvido com o objetivo de comparar uma solução NoSQL com uma solução NewSQL a fim de identificar qual se encaixaria melhor em uma aplicação de analise de dados utilizada pela Scania.

O departamento de TI da Scania utiliza um banco de dados relacional para armazenar os dados de todos os veículos e prover analise em cima destes dados. Por ser utilizado um banco relacional tradicional, esta aplicação consome grande volume de memoria e tempo, por isso foi proposto uma solução distribuída para solucionar este problema. Foi realizada uma analise pratica comparando uma solução NoSQL com uma solução NewSQL, a fim de identificar qual seria mais adequada para a solução do problema. A comparação foi feita analisando o

desempenho dos dois bancos em um caso de uso comum e com uma tabela comparando características dos sistemas.

Nos testes realizados o banco NewSQL se mostrou superior ao banco NoSQL, entretanto o teste mostra apenas uma pequena parte de todo o processo não devendo ser considerado como fator decisivo. Os bancos utilizados foram MySQL Cluster como exemplo de banco NewSQL e o HBase como exemplo de banco NoSQL.

7 ANALISE COMPARATIVA ENTRE OS MODELOS RELACIONAIS, NEWSQL E NOSQL DE BANCOS DE DADOS

Neste capítulo está apresentada duas analise comparativa entre bancos de dados dos três modelos discutidos neste trabalho. A primeira análise é feita em forma de comparação direta das características de bancos de dados dos três modelos através de uma tabela. A segunda análise foi feita através de execuções de *benchmarks* voltados para bancos de dados, nesta etapa os bancos foram separados em dois grupos para as aferições, o primeiro para bancos relacionais e o segundo para bancos distribuídos, considerando que bancos NewSQL são relacionais e distribuídos este modelo participou dos dois grupos de aferições. Foram selecionados para esta análise dois *benchmarks* já conceituados no mercado, o TPC *Benchmark* C ou apenas TPC-C, desenvolvido pela *Transaction Processing Performance Council* (TPC), focado em transações OLTP, e o *Yahoo! Cloud Serving Benchmark* conhecido como YCSB, desenvolvido pela Yahoo! com o objetivo de padronizar *benchmarks* para bancos de dados baseados na nuvem.

Para o desenvolvimento das analises foram considerados dois bancos de dados de cada modelo. Todos os bancos selecionados são de código aberto ou possuem algum tipo de licença livre e foram selecionados com base no ranking do site DB-Engines. Os bancos selecionados são: MySQL e PostgreSQL, representando o RDBMS, MongoDB e Cassandra como NoSQL e para NewSQL foram utilizados o MySQL Cluster e o NuoDB.

A seção 7.1 mostra a análise feita através de uma matriz, comparando características dos três modelos de bancos de dados. A seção 7.1 descreve o conjunto de *benchmarks* que foram utilizados nas comparações. A seção

7.1 COMPARAÇÃO DIRETA ENTRE BANCOS DE DADOS DOS TRÊS MODELOS

Nesta seção é apresentada a comparação entre os bancos de dados MySQL, PostgreSQL, NuoDB, MySQL Cluster, MongoDB e Cassandra. Na tabela 3 é possível analisar as características que definem cada modelo, por exemplo, os bancos NoSQL, possuem métodos de particionamento e de replicação, necessários para a elasticidade dos bancos, enquanto os bancos relacionais não possuem estás características, porém os SGBD NoSQL não possuem transações ACID e uma

linguagem de consulta padrão como o SQL, que são necessárias para garantir segurança dos dados e agilidade de desenvolvimento de consultas complexas.

Tabela 3 – Tabela de comparação entre modelos de bancos de dados

	RDBMS		NewSQL		NoSQL	
	MySQL	PostgreSQL	NuoDB	MySQL Cluster	MongoDB	Cassandra
Conceitos de transações (ACID)	ACID	ACID	ACID	ACID	Atomicidade em operações em apenas um documento	Atomicidade e Isolamento são suportados em algumas operações
Linguagem de consulta, definição e manipulação	SQL	SQL	SQL	SQL	Própria	CQL
Método de partição	Não possui	Não possui	Dados dinamicamente alocados nos nós de leitura e escrita	Particionamento horizontal (Sharding)	Particionamento horizontal (Sharding)	Particionamento horizontal (Sharding)
Método de replicação	Mestre-escravo	Mestre-escravo	Gerenciado de forma transparente pelo SGBD	Mestre-escravo	Mestre-escravo	De acordo com a estratégia configurada e do fator de replicação
Índices	Possui	Possui	Possui	Possui	Possui	Restrito
Schema	Possui	Possui	Possui	Possui	Livre	Livre
Store Procedures/Triggers	Possui	Possui	Possui	Possui	Não possui	Possui
Chaves	Possui	Possui	Possui	Possui	Não possui	Não possui
Método de armazenamento	Relacional	Relacional	Relacional	Relacional	Documento	Orientado a colunas
Disponibilidade	Baixa	Baixa	Alta	Alta	Alta	Alta
Tolerância a Partição	Fraca	Fraca	Forte	Forte	Forte	Forte
Atomicidade	Possui	Possui	Possui	Possui	Operações em apenas um documento	Em operações singulares
Consistência	Possui	Possui	Possui	Possui	Consistência eventual ou imediata	Consistência eventual ou imediata
Isolamento	Possui	Possui	Possui	Possui	Manual	Em operações singulares
CAP	CA	CA	CAP	CAP	AP	AP
Durabilidade	Possui	Possui	Opcional	Possui	Opcional	Possui
Tipagem	Forte	Forte	Forte	Forte	Fraca	Fraca
Controle de concorrência	Possui	Possui	Possui	Possui	Possui	Possui

Fonte: Própria.

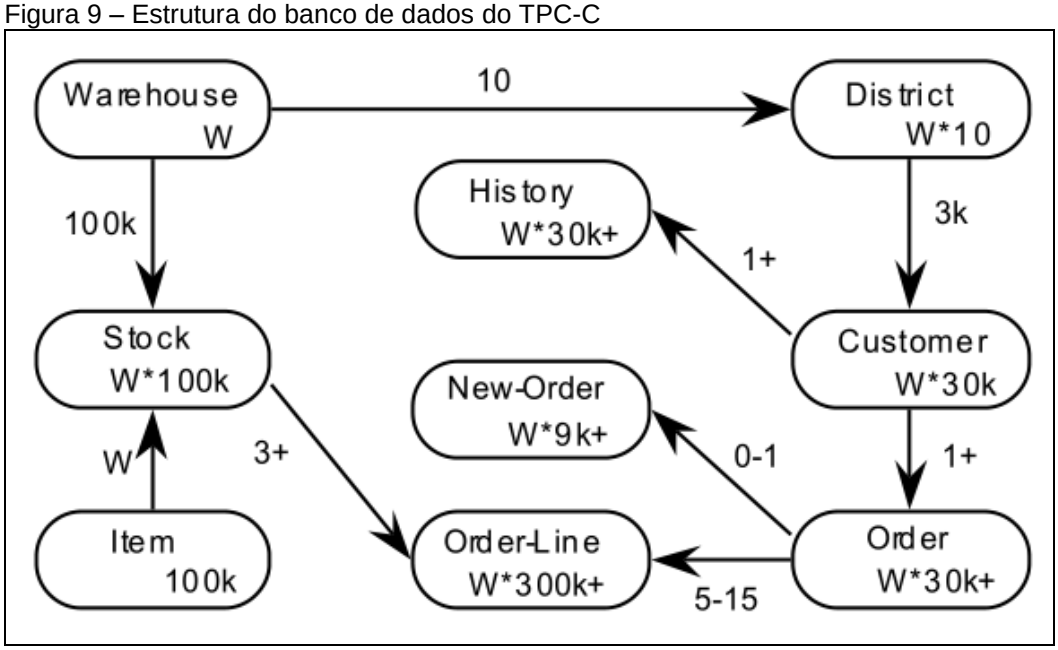
Através da tabela 3 é notável também a proposta dos bancos NewSQL, em atenderem características que os dos dois modelos tradicionais atendem. Bancos NewSQL possuem métodos de particionamento e replicação e também possuem suporte a SQL e garantias das propriedades ACID. Nas seções a seguir, estão descritos características, conceitos e resultados de uma series de testes práticos que foram feitos com estes bancos de dados. As informações foram coletadas diretamente dos desenvolvedores dos SGBD e do site DB-Engines.com.

7.2 TPC-C

O *benchmark* TPC-C foi desenvolvido pela *Transaction Processing Performance Council* (TPC), uma instituição de padronização e de desenvolvimento de *benchmarks* para aplicações OLTP (TPC, 2001). A primeira versão do TPC-C foi liberada em 1992 e hoje está na versão 5.11que foi liberada em 2010.

O TPC-C é um teste de carga OLTP que simula atividades comuns em aplicações OLTP reais através de intensas transações de consultas e atualizações. O TPC-C simula múltiplos terminais, onde cada um equivale a uma conexão ao banco de dados e executam múltiplas transações de consulta e atualizações simultaneamente no banco de dados. O banco consiste de tabelas com variedades de tamanhos, atributos e relacionamentos, a figura 9 é representação do modelo ER do banco de dados utilizado no *benchmark.*

Figura 9 – Estrutura do banco de dados do TPC-C



Fonte: TPC (2010).

O TPC-C simula um sistema de vendas com suas principais transações, como pedido, pagamento, controle de situação do pedido, entrega e controle de estoque, no *benchmark* as transações são chamadas pelas suas nomenclaturas em inglês*,* cada uma destas cinco transações possui um peso, medidos em porcentagem, que representa a frequência em que a transação deve ser executada, as transações e pesos são:

a) *newOrder,* que representa a realização de um novo pedido, geralmente configurado por padrão do *benchmark,* está operação não possui um peso mínimo atribuído para ela, nos arquivos de configuração da suíte OLTP-Bench, está operação possui peso de 45%, é considerada uma transação de escrita e leitura de peso médio e com alta frequência de execução e tempo de resposta restrito;
b) *payment,* simula uma transação de pagamento, possui peso mínimo padrão de 43%, é considerada uma transação de escrita e leitura leve, também com alta frequência de execução e tempo de resposta restrito;
c) orderStatus, é a verificação da situação do pedido realizado na transação *NewOrder*, possui peso mínimo de 4% e representa uma transação de leitura de peso médio com baixa frequência de execução.
d) *delivery,* representa a entrega de um lote de 10 pedidos, ainda não entregues, cada ordem é processada individualmente com um processo completo de escritas e leituras ao banco de dados, possui baixa frequência de execução, e o tempo de resposta não é restrito, também possui peso mínimo de 4% das transações executadas.
e) *stockLevel,* considerada uma transação pesada, somente de leitura e de baixa frequência, esta transação, com peso 4%, consiste em verificar o nível de estoque vendido recentemente.

Para determinar a desempenho do *benchmark*, o TPC-C mede a quantidade de transações do tipo *NewOrder* executadas dentro do intervalo de tempo definido antes da execução do *benchmark* (TPC, 2010)*.* Nesta analise a execução do TPC-C foi feita através da suíte de *benchmarks* OLTP-Bench, um projeto de código aberto que visa unir em um único *framework* diversos *benchmarks* padronizando os relatórios e conexões, na subseção seguinte está detalhado um pouco mais sobre está suíte e o motivo da sua escolha para este projeto.

7.2.1 OLTP-Bench

O projeto OLTP-Bench é uma suíte, de código aberto, de *benchmarking* de bancos de dados que visa simular aplicações do mundo real através de diversos testes e oferece suporte para SGBD relacionais, distribuídos e de serviços (Database-as-a-service, DBaaS) (CURINO et al, 2012). O projeto foi desenvolvido com base em diversas características que definem um teste de bancos de dados moderno, que os desenvolvedores do projeto junto com outros especialistas em bancos de dados acordaram (DIFALLAH, et al, 2013).

O *framework,* escrito em Java, é composto por quinze *benchmarks,* alguns já existentes e difundidos no mercado, outros desenvolvidos pelos criadores do *framework,* infelizmente, até a versão atual do OLTP-Bench, o *framework* suporta apenas bancos de dados relacionais com conexão JDBC, não oferecendo suporte a bancos NoSQL, porém está previsto para trabalhos futuros está compatibilidade (DIFALLAH, et al, 2013). Além do TPC-C o OLTP-Bench também já possui o YCSB e outros testes, que variam entre três classes de testes:

a) *benchmarks* transacionais: possui testes tradicionais para aplicações de processamento de transações em tempo real, do inglês Online Transaction Processing (OLTP), através de relacionamentos complexos e transações intensas de escritas, o TPC-C está incluso nesta classe;
b) *benchmarks* orientados a web: são testes baseados em aplicações voltadas para a nuvem, como redes sociais e operações que envolvem relações de muitos para muitos baseadas em grafos sem acessos uniformes;
c) *benchmarks* de funcionalidade: são testes voltados para testes de funcionalidades individuais do sistema também conhecidos como *micro benchmarks,* são mais simples e leves que os *benchmarks* das outras classes. O YCSB faz parte desta classe. (DIFALLAH, et al, 2013).

Desta forma temos uma ferramenta pra execução de diversos testes porém para a forma de executa-los está padronizada. Outra vantagem é que é necessário apenas informar em qual banco o teste deve ser feito, que a própria ferramenta se encarrega de criar as tabelas, chaves e índices necessários para a execução dos

testes. Além de em trabalhos futuros, será possível avaliar os três modelos de bancos de dados utilizando uma única plataforma neutra.

7.3 YAHOO! CLOUD SERVING BENCHMARK

O *Yahoo! Cloud Serving Benchmark,* ou YCSB, é considerado uma coletânea de *micro benchmarks* que representam aplicações de manipulação e gerenciamento de dados considerados simples, mas que requerem alta escalabilidade, como aplicações desenvolvidas para Web. O *benchmark* é uma representação de uma aplicação do tipo chave-valor simples (DIFALLAH, et al, 2013).

