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

Investigando o uso de bancos de dados não convencionais para gerenciar informações da administração pública

Diogo Araújo Pacheco Wanzeller

Monografia apresentada como requisito parcial
para conclusão do Curso de Computação — Licenciatura

Orientador
Prof. Dr. Rodrigo Bonifacio de Almeida

Brasília
2013

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Curso de Computação — Licenciatura

Coordenador: Prof. Dr. Flávio de Barros Vidal

Banca examinadora composta por:

Prof. Dr. Rodrigo Bonifacio de Almeida (Orientador) — CIC/UnB
Prof. Dr. Professor I — CIC/UnB
Prof. Dr. Professor II — CIC/UnB

CIP — Catalogação Internacional na Publicação

Wanzeller, Diogo Araújo Pacheco.

Investigando o uso de bancos de dados não convencionais para gerenciar informações da administração pública / Diogo Araújo Pacheco Wanzeller. Brasília : UnB, 2013.
57 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de Brasília, Brasília, 2013.

1. big-data, 2. NoSql, 3. Banco de Dados

CDU 004.4

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

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

Investigando o uso de bancos de dados não convencionais para gerenciar informações da administração pública

Diogo Araújo Pacheco Wanzeller

Monografia apresentada como requisito parcial
para conclusão do Curso de Computação — Licenciatura

Prof. Dr. Rodrigo Bonifacio de Almeida (Orientador)
CIC/UnB

Prof. Dr. Professor I
CIC/UnB

Prof. Dr. Professor II
CIC/UnB

Prof. Dr. Flávio de Barros Vidal
Coordenador do Curso de Computação — Licenciatura

Brasília, 28 de janeiro de 2013

Dedicatória

Dedico a....

Agradecimentos

Agradeço a....

Abstract

A ciência...

Palavras-chave: big-data, NoSql, Banco de Dados

Abstract

The science...

Keywords: big-data, NoSql, Data Bases

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A sociedade está lidando com uma quantidade de dados cada vez maior. Hoje, por menor que seja o dado ele se torna importante pelas informações que podem ser extraídas a partir dele. Se pararmos para pensar estamos envolvidos por uma quantidade de dados enorme. Como a necessidade de extrair informação é comum em um mundo globalizado e informatizado, os cientistas e engenheiros se veem obrigados a desenvolvem novas maneiras de medir eventos. Sensores, câmeras de trânsito, dados da web, genes, dados geográficos, dados de compras, dados de pesquisas, e muitas outras informações se tornaram o diferencial nesse mundo competitivo e dinâmico, e precisam de tratamento e atenção [12]. Conforme Borkar et al exemplificou, hoje empresas estão monitorando compras de clientes, pesquisas de produtos, sites de relacionamento e diversas outras fontes para aumentar a eficácia do seu marketing e dos serviços ofertados aos clientes, que cada vez mais precisam ser diferenciados e inovadores; governos e empresas estão rastreando conteúdos de blogs e tweets para realizar análises de sentimentos e organizações públicas de saúde estão monitorando artigos de notícias, tweets, e tendências de pesquisas na web para acompanhar o progresso de epidemias [10].Essa enorme quantidade de dados já gerou, e ainda gera, muitos desafios para cientistas e estudiosos. Não é de hoje que o mundo da TI vem enfrentando grandes problemas com essa grande e heterogênea massa de dados.

Ao decorrer da evolução dos computadores e com a informatização do mundo, podemos perceber que, ao longo do tempo, a definição de "grande" foi mudando significativamente. Poucos anos atrás falar em terabytes era coisa de outro mundo, e atualmente a grande maioria das pessoas já possuem dispositivos de armazenamento com capacidade superior a um terabyte. Hoje, para as grandes empresas, já é normal gravar dados na ordem de petabytes . Tabela 3.1

Como exemplo podemos citar a administração pública, que possui um grande volume de documentos que precisam ser armazenados com qualidade e cuidado, pois fazem parte dos chamados arquivos permanentes. Esses arquivos ocupam cada vez mais espaço e, devido a sua característica, precisam ser mantidos em locais próprios e com características bem definidas, já que precisam durar por um longo período de tempo. Inúmeras vezes os órgãos precisam consultar esses arquivos e essas consultas, além de contribuírem para a deteriorização dos documentos, são muitas vezes lentas e dificeis de se realizar [6].

A dificuldade de manter esses papéis fez com que o governo federal incentivasse a organização pública a iniciar um processo de digitalização dos documentos para que uma

cópia digital desse arquivos fosse mantida pelos órgãos. A digitalização dos arquivos não só possibilita a preservação dos documentos, pois restringe o manuseio dos originais, quanto também facilita o acesso, já que passa a permitir acessos tanto locais quanto remotos e também acessos simultâneos. O processo de digitalização é complexo, demorado e, além de um controle de work flow bem definido, necessita de grandes investimentos de software e hardware para que o resultado tenha uma boa qualidade [6].

Como podemos ver, as finalidades da extrasão desses dados são diversas e podemos retirar informações preciosas dos dados que nos cercam, mas isso nem sempre é trivial e muitas vezes envolve muita tecnologia e estudo. Hoje, além de lidarmos com uma grande quantidade de dados, as vezes nos deparamos com dados que possuem grande variedade e fluxo. Essa massa de dados com características singulares é chamada de big data.

Podemos dividir as tecnologias que sustentam big data em duas: as envolvidas com análise dos dados, como Hadoop e MapReduce e tecnologias de armazenamento e processamento dos dados [23]. Na parte de armazenamento podemos citar os bancos de dados NoSql (Not only SQL), que surgiram a partir da necessidade de inovar no que diz respeito ao armazenamento de big data e distribuição de dados.

Dado esse cenário, os SGBDs relacionais não foram capazes de resolver todos os problemas e com o surgimento desse novo paradigma de modelagem, os sitemas de banco de dados NoSql estão sendo cada vez mais utilizados. Como essa tecnologia não-relacional é relativamente recente, existem poucos estudos comparativos que mostre em qual cenário se aplica o uso de uma tecnologia NoSqll e até que ponto ela é melhor que um banco de dados convencional.

