Abordagem NoSQL – uma real alternativa

Renato Molina Toth
Universidade Federal de São Carlos – *Campus Sorocaba*
Sorocaba, São Paulo
email: renatomolinat@gmail.com

***Abstract*— Nas grandes aplicações web, desktop ou até mesmo móveis é cada vez mais comum o grande volume de dados, e com certeza sabemos do fato: no futuro teremos mais informações e dados para armazenar. Em grande parte dos sistemas e aplicações feitas por desenvolvedores de todo planeta usa-se como padrão um modelo relacional de base de dados. Entretanto, com o crescimento de sistemas críticos que necessitam de alto desempenho e configurações menos árduas de serem efetuadas nos sistemas de banco de dados em sistemas distribuídos, foi introduzido o conceito e modelo de base de dados não relacional, denominado NoSQL. Sua abordagem veio para resolver os problemas dos sistemas que precisam de escalabilidade e agilidade em suas inserções, buscas e alterações.**

***Keywords*— NoSQL, Escalabilidade, Banco de dados Relacionais.**

I. Introdução

Indubitavelmente, o crescimento da quantidade de dados e informação na web é indiscutível e perceptível no nosso dia a dia. Entretanto, se as aplicações possuem um grande volume de dados é possível que se tenha grandes problemas em relação à infra-estrutura. Atualmente, uma das abordagens mais adotadas é a distribuição horizontal (scale out), isto é, adicionar mais servidores para que atenda a aplicação. Entretanto, isso pode acarretar em um problema grande, pois, não é uma tarefa fácil efetuar a adição de um servidor em um sistema distribuído, devido a complexidade de configuração em uma aplicação que usa o banco de dados relacional.

Quando tratamos de números relativamente grandes de registros, o desempenho pode ser prejudicado quando usamos uma abordagem de banco de dados relacional, diminuindo a interação com o usuário.

O NoSQL surgiu para resolver essa problemática, mostrando uma abordagem diferente de persistência de dados, baseada em disponibilidade, desempenho e escalabilidade dos dados.

Atualmente a diversidade de tipos de modelos e números de banco de dados não-relacionais (NoSQL) é grande, cada um possuindo conceitos e particularidades diferentes, proporcionando ao desenvolvedor uma gama enorme, podendo atender à necessidades distintas.

Nesse artigo iremos apresentar uma visão mais apurada das particularidades dos principais modelos de banco de dados não relacional. Expondo os métodos de funcionamento, os conceitos teóricos e uma reflexão sobre em que determinada circunstância o modelo é recomendado para ser utilizado na aplicação.

Os modelos de dados NoSQL que iremos abordar a fundo nesse artigo são:

- Baseados em documento (document-store)
- Chave/valor(key/value)
- Grafo (Graph)

As ferramentas que implementam base de dados com conceitos NoSQL são predominantemente opensource[2], refletindo positiva e diretamente no uso do mercado e no crescimento da comunidade de desenvolvimento e aprimoramento de ferramentas para a abordagem.

Existe uma grande tendência de software com o código aberto, pois eles ganham um incentivo de grande número de desenvolvedores rapidamente, uma vez que permitem usuários realizarem adaptações em sistemas com um baixo custo, suprindo sua necessidade.

Outro ponto interessante e poderoso nessa abordagem é o nível de abstração[2] que ela possui, dando a possibilidade dos desenvolvedores se focarem principalmente na lógica da aplicação, fazendo com que a persistência tenha interfaces quase triviais.

Na seção II será abordada a problemática da escalabilidade com uma abordagem diferente do modelo tradicional proposto.

A seção III descreve os tipos de modelos de base de dados não relacionais, suas características, vantagens e desvantagens.

Por ser um assunto relativamente novo, existe muita divergência em perspectivas para o futuro ou na realidade atual desses modelos, essa discussão é apresentada na seção IV.