O objetivo do YCSB é implementar um *benchmark* padrão para os diferentes tipos de bancos de dados para web e NoSQL existentes, utilizando testes e cargas de trabalhos comum em sistemas diferentes, focados em performance e escalabilidade e no futuro, disponibilidade e replicação. A ferramenta atualmente se apoia em duas camadas de *benchmark*:

a) camada 1 – performance: esta camada foca no desempenho do banco de dados, medindo a curva de latência e de rendimento do sistema. O YCSB mede a latência do banco de dados enquanto aumenta o rendimento até o banco de dados estar saturado e não conseguir mais aumentar o rendimento, mantendo o mesmo hardware. Para isso é utilizado um gerador de carga de trabalho, que é responsável por definir e carregar os dados no banco de dados e depois executar as operações no banco, enquanto mede o desempenho do mesmo;
b) camada 2 – escalabilidade: considerado um fator chave para bancos voltados para a nuvem, a capacidade de escalabilidade horizontal elástica, permite que o SGBD manuseie mais carga e cresça com mais facilidade. Esta camada analisa o impacto no desempenho do banco de dados a cada novo nó adicionado ao conglomerado através de duas métricas, a primeira delas, *scaleup,* mede o desempenho do sistema ao adicionar um novo computador ao conglomerado, adicionando o computador com os testes parados, iniciando os testes sempre do zero a cada novo nó adicionado. E a segunda métrica, *elastic speedup,*

> mede o impacto no desempenho ao adicionar um novo nó, com os testes em execução. (COOPER, 2010).

O YCSB nativamente já apresenta suporte a diversos bancos de dados NoSQL como o Apache Cassandra, MongoDB, DynamoDB, HBase, Hypertable entre outros, além de permitir outros bancos NoSQL o *framework* também possui suporte a conexão JDBC, desta forma poderemos aferir também outros tipos de bancos como NewSQL e bancos relacionais tradicionais. Os testes são separados em seis micros testes onde cada um simula uma situação real para bancos distribuídos. Os testes, chamados em inglês de *workloads* que em uma tradução direta seria o equivalente a ‘carga de trabalho’ são nomeados por letras que vão de ‘a’ à ‘f’, que representam diversas aplicações existentes de maneira bastante simplificada. O *framework* também permite a criação de novos *workloads*. Os seis *workloads* básicos são:

a) *workload a: updates,* o teste mescla 50% de operações de leitura com 50% de escrita, uma aplicação exemplo é a gravação de dados de ações recente uma sessão de algum aplicativo;
b) *workload b:* maioria de leituras, neste teste a composição é 95% de leitura e apenas 5% de escrita, exemplo: utilização de *tags* em blogs, inserir a tag é uma escrita porém a grande maioria dos casos se utiliza das tags para leitura;
c) *workload c:* somente leitura, 100% leituras no banco de dados, analise de dados;
d) *workload d:* leitura dos últimos, consiste em um teste onde os últimos dados inseridos são os mais lidos, como exemplos *status* de redes sociais, onde o ultimo é o mais interessante;
e) *workload e:* intervalos curtos, realiza consultas de intervalos de dados, não apenas registros únicos;
f) *workload f:* lê-modifica-escreve, um teste onde o *framework,* lê uma informação, modifica a mesma e depois a grava no banco.

Neste projeto utilizamos para teste apenas o *workload b, pois* o foco do trabalho não é avaliar e comparar os despenhos dos bancos de dados e sim dar uma visão geral dos três modelos e o que cada um tem a oferecer, quais suas características, diferença entre modelos, seu funcionamento, que está sendo demonstrados através dos *benchmarks,* pós e contras e etc..

7.4 AMBIENTE DE TESTE

Nesta seção está detalhado o ambiente de teste utilizado para as avaliações, desde os hardwares utilizados, suas configurações até os softwares que envolveram todo o processo de testes.

7.4.1 Hardware

Para esta avaliação, foram utilizados um desktop e dois notebooks pessoais com as seguintes configurações:

a) *desktop:* Sistema Operacional Windows 7, 2GB de memoria RAM, processador Intel Dual Core 2,2GHz e 320GB de armazenamento;
b) *notebook* 1: Sistema Operacional Windows 8.1, 8GB de memoria RAM, processador Intel Core i7 *Quad Core* 2,2GHz e 720GB de armazenamento;
c) *notebook* 2: Sistema Operacional Windows 8, 4GB de memoria RAM, processador Intel Core 2 Duo 2,1GHz e 240GB de armazenamento;

Para as avaliações feitas com um único nó, todos os testes foram realizados utilizando o *notebook* 1 como computador servidor de bancos de dados e cliente da aplicação, por se tratar do equipamento com maior capacidade de processamento e memoria. As avaliações que utilizaram dois nós foram feitas utilizando o *desktop* como cliente enquanto os *notebooks* 1 e 2 serviram como servidores de banco. Os testes foram executados sob uma rede ethernet de 100Mbs

7.4.2 Software

Como já mencionado na subseção anterior, todos os computadores utilizados rodavam sistema operacional Windows. Nesta subseção estão listados os outros softwares utilizados, desde bancos de dados e suas versões, até os softwares que auxiliaram o desenvolvimento do projeto, atentando que nem todos os softwares listados aqui são obrigatórios para a execução dos testes, alguns serviram apenas como facilitadores do processo.

a) *java SE Runtime Enviroment 7;*

b) *java SE Development Kit 7;*
c) *apache Ant 1.9;*
d) *apache Maven 3.2.2;*
e) *python 2.7.8;*
f) *git 1.9.4;*
g) *mySQL 5.5.39;*
h) *postgreSQL 9.3.5-1;*
i) *nuoDB 2.0.4;*
j) *mySQL Cluster gpl 7.3.6*
k) *mongoDB 2.6.4;*
l) *apache Cassandra 2.1.0;*
m) OLTP-Bench;
n) YCBS 4.

Com o equipamento e softwares definidos e configurados, a próxima seção descreve a metodologia utilizada para efetuar os testes detalhando as configurações utilizadas, formas de aferições, cenários utilizados e objetivos dos testes.

7.5 METODOLOGIA

Após definir os bancos de dados e testes que seriam feitos, foram estudados e analisados os dois *benchmarks* de forma a obter conhecimento sobre estes, para definir com maior clareza quais parâmetros seriam utilizados nas avaliações, desta forma foram definidos os objetivos dos testes e os cenários avaliados.

7.5.1 Objetivos

O objetivo principal desta analise foi conhecer um pouco sobre cada um dos modelos de bancos de dados na teoria e na pratica, avaliando suas características notando os prós e contras de cada modelo identificando situações onde cada banco de dados se enquadra melhor, de maneira imparcial não buscando determinar qual dos modelos é o melhor ou o pior.

E de maneira especifica os objetivos deste projeto foram:

a) analisar a fundamentação teórica por trás dos bancos de dados e seus modelos;
b) comparar diretamente características de cada banco de dados, permitindo visualizar as principais diferenças entre cada modelo;
c) verificar na pratica o comportamento de cada modelo de bancos de dados, avaliando suas usabilidades, facilidades de uso e desempenhos;
d) conhecer métodos de avaliação de bancos de dados e em que casos devem-se basear nesses métodos ao escolher um SGBD;
e) ampliar os conhecimentos técnicos e teóricos sobre o modelo NewSQL, por ser um modelo recente e que une características de dois modelos distintos de bancos de dados, criando uma base de conhecimento para pesquisas futuras.

7.5.2 Cenários de teste

Nesta etapa estão descritos os cenários utilizados para cada *benchmark* e bancos de dados utilizados, separados por *benchmark*. É importante salientar que para todos os testes não foi feito nenhuma otimização dos bancos de dados ou instalados quaisquer aplicativos que possam influenciar no desempenho do banco.

7.5.2.1 TPC-C

Para as execuções dos testes TPC-C, como já mencionado, foi utilizado o *framework* OLTP-Bench. A execução de um *workload* através do OLTP-Bench é bastante simplificada, devido toda a configuração necessária estar mantida nos arquivos de configuração de cada teste. No arquivo de configuração, basicamente, é definido o banco de dados e suas informações de conexão, o fator de escala, numero de terminais virtuais, que são equivalentes a conexões no banco, tempo e pesos de cada processo.

Os testes foram aplicados nos bancos MySQL (InnoDB), PostgreSQL, NuoDB e MySQL Cluster (NDBCluster). Os bancos não tiveram nenhum tipo de otimização, a única alteração nas configurações de cada banco foram os números

de conexões máximas para acima de cem conexões. Todos foram executados utilizando as mesmas configurações.

Foram feitos quatro baterias de testes, onde cada bateria é composta por dez execuções idênticas e gerado a média dos resultados. As diferenças entre cada uma das baterias de teste é apenas a quantidade de terminais virtuais, que equivalem ao numero de conexões ao banco de dados. A primeira bateria com apenas uma conexão, a segunda com 10, a terceira com 50 e a ultima com 100 conexões simultâneas. Cada um dos 10 testes de cada bateria teve duração de 60 segundos e fator de escala 10, no TPC-C este valor representa a quantidade de armazéns que serão carregados no banco de dados antes de iniciar os testes. Conforme a figura 9, os números de registros de cada tabela irão variar de acordo com a quantidade de armazéns. Para bancos NewSQL os testes foram feitos duas vezes, uma utilizando apenas um nó, outra com um cluster de dois nós. Desta forma tivemos os seguintes cenários avaliados:

a) primeiro cenário: Banco MySQL, quatro baterias de 10 testes, cada uma das baterias variando entre 1, 10, 50 e 100 conexões;
b) segundo cenário: Banco PostgreSQL. quatro baterias de 10 testes, cada uma das baterias variando entre 1, 10, 50 e 100 conexões;
c) terceiro cenário: Banco NuoDB, oito baterias de 10 testes, sendo quatro baterias com um nó e quatro com dois nós configurados, cada uma das baterias variando entre 1, 10, 50 e 100 conexões;
d) quarto cenário: Banco MySQL Cluster, oito baterias de 10 testes, sendo quatro baterias com um nó e quatro com dois nós configurados, cada uma das baterias variando entre 1, 10, 50 e 100 conexões.

Para o banco MySQL Cluster é necessário registrar algumas configurações que foram necessárias para realizar os testes. Por utilizar o mesmo gerenciador do MySQL, tendo de diferente apenas o motor de banco de dados, enquanto MySQL comum utiliza os motores MyISAN e InnoDB, o MySQL Cluster utiliza o morto NDBCluster, a maneira de conectar no MySQL Cluster é a mesma que a do MySQL, desta forma ao carregar os dados, as tabelas criadas, por padrão serão do tipo InnoDB, devido a isso foi necessário alterar o script de criação de tabelas para fossem criadas com o motor correto.

No anexo A está descrito um arquivo de configuração do teste TPC-C e no anexo B os comandos para executa-lo.

Os testes TPC-C foram executados da seguinte forma:

a) criação: criar o banco de dados ‘TPC-C’ no SGBD que será analisado;
b) arquivos de configuração: criar os arquivos de configuração para cada quantidade de conexões que serão feitas com a configuração de conexão do banco;
c) carga de dados: executar o comando para carregar dados no banco, este comando também já é responsável por criar as tabelas, chaves e índices, a carga de dados só foi feita uma vez para os testes TPC-C;
d) execução dos testes: executar 10 vezes os testes de cada cenário;
e) analise dos resultados: cada teste gera uma serie de arquivos de logs e resultados dos testes, o arquivo com os resultados de rendimento e latência média é o arquivo com extensão ‘.res’.

O arquivo de resultados do OLTP-Bench é um arquivo de texto separado por vírgula conhecido como Comma Separated Values ou CSV. Durante os testes foi notado um ‘*bug’* na geração destes arquivos. Todos os valores de resultado, com exceção do tempo, possuem três casas decimais, como no Brasil o separador de casas decimais também é a vírgula, não existe distinção entre decimais e colunas de maneira clara e visível.

7.5.2.1 YCSB

Os testes com o YCSB foram executados utilizando o *framework* original desenvolvido por Brian Cooper do Yahoo! utilizando o *workloadB*. Ao compilar a suíte através do *Apache Maven,* já são gerados juntos os drives para as conexões com os diversos bancos suportados. Por ter os drives de conexão gerados pelo próprio YCSB ao executar os testes, muitas vezes basta apenas informar o *workload* que será gerado e o nome do banco que será testado, deixando o processo bem simples, porém para conexões JDBS os parâmetros de conexão são passados manualmente, devido cada banco ter sua conexão JDBC distinta.

Para os testes os parâmetros *recordcount* e *operationcount* do *workload* de 1000 para 100000. Os demais parâmetros foram manipulados através dos comandos de execução. Seguindo o mesmo modelo dos testes com o TPC-C, as baterias foram divididas pelas quantidades de conexões no banco, neste caso, controlada pelo parâmetro *threadcount,* que foi manipulado ao ser executado cada

teste, utilizamos as mesmas variações que os testes anteriores, 1, 10, 50 e 100 conexões. Outro parâmetro que foi controlado é o *maxexecutiontime,* este parâmetro controla o tempo de execução de cada teste, utilizamos 60 segundos neste parâmetro. No caso do YCSB o tempo não é absoluto, o YCSB executa a quantidade de operações descrita no parâmetro *recordcount,* caso tenha algum tempo informado, o YCSB termina a execução no momento em que um dos parâmetros atingirem o limite. Por todos os bancos utilizados serem distribuídos, todos os testes foram feitos com cluster de nó único e de 2 nós, os cenários são os seguintes:

a) primeiro cenário: Banco Apache Cassandra, oito baterias de 10 testes cada, sendo quatro baterias com um nó e quatro com dois nós configurados, cada uma das baterias variando entre 1, 10, 50 e 100 conexões;
b) segundo cenário: Banco NuoDB, oito baterias de 10 testes cada, sendo quatro baterias com um nó e quatro com dois nós configurados, cada uma das baterias variando entre 1, 10, 50 e 100 conexões;
c) terceiro cenário: Banco MongoDB, oito baterias de 10 testes cada, sendo quatro baterias com um nó e quatro com dois nós configurados, cada uma das baterias variando entre 1, 10, 50 e 100 conexões;
d) quarto cenário: Banco MySQL Cluster, oito baterias de 10 testes cada, sendo quatro baterias com um nó e quatro com dois nós configurados, cada uma das baterias variando entre 1, 10, 50 e 100 conexões.

