UM ESTUDO EXPLORATÓRIO ACERCA DE BANCO DE DADOS NOSQL COMPARADO AOS BANCOS DE DADOS RELACIONAIS

Pedro Henrique Ravagnani Pintar

xpehen@outlook.com

ABSTRACT: with the emergence of large amount of data and the need for handling them, the constant technological advances showed the need to solve this problem , so that there was the emergence of NoSQL term . The main objective of this work is the comparison between the relational database system, most commonly using for data storage and nosql database system, which has emerged as an alternative solution to manage large amounts of data, based in their concepts, handling and storage of data. As a result it was noted that the different modeling and structuring of data, resulted in performance and storage thereof, especially regarding the great mass of data generated.
RESUMO: com o surgimento de grande quantidade de dados e a necessidade de manipulação dos mesmos, o constante avanço tecnológico apresentou a necessidade de solucionar essa problemática, fazendo com que houvesse o surgimento do termo NoSQL. O principal objetivo deste trabalho é realizar a comparação entre o sistema de banco de dados relacional, mais comumente utilizando para armazenamento de dados e o sistema de banco de dados nosql, que surgiu como uma solução alternativa para gerenciar grandes quantidades de dados, baseando-se em seus conceitos, manipulação e armazenamento dos dados. Como resultado notou-se que as diferentes modelagens e estruturação dos dados, acarretaram no desempenho e no armazenamento dos mesmos, principalmente com relação a grande massa de dados gerados nos dias de hoje.

PALAVRAS-CHAVE: Banco; Dados; NoSQL; Relacional; ACID.

1. Introdução

Sem dúvida alguma, o crescimento da quantidade de dados e informação na Web, gerados em sua maioria pelas redes sociais, é indiscutível e perceptível nos dias atuais. Esse fator, bem como a necessidade de suporte a tipo de dados complexos, semiestruturados e não estruturados, influenciaram as pesquisas sobre modelos de bancos de dados que suportem tais fatores.

Entretanto, se as aplicações possuem um grande volume de dados, com diferentes formatos, é possível que se tenha grandes problemas em relação ao armazenamento, processamento e manipulação dos mesmos.

Atualmente, a abordagem de banco de dados mais adotada é a Relacional, que garantem as Propriedades ACID (Atomicity, Consistency, Isolation e Durability) dando total suporte ao armazenamento, processamento e manipulação de "dados estruturados", mas que deixam a desejar quando se trata dos fatores acima mencionados.

Com o surgimento dos Bancos de Dados Objeto-Relacional, que é um modelo híbrido, passou a ser possível o suporte a tipos complexos de dados, porém, mesmo assim, esses modelos não atendem as necessidades atuais, da necessidade de manipular grandes volumes de dados, principalmente quando se trata de dados semiestruturados e não estruturados. Os Bancos de Dados NoSQL (Not Only SQL) surgiram para resolver essa problemática, mostrando uma abordagem diferente de persistência de dados, baseada no paradigma BASE (Basically Available, Soft-state or Scalable, Eventually Consistency), disponibilidade, desempenho, escalabilidade e consistência 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, de acordo com quatro categorias, orientado a coluna, orientado a documentos, orientado a grafos e orientado a armazéns chave-valor.

Com base nesse breve panorama, essa pesquisa tem por objetivo fazer um estudo acerca de Bancos de Dados NoSQL no sentido de apresentar um comparativo entre essa tecnologia e a Abordagem de Banco de Dados Relacionais. Pretende-se nesse comparativo mostrar as novas soluções de bancos de dados desenvolvidas para atender as novas necessidades e tendências do mercado, tais como armazenamento e manipulação de Dados Não Estruturados, bem como ao crescente volume de dados

gerados diariamente, necessidades que os Modelos Relacionais não dão suportam totalmente.

Sendo assim foi realizado um estudo acerca dessa tecnologia, explorando as principais categorias de bancos de dados NoSQL existentes no mercado, bem como os fatores que culminaram na criação ou no desenvolvimento desses tipos de bancos de dados.

Apresentando exemplos práticos de Sistema de Banco de Dados NoSQL, que fazem uso do paradigma BASE, realizando estudos de casos, bem como traçar um comparativo com os bancos de dados elaborados com base na Abordagem Relacional, que fazem uso total das Propriedades ACID.