Muito se fala que o NoSQL possui desempenho incrivelmente maior do que o modelo relacional, entretanto, os estudos de benchmark e análises mais profundas são escassos, a fim de mostrar dados mensuráveis a seção V irá apresentar um estudo de caso real de uma grande corporação.

R. M. Toth – Universidade Federal de São Carlos – Campus Sorocaba, Sorocaba, São Paulo, Brasil, renatomolinat@gmail.com

II. ESCALABILIDADE

A. *Banco de dados relacionais*

É verdade que a escalabilidade pode ser alcançada em banco de dados com o modelo relacional, entrentato, pode ser uma tarefa de custo elevado e uma tarefa bastante complexa, como comentado anteriormente.

Quando é necessário o aumento de infra-estrutura para um bancos de dados é normal usar como primeiro recurso uma distribuição vertical[11] (scale up) de servidores, ou seja, quanto mais dados, aumenta-se a configuração física do equipamento (memória, processador e hard-disk). Essa pode ser uma solução excelente se tratando de curto prazo, contudo, no futuro o problema da escalabilidade pode voltar a aparecer uma vez que o hardware possui limitações físicas e tecnológicas.

Outra possível solução é aplicar a técnica de distribuição horizontal (scale out), ou seja, quanto mais dados, se deve aumentar a quantidade de servidores, contudo, não necessariamente computadores com alto poder de processamento. A grande problemática dessa da técnica é que a configuração para se integrar com os demais servidores já existentes no sistema distribuído com banco de dados relacional, podendo tomar o processo extremamente complexo, caro e demorado.

Um dos motivos da dificuldade é que um modelo relacional é muito ligado a um esquema com estrutura pré-definida[3]. É muito conveniente usar esse modelo quando não existem muitas modificações nos registros da estrutura de dados, entretanto, se o cenário estiver ao contrário, e na maioria das vezes está, o processo se torna complexo e muitas vezes inviável.

B. *ACID x BASE*

Como se sabe a técnica de distribuição vertical é limitada, então é necessário usar modelos de infra-estrutura com distribuição horizontal.

Os bancos de dados relacionais a fim de garantir a integridade de dados utilizam o conceito ACID como propriedades em suas estruturas e transações.

As propriedades ACID (*Atomicity*, *Consistency*, *Isolation*, *Durability*) garantem que transações sejam feitas com segurança, garantindo atomicidade de cada uma delas, consistência dos dados, isolação e integridade durante todas as transações.

Entretanto, garantir todos esses atributos pode acarretar em configurações complexas, como já citado anteriormente, e um péssimo desempenho em uma transação ou consulta. Os conflitos surgem entre os diferentes aspectos da necessidade de garantir as propriedades ACID em sistemas distribuídos.

O teorema de CAP (também chamado de Teorema de Brewer's[12]) propõe um modelo mais realístico do que acontece na prática, para entende-lo teremos que entender a fundo o que as propriedades ACID propõem.

Sabemos que as propriedades descritas acima necessitam garantir os seguintes requisitos:

- Todos os nós da rede necessitam ter a mesma versão do arquivo (Consistência).
- Todos os clientes podem sempre encontrar pelo menos uma cópia dos dados solicitados, mesmo que um cluster estiver desligado (Disponibilidade).
- O sistema continua com seus dados, propriedades e características mesmo se for implantado em servidores diferentes, transparente para o cliente (Tolerância a partições).

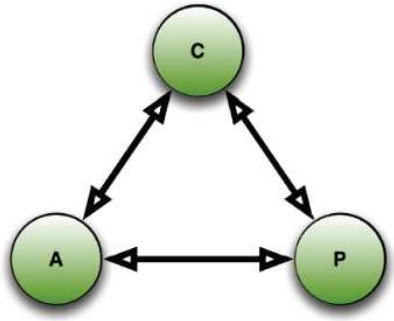

Figura 1 – Ilustrando a interação do teorema de CAP[12]

O teorema de CAP postula que apenas dois desses três itens, citados acima, podem ser alcançados plenamente, ao mesmo tempo[12].