Esse trabalho surgiu de uma dúvida arquitetural para a implementação de um sistema que armazenasse todos os dados de servidores públicos federais. Como dito anteriormente, o governo tem orientado os órgãos a digitalizarem os seus arquivos e essa orientação fez com que os órgão se juntassem, devido ao elevado custo para implementação, para darem início a essa conversão dos arquivos físicos em digitais.

O conceito de documentos descentralizados em pastas funcionais será substituído por repositórios de dados e informações de origem primária, auditáveis e não replicados. Com isso teve origem o Projeto de Assentamento Funcional Digital – AFD, que objetiva a criação de um Dossiê, em mídia digital, que será tratado como Fonte Primária de Informação de dados cadastrais do Servidor Público Civil Federal e que substituirá a tradicional Pasta Funcional ou Assentamento Funcional. No site do SIGEPE (Sistema de Gestão de Pessoas) [7] alguns pontos de melhoria com a criação do AFD como: "A criação do Assentamento Funcional Digital (AFD) possibilitará a diminuição drástica do volume de papeis armazenados e tramitados. O AFD constituirá de um banco referencial, de dados e imagens das pastas funcionais, com indexadores para localização dos documentos de maneira online [1]."

Para que a base de dados possa cumprir com o seu proprósito, ela precisa ter um alto nível de performace, e também, grande escalabilidade. Sendo assim, esse trabalho tem como objetivo analisar a diferença de performace entre bancos de dados NoSql e um banco de Dados Relacional para a gerência dos dados dos servidores públicos.

Para cumprir esse objetivo, desenvolveremos um webservice que possua métodos simples de inserção, consulta, exclusão e atualização de dados. A camada de persistência será implementada em diferentes SGBDs. Testaremos o desempenho da nossa aplicação

usando um banco de dados relacional e compararemos o resultado com o de quatro bancos de dados NoSQL com diferentes tipos de modelagem.

Capítulo 2

NoSql

Nesse capítulo iniciaremos com a definição de o que é o termo NoSql e, após vermos como essa nova forma de se pensar em banco de dados surgiu, conheceremos os principais tipos de banco de dados não relacionais e como eles armazenam os seus dados.

2.1 Definição

O termo NoSql é a junção de duas palavras. No and SQL. Ao pé da letra significa que é uma tecnologia/produto que trabalha de forma contrária à tecnologia dos banco relacionais (adeptos do SQL). O termo é usado com o sentido de Não Relacional. Atualmente o termo NoSql é traduzido para "Not only Sql", ou seja "Não só Sql". Saindo da definição, NoSql é um termo genérico para uma classe definida de banco de dados não-relacionais que armazenam os dados de forma diferente da conhecida modelagem relacional e que surgiram com o propósito de sanar algumas dificuldades encontradas com o modelo relacional. NoSql não é um produto, mas a uma classe de produtos e conceitos de armazenagem e manipulação de dados.

O que diferencia os bancos de dados NoSql dos relacionais são os seus modelos de dados sem um schema definido. Os bancos de dados NoSql podem ser classificados, segundo seu modelo de dados, em quatro grupos: chave-valor, orientados a documentos, orientados a colunas e baseados em grafos [11, 16].

2.2 História

Bancos de dados que usam a modelagem não relacionais não são novidades. Conforme discutido no livro (NoSql Professional) eles surgiram junto com as primeiras máquinas de computar. Bases não relacionais ficaram conhecidas e cresceram por causa do uso de mainframes e em dominios específicos como o armazenamento de credenciais para autenticação. Esse NoSql que conhecemos hoje é uma nova visão, ou como diz fulano em (NoSql Professional), uma reincarnação que nasceu no mundo de aplicações web que necessitam de recursos escaláveis para tratar de sua enorme massa de dados. Apesar de o paradigma NoSql já ter sido criado há algum tempo nenhum ele só tomou as proporções atuais depois que grandes empresas como Google, Amazon e Facebook começam a usar em suas arquiteturas [16].

Ao utilizar SGBD's relacionais com grandes quantidade de dados surgem problemas como falta de eficiência no processamento, uma paralelização não efetiva, alto custo e scalabilitade limitada. Sendo um gigante da internet, o Google, se não for a empresa que manipula a maior quantidade de dados, é com certeza uma das maiores e ao se deparar com essa problemática construiu a sua própria infraestrutura para que o seu mecanismo de busca e outras aplicações pudessem tratar a massa de dados de forma eficiente.

Com o lançamento de artigos pelo Google que explicavam em partes como o problema foi solucionado, desenvolvedores de software livre criaram o primeiro motor de busca de código aberto que replicava algumas característica da infraestrutura do Google, o Lucene. Logo depois, os principais desenvolvedores do Lucene se juntaram ao Yahoo e com a ajuda de diversos outros desenvolvedores criaram uma estrutura que imitada todas as peças da infraestrutura de computação distribuida do Google. Essa solução livre é o Hadoop. Nessa mesma época surgiu a idéia do NoSql.

O sucesso do Google e o Hadoop ajudaram a impulsionar novos conceitos de computação distribuída, NoSql e o próprio projeto Hadoop. Um ano após o lançamento dos artigos do Google outra gigante da internet resolveu compartilhar o seu caso de sucesso. Em 2007 a Amazon mostrou ao mundo sua solução de base de dados distribuída, disponível e consistente que se chama Dynamo.

Após Google e Amazon mostrarem para o mundo que o NoSql dava certo começaram a surgim diversos outros produtos nessa linha. O NoSql e os conceitos de manipulação de Big Data ganharam espaço e forma surgindo diversos casos de uso de sucesso de grandes companias como o Facebook, Netflix, Yahoo, EBay, Hulu, IBM e diversas outras.

