FACULDADE DE ADMINISTRAÇÃO E NEGÓCIOS DE SERGIPE – FANESE

CURSO DE PÓS GRADUAÇÃO EM BANCO DE DADOS

SÉRGIO MURILO DA SILVA OLIVEIRA SOBRINHO

NOSQL:

A APLICAÇÃO EM AMBIENTES CORPORATIVOS

Aracaju – SE

2016.1
SÉRGIO MURILO DA SILVA OLIVEIRA SOBRINHO

NOSQL:

A APLICAÇÃO EM AMBIENTES CORPORATIVOS

Trabalho de Conclusão de Curso apresentado ao Núcleo de Pós-Graduação e Extensão – NPGE, da Faculdade de Administração de Negócios de Sergipe – FANESE, como requisito para a obtenção do título de Especialista em Banco de Dados.

Aracaju - SE

2016 / 01

NOSQL: A APLICAÇÃO EM AMBIENTES CORPORATIVOS

Sérgio Murilo da Silva Oliveira Sobrinho[1]

RESUMO

O modelo relacional de banco de dados se faz presente como o único ou o mais utilizado na grande maioria de aplicações que fazem o uso de uma estrutura para o armazenamento de seus dados. Mas, com o passar do tempo, o grande crescimento do volume de dados criou a necessidade de uma busca de novos paradigmas e estruturas que comportem de melhor forma e trabalhem melhor com esse grande volume de informações. Quesitos como a escalabilidade e armazenamento distribuído, somados a um grande volume de informações armazenadas, vem tornando o uso do modelo relacional bastante custoso a ponto de se tornar inviável em algumas situações especificas. É neste contexto que se surge um novo paradigma de armazenamento e processamento de dados, os bancos de dados não relacionais de alto desempenho, os bancos NoSql. Este artigo qualitativo e bibliográfico aborda a utilização de NoSql em ambientes corporativos como estrutura de banco de dados como solução de aplicações que fazem necessário grande escalabilidade em conjunto ao grande volume de dados comportado. Dentre os vários modelos de bancos não relacionais apresentados, é prototipado uma implementação de um sistema utilizando um banco de dados orientado a documentos, o mongodb, e discutido a viabilidade de execução deste protótipo.

Palavras-chave: NoSql;Não relacional; banco de dados;mongodb

1. INTRODUÇÃO

Para fornecer uma maneira eficaz de múltiplos acessos a diversos usuários, aplicações de grande porte necessitam que seus bancos de dados te ofereçam total suporte e escalabilidade para a execução. Fazendo o uso deste contexto,

[1] Aluno da Faculdade FANESE

requisições são feitas ao banco de dados em questão, fazendo a manutenção da informação requisitada.

O uso de servidores com configurações robustas e bom desempenho se faz necessário para suportar tal estrutura. Em alguns casos, mesmo com a utilização de um alto poder de processamento, o modelo relacional de banco de dados não tem atendido com satisfação devido à geração de um grande volume de dados e a necessidade da alocação de muito recurso para a execução da tarefa requisitada. No modelo relacional, após um grande crescimento no volume de informações, é comum haver uma demora no tempo de resposta à aplicação.

A busca por escalabilidade quando se tratar de grande volume de dados demandou a criação de uma estrutura de banco de dados que oferecesse melhor desempenho. Partindo desta necessidade, surge o conceito dos bancos não relacionais, os bancos de dados NoSql (*NotOnly SQL*).

O termo NoSql, foi utilizado primeiramente como nome de um banco de dados não relacional de código aberto, em 1998. Seu autor, Carlos Strozzi (1998) afirmava que o NoSql “é completamente distinto do modelo relacional e, portanto, deveria ser mais apropriadamente chamado "NoREL" ou algo que produzisse o mesmo efeito”. No artigo: “BigTable: A Distributed Storage System for Structured Data, publicado pelo Google em 2006”, novamente é referenciado o conceito NoSql. Seguindo a tendência, empresas com Amazon, em 2007, e Facebook, em 2008, lançaram outros modelos de banco NoSql, os bancos Dynamo e Cassandra, respectivamente. A partir daí o surgimento de várias outras implementações de bancos NoSql vem sendo apresentadas. Em 2009, vem ser lançada a primeira versão pública do MongoDB, que começou a ser desenvolvido em 2007 pela 10gen.