No anexo C e D estão descritos o *workload B*, e os comandos de execução do YCSB. Os testes YCSB foram executados da seguinte forma:

a) criação: criar o banco de dados ‘YCSB’ no SGBD que será analisado
b) carga e execução: executar o comando de carga de banco de dados e executar os testes 10 vezes para cada cenário. Diferente do TPC-C, o YCSB não necessita de arquivos de configurações para cada banco, o banco é definido por parâmetro no momento da execução dos testes, assim como o numero de conexões. Para estes testes, a cada bateria de 10 execuções foi feito uma nova carga de dados;
c) analise dos resultados: No YCSB o arquivo de saída é determinado pelo próprio usuário por parâmetro no momento da

execução. O arquivo gerado é um arquivo de texto, onde informações como rendimento, latência média e outras já estão calculadas.

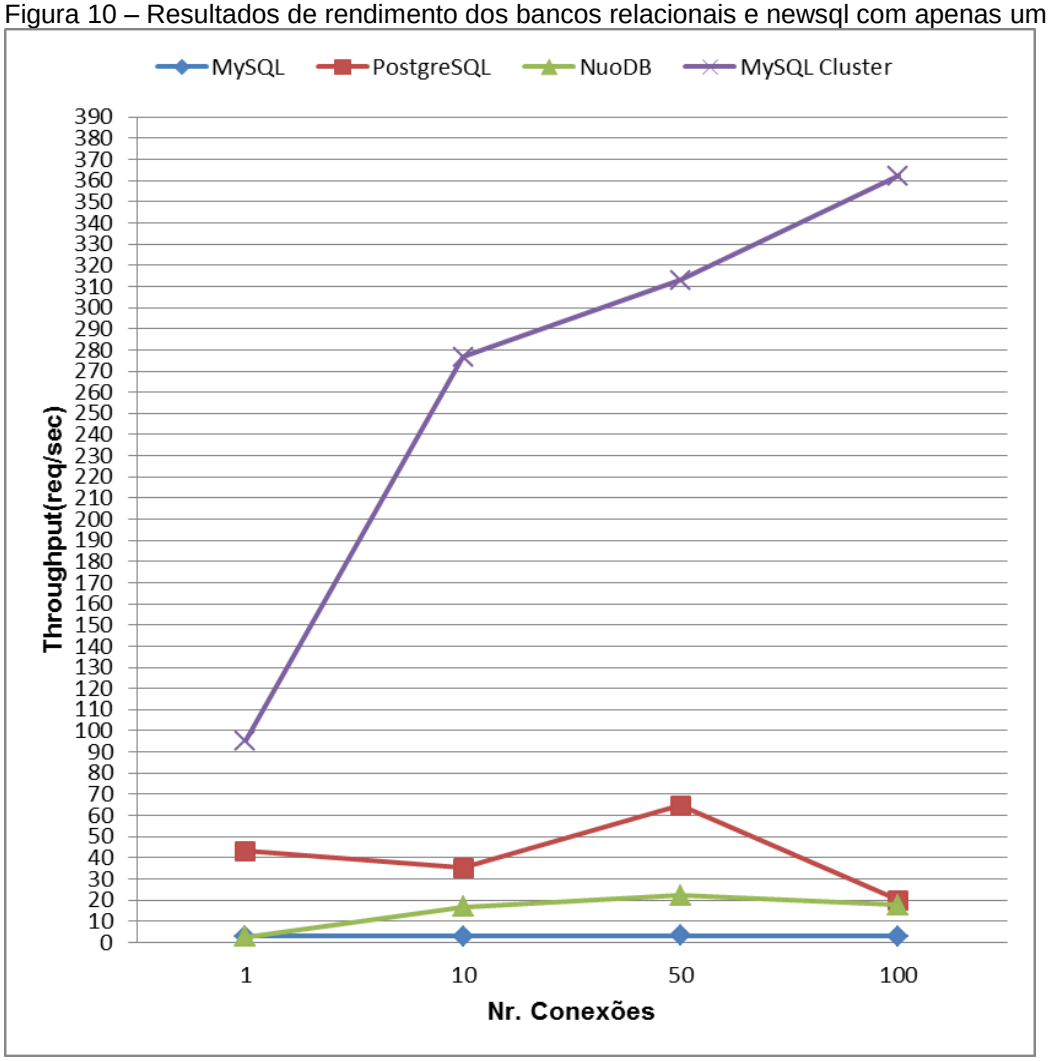
8 RESULTADOS OBTIDOS

Depois de realizadas as baterias de testes, os relatórios dos *benchmarks* foram somados e calculados as médias de rendimento por segundo e latência média do banco e gerados os gráficos de resultado finais.

8.1 TPC-C

Para o *benchmark* TPC-C as analises foram separadas em dois grupos o primeiro composto pelos bancos relacionais tradicionais mais os bancos NewSQL com apenas um nó e no segundo grupo os bancos NewSQL com um e dois nós. As informações dos gráficos para bancos NewSQL no segundo grupo, são as mesmas do primeiro, apenas foram replicas para facilitar analise do banco ao adicionar um novo nó.

Figura 10 – Resultados de rendimento dos bancos relacionais e newsql com apenas um nó.

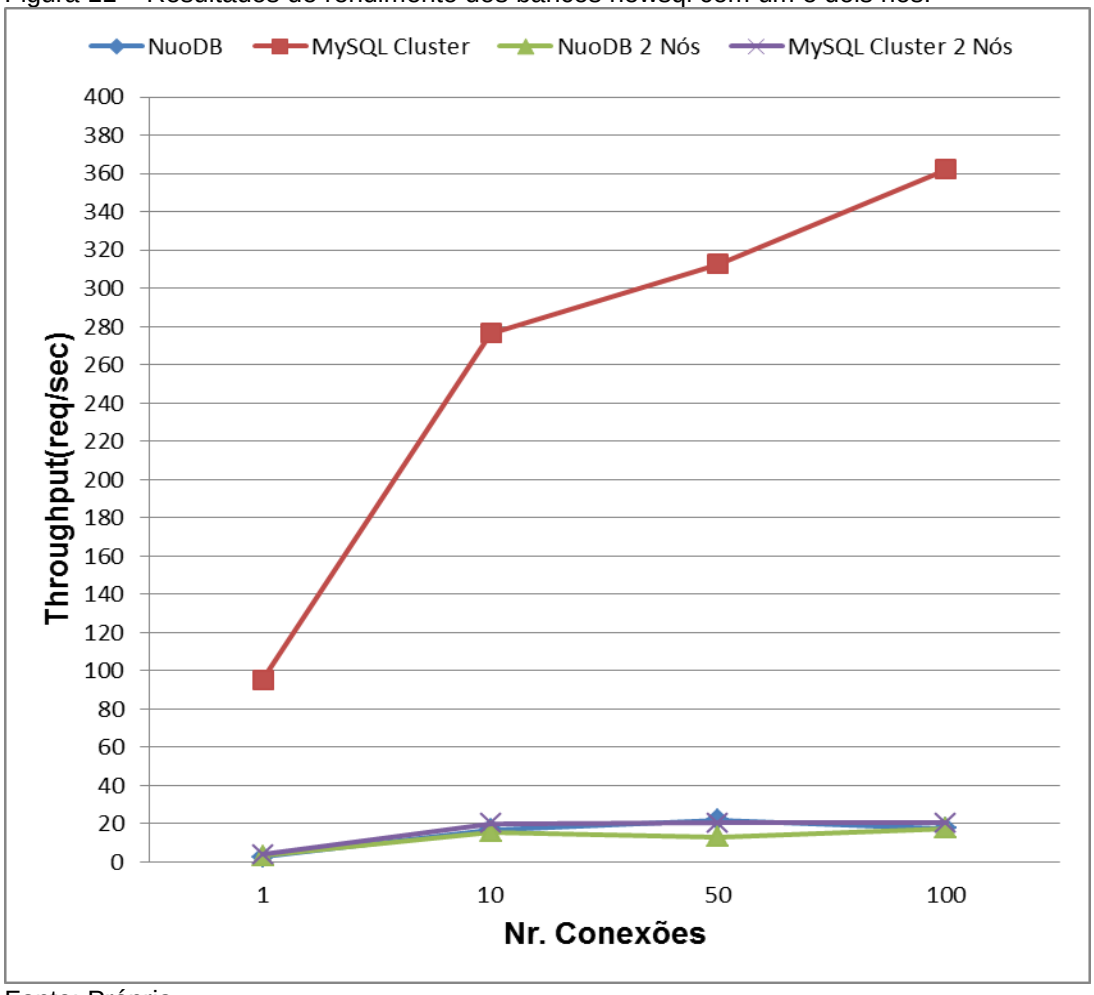


Fonte: Própria.

A figura 10 demonstra o comportamento do rendimento (*throughput*) do banco de dados ao aumentamos o numero de conexões. A tendência seria ter um aumento gradativo do rendimento do banco de dados até que o numero de conexões ao banco sature o seu processamento. Situação que fica notável nos bancos PostgreSQL e NuoDB, onde tiveram seus melhores resultados nos testes com 50 conexões, após isso houve perda de desempenho.

Outro fator visível na figura 10 é a disparidade de desempenho entre o banco de dados MySQL Cluster e os outros bancos, isto ocorreu devido ao fato de que o motor *NDBCluster* é desenvolvido para ter alta performance e disponibilidade para ser utilizados em ambientes distribuídos através da arquitetura *shared-nothing* enquanto o motor *InnoDB* é um motor relacional tradicional, focado em dados estruturados e com transações seguras e garantia a ACID através da arquitetura *shared-disk.* Logo fica claro que a comparação de desempenhos entre os estes bancos é injusta e 'desnecessária' já que cada banco foi desenvolvido com um proposito diferente.

Figura 11 – Resultados de rendimento dos bancos newsql com um e dois nós.

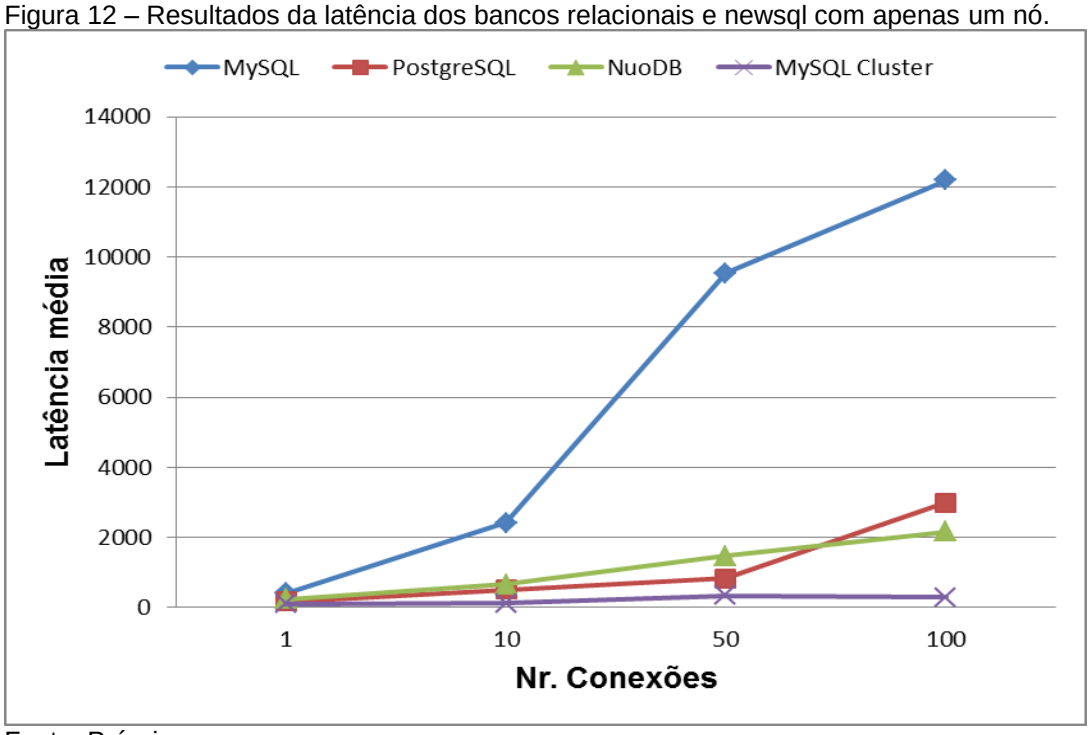


Fonte: Própria.