2.3 Os Principais Tipos de Banco de Dados NoSql

2.3.1 Chave-Valor

Bancos de dados NoSql que usam a modelagem Chave-Valor armazenam os dados indexados por um valor chave. A base é similar a um dicionário, onde os dados são endereçados por uma única chave. Uma vez que os dados são armazenados, é através das suas chaves a única forma de recuperá-los. Os valores sao isolados e independentes um dos outros, sendo necessário tratar isso na aplicação. Por isso banco chave-valor são livres de schema. Isso permite que novos tipos de dados sejam inseridos em tempo de execução sem que o banco entre em conflito e sem influenciar na disponibilidade do sistema [16, 18].

Alguns exemplos de banco de dados que usam esse tipo de modelagem são: RIAK, LevelDB, Voldemort, redis [5].

2.3.2 Orientados a Documentos

A modelagem orientada a documentos armazena os dados encapsulados em pares de chave-valor em JSON ou em outro padão semelhante. Dentro dos documentos as chaves devem ser únicas. Cada documento recebe um identificador que também é único dentro de uma coleção de documentos. Os documentos são as unidades básicas e não têm uma estrutuda definida como nas tabelas do modelo relacional, ou seja, não tem um schema de dados definido. Ao armazenar os dados em JSON há uma vantagem adicional que é o

suporte a tipos de dados, o que torna a forma de armazenamento mais amigavel para os desenvolvedores [11, 16].

Os exemplos mais significativos são: CouchDB, MongoDB e Riak [16].

Listing 2.1: Exemplo de arquivo do CouchDB

```
{
    "Subject": "I like Plankton",
    "Author": "Rusty",
    "PostedDate": "5/23/2006",
    "Tags": ["plankton", "baseball", "decisions"],
    "Body": "I decided today that I don't like baseball. I like plankton."
}
```

2.3.3 Orientados a Colunas

Nesse tipo de modelagem o paradigma passa a ser de orientação a atributos(colunas).Ao contrário da modelagem chave-valor, agora os dados são armazenados usando tabelas sem um schema definido, mas sem suporte a associação entre elas . Figura 2.1 Segundo Jing Han et all, um banco orientado a colunas tem as seguintes caracteristicas [15]:

1. Os dados são armazenados em colunas
2. Cada coluna de dado é um índice do banco
3. Acessar somente colunas faz com que haja redução de I/O nos resultados das consultas
4. Consultas simultâneas, isto é, cada coluna é tratada por um processo
5. Possuem o mesmo tipo de dados, características semelhantes e boa taxa de compressão

Em geral esse tipo de banco é mais vantajoso para aplicações de agregação e data warehouses. Alguns exemplos são: Cassandra e Hypertable [5].

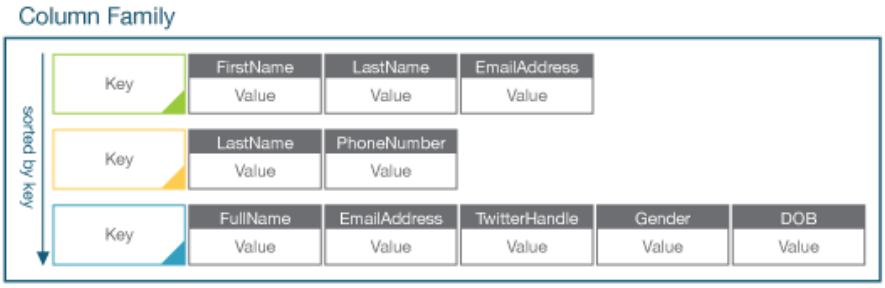

Figura 2.1: Modelagem orientada a colunas

Tabela 2.1: BSON - Tipos Suportados

Tipo	Número
Double	1
String	2
Object	3
Array	4
Binary Data	5
Object id	7
Boolean	8
Date	9
Null	10
Regular Expression	11
JavaScript	13
Symbol	14
JavaScript (with scope)	15
32-bit integer	16
Timestamp	17
64-bit integer	18
Min key	255
Max key	127

2.3.4 Baseados em Grafos

Nessa categoria os dados são armazenados em nos de um grafo cujas arestas representam o tipo de associação entre esses nós. Esse tipo de banco é especializado em manter dados fortemente ligados. O twitter armazenar as relações entre os seus usuários no seu próprio banco de dados baseados em grafos, o FlockDB, que é otimizado para listas de relações muito grandes, leituras e escritas [16]. Alguns exemplos são: Neo4J, infoGrid e FlockDB [5].

2.4 MongoDB

Essa seção foi baseada no site oficial do MongoDB [4] exceto quando explicitamente citado.

MongoDB é um banco de dados NoSQL, de código aberto, orientado a documentos, schema-free e escrito em C++. Os dados são persistidos em coleções de dados que são representados usando o BSON, um formato binário similar ao JSON (Figura 2.2). O MongoDB tem suporte a todos os tipos de dados JSON como string, inteiro, boleano, double, array e objeto. Por usar codificação BSON o MongoDB suporta alguns tipo de dados adicionais como data, binary data, regular expression e code [24]. Na Figura 3.1 podemos ver os tipos suportados pelo BSON.

Como não usa o mesmo formato de armazenamento dos SGBDS relacionais, o MongoDB armazena os seus dados em coleções, que são equivalentes às tabelas. Uma Coleção

pode ter um ou mais documentos; são equivalentes as linhas em uma tabela de um banco de dados relacional. Cada documento tem um ou mais campos, o que corresponde a uma coluna.