Este artigo tem como objetivo analisar situações onde se tem como melhor opção a utilização de NoSql em aplicações corporativas. Estando estruturado conforme a seguir: A seção 2 apresenta uma visão geral a respeito dos bancos de dados NoSql e seus aspectos. A seção 3 apresenta a estrutura do banco comumente adotado por ambientes corporativos. A seção 4 apresenta as deficiências enfrentadas no modelo relacional. A seção 5 apresenta um comparativo direto entre o modelo de dados relacional e o modelo não relacional. A seção 5 apresenta uma implementação de NoSql como melhor alternativa.

2. UMA VISÃO GERAL

Como proposta de solução dos problemas enfrentados por aplicações que processam grandes volumes de dados, surge os bancos de dados NoSql. Foram criados para solucionar as deficiências apresentadas em contextos específicos pelo modelo relacional que exigem maior disponibilidade e escalabilidade por parte de servidores de banco de dados.

Em outras palavras, o NoSql é um paradigma alternativo ao modelo relacional, criado para suprir necessidades em que o modelo relacional possui deficiência, não sendo utilizado como um substituto dos bancos relacionais, mas como uma ferramenta que pode ser utilizada em conjunto para suprimir as necessidades em situações especificas.

Dentre principais aspectos do modelo NoSql (não relacional), estão a desnormalização dos dados e alto desempenho no armazenamento paralelo de dados em grande volume. Possui também uma grande capacidade de expansão horizontal, utilizando servidores de banco de dados com processamento menor lado a lado, tratando assim a leitura e escrita de um grande número de dados. Segundo Ianni (2013), os BDs NoSql “...permitem uma escalabilidade mais barata e menos trabalhosa, pois não exigem máquinas extremamente poderosas e sua facilidade de manutenção permite que um número menor de profissionais seja necessário. ”

VIEIRA et. al. (2012) afirma que “O processamento de dados distribuídos é possível através da utilização do conceito de MapReduce, que realiza o mapeamento dos dados para os servidores que irão processá-los e gerar um resultado”.

> Quanto ao tipo de controle de consistência, os BDs NoSQL implementam o modelo BasicallyAvailable, Soft-state, Eventuallyconsistency (BASE), que fornece à aplicação uma visão de que está disponível o tempo todo para consistência, o que não ocorre na realidade, realizando a consistência no momento adequado. (QUICOLI, 2013).

As várias implementações de bancos NoSql fazem o uso de estruturas distintas, existindo quatro modelos mais comuns, são eles:

• **Armazenamento Chave-Valor**: Fazem o uso do conceito chave/valor, conhecido também como tabela hash, na qual há uma chave única e um indicador de um dado ou de um item em particular, consistindo assim em registros e evitando a redundância de dados.

O modelo chave-valor é o mais simples e fácil de implementar. Mas é ineficiente quando o objetivo de sua utilização é apenas consultar ou atualizar parte do valor do dado.

Exemplos	Redis, DynamoBD, AzureTableStorage
Aplicações	Cache de conteúdo, logging, etc
Modelo de dados	Coleção de pares de chave-valor
Pontos fortes	Pesquisas rápidas
Pontos fracos	Dados não possuem schema

• **Armazenamento WideColumnStore**: Neste modelo, o armazenamento de dados em famílias de colunas, apesar de não implementar o conceito de tabelas do modelo de banco de dados relacional.

As linhas são agrupadas conforme o contexto. Foram criadas para armazenar e processar grandes quantidades de dados distribuídos em muitas maquinas. Ainda

existem chaves, mas elas apontam para colunas múltiplas. As colunas são organizadas por família da coluna.