Para que seja possível se trabalhar com um sistema distribuído foi proposto enfraquecer o requisito de consistência de dados, se focando em melhorar a disponibilidade e a tolerância a criar mais partições.

Foi dessa idéia que nasceu o conceito de **BASE** (*Basically Available*, *Soft state*, *Eventual consistency*) com propriedades e filosofia oposta.

Esse novo conceito considera um cenário que provê transações distribuídas, tolerância a falhas de consistência, e replicação otimista em um sistema distribuído.

C. *Banco de dados não relacionais*

A partir do conceito de BASE foram criados diferentes de tipos de bancos de dados não relacionais, que devido à sua filosofia, não garantem consistência dos dados de uma aplicação.

Empresas grandes como Google e a Amazon aproveitam bastante a vantagem de BASE, utilizando um modelo NoSQL, aplicado ao conceito de distribuição horizontal(scale out).

Como não existe garantia de consistência, torna-se mais fácil efetuar configurações de particionamento no modelo e por conseqüência, este, torna-se mais escalável, e eficiente em questão de desempenho (iremos ver detalhes na próxima seção) e de custo reduzido.

III. POR DENTRO DOS TIPOS DE BANCO DE DADOS NÃO RELACIONAIS

A. *Chave-Valor (key/value store)*

O modelo apresenta a proposta que permite que o desenvolvedor efetue a persistência de dados totalmente livre de definições de estrutura para esquema.

Como o nome já diz e sugere, trata-se de uma abordagem parecida com uma tabela hash. A informação do conteúdo é armazenada e um índice em forma de um tipo primitivo de dado é usado para mapeá-lo. O conteúdo pode ser armazenado em qualquer formato, seja uma string, inteiro, matriz, ou objeto.

Pelo fato de ter uma chave de indexação para mapear um valor, a inserção e a recuperação se torna mais fácil e com interface simplificada.

Na maioria dos casos a interface segue dois principais métodos (inserir e recuperar) que seguem o padrão a seguir:

- Put(chave, valor) – método que insere um registro
- Get(chave) – método que recupera um registro

Embora as vantagens de velocidade no armazenamento e recuperação de dados sejam indiscutíveis, é necessário considerar a quantidade de dados que esse modelo gera, uma vez que existe muita redundância e replicação de dados.

Um exemplo real pode se basear em um problema para armazenar as lojas de um shopping Center divididos em diversos tipos de departamentos, teremos muitos dados replicados para cada registro. Por exemplo, para cada registro de uma loja no shopping é necessário replicar o atributo departamento, sendo que em um modelo relacional podemos trabalhar com identificadores e aplicar conceitos normalização no esquema evitando que se repliquem os dados descritivos de uma entidade desnecessariamente.

Esse modelo é um pouco mais consolidado, uma vez que grandes empresas, como a própria Amazon, utilizam em seus sistemas, fornecem serviços, apis e outras ferramentas.

Uma ferramenta disponibilizada para os desenvolvedor é o Amazon's SimpleDB, o qual trata-se de um serviço na web que disponibiliza um banco de dados que usa o modelo de chave/valor com funções de indexamento para arquivos genéricos em uma nuvem.

B. *Documento (Document Store)*

Um banco de dados baseado em documento é, na sua essência, parecido com a abordagem chave/valor com uma grande exceção[9]. Em vez de armazenar qualquer arquivo binário como valor de uma chave é requisitado que o valor dos dados que serão armazenados possua um formato que o banco possa interpretar. Na maioria das vezes podem ser arquivos de extensão xml, json, json-blob ou qualquer formato descritivo, variando muito de ferramenta para ferramenta.

O fato de definir uma estrutura para um arquivo, mesmo que semanticamente váriavel, faz com que banco de dados orientados a documentos sejam um pouco parecidos com banco de dados relacionais[10].