Diferente do que a maioria das pessoas estão acostumadas, o MongoDB não trabalha com uma estrutura de dados bem definida (schema), ou melhor dizendo, ele usa schemas dinâmicos. Com ele é possível criar coleções sem que a estrutura, campos ou tipos de valores dos documentos estejam definidos. Essa forma flexível de armazenar os dados nos permite trabalhar com estruturas e dados bastante heterogêneos.

```
var mydoc = {
               _id: ObjectId("5099803df3f4948bd2f98391"),
               name: { first: "Alan", last: "Turing" },
               birth: new Date('Jun 23, 1912'),
               death: new Date('Jun 07, 1954'),
               contribs: [ "Turing machine", "Turing test", "Turingery" ],
               views : NumberLong(1250000)
             }
```

Figura 2.2: Documento BSON usado no MongoDB

Quanto mais controle, mais custosa é uma operação para o sistema gerenciador de banco de dados. Os banco de dados NoSQL, como dito anteriormente, foram criados para suprir algumas características que os banco de dados relacionais não atendiam. Uma dessas características é a velocidade com que operações de consulta, escrita, atualização e exclusão são executadas. Para que a velocidade dessas transações fosse aumentada foi preciso retirar alguns controles, e com isso os banco de dados NoSQL não se comprometem com todas as características ACID.

O MongoDB não provê transações ACID, mas possui alguns recursos transacionais básicos. Operações atômicas são possíveis no escopo de um único documento. Na tabela abaixo temos alguns exemplos de operações em SQL e suas correspondentes no MongoDB.

2.5 Cassandra

Essa seção foi baseada no livro "Cassandra: The Definitive Guide" [17] exceto quando explicitamente citado.

Cassandra é um SGBD NoSQL, de código aberto, do tipo chave-valor e que foi inicialmente desenvolvido pelo Facebook. Com a recente popularização, várias empresas estão achando casos em que o Cassandra pode ser utilizado. Atualmente o Cassandra é usado por por grandes empresas como o Netflix, eBay, Twitter e Cisco [2]. A maior instalação conhecida, com mais de 150 TB e mais de cem máquinas, é a do Facebook.

Cassandra se tornou open-source em julho de 2008 quando o Facebook o mostrou para o mundo e após ser encubado pela apache se tornou bastante popular graças as suas excelentes características técnicas. Ele executa escritas muito rápidas, armazena terabytes de dados e possui uma arquitetura simétrica e descentralizada.

Conforme definido no livro [definitive guide] Apache Cassandra é banco de dados orientado a colunas, open-source, distribuído, de consistência ajustável, descentralizado e elasticamente escalável que é baseado no Amazon Dynamo e no Google Bigtable.

Cassandra representa sua estrutura de dados em tabelas multidimensionais e esparsas (Figura 2.3). No Cassandra podemos ter linhas com uma ou mais colunas, diferentemente de banco de dados relacionais, porem cada linha deve possuir uma chave que a torna acessível. Essa flexibilização do schema nos permite decidir a estrutura de armazenamento da aplicação dinamicamente. Não é preciso saber todas as características e campos do modelo de dados antes do início do projeto.

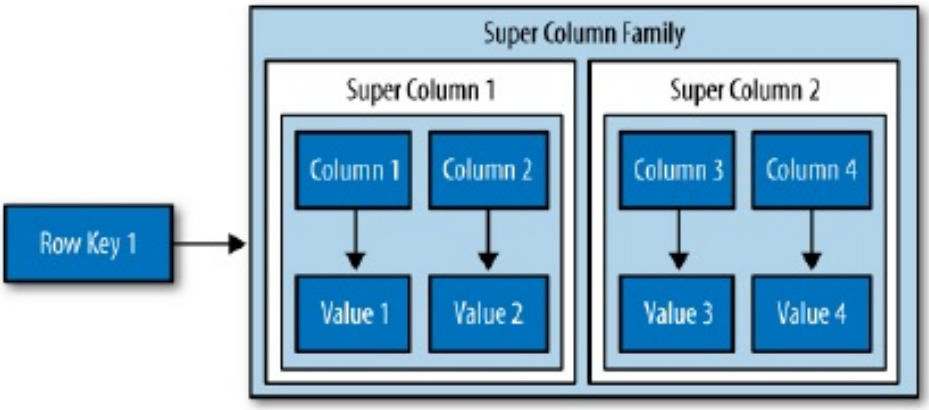

Figura 2.3: Modelo de dados do Cassandra

Apesar de ser necessário definir um agrupador chamado keyspace, que contem os aglomerados de colunas, as 'tabelas' não precisam de uma definição de quais as colunas vão armazenar. Isso torna o Cassandra schema-free e permite que criemos colunas a qualquer hora. O keyspace é como um namespace lógico que agrupa algumas características e propriedades.

Capítulo 3

Big Data

A quantidade de informação que está disponível para a humanidade é enorme e a medida que o conhecimento humano se expande, maior é a quantidade dessa informação que precisa ser armazenada e analizada. Além da quantidade, o fluxo e variedade dessas informações constantemente desafiam a indústria e a academia a medida em que a quantidade de Big Data aumenta exponencialmente. Nesse capítulo veremos uma definição detalhada de o que é Big Data, as tecnologias que apoiam esse domínio.

3.1 O que é Big Data?

Em um estudo divulgado em 2011 o tamanho do universo digital quebrou a barreira dos zettabytes e esse número está crescendo rapidamente [3]. Cientistas de diversas áreas estão vendo o grande potencial de conhecimento que se pode adiquirir pela análise a armazenamento de informação digital. Conforme já dito anteriormente o conceito de 'grande (big)' foi mudando no decorrer da nossa história. Na década de 70, grande significada megabytes; ao longo do tempo cresceu para gigabytes e em seguida, a terabytes. Atualmente já podemos dizer que grande é petabyte e até mesmo exabytes [10]. Contudo o conceito de Big Data não se dá somente por tamanho ou domínio, mas sim por um conjunto de características que o difere de uma base de dados comum.