Exemplos	Cassandra, Riak
Aplicações	Sistemas de arquivos distribuídos
Modelo de dados	Famílias de colunas
Pontos fortes	Pesquisas rápidas, boa distribuição de armazenamento de dados
Pontos fracos	API de baixo nível

• **Armazenamento orientado a documentos**: É composto por documentos no lugar de registros, onde estes são baseados em eXtensibleMarkupLanguage (XML) ou JavaScriptObjectNotation (JSON). Este último sendo o mais usado em implementações orientada a documentos. Os documentos dos bancos dessa categoria, são coleções de chaves e atributos, podendo esses, serem multivalorados ou não.

Em geral, bancos orientados a documentos não possuem esquema, ou seja, os documentos não precisam possuir uma estrutura em comum para serem armazenados, característica esta que faz com que bancos orientados a documentos sejam uma das melhores opções na implementação de armazenamento de dados semiestruturados.

Exemplos	MongoDB, CouchDB, RavenDB
Aplicações	Aplicações WEB em geral
Modelo de dados	Coleções de coleções de chave-valor

Pontos fortes	Tolerante a dados incompletos
Pontos fracos	Query performance, sem sintaxe de query padrão

• **Armazenamento orientado a grafos**: Neste modelo, é utilizado a estrutura de grafo no armazenamento de dados. Os dados são classificados e armazenados como entidades, assim como, suas relações são estabelecidas por meio de relacionamentos.

O modelo de grafos é mais interessante que outros quando informações sobre a interconectividade ou a topologia dos dados são mais importantes, ou tão importante quanto o valor do dado propriamente dito. O modelo orientado a grafo possui três componentes básicos: os nós (vértices do grafo), os relacionamentos (arestas) e as propriedades (ou atributos) dos nós e relacionamentos. Neste caso, o banco de dados pode ser visto como um multigrafo rotulado e direcionado, onde cada par de nós pode ser conectado por uma ou mais arestas.

Exemplos	Neo4J, InfoGrid, InfiniteGraph
Aplicações	Redes sociais, recomendações
Modelo de dados	Grafos de propriedades (nós)
Pontos fortes	Conectividade, algorítimos grafos, p.e. caminho mais curto, relacionamentos n degrees
Pontos fracos	Não é fácil de agrupar, ter que atravessar todo o grafo para obter uma resposta definitiva

O volume de dados a ser armazenados e a forma como eles serão melhor processados é quem vai definir qual a melhor escolha de modelo de banco de dados NoSql a se utilizar na aplicação. O crescimento constante no volume de dados e a necessidade de uma alta escalabilidade torna o NoSql a melhor solução para o tratamento desses dados.

A possibilidade de uma aplicação fazer o uso de um banco de dados NoSql e um banco de dados relacional simultaneamente, faz com que o NoSql seja implementado apenas em cenários que haja a necessidade de manipulação de uma grande carga de dados, e os demais cenários continuem utilizando o paradigma relacional.

Nestes casos, são aplicados o desenvolvimento conhecido como desenvolvimento hibrido, onde um banco relacional trabalha em conjunto com um banco NoSql.

3. A ATUAL ESTRUTURA UTILIZADA EM CORPORAÇÕES

Por atender a necessidade da grande maioria das aplicações, o banco de dados relacional é frequentemente o mais utilizado se comparado ao banco NoSql. Dentre as várias opções de modelo relacional, estão: MySql, Oracle, SQLServer, PostgreSQL.

O modelo de banco de dados relacional foi proposto por Edgar Codd, um pesquisador da IBM, em torno de 1970, sendo descrito no artigo "Relational Model of Data for Large Shared Data Banks". Desde então o modelo relacional é o modelo de banco de dados dominante nas aplicações comerciais até hoje.

Sempre contando com comunidades bastante ativas e enorme suporte, os bancos relacionais possuem uma grande gama de profissionais que desenvolvem neste ambiente, liderando a utilização nos vários segmentos de sistemas existentes.