Esse tema é relevante, e justifica-se o mesmo, já que é uma tecnologia mundialmente em discussão e que deverá proporcionar soluções práticas e essenciais para empresas que necessitam armazenar grandes volumes de dados, sendo boa parte dessas semiestruturas e não estruturados.

2. Banco de Dados Relacional

Sendo comumente utilizado pela grande parte das aplicações nos dias de hoje, o modelo relacional surgiu para resolver à problemática quanto à manipulação e armazenamento de dados, para a essa finalidade ela se apoia totalmente a álgebra relacional para exercer diversas funções.

A estrutura fundamental do banco de dados relacional é a relação, também conhecida como tabela, onde os dados são organizados por um ou diversos atributos que fazem uso da metainformação para traduzir o tipo de dado que será armazenado, esse conjunto de dados por sua vez são organizados em registros (linhas).

Dessa maneira o modelo relacional não mais possui um caminho pré-definido para que se realize acesso aos dados como os modelos que os precederam.

Essa estrutura de relação, entretanto faz com que haja a necessidade de que algumas restrições sejam impostas para evitar aspectos indesejáveis, como a repetição de informações, perda de informação e a incapacidade de apresentar a informação completa.

Outra grande característica importante para o modelo é a utilização de restrições de integridade que fazem o uso de chaves primárias e chaves estrangeiras a fim de garantir a integridade dos dados. A chave primária que permite a identificação única de um

registro dentro de uma relação e aumento no desempenho de consultas, e a chave estrangeira, que por sua vez permite com que o usuário realize relações entre tabelas distintas.

Através disso, os sistemas gerenciadores de bancos de dados relacionais trouxeram a liberdade para que o usuário não se preocupe mais com a garantia e integridade de seus dados, controle e coerência, recuperação de falhas e segurança dos dados, fazendo com que todo o trabalho seja realizado e gerenciado pelos SGBDs.

Para que os itens acima mencionados sejam fielmente seguidos é preciso que se mantenham as propriedades conhecidas pelo acrônimo de ACID (Atomicity, Consistency, Isolation, Durability), propriedades que iremos detalhar futuramente.

Dessa maneira, em um cenário onde o número de dados cresce constantemente, como por exemplo, uma aplicação Web de uma grande empresa, é perceptível a dificuldade na organização dos dados utilizando o modelo relacional em sistemas que trabalham com o particionamento desses dados, sendo necessário fazer com o banco de dados relacional se torne cada vez mais escalável, de maneira que consiga suportar o constante crescimento de dados gerados. Uma das possíveis soluções para esse caso é uma técnica denominada escalonamento vertical, onde seria feito uma atualização dos servidores. Entretanto em algum momento, os dados gerados seriam cada vez maiores, fazendo com que seja necessária uma nova atualização, outra solução seria a distribuição do banco em diversos servidores, utilizando de uma técnica denominada escalonamento horizontal, onde novas máquinas são adicionadas para que assim seja realizada a distribuição de dados, situação na qual os bancos de dados NoSQL, podem intervir de maneira mais eficiente.

A questão é que os bancos de dados relacionais não são tecnologias que possuem a finalidade de funcionar com particionamento de dados, por esse fato não são eficientes quando se trata de trabalhar de maneira distribuída.

Assim, as limitações apresentadas pelo modelo relacional fez com que em busca por resposta a essas problemáticas, houvesse um foco maior em soluções não relacionais, como, por exemplo, os banco de dados NoSQL.

3. Banco de Dados NoSQL

Com o surgimento dos Bancos de Dados Objeto-Relacional, que é um modelo híbrido, passou a ser possível o suporte a tipos complexos de dados, porém, mesmo assim, esses

modelos não atendem as necessidades atuais, da necessidade de manipular grandes volumes de dados, principalmente quando se trata de dados semiestruturados e não estruturados.

O modelo de dados NoSQL, surgiu como uma solução alternativa ao uso do modelo relacional, aonde diversos projetistas de banco de dados começaram a desenvolver novas estratégias de desenvolvimento com o objetivo de flexibilizar as estruturas e regras, uma vez que a estrutura do banco de dados relacional fosse pouco flexível.

Inicialmente as implementações do banco de dados NoSQL surgiu em grandes empresas como Amazon, que utilizava do banco de dados não relacional Dynamo. Mas acima de tudo a maioria dos bancos de dados NoSQL atuais, segundo Leavitt(2010), são open source, ou seja, possuem seu código aberto.