Segundo Gartner big data é definido, em geral, como uma massa de dados de grande volume, velocidade e variedade de informações que exigem formas inovadoras de processamento para maior visibilidade e tomada de decisão [14]. A maioria dos estudiosos compartilham dessa mesma definição e dizem que Big Data é caracterizado por no mínimo três V's. Volume, variedade e velocidade. [19, 22]

Volume é a característica mais fácil de se perceber. Geramos enormes quantidades de dados todos os dias, e essa quantidade só tende a aumentar. Redes sociais, dispositivos móveis que guardam nossas informações, sites que armazenam nossas preferencias, dispositivos de busca que indexam as páginas da web e a popularização da computação em nuvem nos colocam em uma época de grande volume de dados, uma época em que tudo é informação, tudo é valioso, tudo pode ser extraído. Cada dia fica mais comum grandes empresas terem de ligar com dados na ordem de petabytes. Variedade é outra característica que é de fácil percepção, pois os dados são de diversas naturezas como email, dados gerados por mídias sociais (blogs, Twitter, Youtube, Facebook, Wikis), documentos eletrônicos, apresentações, fotos, mensagens instantaneas, dados médicos, videos, etc. A

Tabela 3.1: Tabela de bytes

Nome	Tamanho	Abreviação
Kilobyte	10^3	KB
Megabyte	10^6	MB
Gigabyte	10^9	GB
Terabyte	10^{12}	TB
Petabyte	10^{15}	PB
Exabyte	10^{18}	EB
Zettabyte	10^{21}	ZB
Yottabyte	10^{24}	YB

característica de velocidade é explicada quando precisamos processar os dados praticamente em tempo real como em controle de tráfego, detecções de fraudes e propagandas dinâmicas na web. Os dados são cada vez mais usados para tomadas de decisão em tempo real [9].

Dada a problemática do armazenamento, ao se deparar com os limites de técnicas e ferramentas disponíveis o mercado tratou de criar suas próprias soluções de gerenciamento de dados, em sua maioria não relacional. Usando a tecnologia apropriada, profissionais capacitados podem transformar grandes massas de dados em informações muito valiosas. Muitos sistemas comercias relacionais se dizem capazes de lidar com vários petabytes de base de dados (Greenplum,Netezza,Teradata, ou Vertica). Apesar dessa quantidade de dados atender a grande maioria das empresas, existem empresas de grande porte como o Google e o Facebook que não são atendidas e precisaram criar suas próprias soluções, além disso, sistemas open source como Postgres não tem o mesmo nível de escalabilidade que os comerciais [19].

3.2 Tecnologias de Apoio

3.2.1 Bases Relacionais

Quando pensamos em armazenamento de dados em SGBDs logo associamos essa idéia ao método tradicional que inclui bancos de dados como MySQL, PostgreSQL, modelagem relacional e esquemas de dados bem definidos. O modelo de dados relacional foi introduzido por Ted Codd, da IBM Research, em 1970, em um artigo que conseguiu atrair grande atenção devido a simplicidade e base matemática. Os SGBDs relacionais mais populares atualmente são o DB2 e Informix Dynamic Server (IBM), o Oracle e Rdb (Oracle), o Sybase SGBD (Sybase) e o SQLServer e Access (Microsoft). Ainda temos os de código aberto como o MySQL e PostgreSQL.

O modelo relacional representa o banco de dados como uma coleção de relações. Uma relação é como se fosse uma tablea de valores ou um arquivo de registros. Cada tabela é formada por uma ou mais colunas de dados. Por sua vez, cada linha na tabela contém uma instância única de dado para as categorias de colunas definidas. No modelo relacional

é possível criar conexões entre as tabelas e os campos e os formatos dos valores são bem definidos, ou seja, possui um schema de dados [13, 18].

Os SGBDs (Sistemas Gerenciadores de Banco de Dados) relacionais proveem diversas garantias aos seus usuários como: validação, verificação e garantias de integridade dos dados, controle de concorrência, recuperação de falhas, entre outros. As propriedades que esses bancos têm de prover são a atomicidade, consistência, isolamento e a durabilidade (ACID).Todas essas características mantém os SGBDs como principal solução na maioria dos ambientes computacionais, mas não impediram o surgimento de problemas, em alguns casos, causados pela rígida estrutuda definida pelo layout das tabelas, nomes e tipos das colunas.

As abordagens mais usadas para manipular grandes bases nesse tipo de estrutura são os data warehouses e data marts. Um data warehouse é um banco de dados relacional usado para armazenar, analizar e gerar relatórios sobre os dados. O data mart é a camada usada para acessar o data warehouse. As duas abordagem usadas para se armazenar dados em um data warehouse são a normalização e a modelagem dimensional [8].

Limitações

Com a evolução das aplicações e com requisitos cada vez mais exigentes, foram surgindo casos em que os banco de dados relacionais não escalavam. Operações de joins estão presentes nos menos dos bancos de dados relacionais, e esse tipo de operação é lenta.Para que SGBDs relacionais consigam garantir consistência para os dados eles usam o conceito de transações, o que requer um bloqueio nos dados durante um certo período de tempo. Dessa forma, quando o banco recebe várias requisições simultâneas em um mesmo dado os usuários são obrigados à esperarem em uma fila [17].

A necessidade de transformar os dados em tabelas causa um aumento na complexidade da operação pois requer o uso de complexos algoritmos de mapeamento e estrutura. Mesmo quando uma base de dados pode ser coberta pelo modelo relacional, as vezes as diversas garantias providas por esse modelo gera uma sobrecarga que não seria necessária para tarefas simples. O schema rigoroso pode ser pesado para aplicações que precisam de velocidade, como aplicações web e blogs que possuem diversos tipos de atributos. Textos, comentários, imagens, vídeos fonte, código e outras informações precisam ser armazenadas em diversas tabelas, e como as aplicações na web são muito ágeis, precisam ser amparadas por uma base de dados igualmente ágil e com um schema de fácil adaptação [16].

O considerável aumento na quantidade de dados deve ser considerado por grandes empresas como Facebook, Amazon e Google. Além de tratar terabytes/petabytes de dados, realizar requisições de leitura e escrita na base a todo o momento essas empresas devem se preocupar com o tempo que essas transações estão levando, ou seja, a latência. Para tratar esses requisitos é preciso mantes milhares de máquinas com um hardware moderno e veloz. Por ter que cumprir com os requisitos de ACID e manter os dados normalizados, um modelo relacional não é adequado para esse cenário, visto que as operações de join bloqueiam os dados e influenciam negativamente no desempenho da aplicação.