Fazendo o uso de tabelas, com linhas e colunas identificadas por uma chave única, é feito o armazenamento de dados no modelo relacional. Por meio destas chaves é possível fazer o relacionamento com outras tabelas.

> Para a criação da estrutura dos bancos de dados relacionais é utilizada a linguagem Data DefinitionLanguage (DDL), para manipular as informações Data ManipulationLanguage (DML) e por fim para atribuir permissões dos usuários às tabelas criadas é utilizada a Data ControlLanguage (DCL). É possível também a criação de índices que permitem uma forma mais rápida de buscar informações nas tabelas [NETO, 2013].

As principais características oferecidas pelo modelo relacional são: Atomicidade, consistência dos dados, isolamento e Durabilidade. A atomicidade oferece a garantia de que ou todas as informações são processadas ou a transação do dado é descartada em caso de alguma falha.

A consistência de dados garante que ao iniciar a manipulação do dado qualquer alteração feita deve estar consistente no banco de dados, O isolamento permite que as transações sejam realizadas de maneira individual, sem a dependência interna de outra e a durabilidade garante que após a realização bem-sucedida de uma alteração no banco de dados, esta não seja mais desfeita.

A linguagem chamada de SQL foi desenvolvida para trabalhar em conjunto com os bancos relacionais. Inspirada na álgebra relacional e também desenvolvida pela IBM, o SQL é uma linguagem declarativa para o uso em banco de dados relacionais.

Com a sua facilidade de uso e simplicidade de expressão, o SQL se transformou na linguagem de consulta de dados com a maior utilização no mundo, o que ajudou na consolidação do modelo relacional de banco de dados.

Com todas essas características e recursos, o modelo relacional assumiu uma posição de destaque e predominância nos diversos ambientes que fazem uso de banco de dados para o armazenamento de informações. No entanto, a complexidade estrutural deste modelo acarretou no surgimento de alguns

problemas, principalmente relacionados ao crescente volume de dados que são armazenados atualmente.

4. AS DEFICIÊNCIAS ENFRENTADAS NO MODELO RELACIONAL

Com pequenas mudanças desde o seu surgimento para atender a novas exigências, o modelo relacional dificilmente irá suprir totais necessidades do mercado. A utilização de um modelo mais flexível e escalável, vem motivando o uso do NoSql para suprir tais necessidades apresentadas pelo modelo relacional.

Como exemplo, em uma determinada aplicação web que faça o uso de um banco de dados relacional para o armazenamento de seus dados, com o constante aumento em sua carga de dados, a aplicação começa a ter uma queda em seu desempenho. Para que seja sanado essa queda no desempenho de consumo de dados existem apenas duas alternativas: o upgrade no servidor onde está hospedada ou aumentar o número de servidores.

Tais alternativas se tornariam apenas temporárias, se fosse levado em consideração o constante aumento no fluxo de dados, transferindo o problema para o próprio acesso a base de dados. A solução neste caso, portanto, passaria a ser escalar o banco dados, distribuindo-o em várias maquinas, o que não seria uma tarefa tão simples.

Outro ponto a ser exposto é que, o modelo relacional possui um alto custo para sua escalabilidade, tanto em sua forma horizontal, que pode ser feita particionando a estrutura de dados, ou em sua forma vertical, que é feita aumentado a capacidade de processamento do servidor, fazendo necessário um alto gasto com a compra de novos servidores com mais capacidade de processamento.

Atendendo aplicações de pequeno, médio e grande porte, no geral, o modelo relacional não foi concebido visando o armazenamento de grandes volumes de dados de forma distribuída e oferecendo um alto processamento de grande volume informações.

O escalonamento, feito em sua maioria de forma vertical, faz com que muitas vezes se atinja o limite do hardware sem obter o desempenho desejado. Situações como essa demandam indispensável para o processamento e armazenamento de dados distribuídos.