O CouchDB é um dos bancos de dados orientado a documento amplamente usados no mercado, possuindo versões para para web, desktop e mais recentemente para embarcados em plataformas móbile, como o Android e iOS.

A figura 2 mostra um exemplo trivial de um documento, isto é, um registro de uma estrutura de dados criada semanticamente para armazenar uma agenda.

```
{
 type: 'contact'
 first: 'Paulo'
 lastname : 'Abreu'
 email : [ 'home': 'pauloabreu@yahoo.com',
           'work': 'pauloworkholic@work.com' ],
 telephone : [ 'home': '+810001000'],
 address: []
}
```

Figura 2 – Um registro de uma base de dados em CouchDB

C. *Grafo (graph)*

Tipicamente, grafos podem ser definidos como uma abstração matemática que podem ser representados através de vértices e arestas, representando caminhos [6].

No cenário atual de modelo de base de dados baseados em grafos, temos banco de dados relacionais e estruturas de web semântica.

Em um banco de dados relacional é necessário abstrair a normalização da modelagem ao nível máximo para representar um grafo, o que torna o tempo da consulta grande, devido ao número de consultas que será necessário ser feito. Isso acontece especialmente em estruturas recursivas de grafos, como percursos em largura ou em profundidade, pois é necessário guardar a referência a cada interação do percurso.

Entretanto em uma base de dados não relacional onde a semântica é baseada em grafos, pode ser mais eficiente já que teremos uma estrutura respeitando as seguintes regras:

- Um vértice ou aresta possui chave e conteúdo.
- A chave de um vértice descreve o conteúdo do próprio vértice.
- A chave de uma aresta descreve o conteúdo de um relacionamento entre vértices.
- O conteúdo de um vértice possui dados sobre o vértice.
- O conteúdo de uma aresta possui dados referentes a um relacionamento entre dois vértices.

Esse modelo pode tornar consultas rápidas, devido à possibilidade da utilização de propriedades de grafos, medidas de centralidade (pagerank) e algoritmos de menor caminho.

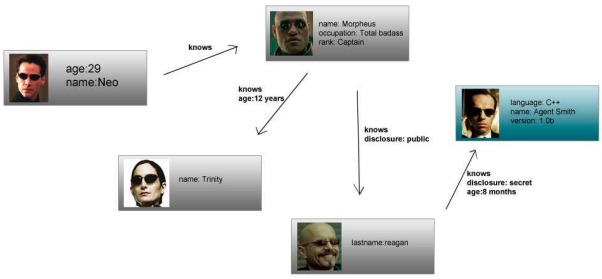

Figura 4 – Exemplo de modelo de base de dados em grafos

É importante observar que o valor presente no conteúdo dos nós não segue um padrão de estrutura de dados, isto é, pode-se armazenar qualquer tipo de dados dentro do conteúdo.

A maioria das ferramentas de base de dados não relacionais baseadas em grafos possuem um alto nível de abstração, tornando a fase de desenvolvimento mais fácil.

Por ser uma abordagem que se baseia em guardar informações relevantes sobre uma determinada relação entre dois nós, pode-se considerar um modelo recomendado para redes sociais e sistemas de recomendação[x].

Indubitavelmente, se trata de um dos conceitos mais complexos, quando falando de banco de dados NoSQL, devido a seguir modelos matemáticos complexos.

Um ponto positivo é não existir tanta replicação de dados como nos outros modelos, fato que acontece por se aproveitar do relacionamento entre os registros.

O gráfico (figura 3) a seguir mostra uma relação entre o tamanho dos registros(y) pela complexidade(x), do tipo de modelo NoSQL.