Outro requisito fundamental para as grandes empresas é a disponibilidade de seus serviços. Para isso a base de dados deve ser facilmente replicável e fornecer uma forma automática de tratamento à falha de bases ou do datacenter. Esses SGBDs também devem ser capazes de balancear a carga em várias máquinas para não sobrecarregar um único

servidor. Bancos relacionais priorizam a consistência em detrimento à disponibilidade e também possuem um mecanismo de replicação limitado.

Esses problemas podem ser resolvidos de algumas formas.Primeiramente optamos por um upgrade simples de hardware. Se o problema persistir a próxima opção seria adicionarmos novos servidores ao cluster, porém com os problemas de consistência e replicação durante o uso regular e em cenários de falha. A próxima etapa seria melhorar a configuração do gerenciador de banco de dados. Caso as opções de melhoria no SGBD se esgotem é preciso melhorar a aplicação. Verifica-se o desempenho das consultas, criamos índices e etc. Se o desempenho ainda não for satisfatório então talvez coloquemos uma camada de cache, mas que também gera um problema de consistência. Se mesmo assim o desempenho não atender as expectativas, então é necessário pensarmos novamente no SGBD. A última opção seria uma desnormarlização do banco, mas assim se estaria indo contra os princípios da modelagem relacional e das regras normais [17].

Dado toda essa problemática surge uma opção. Bancos de dados que não seguem o paradigma relacional. Os dados não são normalizados.

Capítulo 4

Testes

Nesse capítulo iniciaremos com a definição de o que é o termo NoSql e, após vermos como essa nova forma de se pensar em banco de dados surgiu, conheceremos os principais tipos de banco de dados não relacionais e como eles armazenam os seus dados.

4.1 Definição

Segundo o dicionário de termos da IEEE, teste é definido da seguinte forma:

- Teste: atividades nas quais um sistema ou um componente é executado sob determinadas condições e os resultados são observados ou gravados, e uma avaliação é feita observando determinado comportamento do sistema ou do componente;

4.2 Teste de Software e Qualidade de Software

O teste de software está diretamente ligado com a qualidade do software que está sendo desenvolvido. Podemos ver essa ligação já na definição de qualidade de software.

- Qualidade de Software: Conformidade a requisitos funcionais e de desenvolvimento explicitamente declarados, a padrões de desenvolvimento claramente documentados e a características implícitas que são esperadas de todo software profissionalmente desenvolvido.

Dentro da qualidade de software temos a atividade de garantia de qualidade de software e esta compreende uma variedade de tarefas associadas a sete grandes atividades, entre elas a atividade de testes:

1. Aplicação de métodos técnicos;
2. realização de revisões técnicas formais;
3. Atividades de testes de software;
4. Aplicação de padrões;
5. Controle de mudanças;

Tabela 4.1: Tipos de teste e sua característica de qualidade correspondente

Tipos de Teste	Características de qualidade
Funcionalidade	Funcionalidade
Interfaces	Conectividade
Carga	Continuidade, Performance
Produção	Operabilidade
Recuperação	Recuperação
Regressão id	Todas
Segurança	Segurança

6. Medição;
7. Manutenção de registros e reportagem;

Então podemos estar certos de que se queremos um software que atenda aos requisitos especificados, funcionais e não funcionais, que possua uma quantidade de erros reduzida e um desempenho que atenda ao usuário, uma tarefa que não pode ser despensada é o teste da aplicação. Como já dito anteriormente, teste de software e qualidade de software estão intimamente ligados, na tabela 4.1 podemos ver quais as características de qualidade são verificadas por determinados tipos de testes.

4.3 Tipos de Testes

O teste de software nos permite trabalhar com diversas estratégias e em diferentes níveis da aplicação. Emerson Rios e Trayahú Moreira [21] dizem que muitas vezes os tipos de software se sobrepõem, sendo até mesmo as suas definições abrangentes ou específicas, confome sua execução. Nessa seção listaremos os principais tipos de testes descritos por esses autores.

4.3.1 Aplicados a cada estágio de teste

Testes Caixa Preta

Esse tipo de teste tem como objetivo verificar as funcionalidades da aplicação e a aderência aos requisitos, do ponto de vista do usuário, sem se basear no código ou lógica interna da aplicação.

Testes Caixa Branca

Os testes de caixa branca avaliam o código, a lógica interna do componente, as configurações e outros elementos técnicos.

4.3.2 Estágios (ou Níveis) de teste

Testes unitários

Esse é o tipo de teste que analisa o estágio mais baixo da aplicação. São aplicados nos menores componentes de código criados, verificando o atendimento as especificações e funcionalidades. Verificam o funcionamento de um pedaço do sistema, componente ou programa, isoladamente. Geralmente são realizados pelos próprios desenvolvedores.

Testes de integração

Esse teste visa testar se as interações estre os componentes da aplicação está resultando em algum tipo de erro. Tem como objetivo assegurar que as interfaces funcionem corretamente e que os dados são processados corretamente.Componentes podem ser pedaços de código, módulos, aplicações distintas, clientes e servidores etc. Esse tipo de teste possui várias estratégias. Podemos testar a integração desde os componentes de mais baixo nível (Booton-up) até o sistema como um todo (Teste de sistema). Para o nosso trabalho nos atentaremos ao teste de sistema.

Testes de sistema

Esse teste é executado sobre o sistema como um todo, ou um subsistema, dentro de um ambiente operacional controlado. Deve ser simulada a operação normal do sistema, sendo testadas todas as suas funções de forma mais próxima possível do que irá ocorrer no ambiente de produção. É nesse estágio que deve-se realizar os testes de carga, performance, usabilidade, compatibilidade, segurança e recuperação.