Bancos de Dados NoSQL consistem em Sistemas Gerenciadores de Bancos de Dados Não-Relacionais projetados para gerenciar grandes volumes de dados e que disponibilizam estruturas e interfaces de acesso simples [Sousa et al. 2010].

Segundo Cattell (2010), os Bancos de Dados NoSQL proporcionam um grande número de operações de leitura e escrita por segundo, característica comum em aplicações Web modernas.

Conforme McMurtry (et al. 2013), o suporte a tipos de dados complexos, semiestruturados ou não estruturados também favorece o uso destes Banco de Dados, que atualmente são categorizados em chave-valor, documento, família de colunas e baseado em grafos. Sendo assim diferentes do modelo relacional que só possuem um único modelo para armazenar seus dados, os banco de dados NoSQL possuem diversos.

O sistema chave-valor (key-value) é considerado simples e permite que nós acessemos os dados através de uma tabela de hash, na qual existe uma chave e um indicador para determinado dado. É caracterizado pela facilidade ao ser implementado, permitindo que os dados sejam acessados rapidamente, aumentando assim a disponibilidade de acesso aos dados. Um problema enfrentado por esse tipo de banco de dados é que o mesmo não permite a recuperação de objetos utilizando-se de consultas mais complexas. Um exemplo de banco de dado que utiliza esse modelo chave-valor é o Dynamo que foi desenvolvido pela Amazon.

Também existe o modelo orientado a documentos, este por sua vez armazena coleções e documentos. Diferente do banco de dados chave-valor onde se cria uma única tabela hash, neste modelo temos um agrupado de documentos, sendo que estes são por sua vez

um conjunto de campos e seus respectivos valores. Uma grande vantagem do modelo é a ausência de esquemas pré-definidos (schema free), fazendo com que seja possível realizar alterações no documento sem que outros documentos sejam afetados, outra característica importante é que não é necessário armazenar valores de dados vazios para campos que não possuem um valor. Temos como exemplo de sistema de banco de dados que utilizam esse modelo o CouchDB e o MongoDB.

Demonstrando maior complexidade que o de chave-valor, o modelo orientado a coluna (column Family) foi criado para armazenar e processar grandes quantidades de dados, sendo estes distribuídos em diversas máquinas. As colunas são organizadas por famílias de colunas, onde o objetivo é reunir colunas que armazenam o mesmo tipo de informação, aqui também existem chaves, mas essas apontam para atributos ou colunas múltiplas. Sendo assim, uma determinada chave apontará para diversas colunas e diversos dados pertencentes a essas colunas, fazendo com que a informação seja apresentada corretamente, isso se faz necessário uma vez que as escritas e leituras são atômicas, ou seja, os valores associados a uma linha são considerados no tempo de sua execução. Para esse modelo podemos citar o BigTable criado pela Google e o Cassandra, utilizando e desenvolvido pelo Facebook.

O modelo orientado a grafos possui um alto desempenho, ele possui três componentes, os nós, os relacionamentos e as propriedades. Dessa maneira cada par de nós poderá ser conectado por diversas arestas, tornando a utilização do modelo muito útil quando se é necessário realizar consultas complexas. Como exemplos podem ser citados o Infinite Graph e o Neoj4.

É possível ver entre os modelos características em comum, onde todos possuem maior escalabilidade em relação aos bancos de dados relacionais, alta disponibilidade e são livres de esquema. Apesar de a maioria das soluções serem distribuídas, existem modelos que promovem o particionamento e a replicação dos dados, como o MongoDB e o CouchDB, já outros podem deixar essa tarefa para o cliente, como é o caso do Amazon SimpleDB.

A consistência de um banco de dados NoSQL é considerada eventual, uma vez que priorizam a disponibilidade do sistema em detrimento da consistência, prioridades que não são possíveis de serem obtidas de forma absoluta e simultânea com a tolerância ao particionamento. Dessa forma não se pode garantir que a escrita de um dado gere leituras atualizadas do mesmo por outros processos simultâneos.

Segundo Orend (2010), a consistência eventual não garante que dois ou mais processos enxerguem a mesma versão de um determinado item de dado ao mesmo tempo, comportamento esse que é normalmente causado pela replicação de dados em diversos nós.