Na figura 11 estão descritos os resultados das comparações dos bancos NewSQL com um e com dois nós. Era previsto que ao aumentar a quantidade de nós o desempenho geral do SGBD aumentasse também, porém os testes feitos descrevem o contrario, ao aumentar o numero de nós físicos, o desempenho se manteve o mesmo e em alguns casos piorou, nitidamente visto nos resultados do MySQL Cluster, que teve uma perda de desempenho bastante considerável ao ser executado utilizando duas máquinas. Acredito que isto foi resultante das configurações feitas e do ambiente em que foram analisados, já que se tratam de equipamentos de uso pessoais não preparados para tal finalidade, além da diferença de desempenho entre os próprios nós do *cluster.*

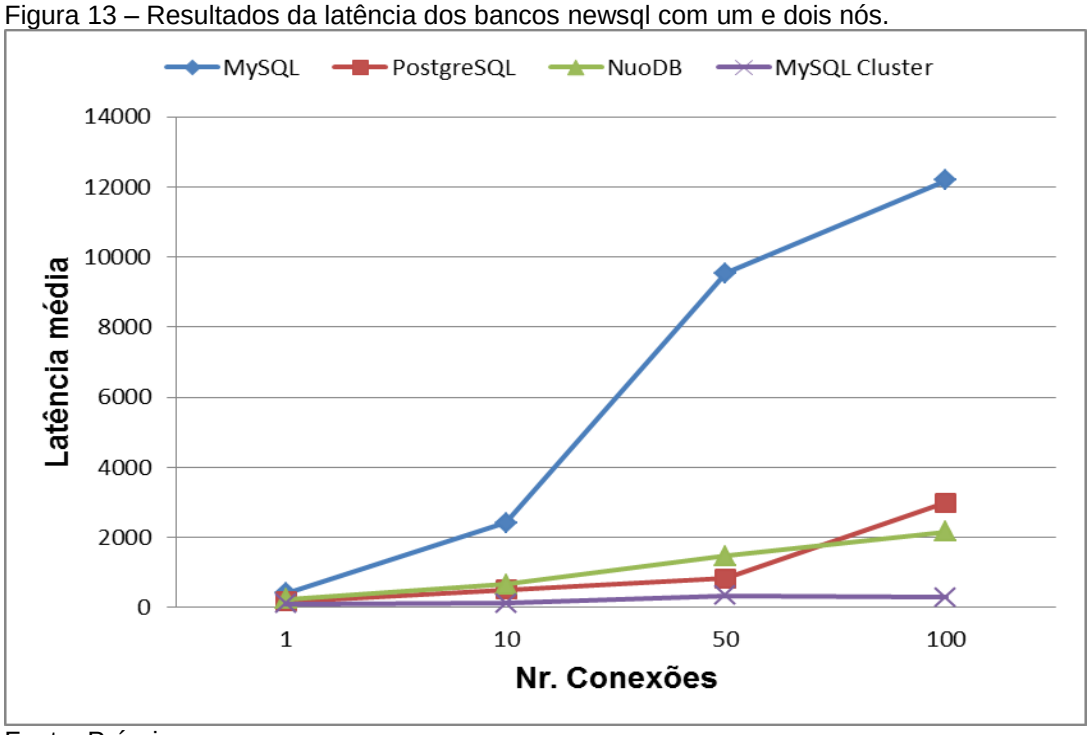
Outro ponto observado foi à latência média do banco de dados durante os testes, isto é, o tempo entre o inicio de uma transação até o seu resultado. Como esperado, a figura 12 e 13 mostram que quantos mais conexões simultâneas realizando operações no banco, maior é a sua latência.

Figura 12 – Resultados da latência dos bancos relacionais e newsql com apenas um nó.



Fonte: Própria.

Figura 13 – Resultados da latência dos bancos newsql com um e dois nós.



Fonte: Própria.

8.2 YCSB

Os resultados do *benchmark* YCSB foram agrupados por banco de dados, comparando os resultados dos testes com um único nó e dos testes com dois nós. Nestes testes foram observados o rendimento banco e sua latência média.

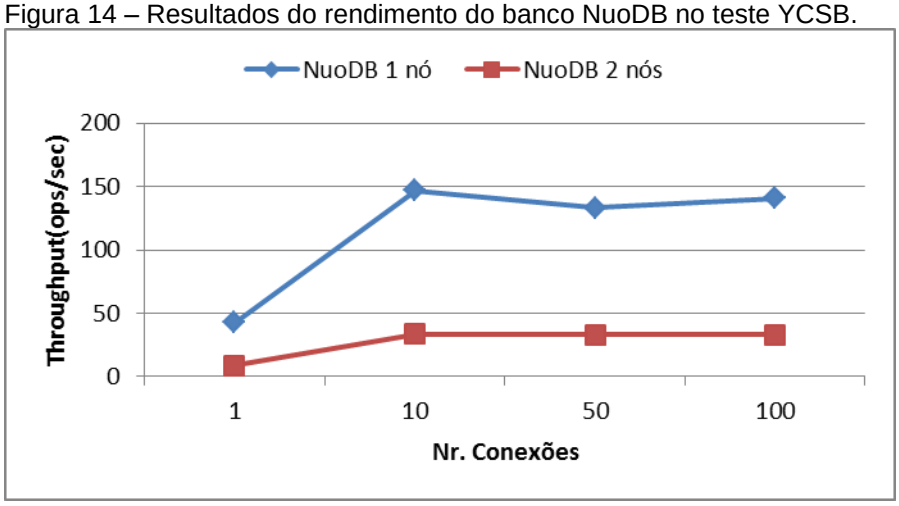
Como o *workload* utilizado (*workload b)* é focado em principalmente leitura de dados, anotamos a latência das atividades relacionadas à leitura apenas, porém nos arquivos de logs gerados pelos testes é possível consultar além da latência média as latências mínimas e máximas, tanto para operações de leitura quanto para a de escrita. Lembrando que estes testes foram executados para ter-se uma noção não só do desempenho, mas principalmente para obter conhecimento mais aprofundado sobre cada um dos bancos estudados. Nas subseções a seguir estão descritas as evoluções dos bancos de dados avaliados no teste YCSB conforme o numero de *threads* era aumentado.

8.2.1 NuoDB

Nas figuras 14 e 15 analisamos os resultados obtidos nos testes com o NuoDB. Nos gráficos podemos perceber que ao adicionar um novo nó ao *cluster* o

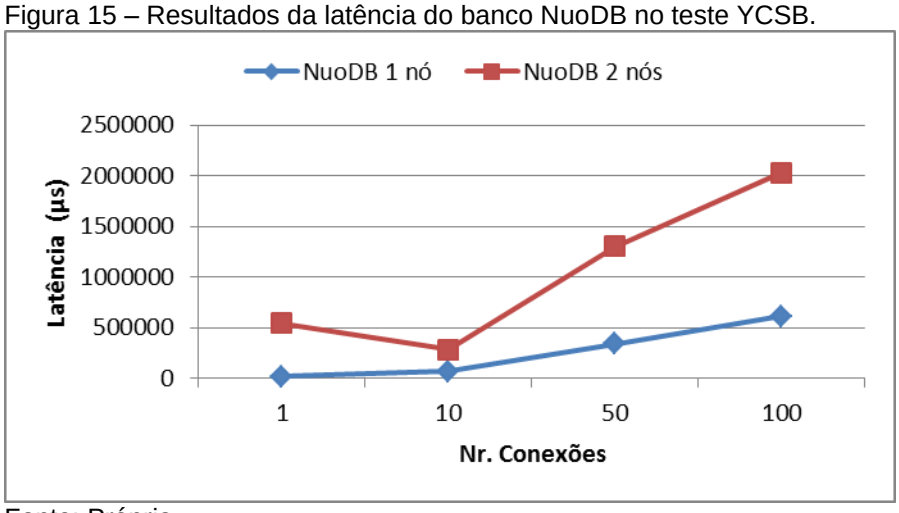
desempenho geral do banco foi prejudicado. Acredito que tal resultado possa é causado por alguns fatores distintos, como o ambiente precário onde os dois servidores de banco possuem diferenças consideráveis de hardware, ou problemas de administração de memoria e carga do próprio SGBD.

Figura 14 – Resultados do rendimento do banco NuoDB no teste YCSB.



Fonte: Própria.

Figura 15 – Resultados da latência do banco NuoDB no teste YCSB.

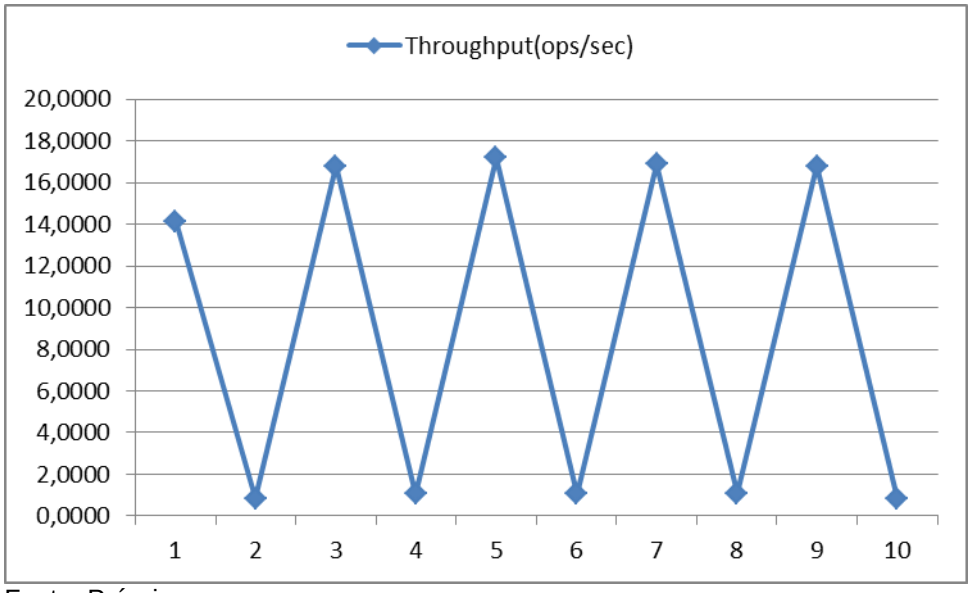


Fonte: Própria.

Apesar da perda geral de desempenho de forma geral, ao acrescentar um novo nó físico ao banco, os gráficos que utilizando dois nós ao aumentar de uma conexão para dez, o SGDB teve acréscimo real de desempenho, onde tanto o rendimento (*Throughput*) quanto a latência tiveram resultados positivos, em outras palavras, aumento da quantidade de transações por segundo (TPS) e diminuição da latência.

Outro ponto que chamou atenção nos resultados do NuoDB foi seus resultados individuais nos testes com dois nós e apenas uma conexão, os resultados foram colocados no gráfico à figura 16.

Figura 16 – Resultado dos testes com dois nós e apenas uma conexão do banco NuoDB no teste YCSB.



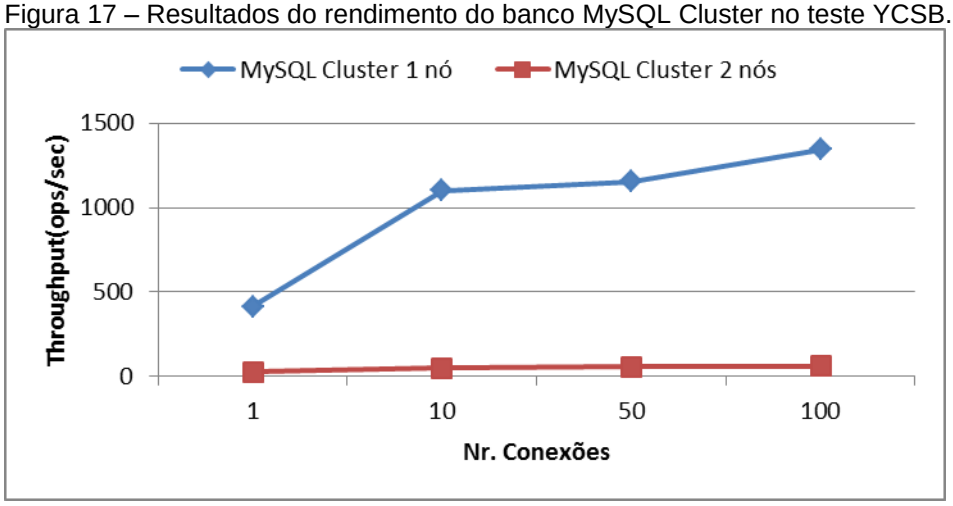
Fonte: Própria.

O gráfico mostra uma variação uniforme entre aproximadamente 17 transações por segundo e uma transação, a latência do banco também sofre o mesmo padrão de variação. Considerando que os dez testes foram executados de forma sequencial, sem tempo de espera entre cada teste e com apenas uma conexão, acredito que a cada teste o SGBD delegou o conjunto de operações para apenas um nó, desta forma quando o *notebook* 2 era requisitado, o resultado do processamento foi consideravelmente pior que os testes executados pelo *notebook 1*. Nos testes com mais conexões este fator não é visível provavelmente por trabalharem já com 10 conexões ou mais, dessa forma os resultados ficam balanceados entre os processamentos do *notebook 1* e do *notebook 2.* Para ter um melhor entendimento sobre este problema, novos testes teriam que ser feitos, com posse de um ambiente de testes mais homogêneo. Desta forma, com base nos testes não é possível afirmar com certeza o aumento de desempenho a cada novo nó adicionado ao *cluster.*

8.2.2 MySQL Cluster

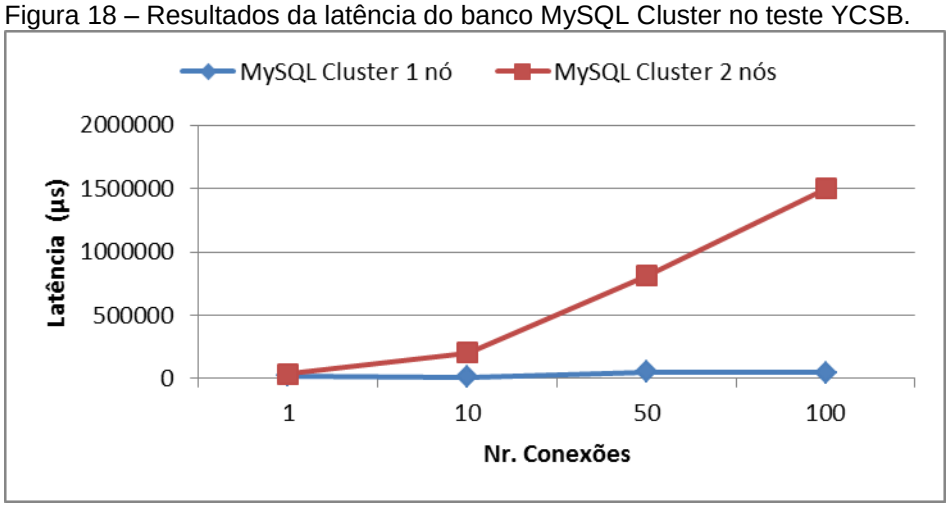
Nas figuras 17 e 18 estão os valores atingidos pelo MySQL Cluster, que semelhante ao NuoDB, teve perda de desempenho ao ser executado com um nó a mais, neste caso a perda foi muito maior que o NuoDB. Enquanto os testes *single-node* atingiram aproximadamente 1346 operações por segundo, ao adicionar um novo nó o máximo atingindo foi em torno de 61 operações.