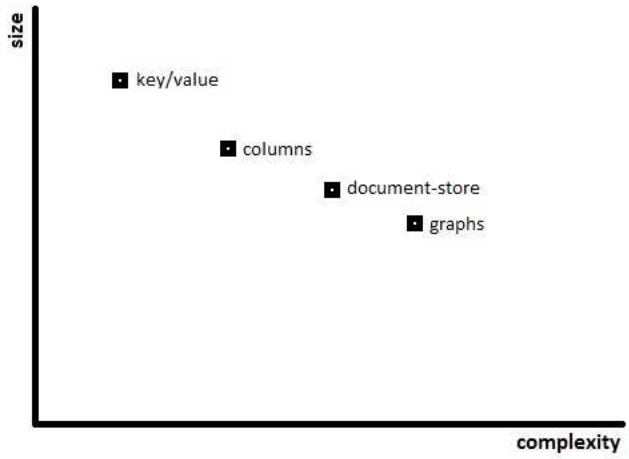

Figura 4 - Relação entre complexidade e tamanho

IV. MITOS DO NOSQL

Muito se diz sobre o NoSQL ser escalonável, possuir um baixo custo com alto grau de abstração, mas quanto isso seria tão inovador para afetar diretamente o cenário atual?

A partir dessas questões são criados mitos e especulações exageradas na maioria das vezes.

A. NoSQL é escalonável

Em todo momento do artigo foi abordado que NoSQL se trata de uma abordagem incrivelmente escalonável, entretanto, dizer que um sistema se auto escanola trivialmente é um sonho. Na verdade é uma questão de referêncial entre dois modelos. O NoSQL é mais fácil de ser escalonado em relação ao modelo de banco de dados relacional.

B. Não precisamos mais de DBA

Se possuimos um modelo que nos proporciona um cenário no qual temos escalabilidade e de alta abstração, então faz sentido dizer que o profissional com o perfil de DBA não seja mais necessário existir. Falso. É claro que precisaremos de profissionais que gerenciem a camada de persistência, entretanto, com a facilidade de configurações e maior abstração dos bancos de dados não relacionais, é necessário que o perfil do profissional se adapte as necessidades.

C. Econômico

Anteriormente foi citado que o custo seria algo a favor de base de dados NoSQL, entretanto, é importante ter cuidado com algumas situações.

O custo em usar um banco de dados relacional pode ser alto devido à escala ou a licenças envolvidas. Em muitos casos a solução relacional atende perfeitamente todas as necessidades do cliente e ainda sim pode ser considerada de menor custo.

Bancos de dados open source como MySQL e PostgreSQL são utilizados, sem problemas, por um grande número de sistemas. É uma questão de analisar o tamanho, licenças e necessidade.

V. AVALIAÇÃO DE DESEMPENHO

Muito se diz sobre desempenho em NoSQL, entretanto, hoje existem poucos benchmarks no mercado efetuando comparações.

Um dos estudos de caso mais consistentes e completos existentes é sobre o serviço Yahoo! Cloud Serving. O estudo de caso consistiu em usar o serviço com tipos de banco de dados distintos.

Inicialmente é necessário criar um modelo de benchmark, escolhendo atributos para ocorrer à comparação métrica entre os modelos de base de dados. Nesse caso, em particular, foi usado o indicador de latência.

O método de benchmark consistiu em avaliar a latência através da carga de trabalho de requisições.

Para isso foi criado uma aplicação em java que tinha essa responsabilidade, isto é, preparar e executar um grande número de transações em um determinado intervalo de tempo[3].

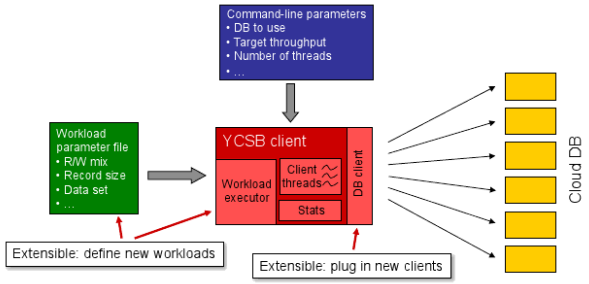

Figura 5 – Mostra o esquema do cliente para efetuar o benchmark. Existem módulos para definir a carga de trabalho (verde), parametros das cargas de trabalhos (azul) e o cliente (vermelho) que une os dois conceitos, fazendo requisições ao banco de dados na nuvem(amarelho).