5. BANCOS RELACIONAIS VS BANCOS NOSQL

Ao fazer a análise dos dados, para uma possível aplicação de um modelo não relacional ao invés da aplicação do modelo relacional tradicional, se faz necessário a consideração de quesitos básicos, como exemplo, formas e critérios adotados para o escalonamento, a disponibilidade de dados, assim como sua consistência.

O quesito escalonamento é primordial, pois é nesse ponto em que justamente o modelo não relacional destacam suas principais vantagens em relação ao modelo relacional, por terem sido concebido com tal finalidade, diferente do modelo relacional que possui uma estrutura com menor flexibilidade e pouco adaptada para os cenários em que se faz necessário o escalonamento.

Para uma determinada aplicação, com arquitetura simples, para se promover opções de atendimento ao número crescente de fluxo dados armazenados, os responsáveis pela aplicação poderiam optar pelo upgrade no servidor, solução também conhecida como Escalonamento Vertical, ou pelo aumento na quantidade de servidores, solução também conhecida como Escalonamento Horizontal.

Caracterizado por sua simplicidade de aplicação, o Escalonamento Vertical tem sido comumente o mais indicado para as camadas de banco de dados, por sua vez o Escalonamento Horizontal tem sido mais aplicado nas camadas das aplicações, principalmente para as aplicações web.

O próximo passo a se tomar consiste em tentar escalonar o próprio banco de dados. Basicamente, a proposta é distribuir o banco em várias máquinas, fazendo o uso de um particionamento dos dados. Processo esse conhecido como *sharding*. Entretanto, esta abordagem esbarra nas características individuais do SGBD, dada a dificuldade em se adaptar a toda a estrutura lógica do Modelo Relacional à solução

que foi proposta. A grande dificuldade na aplicação do processo de *sharding* em um banco de dados relacionais está relacionada com alguns fatores importantes.

Primeiramente, enquanto os dados de um SGBD relacional obedecem ao critério de normalização, o processo de *sharding* se caracteriza justamente pelo inverso: a desnormalização dos dados. Dados que são utilizados em conjunto são armazenados em conjunto.

Um segundo ponto a ser levado em questão é que, existe uma mudança de paradigma em relação ao processo de escalonamento. Enquanto os SGBDs relacionais aplicam a estratégia de escalonamento vertical, ou seja, reforçar o servidor; o processo de *sharding* tem como o objetivo trabalhar com o escalonamento horizontal, paralelizando seus dados em vários servidores.

Um terceiro ponto é que, o próprio volume de dados por máquina é sensivelmente minimizado devido ao processo de distribuição. Conjuntos de dados menores são mais fáceis de serem acessados, atualizados e gerenciados.

Por último, o fator não menos importante, o nível de disponibilidade do sistema é otimizado em relação à abordagem relacional, justamente pelo critério referente ao fato de que a queda de uma das máquinas não irá causar a interrupção de todo o sistema.

Já se tratando dos bancos de dados não relacionais, estes foram especialmente projetados para atender características como essas discutidas, de forma mais natural.

Um exemplo é o MongoDB o qual inclui um módulo de sharding automático, permitindo a construção de cluster de banco de dados escaláveis horizontalmente projetado para incorporar novas máquinas de uma forma dinâmica. Em uma arquitetura como esta, um cluster consiste de dois ou mais blocos de servidores que recebem o nome de shards, um ou mais servidores de configuração – os quais armazenam os metadados do banco – e qualquer número de processos roteadores chamados de mongos através dos quais os servidores da aplicação se conectam.

Tendo como um exemplo o cenário: possuem inicialmente dois servidores, armazenando em cada parte distinta do banco de dados, a aplicação fará conexão

ao cluster de nós através de um dos mongos o qual é responsável pela distribuição de operações ao devido destino.

O shard, como é conhecido o local de destino, consiste de dois ou mais servidores, cada um possuindo em si uma réplica da porção do banco armazenado. O gerenciamento de falhas se dá através da substituição automática do servidor falho por um novo servidor. Assim, cada shard sempre estará on-line. Apesar de, para a aplicação, o cluster continuar sendo visto como um banco de dados não-distribuído, a capacidade do sistema é otimizada.