Figura 17 – Resultados do rendimento do banco MySQL Cluster no teste YCSB.



Fonte: Própria.

Figura 18 – Resultados da latência do banco MySQL Cluster no teste YCSB.



Fonte: Própria.

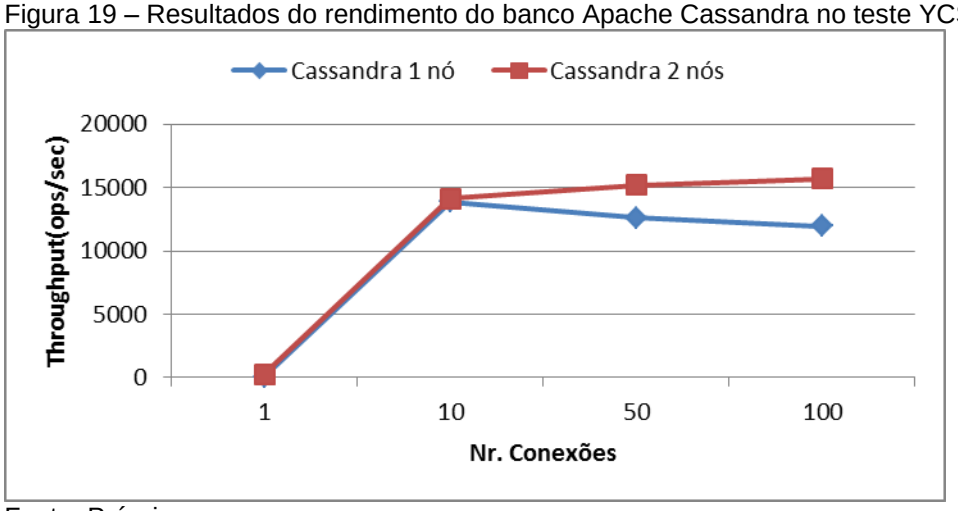
8.2.3 Cassandra

Nos testes práticos o *Apache Cassandra,* ou apenas *Cassandra,* apresentou melhores resultados que os bancos NewSQL, apresentando um crescimento

bastante satisfatório no gráfico de rendimento, representado na figura 19, e o oposto no gráfico de latência da figura 20.

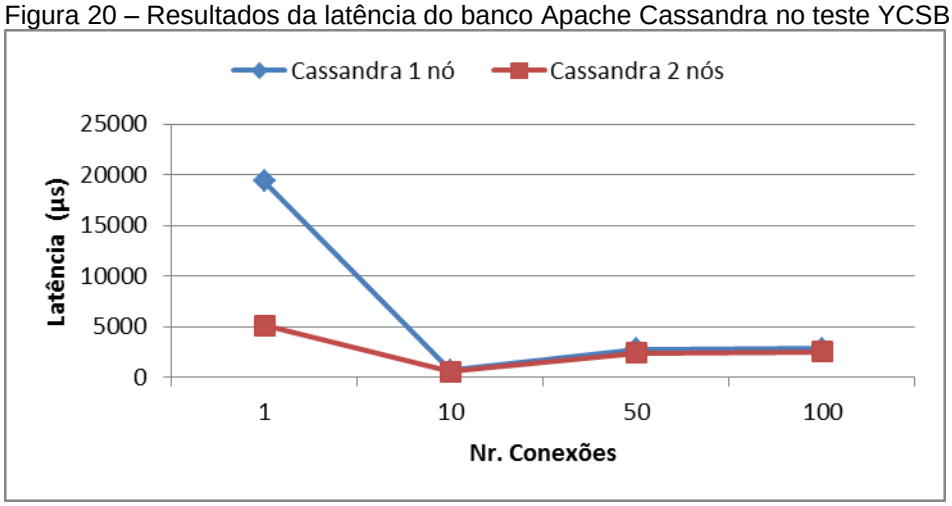
A primeira bateria realizada apenas com uma conexão o Cassandra atingiu uma média de 252 processos por segundo, porém quando testado com 10 conexões ou mais conseguiu atingir mais de 1500 TPS.

Figura 19 – Resultados do rendimento do banco Apache Cassandra no teste YCSB.



Fonte: Própria.

Figura 20 – Resultados da latência do banco Apache Cassandra no teste YCSB.



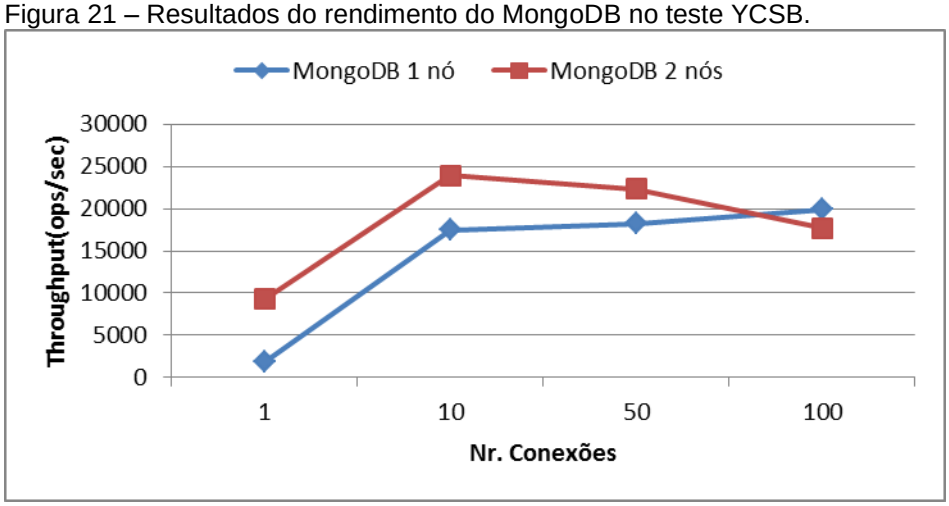
Fonte: Própria.

Outro ponto que merece ser observado nos resultados do Apache Cassandra é o desempenho entre os testes com apenas um nó e com dois nós conectados, nos cenários com dois nós no *cluster* o Cassandra obteve tanto a latência quanto o rendimento de dados melhor que os testes feitos em um único nó. Comprovando a elasticidade e o aumento de desempenho ao adicionar novos nós ao cluster.

8.2.4 MongoDB

Seguindo um resultado semelhante ao do Apache Cassandra, o MongoDB também apresentou melhorias de desempenho ao acrescentar um novo nó ao banco de dados, além de apresentar resultados de rendimento melhores que o do Cassandra, conforme mostra a figura 21.

Figura 21 – Resultados do rendimento do MongoDB no teste YCSB.

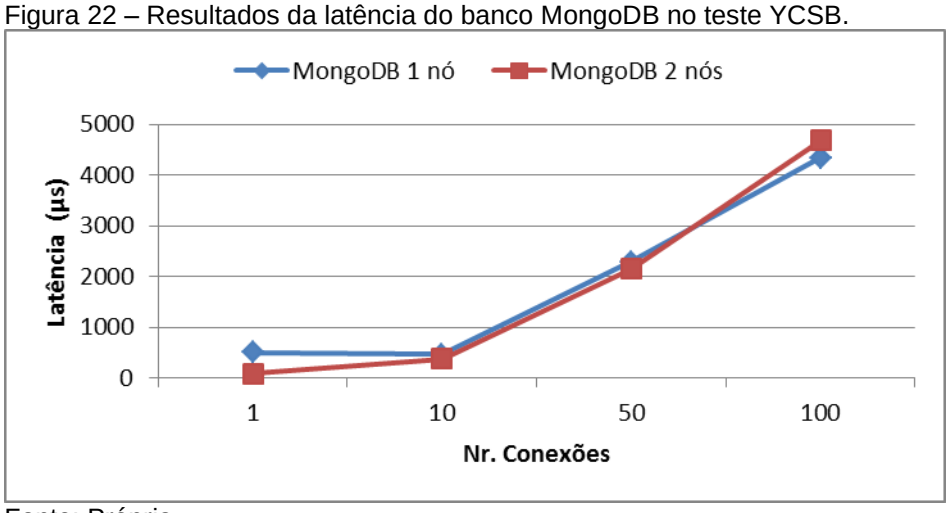


Fonte: Própria.

Porém ao atingirmos os testes com o numero de conexões acima de 50 o desempenho começou a regredir, provavelmente devido à quantidade de conexões e o hardware utilizado saturarem o banco com facilidade, ficando levemente abaixo dos testes com apenas um nó.

A figura 22 deixa clara a perda de desempenho a partir de 50 conexões quando além do numero de operações por segundo ter diminuído, a latência média do banco aumenta consideravelmente.

Figura 22 – Resultados da latência do banco MongoDB no teste YCSB.



Fonte: Própria.

Durante o inicio dos testes com o MongoDB, ao tentar executar a carga do banco de dados com o cluster com dois nós, o YCSB! emitia um erro de duplicidade

de chaves e não carregava o banco de dados. Ao parar um dos nós do cluster, deixando-o apenas com um nó, o YCSB! realizou a carga de dados corretamente sem problemas, após terminar a carga de dados, o segundo nó foi reconectado ao cluster e sincronizado automaticamente pelo SGBD.

8.3 LIMITAÇÕES

Alguns pontos que limitaram os testes e resultados obtidos que merecem ser pontuados.

a) limitação de tempo de execução dos *benchmarks*: os testes foram limitados a 1 minuto, tempo padrão do teste TPC-C do OLTP-Bench. Como o objetivo principal dos testes, era o conhecimento geral do banco e não realmente um teste de desempenho, optou-se por este tempo de execução, pois facilitaria a execução de vários testes, analisando os parâmetros oferecidos pelos testes conhecendo mais a fundo as ferramentas utilizadas;
b) hardware utilizado: os hardwares utilizados são de uso pessoal, não estruturados para este tipo de aplicações, o caso do *notebook 1* apresenta boa especificação pra computadores pessoais, porém está longe de ter o desempenho e a memoria de um servidor, os outros computadores utilizados possuem hardware abaixo da média de mercado atual impedindo que os SGBDS demonstrem seu máximo potencial;
c) quantidade de nós utilizados: foi utilizado um cluster de apenas dois nós para os bancos NoSQL e NewSQL, e rede *ethernet* 100Mbs, em cenários reis de aplicações para estes modelos o numero de nós do conglomerado são bem maiores que o utilizado, além da infraestrutura de rede.

9 CONCLUSÃO

Não há duvidas de que a expansão da computação na nuvem mudou e continua mudando a maneira como pensamos no momento de desenvolver novas aplicações e junto com isso como devemos tratar e armazenar informações. Este advento mudou não somente a maneira de programar ferramentas voltadas para a nuvem, mas em diversas áreas da computação. Um exemplo disto são aplicações de BI, que antigamente eram suportadas por *datawarehouses* atualmente, este tipo de conceito já começa a ficar ultrapassado, devido ao surgimento do *big data* que também só foi possível devido ao surgimento dos bancos em memoria, NoSQL, focados no desempenho e não em segurança, junto com surgimento de ferramentas de processamento de dados em paralelismo como o *Apache Hadoop* e o *Apache Spark*.

Atualmente estão quase que em constante conexão com os bancos de dados NoSQL assim como em bancos de dados relacionais. Redes sociais, mensagens instantâneas, sites de busca e uma diversidade enorme de serviços online utilizam tecnologia NoSQL, devido à necessidade de desempenho e disponibilidade, nem que pra isso segurança e facilidade de manuseio dos dados seja perdida. Ao mesmo tempo, que bancos, supermercados, fábricas de todos os tipos, utilizam sistemas que se apoiam na segurança e na facilidade de manuseio dos bancos relacionais mesmo que não tenha um desempenho tão alto. São aplicações muito distintas uma das outras, logo as soluções em banco também são distintas. Porém existem casos de soluções que mesclam a necessidade de desempenho com facilidade de manuseio dos dados através das linguagens de consulta e segurança, como por exemplos, aplicações de comercio eletrônico, onde o existe a necessidade de disponibilidade e segurança dos dados ou aplicações de BI, que precisam de desempenho e facilidade de acesso aos dados. São aplicações que podem ser atendidas pelos dois modelos de bancos de dados, porém muito trabalho em nível de programação deve ser feito para compensar um ponto ou outro devido à limitação de cada modelo. Para estes casos, começaram recentemente a surgir os bancos NewSQL, quem buscam atender situações onde o desempenho e a disponibilidade são tão importantes quanto segurança e facilidade de acesso aos dados.

Bancos NewSQL surgiram a muito pouco tempo e começou a chamar a atenção a menos tempo ainda, existem diversos projetos semelhantes a este sendo feitos ainda buscando explorar e conhecer mais deste modelo. Graças a isso não é possível ainda dizer se é um modelo que surgiu pra tomar mercado e se estabelecer, neste projeto percebeu-se que os bancos deste modelo possuem características dos dois modelos mais tradicionais de banco de dados, porém nos testes práticos foi possível perceber um pouco de imaturidade, tanto por parte do banco quando das interfaces disponíveis e que alguns pontos que são afirmados pelos fabricantes, como o aumento de desempenho a cada novo nó adicionado, não é tão verdade assim, deve ter uma serie de outras atividades que devem ser feitas para que o banco possa aproveitar o desempenho desse novo nó e render mais.

É fato que existe a tendência de um crescimento deste modelo proposto pelos bancos NewSQL, um dos fatores que monstra isto é a ultima versão do banco de dados da Oracle, o Oracle 12c, onde este já começa apontar seu foco para a nuvem. Existe também um caso com o PostgreSQL onde o mesmo foi adaptado para trabalhar como NewSQL. Com isso podemos dizer que talvez este seja o futuro para todos os bancos relacionais tradicionais, porém ainda é cedo para afirmar com total certeza e enquanto isso bancos relacionais e NoSQL continuarão a manter seu mercado sem maiores problemas.

9.2 TRABALHOS FUTUROS

Este trabalho analisa e apresenta uma analise comparativa entre os três modelos de bancos de dados principalmente a nível teórico. As analise de desempenho foram apenas para ter-se uma noção do funcionamento geral dos bancos de dados. Desta forma, esta analise genérica, entre os três modelos, abre portas para diversos outros trabalhos relacionais a área.