A. Carga de trabalho

Para definir uma carga de trabalho é necessário escolher uma combinação de transações e parâmetros com astucia, fazendo com que o existam diferentes tipos de consultas que exigam o máximo do banco de dados em diferentes aspectos.

B. Experimento

Nesse experimento foram descritos dois planejamentos de carga de trabalho com mistura de consultas de escritas e leituras, requisições e tamanho de registros. Além disso, o cliente de benchmark foi escrito especialmente para testar uma particular configuração de hardware.

Foram utilizados os bancos de dados Cassandra (NoSQL-Tabular), Hbase(NoSQL-Tabular), Sherpa(NoSQL - Sem categoria definida – usa como base um modelo em arvore B). Nos gráficos a seguir podem ser observados resultados em relação a latência, em diferente tipos de consultas.

1) Fase 1 - Carga de trabalho A

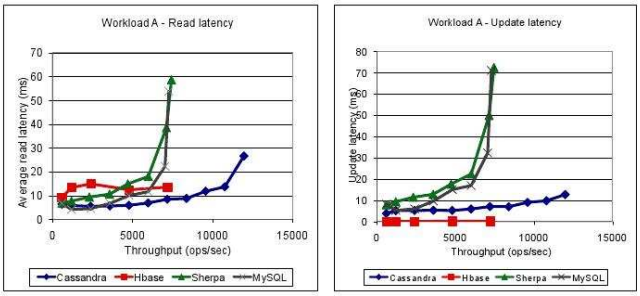

Figura 6 - Comportamento do desempenho de base de dados ao ser executado atráves de uma carga de trabalho.

O banco de dados Cassandra e HBase são otimizados para escrita, e atinge um maior rendimento e menor latência. Entretanto, o Hbase deixa desejar um pouco na leitura, por necessitar realizar a tarefa de reconstruir registros em cachê.

2) Fase 2 - Carga de trabalho B

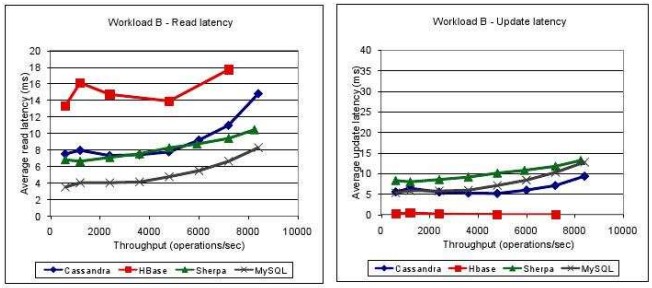

Figura 7 - Comportamento do desempenho de base de dados ao ser executado atráves de uma carga de trabalho.

Novamente o Hbase e o Cassandra possuem melhores resultados na média, entretanto o Hbase possui a mesmo problema na reconstrução dos registros, como já observado anteriormente.

Podemos observar que o Sherpa apresentou um ótimo desempenho na média, já que possui sua estrutura baseada em uma árvore-B, a qual apenas necessita de uma consulta para ler um registro, não precisando reconstruir registros.

C. Resultados

Pode ser observado que nos dois casos o banco de dados relacional (MySQL) tem performance constante nas duas fases do experimento e na maioria das vezes maior latência, em relação aos banco de dados NoSQL, principalmente após a leitura.

Podemos concluir que para esse benchmark a abordagem NoSQL possui, na maioria dos casos, maior performance que uma base de dados relacional.

VI. CONCLUSÃO

Este artigo demonstrou que o NoSQL pode ser uma alternativa real para banco de dados que necessitam de uma alta escalabilidade, desempenho e que podem possuir tolerância a falhas.