Atualizações e leituras de dados são distribuídas através dos dois servidores shard.

Outra comparação interessante entre os bancos de dados relacionais com os bancos de dados não relacionais é na questão de controle de concorrência. Enquanto no modelo relacional o processo de controle de concorrência faz o uso de bloqueios (locks) para que seja garantido que dois acessos simultâneos não estejam atualizando o mesmo item de dados, em soluções de bancos não relacionais são aplicadas outras estratégias que permitem um grau mais elevado de concorrência.

Ao se substituir um modelo relacional por uma solução NoSQL, é preciso ter em mente que a arquitetura irá perder em consistência, mas pode obter ganho na flexibilidade, performance e disponibilidade. Tal ideia se baseia no Teorema de Eric Brewer conhecido como Teorema CAP (*Consistency, Availability e Partition Tolerance*). O qual afirma “É impossível para um sistema computacional distribuído garantir, de forma simultânea, consistência, disponibilidade e tolerância ao particionamento. ”

Segundo esse teorema, um sistema distribuído pode garantir apenas duas dessas três características simultaneamente. O tipo de consistência utilizada nos bancos de dados NoSQL é chamada de Consistência Eventual: o sistema de armazenamento garante que se nenhuma nova atualização for realizada sobre o objeto, eventualmente todos os acessos a esse objeto retornarão o último valor atualizado.

Quando uma escrita for realizada no banco, não se pode garantir que, a partir daquele momento, todos os outros processos terão acesso apenas ao dado

atualizado. Estes processos estarão apenas eventualmente capacitados para perceber tais mudanças.

6. A IMPLEMENTAÇÃO DO NOSQL

Optando por uma aplicação que registra dados de entrada/saída de pessoas como ambiente para a implementação, serão avaliados como critérios: o tempo de desenvolvimento, facilidade de desenvolvimento/manutenção e integridade de dados.

A aplicação permitirá o cadastro de pessoas, incluindo todos os seus dados, foto, e histórico de entrada e saída no local onde o sistema será aplicado.

Como linguagem de programação para o desenvolvimento foi escolhido o JavaScript por ser uma das mais utilizadas atualmente em ambiente web, gratuita e por ser a mesma linguagem utilizada para a implementação do banco de dados. Quando ao banco de dados optou-se pelo MongoDB, por ser um banco de dados que utiliza javascript em seu desenvolvimento, o que facilita a utilização de apenas uma linguagem em todo o desenvolvimento do sistema, seja ela na aplicação ou banco de dados.

Não existindo o paradigma de relacionamento no NoSql, o mongodb armazena seus dados no formato de JSON (JavaScript Object Notation), que será o mesmo utilizado pela aplicação para a manipulação de dados.

Por não possuir controle de chaves e funções para a integridade dos dados armazenados, todo este controle fica por conta da aplicação, tornando a implementação mais complexa, o que demanda mais tempo de desenvolvimento em comparação a uma mesma implementação no modelo relacional.

O MongoDB, mantido pela 10gen, possui uma solução paga que te oferece um suporte para atualizações, validação de um plano de implementação, sugestões de configuração. Por outro lado, existe também uma solução gratuita e uma comunidade bastante ativa para ajuda e suporte. Atualmente no mercado, a

deficiência em profissionais qualificados, e a grande maioria de manuais e tutoriais estão disponíveis em inglês.

Como o paradigma NoSql não oferece a solução ACID (Atomicidade, Consistência, Isolamento e Durabilidade), a integridade dos dados fica por conta da implementação da aplicação. Tendo como foco o armazenamento de grandes volumes de dados, o NoSql depende bastante de uma boa implementação para que seja proveitoso o seu uso.

Em uma implementação bem aplicada, a escalabilidade dos dados desta aplicação está garantida de forma mais simples, graças ao recurso de sharding oferecido nativamente pelo mongodb. No surgimento da necessidade de escalar a aplicação, será optado pelo escalamento horizontal.