Em primeiro lugar, ampliar as condições do ambiente de teste e a execução de testes, focando na comparação de desempenhos não só dos bancos analisados mas também de outros SGBD, sempre separando as analises de acordo com o objetivo do SGBD, além da execução de outros *benchmarks,* como teste de carga e de estresse.

Analisar a fundo o comportamento dos bancos ao trabalharem em uma rede de computadores homogênea e em uma rede heterogênea e verificar se os

resultados obtidos nos testes deste projeto foram realmente prejudicados por questão de *hardware* e rede.

Outro ponto a ser trabalhado no futuro, são implementações em aplicações reais com algum bancos de dados NewSQL. Bancos de dados NewSQL ainda são ferramentas relativamente novas pertos dos SGBDS NoSQL e Relacionais tradicionais, porém demonstram um novo caminho a ser seguido no mercado de banco de dados.

Outra das oportunidades de trabalhos futuros neste tema, é a possibilidade de interligar os três modelos através de uma solução de aplicações integradas, onde cada aplicação do conjunto utilize um banco de dados apropriado para seu objetivo, por exemplo um sistema ERP, que integre com comercio eletrônico, aplicações de inteligência de negocio e central de atendimento ao consumidor. Onde cada aplicação pode ser atendida com mais eficácia por determinado modelo de banco de dados.

REFERÊNCIAS

Aslett, M. How will the database incumbents respond to NoSQL and NewSQL?. The 451 Group, 2011.

AsletT, M. NoSQL, NewSQL and Beyond: The answer to SPRAINed relational databases. Disponível em: <http://blogs.the451group.com/information_management/2011/04/15/nosql-newsql-and-beyond/>. Acesso em 10 de jun de 2014.

AsletT, M. What we talk about when we talk about NewSQL. Disponível em: < http://blogs.the451group.com/information_management/2011/04/06/what-we-talk-about-when-we-talk-about-newsql/>. Acesso em 10 de jun de 2014.

Brewer, E. Towards Robust Distributed System. Symposium on Principles of Distributed Computing (PODC), 2000.

Burd, G. What Is NoSQL? Why NoSQL?. Usenix, 2011.

Chen, P. The Entity Relationship Model—Toward a Unified View of Data, ACM Transactions on Database Systems, 1976.

Codd, E. F. A relational model of data for large shared data Banks. Comunications of the ACM, v.13, e.6, Junho 1970.

Codd, E. F. The relational model for database management : version 2. Addison-Wesley Publishing Company, Inc, 1990.

Cooper, B. et al. Benchmarking Cloud Serving Systems with YCSB, Yahoo!, Research, Santa Clara, CA, USA, 2010. Disponível em <http://labs.yahoo.com/files/ycsb.pdf>. Acessado em 20 out. 2014.

Cooper, B. Yahoo! Cloud Serving Benchmark Overview and Results, Yahoo!, Research, Santa Clara, CA, USA, 2010. Disponível em <http://labs.yahoo.com/files/ycsb-v4.pdf>. Acessado em 20 out. 2014.

Cooper, B. YCSB. Core Workloads. Disponível em < https://github.com/brianfrankcooper/YCSB/wiki/Core-Workloads>. Acesso em 15 out 2014;

Cooper, B. YCSB. Running a Workloads. Disponível em < https://github.com/brianfrankcooper/YCSB/wiki/Running-a-Workload>. Acesso em 15 out 2014;

Curino, C. el al. Benchmarking OLTP/Web Databases in the Cloud: The OLTP-Bench Framework, 2012, Clouddb2012. Disponivel em <www.cs.cmu.edu/~pavlo/static/papers/oltpbench.pdf>. Acesso em 14 out. 2014.

Date, C. J. Introdução a Sistemas de Banco de Dados. Rio de Janeiro: Elsevier Editora, 2004.

Difallah, D. el al. OLTP-Bench: An Extensible Testbed for Benchmarking Relational Databases. Dez. 2013. Proceedings of the Very Large Database Endowment, vol.7, no. 4, pg. 277-288.

Dubray, J. NoSQL, NewSQL e Além. Disponível em <http://www.infoq.com/br/news/2011/06/newsql>.Acesso em 18 mar. 2014.

Edlich, S., et al. NoSQL: Einstieg in die Welt nichtrelationaler Web 2.0 Datenbanken. Hanser Fachbuchverlag, 2010.

Furtado. G. A história dos bancos de dados. Disponível em <http://www.dicasdeprogramacao.com.br/a-historia-dos-bancos-de-dados/>. Acesso em 25 mar. 2014.

Gilbert S.; Lynch , N. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. SIGACT News, 2002.

Haerder, T.; Reuter, A. - Principles of transaction-oriented database recovery. ACM Computing Surveys, 1983.

Grolinger et al. Data management in cloud environments: NoSQL and NewSQL data stores. Journal of Cloud Computing: Advances, Systems and Applications, 2013.

Harrison, G. 10 things you should know about NoSQL databases. Disponível em: <http://www.techrepublic.com/blog/10-things/10-things-you-should-know-about-nosql-databases/>. Acesso em 24 mar. 2014.

Heuser, C. A. Projeto de banco de dados. Porto Alegre: Bookman, 2009.

Horowitz, Michael L. – An Introduction to Object-Oriented Dabases and Database Systems. Pittsburgh: Information Technology Center Carnegie Mellon University, 1991.

Kumar, R. et al. Critical Analysis of Database Management Using NewSQL. International Journal of Computer Science and Mobile Computing, v. 3, p.434 – 438, Maio, 2014.

Leavitt, N. Will. NoSQL Databases Live Up to Their Promise? 2010. Computer, 12. ISSN: 0018-9162, pg. 12 – 14.

Michael Stonebreaker. Em: Wikipédia: a enciclopédia livre. Disponível em <http://en.wikipedia.org/wiki/Michael_Stonebraker>. Acessado em 25 mar. 2014.

Moniruzzaman, A. B. M..; Hossain, S. A. NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison. International Journal of Database Theory and Application, Vol. 6, No. 4, 2013.

Morgan, A. My first Cluster running on Windows. Disponível em < http://www.clusterdb.com/mysql-cluster/running-mysql-cluster-over-multiple-servers>. Acesso em 28 out. 2014.

Morgan, A. Running MySQL Cluster over multiple Windows servers. Disponível em < http://www.clusterdb.com/mysql-cluster/running-mysql-cluster-over-multiple-servers>. Acesso em 28 out. 2014.

Murugan, A. S. P. A Study of NoSQL and NewSQL databases for data aggregation on Big Data, Royal Institute of Technology, Etocolmo, Suiça, 2013.

MySQL, MySQL Cluster Documentation Library - Differences Between the NDB and InnoDB Storage Engines. Disponível em <http://dev.mysql.com/doc/mysql-cluster-excerpt/5.1/en/mysql-cluster-ndb-innodb-engines.html>. Acessado em 04 de nov. 2014.

Nascimento, J. NoSQL – você realmente sabe do que estamos falando?. Disponível em: <http://imasters.com.br/artigo/17043/banco-de-dados/nosql-voce-realmente-sabe-do-que-estamos-falando/>. Acesso em 24 mar. 2014.

NuoDB. A Techinical Whitepaper, Versão 1, NuoDB, Inc., Cambridge, MA, 2013. Disponivel em <http://go.nuodb.com/why-you-need-a-distributed-database.html>. Acessado em 02 de Jun. 2014.

Oliveira, C. H. P. SQL: curso prático. São Paulo: Novatec, 2002.

Oliveira, M: PostgreSQL como NewSQL, DevCamp 2014. Diposnível em: < http://www.infoq.com/br/presentations/postgresql-com-newsql> . Acesso em 01 nov. 2014.

Oracle, MySQL Cluster Datasheet, Oracle Corporation, 2013. Disponível em <http://www.mysql.com/products/cluster/mysql-cluster-datasheet.pdf>. Acesado em 04 nov. 2014.

Payandeh, F. Hadoop vs. NoSql vs. Sql vs. NewSql By Example. Disponível em: <http://www.datasciencecentral.com/profiles/blogs/hadoop-vs-nosql-vs-sql-vs-newsql-by-example>. Acesso em 14 mar. 2014.

Porcelli, A. SQL, NoSQL ou NewSQL: Onde armazenar meus dados? - DevInVale 2011. Disponível em: <https://www.youtube.com/watch?v=gx1xT_GzH_g>. Acesso em 17 mar. 2014.

Ramakrishnan , R.; Gehrke, J. Sistemas de gerenciamento de banco de dados. São Paulo: McGraw-Hill, 2008.

Silberschatz, A. et al. Sistema de Banco de Dados. 5. ed. São Paulo: Elsevier, 2006.

Stonebraker, M. New SQL: An Alternative to NoSQL and Old SQL for New OLTP Apps. Disponível em: <http://cacm.acm.org/blogs/blog-cacm/109710-new-sql-an-alternative-to-nosql-and-old-sql-for-new-oltp-apps/fulltext>. Acesso em 14 mar. 2014.

Stonebraker, M. OldSQL vs. NoSQL vs. NewSQL on New OLTP. Disponível em: <https://www.youtube.com/watch?v=uhDM4fcI2aI>. Acesso em 24 mar. 2014.

Stonebraker, M. et. al., The End of an Architectural Era (It's Time for a Complete Rewrite), Proc. 2007 VLDB Conference, Vienna, Austria, 2007.

Stonebraker, M.; Weisberg, A., The VoltDB Main Memory DBMS. Data Engineering, p. 21, 2013.

Transaction Processing Performance Council , TPC Benchmark™ C, 2010. Disponivel em <http://www.tpc.org/tpcc/spec/tpcc_current.pdf>. Acessado em 15 de out. 2014.

Vaish, G. Getting Started with Nosql. Birmingham. Packt Publishing Ltd, 2013.

Vizard, M. In Defense of NewSQL. Disponível em: <http://www.itbusinessedge.com/cm/blogs/vizard/in-defense-of-new-sql/?cs=49255>. Acessado em 25 mar. 2014.

Zollmann, Johannes. NoSQL Databases. 2012.

APÊNDICE(S)

APENCICE A – Artigo

Estudo Comparativo Entre Bancos de Dados Relacionais, NoSQL e NewSQL

Orlando V. Werner[1], Gustavo Bisognin[1]

[1]Universidade do Extremo Sul Catarinense (UNESC) – Criciúma – SC – Brasil

orlando_vw@hotmail.com, gbisog@gmail.com

Abstract. *Databases are important tools for storing and managing information. Actually exists a lot of database models available on the market. This project described theoretical models of relational databases and NoSQL database and a brief presentation of the NewSQL database model, which propose a database with relational and NoSQL features. The objective of this project is form a knowledge base about the three databases models, mainly presenting the characteristics of NewSQL database model. The study compares the characteristics, according to the developers, of the three databases models and performs in practice a functional assessment through comparisons between relational databases and NewSQL using the TPC-C test and between NoSQL and NewSQL databases using the YCSB test, to assess the characteristics of each model.*

Key words*: Database, NoSQL, NewSQL.*

Resumo. *Bancos de dados são importantes ferramentas para armazenamento e gerenciamento de informação. Atualmente existem diversos modelos de bancos de dados disponíveis no mercado. Neste projeto estão descritos os modelos teóricos dos bancos de dados relacionais e NoSQL e uma breve apresentação dos bancos de dados NewSQL, que propõem um banco de dados com características de bancos relacionais e NoSQL ao mesmo tempo. O objetivo do projeto é formar uma base de conhecimento sobre os três modelos de bancos de dados, apresentando principalmente as características dos bancos NewSQL. O estudo compara as caraterísticas, segundo os desenvolvedores, de alguns bancos de dados dos três modelos e executa na pratica uma para avaliação funcional através de comparações entre os bancos relacionais e NewSQL utilizando o teste TPC-C e entre os bancos NoSQL e NewSQL utilizando o teste YCSB, para avaliar as características de cada modelo.*

Palavras-chave*: Banco de dados, NewSQL, NoSQL.*

1. Introdução

Por muitos anos as bases de dados do modelo relacional serviram muito bem para praticamente qualquer aplicação que necessitava de um banco de dados. Porém com a evolução das tecnologias na nuvem notou-se que o modelo tradicional de banco de dados já não estava suprindo a necessidade do mercado.

O mercado passou a ter necessidade de acessar enormes quantidades de dados e em uma velocidade muito maior que as que os bancos relacionais podiam oferecer. Para manter a integridade e a facilidade da linguagem de consulta estruturada, do inglês Structured Query Language (SQL), modelos relacionais exigem grande capacidade de processamento e de espaço em disco, se tornando uma tecnologia lenta para aplicações na nuvem (PAYANDEH, 2013).

Desta necessidade surgiram os bancos de dados conhecidos hoje por NoSQL. Bancos NoSQL são bases de dados não-relacionais que abriam mão das propriedades ACID e da interface SQL, para focarem em desempenho, alta escalabilidade e replicação de dados.
Estes bancos não foram desenvolvidos para substituir os bancos relacionais e sim para suprir as áreas onde o banco de dados relacional não é eficiente. Ao invés de trabalhar com dados relacionados, que requerem mais espaço e são processados por grandes servidores, bancos NoSQL trabalham de uma maneira onde todos os dados de uma informação estão agrupados em um registro e distribuídos de forma horizontal, com outros servidores, exigindo menos capacidade de processamento, desta forma ao invés dos grandes servidores, computadores e servidores de pequeno e médio porte conseguem processar os dados de maneira muito rápida.
Porém nos últimos anos os bancos de alta escalabilidade e desempenho começaram a encontrar alguns problemas. No inicio do crescimento da computação em nuvem, as principais aplicações a necessitarem de bancos neste modelo eram os sistemas de buscas, redes sociais, blogs, entre outros. Sistemas onde o acesso rápido é considerado muito mais importante que a total segurança dos dados. Mas nos tempos atuais o mercado ao mesmo em tempo que migra suas tecnologias para a nuvem, buscando velocidade e alta disponibilidade da informação, requer também segurança e facilidade de acesso a esta informação. Surge assim o conceito de banco de dados NewSQL, um modelo de banco de dados que busca utilizar o melhor do modelo relacional tradicional e do NoSQL (STONEBRAKER, 2014).