Sendo assim o conceito de consistência eventual faz com que o banco de dados NoSQL tenha uma abordagem diferente de persistência de dados, baseando-se no paradigma BASE (Basically Available, Soft-state or Scalable, Eventually Consistency), disponibilidade, desempenho, escalabilidade e consistência dos dados. Utilizando-se do paradigma BASE o sistema poderá ficar, em um determinado momento, em um estado inconsistente, fazendo com que nem todos os usuários consigam visualizar os dados em sua última versão.

4. Propriedades ACID

ACID é o acrônimo para atomicidade, consistência, isolamento e durabilidade, sendo garantida atualmente pelos bancos de dados relacionais, onde tem por finalidade preserva a integridade dos dados, dando total suporte ao armazenamento, processamento e manipulação de dados considerados “estruturados”.

A atomicidade garante que na transação, ou seja, realizada todas as operações, ou nenhuma operação seja realizada, não existindo assim um meio termo, dessa maneira ela garantira que não existam inconformidades com relação aos dados.

Consistência garante que os dados antes de qualquer operação estejam consistentes, e que após a transação também, sem que existam futuros problemas com a integridade dos dados.

Para fazer com que nenhuma operação interfira à outra, o isolamento faz com que determinadas operação não possam ser realizadas simultaneamente, como a alteração de um mesmo dado em um mesmo momento, entretanto existem operações como as consultas que poderão ser realizadas.

A durabilidade garante que um dado gravado ficará mantido no banco de dados até que o mesmo seja excluído ou alterado. Dessa maneira garantido que os dados durem de forma imutáveis até que uma transação de exclusão ou alteração o afete.

Dessa maneira, sendo feito o uso dessas características podemos garantir à consistência, integridade e durabilidade dos dados.

Existem diversas aplicações que dependem do total cumprimento do ACID, uma vez que não possa existir inconsistência de dados, podemos citar como um exemplo as transações realizadas por bancos, onde a conta de origem mostra uma retirada e a conta de destino deverá mostrar um depósito, de maneira que os dois processos deverão obter êxito em sua execução para que se possa validar a operação.

Entretanto o acrônimo ACID não poderá ser aproveitado de maneira eximia pelos bancos de dados NoSQL, sendo livre dessa afirmação algumas poucas exceções. O uso da consistência eventual faz-se necessário a utilização de uma abordagem alternativa ao ACID comumente conhecida como BASE.

Dessa maneira as características do BASE garantira que sendo basicamente disponível o sistema permitira que todos os usuários possam realizar consultas, fazendo assim com que não exista isolamento de dados. Por sua vez, a eventual consistência faz com que os valores armazenados no sistema possam ser alterados a qualquer momento, e uma vez que um dado seja gravado o sistema permanecerá replicando o mesmo para todos os demais nós, criando um curto período de tempo em que esses dados não estarão presentam em todos os nós, acarretando em uma inconsistência no banco, dessa maneira faz-se o uso da consistência eventual.

O teorema CAP de Eric Brewer’s afirma que em sistema de redes distribuídas é impossível com que consigamos manter a consistência, disponibilidade e a tolerância a particionamento simultaneamente, sendo possível atender a apenas duas das três características mencionadas.

Os sistemas que utilizam as tecnologias relacionais tradicionais normalmente não possuem tolerância ao particionamento, para que eles possam garantir a consistência e disponibilidade. Em suma, se uma parte desses sistemas tecnologias relacionais tradicionais está off-line, todo o sistema está off-line.

Sistemas onde a tolerância a particionamento e disponibilidade é de primordial importância não se pode garantir a consistência, uma vez que atualizações possam ser realizadas em ambos os lados da partição, essa estratégia é bem explorada pela tecnologia NoSQL. Exemplos de modelos de dados que utilizam essa estratégia são os modelos chave-valor e família de colunas.

5. Proposta de Trabalho

O presente artigo apresentará uma comparativa entre o modelo de banco de dados relacional e o modelo de banco de dados NoSQL, uma vez que são tecnologias mundialmente em discussão e que deverá proporcionar soluções práticas e essenciais para empresas que necessitam utilizar da mesma. Assim sendo, foram traçadas as principais diferenças entre as características acima apresentadas.