Testes de aceitação

São realizados pelos usuários e visam garantir que a solução atenda aos objetivos do negócio e a seus requisitos, verificando as funcionalidades e a usabilidade do software.

4.3.3 Outros tipos de testes

Testes de carga

Permite avaliar a aplicação sob uma alta carga de dados, repetidas entradas de dados, consultas complexas ou uma grande quantidade simultânea de usuários. Dessa forma é possível medir o nível de escalabilidade da aplicação. Esse tipo de teste deve ser aplicado durante os testes de sistema e também podem ser chamados de testes de estresse.

Testes Back-to-back

É quando o mesmo teste é executado em versões diferentes do software e os resultados são comparados.

Testes de Configuração

É nesse tipo de teste de a execução da aplicação é analisada em diferentes configurações de ambiente.

Testes de Usabilidade

Mede a facilidade de uso da aplicação pelos usuários. É mais comum em aplicações web.

Testes de Segurança

Verifica o quão segura é a aplicação a acesso de usuários não autorizados.

Testes de Recuperação

Mede a qualidade da recuperação do software após falhas de hardware ou outro problemas inesperados.

Testes de Compatibilidade

Verifica se um software é capaz de ser executado em um ambiente determinado.

Testes de Desempenho

Verifica a adequação da aplicação aos níveis de desenpenho e tempo de resposta definidos nos requisitos. Também são conhecidos como testes de performance.

4.4 Planejamento dos Testes

Como o objetivo do trabalho é medir o desempenho da nossa aplicação com o uso de diferentes bancos de dados, restringimos os testes que serão usados no nosso projeto aos testes de carga e performance.

4.4.1 Automação de Testes

Durante muito tempo os testes de software foram feitos manualmente. Os proprios programadores eram encarregados de simular as mais diversas situações [21]. Com o passar do tempo as aplicações se tornaram muito mais complexas e, consequentemente, o processo de teste manual se tornou inviável. Esse cenário foi ideal para que surgissem ferramentas de automação do processo de testes.

As ferramentas de automação de teste visam facilitar o processo de teste e podem auxiliar no desenvolvimento dos testes, execução, manuseio das informações de resultado e a comunicação entre os envolvidos no processo. Utilizando scripts essas ferramentas são capazes de simular a utilização da aplicação por um ou vários usuários e, além disso, podem ser simulados vários cenários de uso. As ferramentas de teste podem ser divididas em três grupos: desenvolvimento, execução 4.2 e acompanhamento/suporte 4.1.

4.4.2 Teste de Performance

Molyneaux fala que do ponto de vista dos usuários, uma aplicação possui boa performance quando ela o permite realizar determinada tarefa sem demora [20]. Ela ainda