2. Banco de dados

Bancos de dados são sistemas para armazenamento de dados eletrônicos, que permitem interações com usuários que podem executar diversas ações nele, comumente chamadas de operações, como consultas, inserção de registros, edição, exclusão, entre outros. Ou seja, um banco de dados é um sistema que visa armazenar quaisquer tipos de dados eletrônicos que tenham relevância dentro de um contexto, para ser utilizado por um sistema de aplicação. (DATE, 2004).
Para realizar operações em um banco de dados é necessário utilizar um sistema próprio para a manipulação e gerenciamento do próprio banco conhecido como Sistema de Gerenciamento de Banco de Dados, ou SGBD. Alguns bancos de dados possuem a alternativa de armazená-los em arquivos e que seja escrito um programa para sua manipulação, porém os SGBD's oferecem uma gama de vantagens quando comparado a coleções de arquivos, como independência de dados, acesso eficiente, integridade, administração dos dados e recuperação de falhas.

3. Bancos de dados relacionais

Bancos de dados relacionais são bancos de dados desenvolvidos com base no modelo relacional onde utilizam uma coleção de tabelas, que são compostas por atributos e entidades, onde atributos são as colunas e as entidades as linhas da tabela, relacionadas entre si e geralmente utilizam linguagem SQL (SILBERSCHATZ, 2006).

Tabela 1. Exemplo de representação de uma tabela de banco de dados relacional

número_conta	nome_agência	saldo
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750

A-222	Redwood	700
A-305	Round Hill	350

Considere a tabela da tabela 1. Nela estão descritas 3 colunas, ou 3 atributos: número_conta, nome_agência e saldo. Para cada um destes atributos existe um conjunto de valores, que é denominado domínio, por exemplo, todos os nomes de agência compõem o domínio do atributo nome_agência. Cada linha da tabela é considerada uma relação e a tabela em si é considerada uma coleção de relações (SILBERSCHATZ, 2006).

3.1. ACID

Um dos principais fatores de um sistema de banco de dados relacional são as propriedades ACID que estão descritas no centro do banco de dados. ACID é um acrónimo para Atomicidade, Consistência, Isolamento e Durabilidade (SILBERSCHATZ, 2006).

Atomicidade é a propriedade que deve garantir que uma operação só será aplicada em definitivo no banco se todas as transações dessa operação obtiverem sucesso (HAEDER; REUTER, 1983).

Consistência é a propriedade que deve garantir que cada transação aplicada no banco de dados irá preservar a estrutura do banco de dados. Esta condição é essencial para o funcionamento da quarta propriedade, a Durabilidade (HAEDER; REUTER, 1983).

Isolamento é a propriedade que garante que quando há mais de uma transação sendo executadas ao mesmo tempo, as transações são invisíveis umas às outras, uma transação não pode interferir em outra (HAEDER, REUTER, 1983).

Durabilidade, depois que uma transação executada no banco de dados é aplicada com sucesso, esta propriedade garante que os dados do banco estarão armazenados no banco independente de mau funcionamento do banco, falhas de outras transações, entre outros (HAEDER; REUTER, 1983).

3.2. Structured Query Language

Structured Query Language (SQL), é uma coleção de comandos que servem para manipular um sistema de banco de dados. Os comandos servem tanto para manipular a estrutura do banco, alterando, criando e removendo tabelas e regras de relacionamento ou para manipulação de dados, onde é possível inserir atualizar e remover, além de consultar (OLIVEIRA, 2002).

Sql pode ser considerado uma "sublinguagem" de programação já que não é uma linguagem autônoma, isto é, não é possível escrever aplicações em sql. Devem-se escrever aplicações utilizando outras linguagens de programação, como c++, java, pascal, entre outras e inserir dentro das aplicações escritas comandos de sql para manipular os dados do banco de dados (OLIVEIRA, 2002).

4. Bancos de dados NoSQL

O termo "NoSQL" surgiu através de Carlos Strozzi ao utilizar o termo para nomear um SGBD que ele havia desenvolvido. Este evitava o uso de SQL como linguagem mesmo sendo um banco relacional. Só recentemente o termo passou a ser interpretado como "Not Only SQL", porém se relaciona a sistemas de bancos de dados não relacionais (ZOLLMANN, 2012).

A maioria dos provedores de bancos de dados NoSQL concordam que o termo 'NoSQL' não é claro, porém chama atenção. A maioria concorda que 'Not Only' não significa negar SQL, mas sim compensar as limitações técnicas da maioria dos bancos de dados relacionais (BURD, 2011).

Os bancos de dados relacionais surgiram e evoluíram em épocas diferentes e estavam sujeitos a limitações tecnológicas da época, e se tornaram uma ferramenta de qualidade para as aplicações típicas daquela época, porém atualmente o modelo de banco relacional apresenta

limitações. Desta forma o NoSQL que surgiu em outro ambiente, operando de maneira diferente do modelo relacional, proporciona soluções mais adequadas para problemas de armazenamento atuais (BURD, 2011).
Uma das vantagens dos bancos NoSQL é a capacidade de manusear dados não estruturados como arquivos de texto, e-mails, dados multimídias como áudios e vídeos e dados de redes sociais de forma eficiente (LEAVITT, 2010).
Segundo Leavitt (2010) existem três tipos populares de bancos de dados NoSQL: Bancos de dados chave-valor, Bancos de dados orientados a colunas e Bancos de dados orientados a documentos.
Sistemas de armazenamento baseados em chave-valor, de maneira básica, são conjuntos de vetores associados, que são formados por chaves e valores, onde cada chave deve ser única fornecendo uma informação não ambígua de valores (ZOLLMANN, 2012).
Ao contrário dos bancos relacionais, onde as tabelas são totalmente estruturadas em linhas e colunas com campos dimensionados de maneira uniforme para cada registro, bancos de dados orientados a colunas possuem uma coluna expansível de dados relacionados (LEAVITT, 2010).
O termo documento neste tipo de banco refere-se a algum tipo de documento de dados estruturados como JSON, BSON ou XML. Enquanto o formato de armazenamento é fixo pelo tipo de documento, a estrutura é flexível, sendo possível armazenar diversos dados diferentes em um mesmo documento, contanto que eles sejam do mesmo tipo (ZOLLMANN, 2012).
Uma das principais diferenças entre o modelo relacional e o NoSQL é a maneira como consistência é tratada. Sistemas NoSQL precisam abrir mão das propriedades ACID para permitir escalabilidade horizontal. A necessidade de abrir mão das propriedades ACID para permitir a escalabilidade, ocorre por conta do Teorema CAP, que foi proposto por Eric Brewer no ano 2000, que se baseia em três propriedades para sistemas distribuídos (ZOLLMANN, 2012).

5. Bancos de dados NewSQL

Atualmente estão surgindo sistemas de gerenciamento de banco de dados que tendem manter a interface SQL e oferecer desempenho e escalabilidade, preservando também noção tradicional de ACID para transações. Para distinguir estas soluções das soluções já conhecidas está classe de bancos de dados está sendo chamada de NewSQL. Estes sistemas devem ser capazes de prover alto rendimento, igual a bancos NoSQL, porém sem a necessidade garantir a consistência através de programação em nível de aplicativo, ou seja o SGBD deve garantir a consistência dos dados, além de oferecer suporte a SQL (STONEBRAKER).
Em 2011 a empresa de análise de mercado 451 Group publicou um relatório onde o analista Matt Aslett mencionou pela primeira vez o termo “NewSQL”, que segundo Aslett (2011) são bancos de dados que prometem garantir escalabilidade e flexibilidade assim como os bancos NoSQL ao mesmo tempo em que dão suporte a consultas SQL e/ou garantia à ACID.
Assim como o termo NoSQL, o termo NewSQL também não deve ser levado ao pé da letra, o termo refere-se ao modelo de banco não ao SQL especificamente (ASLETT, 2011).

6. Analise Comparativa Entre os Modelos Relacionais, NewSQL e NoSQL de Banco de Dados

Neste estudo foram realizadas duas análises. Uma através de uma matriz de comparação de características dos bancos de dados outra através dos *benchmarks* TPC-C e YSCB. Para o desenvolvimento das análises foram considerados dois bancos de dados de cada modelo. Todos os bancos selecionados são de código aberto ou possuem algum tipo de licença livre e foram selecionados com base no ranking do site DB-Engines. Os bancos selecionados são:

MySQL e PostgreSQL, representando o RDBMS, MongoDB e Cassandra como NoSQL e para NewSQL foram utilizados o MySQL Cluster e o NuoDB**.**

Na tabela 2 estão apresentados os resultados das comparações direta entre as características de cada modelo de banco de dados. É possível analisar características que definem cada modelo, por exemplo: os bancos NoSQL possuem métodos de particionamento e de replicação, necessários para a elasticidade dos bancos, enquanto os bancos relacionais não possuem estas características. Já os SGBD NoSQL não possuem transações ACID e uma linguagem de consulta padrão como o SQL, que são necessárias para garantir segurança dos dados e agilidade de desenvolvimento de consultas complexas.

Tabela 2. Tabela de comparação entre modelos de bancos de dados

	RDBMS		NewSQL		NoSQL	
	MySQL	PostgreSQL	NuoDB	MySQL Cluster	MongoDB	Cassandra
Conceitos de transações (ACID)	ACID	ACID	ACID	ACID	Atomicidade em operações em apenas um documento	Atomicidade e Isolamento são suportados em algumas operações
Linguagem de consulta, definição e manipulação	SQL	SQL	SQL	SQL	Própria	CQL
Método de partição	Não possui	Não possui	Dados dinamicamente alocados nos nós de leitura e escrita	Particionamento horizontal (Sharding)	Particionamento horizontal (Sharding)	Particionamento horizontal (Sharding)
Método de replicação	Mestre-escravo	Mestre-escravo	Gerenciado de forma transparente e pelo SGBD	Mestre-escravo	Mestre-escravo	De acordo com a estratégia configurada e do fator de replicação
Índices	Possui	Possui	Possui	Possui	Possui	Restrito
Schema	Possui	Possui	Possui	Possui	Livre	Livre
Store Procedures/Triggers	Possui	Possui	Possui	Possui	Não possui	Possui
Chaves	Possui	Possui	Possui	Possui	Não possui	Não possui
Método de armazenamento	Relacional	Relacional	Relacional	Relacional	Documento	Orientado a colunas
Disponibilidade	Baixa	Baixa	Alta	Alta	Alta	Alta
Tolerância a Partição	Fraca	Fraca	Forte	Forte	Forte	Forte
Atomicidade	Possui	Possui	Possui	Possui	Operações em apenas um documento	Em operações singulares
Consistência	Possui	Possui	Possui	Possui	Consistência eventual ou imediata	Consistência eventual ou imediata
Isolamento	Possui	Possui	Possui	Possui	Manual	Em operações singulares
CAP	CA	CA	CAP	CAP	AP	AP

Durabilidade	Possui	Possui	Opcional	Possui	Opcional	Possui
Tipagem	Forte	Forte	Forte	Forte	Fraca	Fraca
Controle de concorrência	Possui	Possui	Possui	Possui	Possui	Possui

Através da tabela 2 é notável também a proposta dos bancos NewSQL em atenderem características que os dos dois modelos tradicionais possuem. Bancos NewSQL possuem métodos de particionamento e replicação e também possuem suporte a SQL e garantias das propriedades ACID.

6.1. TPC-C e YCSB

O *benchmark* TPC-C foi desenvolvido pela *Transaction Processing Performance Council* (TPC), uma instituição de padronização e de desenvolvimento de *benchmarks* para aplicações OLTP (TPC, 2001). A primeira versão do TPC-C foi liberada em 1992 e hoje está na versão 5.11que foi liberada em 2010.

O Yahoo! Cloud Serving Benchmark, ou YCSB, é considerado uma coletânea de micro benchmarks que representam aplicações de manipulação e gerenciamento de dados considerados simples, mas que requerem alta escalabilidade, como aplicações desenvolvidas para Web. O benchmark é uma representação de uma aplicação do tipo chave-valor simples (DIFALLAH, et al, 2013).

Na próxima sessão serão relatados os resultados obtidos a partir destes dois testes.

7. Resultados obtidos

O objetivo principal desta análise foi conhecer um pouco sobre cada um dos modelos de bancos de dados na teoria e na prática, avaliar suas características, os prós e os contras de cada modelo e identificar situações onde cada banco de dados se enquadra melhor, de maneira imparcial, sem a pretensão de determinar qual dos modelos é melhor ou pior.

Os testes foram separados por modelos de bancos de acordo com o propósito do *benchmark*. Os bancos relacionais foram testados apenas com o TPC-C e os NoSQL com o YCSB enquanto os bancos NewSQL foram avaliados pelas duas ferramentas.

7.1 TPC-C

Nos resultados com o TPC-C, o banco NewSQL MySQL Cluster teve um rendimento bastante superior aos demais bancos averiguados com apenas um nó, atingindo aproximadamente 100 requisições por segundo com 1 conexão e 360 requisições com 100 conexões enquanto o segundo melhor resultado do teste foi com o banco relacional PostgreSQL que obteve um rendimento máximo de aproximadamente 70 requisições por segundo utilizando 50 conexões simultâneas. Porém ao adicionar um novo nó ao MySQL Cluster o rendimento ficou semelhante aos demais bancos, com rendimento máximo de 20 requisições por segundo com 100 conexões. Os resultados do banco de dados NuoDB tiveram valores semelhantes em teste com 1 é com 2 nós, apresentando variação apenas na latência, ambos alcançando 20 requisições por segundo com 100 conexões enquanto o MySQL utilizando o motor InnoDB teve rendimento médio de 10 requisições, independente do numero de conexões simultâneas.