7. CONCLUSÃO

A aplicação de um banco de dados não relacional em substituição de um modelo relacional, deve ser analisada de forma cuidadosa. Algumas informações devem ser levadas em conta na hora da decisão. Fatores como a consistência dos dados, o volume a ser tratado, sua disponibilidade e eventuais escalabilidades, devem ser vistos ponto a ponto. Neste artigo foi apresentado um pequeno comparativo, levando em conta estes fatores, sendo abordado um a um.

A utilização de um banco de dados não relacional torna-se cada vez mais crescente, tendo em vista o crescimento exponencial de dados e a necessidade de armazenar e processar os mesmos, cada vez mais as aplicações demandam grandes volumes de dados e necessitam cada vez mais de um rápido processamento destes dados. Em aplicações em ambientes corporativos menores, a utilização ainda não é bem difundida, por se tratar de uma tecnologia relativamente nova, pela falta de mão de obra qualificada no mercado e poucos profissionais e empresas especializadas em treinamento e consultoria desta tecnologia.

Em contrapartida, vem se tornando uma pratica crescente a utilização de uma implementação hibrida, sendo essa uma aplicação que trabalha simultaneamente

com bancos de dados relacionais e NoSql. Sendo assim, se faz o uso do NoSql apenas em módulos específicos que demandam alto volume de dados e necessitam de um rápido processamento, e mantendo todo o resto da estrutura em um modelo relacional tradicional.

São poucas as empresas que possuem grandes volumes de dados e uma recorrente necessidade de escalabilidade, para que se justifique a utilização de uma aplicação com sua implementação baseada em um modelo de dados NoSql.

ABSTRACT

The relational model database is present as the only or the most used in the vast majority of applications that make use of a structure for storing your data. But over time, the large increase in the volume of data created the need for a search for new paradigms and structures that behave better and work better with this large volume of information. Issues such as scalability and distributed storage, added to a large volume of information stored, is making use of the very expensive relational model to the point of becoming unviable in some specific situations. It is in this context that there is a new paradigm of storage and data processing, non-relational databases, high performance NoSQL banks. This qualitative and bibliographic article discusses the use of NoSQL in corporate environments such as database structure as a solution for applications that do need highly scalable together the large volume of data behaved. Among the various models of non-relational databases presented, it is prototyped an implementation of a system using a database-driven documents, mongodb, and discussed the feasibility of implementing this prototype.

Keywords: NoSql; NotOnly SQL; database; mongodb

REFERÊNCIAS

IANNI, V. Introdução aos bancos de dados NoSQL.**Devmedia**. Disponível em: <http://www.devmedia.com.br/introducao-aos-bancos-de-dados-nosql/26044>. Acesso em: 07 ago. 2015.

java-magazine-114/27500#ixzz2QwZa877Z>. Acesso em: 07 ago. 2015.

MIGUEL, S. B.; CARNEIRO, F. C. F. Repositório de Dados Relacional ou NoSQL?

NASCIMENTO, J. NoSql – você realmente sabe do que estamos falando?**iMasters**. Disponível em: <http://imasters.com.br/artigo/17043/banco-de-dados/nosql-voce-realmente-sabe-do-que-estamos-falando/>. Acesso em: 09 ago. 2015.

“Revista Java Magazine”, n. 114**Devmedia**. Disponível em: <http://www.devmedia.com.br/repositorio-de-dados-relac ional-ou-nosql-revista-ferramenta-certa-para-o-banco-de-dados-nosql/>. Acesso em: 10 ago. 2015.

URANI, A. et al.**MongoDB: Construa novas aplicações com novas tecnologias**. São Paulo, SP: Casa do Código, 2015.

VARDANYAN, M. Escolhendo a ferramenta certa para o banco de dados NoSql.**iMasters**. Disponível em: < http://imasters.com.br/artigo/21781/banco-de-dados/escolhendo--a-