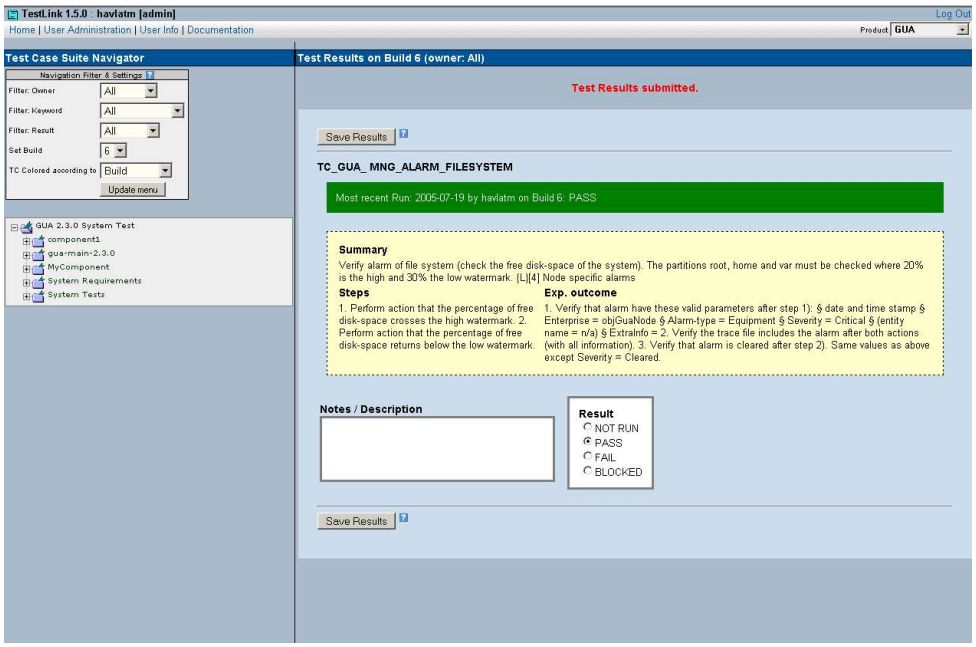

Figura 4.1: TestLink - acompanhamento/suporte

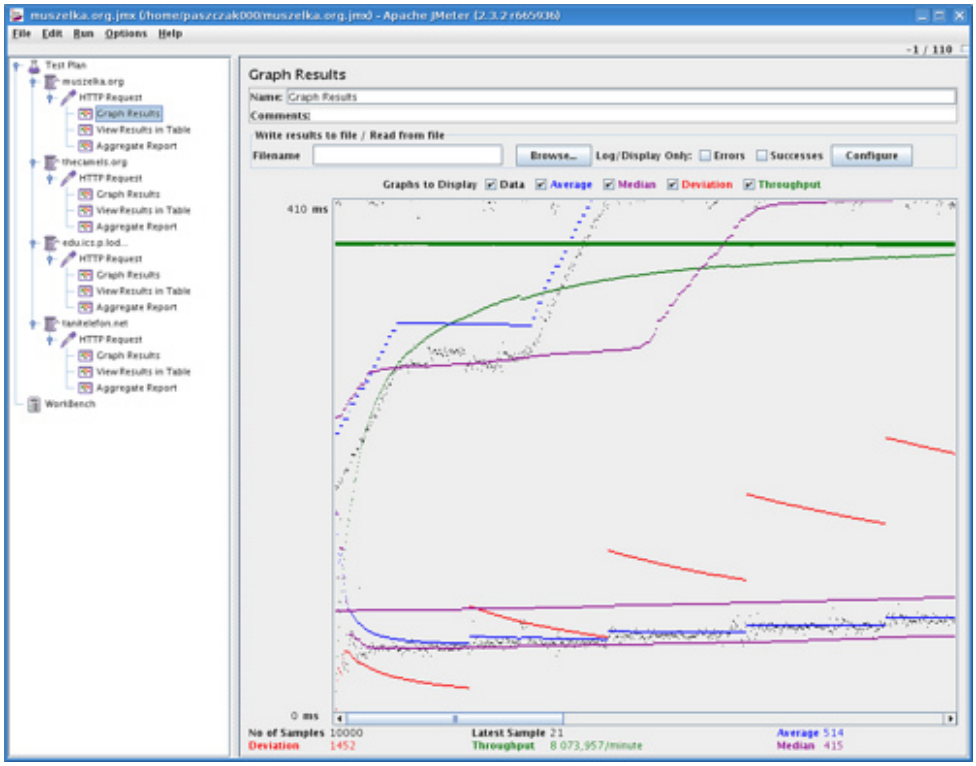

Figura 4.2: JMeter - Ferramenta para execução de testes

diz que em uma aplicação performática o usuário nunca poderá se deparar com uma tela vazia ao realizar operações. O teste de performance é usado para medir o desempenho, em tempo de execução, e com todos os módulos integrados. Conforme Molyneaux, dividiremos os requisitos de performance em dois: orientados a serviço e orientados a eficiência.

Os indicadores de performance orientados a serviço são a disponibilidade e o tempo de resposta. Eles medem a qualidade do serviço que a aplicação está provendo ao usuário. Já os indicadores orientados a eficiência são a vazão e utilização. Vamos definir esses termos:

- Disponibilidade: É a característica de estar disponível para o usuário. Em softwares críticos, qualquer período de indisponibilidade pode gerar grandes prejuísos.
- Tempo de resposta: É o intervalo de tempo entre a requisição e a resposta da aplicação.
- Vazão: É a taxa em que os eventos da aplicação ocorrem.
- Utilização: É a porcentagem da capacidade total de recursos da aplicação que esta sendo usada.

Para que o nosso processo de teste de performance seja bem sucedido precisamos seguir algumas etapas.

1. Escolher uma ferramenta de teste de performance apropriada;
2. Desenvolver um ambiente de teste adequado a realidade dos testes e o mais próximo da realidade;
3. Escolher os objetivos que desejamos alcançar no trabalho;
4. Identificar e criar scripts para as transações críticas para o negócio;

Referências

[1] Apresentação - assentamento funcional digital. Online; accessed 23-Abril-2013. 2

[2] Cassandra official site. Online; accessed 27-Janeiro-2013. 8

[3] Emc digital universe. Online; accessed 26-Janeiro-2013. 10

[4] Mongodb official site. Online; accessed 27-Janeiro-2013. 7

[5] nosql database org. Online; accessed 09-Dezembro-2012. 5, 6, 7

[6] Recomendações para digitalização de documentos arquivísticos permanentes. Online; accessed 23-Abril-2013. 1, 2

[7] Sigepe.org. Online; accessed 23-Abril-2013. 2

[8] K. Bakshi. Considerations for big data: Architecture and approach. In *Aerospace Conference, 2012 IEEE*, pages 1 –7, march 2012. 12

[9] D. Bollier, Communications, and Society Program (Aspen Institute). *The Promise and Peril of Big Data*. Aspen Institute, Communications and Society Program, 2010. 11

[10] Vinayak R. Borkar, Michael J. Carey, and Chen Li. Big data platforms: What's next? *XRDS*, 19(1):44–49, 2012. 1, 10

[11] Ricardo W. Brito. Banco de dados nosql x sgbds relacionais: Análise comparativa. 2010. 4, 6

[12] Andrew Cron, Huy L. Nguyen, and Aditya Parameswaran. Big data. *XRDS*, 19(1):7–8, 2012. 1

[13] Ramez Elmasri and Shamkant B. Navathe. *Sistemas de Banco de dados*. Pearson Addison Wesley, 6 edition, 2011. 12

[14] Gartner. Big data. Online; accessed 14-Dezembro-2012. 10

[15] Jing Han, E. Haihong, Guan Le, and Jian Du. Survey on nosql database. In *Pervasive Computing and Applications (ICPCA), 2011 6th International Conference on*, pages 363 –366, oct. 2011. 6

[16] R. Hecht and S. Jablonski. Nosql evaluation: A use case oriented survey. In *Cloud and Service Computing (CSC), 2011 International Conference on*, pages 336 –341, dec. 2011. 4, 5, 6, 7, 12

[17] E. Hewitt. *Cassandra: The Definitive Guide*. Definitive Guide Series. O'Reilly Media, 2010. 8, 12, 13

[18] N. Leavitt. Will nosql databases live up to their promise? *Computer*, 43(2):12 –14, feb. 2010. 5, 12

[19] Sam Madden. From databases to big data. *Internet Computing, IEEE*, 16(3):4 –6, may-june 2012. 10, 11

[20] Ian Molyneaux. *The Art of Application Performance Testing - Help for Programmers and Quality Assurance.* O'Reilly, 2009. 17

[21] E. Rios and T. MOREIRA. *Teste de software*. Alta Books, 2006. 15, 17

[22] Cezar Taurion. Big data = volume + variedade + velocidade, 2012. Online; accessed 14-Dezembro-2012. 10

[23] Cezar Taurion. Você realmente sabe o que é big data?, 2012. Online; accessed 14-Dezembro-2012. 2

[24] S. Tiwari. *Professional NoSQL*. Wrox Programmer to Programmer. Wiley, 2011. 7