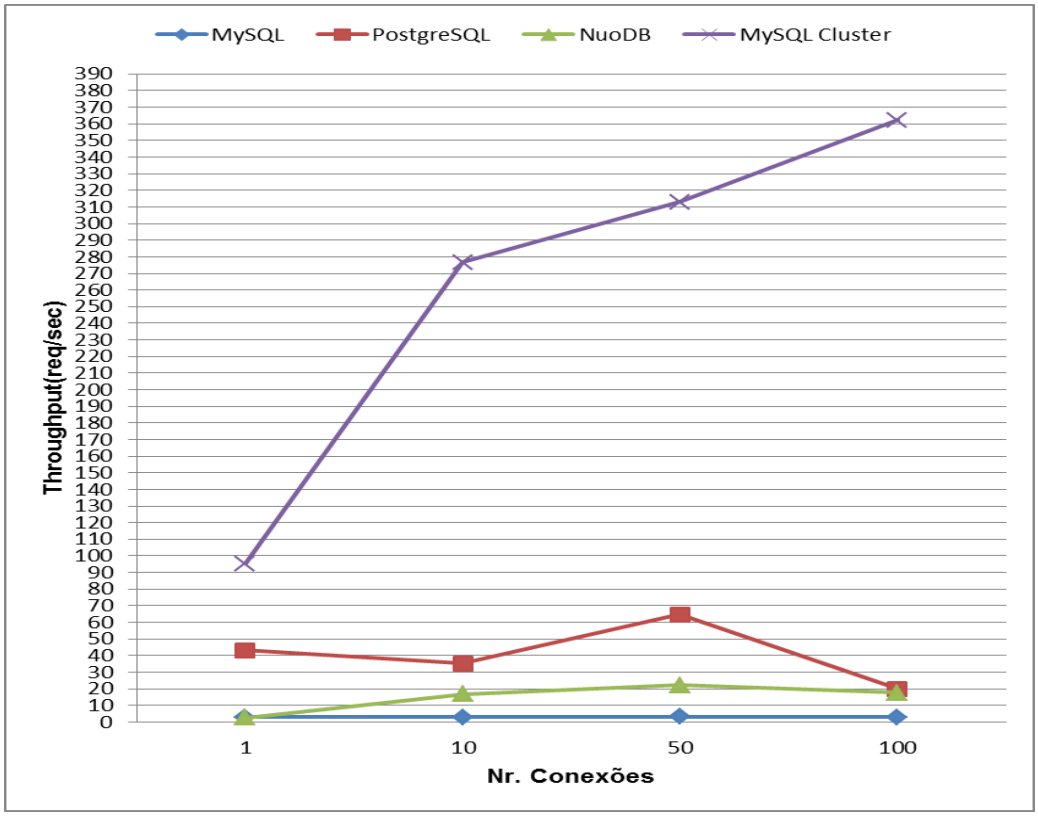


Figura 1. Rendimento dos bancos relacionais e newsql com 1 nó.

7.2 YCSB

O microteste utilizado no YCSB foi o *workload b*, um microteste focado em leituras de registro com uma pequa carga de escrita no banco de dados, os resultados foram agrupados por bancos de dados.

Nos testes os bancos NewSQL não apresentaram ganhos ao ter segundo nó conectado ao cluster, mantando o mesmo rendimento ou tendo o rendimento prejudicado, enquanto os bancos NoSQL apresentaram ganhos notaveis ao ter dois computadores operando juntos o mesmo banco de dados.

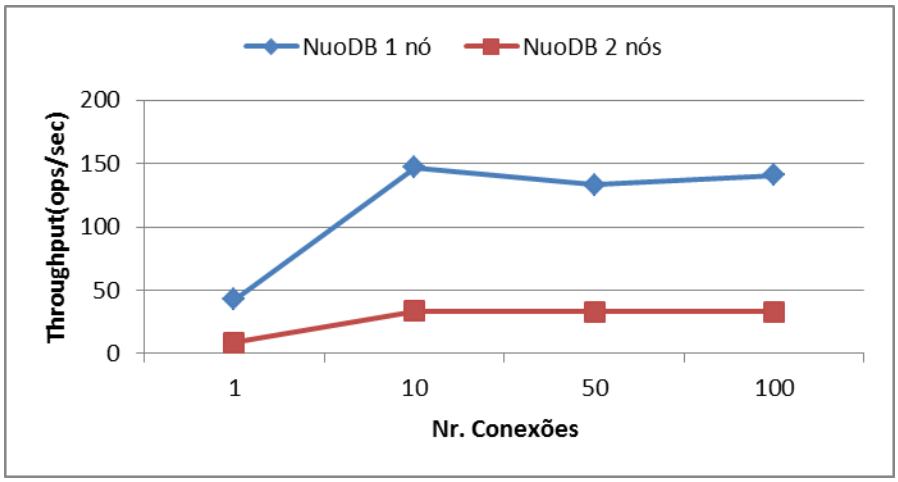


Figura 2. Rendimento do banco NuoDB no teste YCSB.

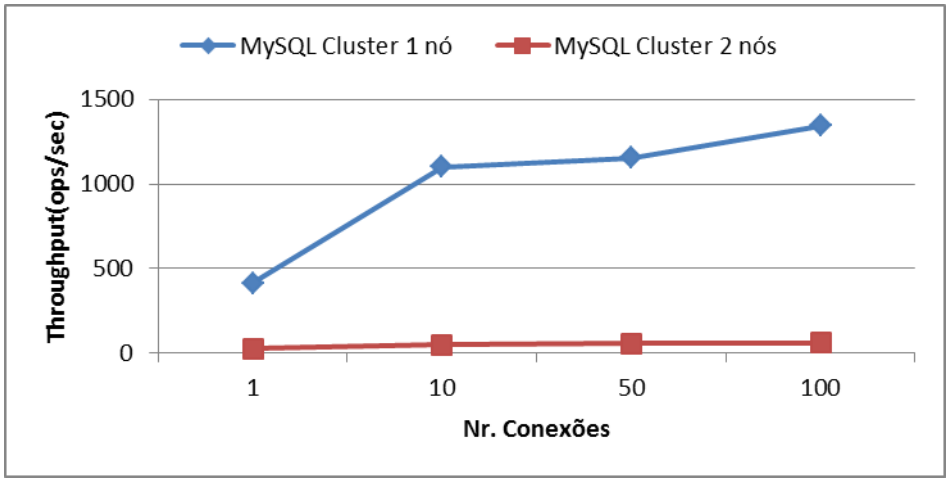


Figura 3. Rendimento do banco MySQlCluster no teste YCSB.

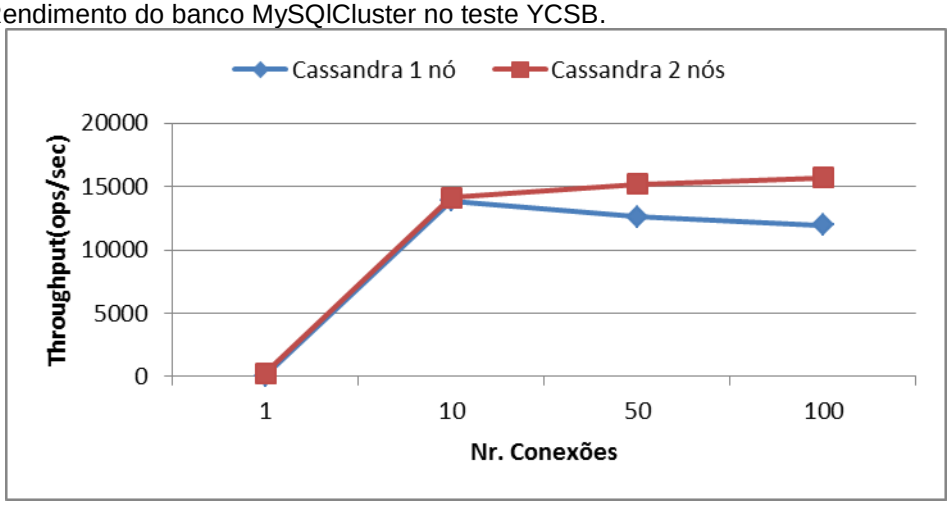


Figura 4. Rendimento do banco Cassandra no teste YCSB.

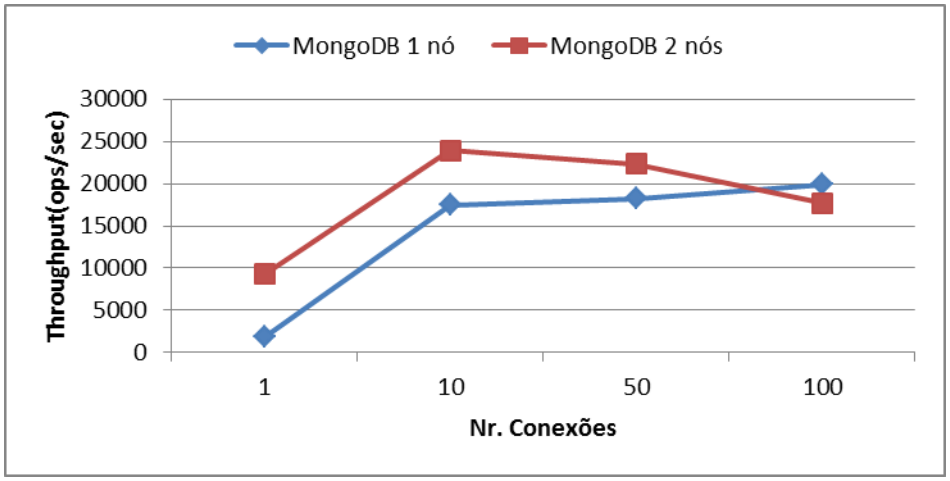


Figura 5. Rendimento do banco MongoDB no teste YCSB.

References

Aslett, M. How will the database incumbents respond to NoSQL and NewSQL?. The 451 Group, 2011.

Burd, G. What Is NoSQL? Why NoSQL?. Usenix, 2011.

Date, C. J. Introdução a Sistemas de Banco de Dados. Rio de Janeiro: Elsevier Editora, 2004.

Haerder, T.; Reuter, A. - Principles of transaction-oriented database recovery. ACM Computing Surveys, 1983.

Leavitt, N. Will. NoSQL Databases Live Up to Their Promise? 2010. Computer, 12. ISSN: 0018-9162, pg. 12 – 14.

Oliveira, C. H. P. SQL: curso prático. São Paulo: Novatec, 2002.

Payandeh, F. Hadoop vs. NoSql vs. Sql vs. NewSql By Example. Disponível em: <http://www.datasciencecentral.com/profiles/blogs/hadoop-vs-nosql-vs-sql-vs-newsql-by-example>. Acesso em 14 mar. 2014.

Silberschatz, A. et al. Sistema de Banco de Dados. 5. ed. São Paulo: Elsevier, 2006.

Stonebraker, M. New SQL: An Alternative to NoSQL and Old SQL for New OLTP Apps. Disponível em: <http://cacm.acm.org/blogs/blog-cacm/109710-new-sql-an-alternative-to-nosql-and-old-sql-for-new-oltp-apps/fulltext>. Acesso em 14 mar. 2014.

Zollmann, Johannes. NoSQL Databases. 2012.

ANEXO(S)

ANEXO A – Exemplo de arquivo de configuração do TPC-C do OLTP-Bench

```
<?xml version="1.0"?>
<parameters>
    <!-- Connection details -->
    <dbtype>mysql</dbtype>
    <driver>com.mysql.jdbc.Driver</driver>
    <DBUrl>jdbc:mysql://localhost:3306/tpcc</DBUrl>
    <username>root</username>
    <password></password>
    <isolation>TRANSACTION_SERIALIZABLE</isolation>
    <!-- Scale factor is the number of warehouses in TPCC -->
    <scalefactor>2</scalefactor>
    <!-- The workload -->
    <terminals>2</terminals>
    <works>
        <work>
          <time>60</time>
          <rate>10000</rate>
          <weights>45,43,4,4,4</weights>
        </work>
          </works>
          <!-- TPCC specific -->
          <transactiontypes>
          <transactiontype>
                    <name>NewOrder</name>
          </transactiontype>
          <transactiontype>
                    <name>Payment</name>
          </transactiontype>
          <transactiontype>
                    <name>OrderStatus</name>
          </transactiontype>
          <transactiontype>
```

```
            <name>Delivery</name>
        </transactiontype>
        <transactiontype>
            <name>StockLevel</name>
        </transactiontype>
        </transactiontypes>
</parameters>
```

ANEXO B – Comandos de execução do benchmark TPC-C do OLTP-Bench

Carregar dados:

```
oltpbenchmark -b tpcc -c config/tpcc_config_mysql.xml --create=true --load=true
```

Executar teste:

```
oltpbenchmark -b tpcc -c config/tpcc_config_mysql.xml --execute=true -s 1 -o MySQL/MySQL_1T
```

ANEXO C – Workload ‘b’ do YCSB

```
# Copyright (c) 2010 Yahoo! Inc. All rights reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License"); you
# may not use this file except in compliance with the License. You
# may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
# implied. See the License for the specific language governing
# permissions and limitations under the License. See accompanying
# LICENSE file.
# Yahoo! Cloud System Benchmark
# Workload B: Read mostly workload
#   Application example: photo tagging; add a tag is an update, but most operations
are to read tags
#
#   Read/update ratio: 95/5
#   Default data size: 1 KB records (10 fields, 100 bytes each, plus key)
#   Request distribution: zipfian
recordcount=100000
operationcount=100000
workload=com.yahoo.ycsb.workloads.CoreWorkload
readallfields=true
readproportion=0.95
updateproportion=0.05
scanproportion=0
insertproportion=0
requestdistribution=zipfian
```

ANEXO D – Comandos de execução do YCSB

Comando de carga:

python bin\ycsb load cassandra-10 -P workloads\workloadb -s > CassandraLoad.log

Comando de execução:

python bin\ycsb run cassandra-10 -P workloads\workloadb -s -t -p threadcount=1 -p maxexecutiontime=60 -p hosts=10.0.0.110 > CassandraRun.log