Para que possamos realizar uma comparativa entre duas tecnologia se faz necessário levar algumas questões em consideração. Escalonamento, consistência, disponibilidade dos dados e tolerância ao particionamento serão as características utilizadas, de maneira que são características essenciais para a estruturação de um banco de dados.

6. Estudo de Caso

A escalabilidade é uma característica importante para se abordar no aspecto comparativo entre banco de dados. Com relação à escalabilidade os banco de dados NoSQL possuem uma grande vantagem em relação aos sistemas gerenciadores de banco de dados relacionais tradicionais, isso ocorre devido ao fato de serem livre de esquemas, devido ao fato de que foram criados com essa finalidade, fazendo com que se tornem flexíveis. Os bancos de dados relacionais possuem uma estrutura não tão flexível e menos adaptada para a situação que faz com que exista certa dificuldade para lidar com situações onde o escalonamento é necessário, uma vez que o mesmo faz o uso da estratégia de escalonamento conhecida como escalabilidade vertical (scale up), aonde se é necessário realizar uma melhoria no servidor.

A partir do momento que a aplicação está sendo demasiadamente acessado por um grande numero de usuários, esse tipo de escalonamento passa a não ser o mais eficiente devido ao seu custeamento, fazendo com que seja necessário escalonar o próprio banco de dados, que consiste basicamente em distribuir o banco em diversas máquinas, particionando os dados e garantindo que a queda de um determinado servidor não acarrete na falta de disponibilidade dos dados, o escalonamento vertical ou Sharding como é comumente conhecido, mostra-se muito complexo para ser implementado quando se faz o uso de um banco de dados relacional, sendo que o modelo relacional obedece aos critérios de normalização de dados e o sharding vai contra isso, pois se caracteriza pela desnormalização dos dados.

Assim sendo o banco de dados NoSQL poderá usufruir de conjunto de dados menores, uma vez que o volume de dados é minimizados devido a distribuição, tornando mais simples com que estes sejam gerenciados, acessados e atualizados.

Em questão a disponibilidade a tecnologia NoSQL se destaca, possuindo maior rapidez nas consultas, paralelismo de atualização de dados. Os banco de dados NoSQL, foram projetados para este fim, e da forma mais simples possível. Dessa forma possui vantagem, pois apresenta uma estrutura capaz de manter o sistema indisponível o menor tempo possível.

Os bancos de dados relacionais não conseguem trabalhar de forma eficiente com a distribuição de dados, o que faz com que não suporte uma demanda muito grande de informações.

No quesito consistência o modelo relacional se mostra forte. Suas regras de consistência são bastante rigorosas no que diz a respeito à consistência de informações, possibilitando uma maior rigidez. A consistência é o ponto mais forte desse modelo.

O banco de Dados NoSQL possui um caráter de consistência eventual, o que não garante que uma determinada atualização, em um dado momento, seja percebida por todos os nós. Assim sendo o banco de dados relacional se sai superior, pois o mesmo mantem sempre a consistência de seus dados.

No que diz a respeito a paradigmas utilizados entre as tecnologias, temos o ACID que força a consistência ao final de cada operação, já o paradigma BASE utilizado pelos bancos de dados NoSQL, permitem que o banco seja eventualmente consistente.

7. Considerações Finais

Devido ao grande crescimento do volume de dados, fato imprescindível em algumas aplicações web, os banco de dados Relacionais apresentaram uma solução mais complicada do que os banco de dados NoSQL, que mostraram-se uma grande alternativa quando nos referimos a escalabilidade e disponibilidade, uma vez que precisemos que o sistema sempre esteja disponível aos usuários.

É fato que a os bancos de dados relacionais já estão no mercado há mais tempo, e que são utilizados em larga escala, também sabemos da solidez de suas soluções. Enquanto isso os banco de dados NoSQL ainda vem conquistando seu espaço no mercado e definindo seus padrões, de forma que a sua utilização ainda não é tão popularizada no

mercado. Além de que existem aplicações que necessitam da consistência de seus dados, tornando esse fator primordial, assim sendo os sistemas gerenciadores de banco de dados se tornariam uma melhor alternativa para lidar com essa problemática.

Dessa maneira foi apresentado que existem diversas situações onde as duas abordagens poderiam ser mais eficiente que a outra, não abrangendo todos os problemas, uma vez que um modelo poderia atender melhor ao problema proposto.