É importante perceber que o modelo proposto, não é uma abordagem ideal que pode substituir todos os tipos de modelos de persistência, mas sim uma opção para as aplicações que possuem o cenário descrito no artigo.

Nos dias de hoje podemos estabelecer um modelo ideal como um sistema de banco de dados hibrido, usando conceitos de diferentes abordagens sob demanda.

Em uma aplicação, quando existe uma parte do sistema que necessita de consistência em seus dados, então é recomendado que se use uma base de dados relacional, e quando necessitar de desempenho em um módulo, então usa-se o NoSQL. Um exemplo real de modelo de dados hibrido foi implementado pela empresa boo-box.

O modelo de negócios da compania é baseado em propaganda integrada com sistemas de recomendação buscando se aproximar dos interesses do usuário, fazendo uma

recomendação otimizada, baseada em conteúdo acessado. Contudo, para buscar promoções de lojas, mercados, marcas e links, que satisfaçam os interesses do usuário, é necessário o acesso as apis dos sistemas dos respectivos estabelecimentos, com o intuito de buscar informações dos produtos e marcas.

Se o modelo de dados fosse inteiramente baseado em um modelo relacional, seria necessário efetuar o acesso a API a cada vez que o usuário acessar uma página diferente e salvar na base de dados, tornando-se muitas vezes um processo demorado. Já com o modelo hibrido propõe-se uma solução com maior desempenho, devido à estrutura, a qual consiste em guardar as requisições feitas temporariamente, como forma de cache, diminuindo assim o acesso as apis externas.

REFERÊNCIAS

[1] Malcolm Davis – “Scale Up vs Scale Out”. Available: http://weblogs.java.net/blog/2006/07/19/scale-vs-scale-out

[2] N.Leavitt, “Will NoSQL Databases Live Up to Their Promise?”, 2010. Available: http://www.leavcom.com/pdf/NoSQL.pdf

[3] A. Popescu, “NoSQL benchmarks and performance evaluations”, 2010. Available:http://nosql.mypopescu.com/post/734816227/nosql-benchmarks-and-performance-evaluations

[4] R. Rees, “NoSQL, no problem - An introduction to NoSQL databases”, Available: http://www.thoughtworks.com/articles/nosql-comparison

[5] C. Monash, “NoSQL Basics, Benefits and Best-Fit Scenarios”, 2010. Available: http://www.informationweek.com/news/software/info_management/showArticle.jhtml?articleID=227701021

[6] P. Neubauer, “Graph Databases, NOSQL and Neo4j”, 2010. Available: http://www.infoq.com/articles/graph-nosql-neo4j

[7] E. Ferreira, “Introdução ao NoSQL parte I”, 2010. Available: http://escalabilidade.com/2010/03/08/introducao-ao-nosql-parte-i/

[8] J. Nascimento, “NoSQL - você realmente sabe do que estamos falando?”, 2010. Available: http://imasters.com.br/artigo/17043/bancodedados/nosql_voce_realmente_sabe_do_que_estamos_falando/

[9] A. Rahien, “That No SQL Thing – Document Databases”, 2010. Available: http://ayende.com/blog/archive/2010/04/11/that-no-sql-thing-ndash-document-databases.aspx

[10] “noSQL (Document-based) or Distributed Cache - a Practitioner's tale.”, 2011. Available: http://javamuse.blogspot.com/2011/03/nosql-document-based-or-distributed.html

[11] C. Curino, E. P. C. Jones, R. A. Popa, N. Malviya, E. Wu, S. Madden, H. Balakrishnan, N. Zeldovich, “Relational Cloud: A Database-as-a-Service for the Cloud”, 2011. Available: http://www.cidrdb.org/cidr2011/Papers/CIDR11_Paper33.pdf

[12] J.Browne, “Brewer's CAP Theorem”, 2009. Available:http://www.julianbrowne.com/article/viewer/brewers-cap-theorem