O artigo teve como finalidade apresentada as principais características do sistema de banco relacional e o sistema de banco NoSql, assim como realizar uma análise comparativa entre as duas tecnologias com base nas características mais comuns de um banco de dados, sendo elas o escalonamento, consistência e disponibilidade, sabendo que essas disputam e completam o mercado.

Um futuro trabalho poderia ser realizando tendo como medida duas aplicações que fazem uso das tecnologias mencionadas, realizando uma análise comparativa da eficiência dessas tecnologias sobre diversas situações.

REFERÊNCIAS BIBLIOGRAFICAS

CATTELL, R. Scalable SQL and NoSQL Data Stores. SIGMOD Record, 39(4):12–27, 2010.

E. F. Codd, “A Relational Model of Data for Large Shared Data Banks”, Communications of the ACM, Volume 13, nº 6, Junho de 1970, p. 377387.

GOLDMAN, Alfredo; KON, Fabio; PEREIRA, Francisco Jr.; POLATO, Ivanilton; PEREIRA, Rosangela de Fátima. Apache Hadoop: conceitos teóricos e práticos, evolução e novas possibilidades. In: SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 2012, Curitiba. Anais do XXXII Congresso da Sociedade Brasileira de Computação, julho, 2012.

LIMA, Claudio de. MELLO, Ronaldo S. 2015. Um Estudo sobre Modelagem Lógica para Bancos de Dados NoSQL. Departamento de Informática e Estatística – Universidade Federal de Santa Catarina, Universidade Federal de Santa Catarian. Santa Catarina, Paraná, 2015.

MARCÁRIO, Carla Geovanna do N.; BALDO, Stefano Monteiro; 2005. O Modelo Relacional. Instituto de Computação – Universidade Estadual de Campinas. Campinas, São Paulo, 2005.

PEREIRA, Felipe S.; BORGES, Hermes P.; RUBENS, Helio; SANTANA, Sonia A. 2013. Utilização de Banco de Dados NoSql em Ambientes Corporativos. Unitri – Centro Universitário do Triângulo. Uberçândia, Mato Grosso, 2013.

POLITOWSKI, Cristiano; MARAN, Vinicius; 2014. Comparação de Performance entre PostgreSQL e MongoDB. Departamento de Ciências Exatas e Engenharias – Universidade Regional do Noroeste do Estado do Rio Grande do Sul. Santa Rosa, Rio Grande do Sul, 2014.

SCHREINER, Geomar A.; DUARTE, Denio; MELLO, Ronaldo dos Santos. 2015. Análise de Abordagens para Interoperabilidade entre Bancos de Dados Relacionais e Banco de Dados NoSQL. Departamento de Informática e Estatística – Universidade Federal de Santa Catarina, Universidade Federal da Fronteira Sul – Campus Chapecó, Santa Catarina, Paraná, 2015.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. Sistema de Banco de Dados. 6. ed. Tradução de Marília Guimarães Pinheiro e Cláudio César Canhette. São Paulo: Makron Books, 2012.

SILVA, Tiago Pasqualini. 2011. Cassandra – Um sistema de Armazenamento NoSQL Altamente Escalável. Universidade Federal de São Carlos – Campus Sorocaba. Sorocaba, São Paulo, 2011.

SOUSA, F. R. C.; MOREIRA, L. O.; MACÊDO, J. A.; MACHADO, J. C. Gerenciamento de Dados em Nuvem: Conceitos, Sistemas e Desafios. In: SBBD 2010. p. 101–130.

TOTH, Renato Molina. Abordagem NoSQL – uma real alternativa. São Paulo: Universidade Federal de São Carlos – Campus Sorocaba, São Paulo, 2011.

VIEIRA, Marcos Rodrigues; FIGUEIREDO, Josiel Maimone; LIBERATTI, Gustavo; VIEBRANTZ, Alvaro Fellipe Mendes. Bancos de Dados NoSQL: Conceitos, Ferramentas, Linguagens e Estudos de Casos no Contexto de Big Data. In: SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 2012, Curitiba. Anais do XXXII Congresso da Sociedade Brasileira de Computação, julho, 2012.

KAUR, K.; RANI, R. (2013) “Modeling and Querying Data in NoSQL Databases”. IEEE. In: International Conference on Big Data, 2013. p.1-7.

WHITE, Tom. Hadoop: The Definitive Guide. 2 ed. Cambridge: O’Reilly, 2010.