Mapeamentos conceituais entre os modelos Relacional e NoSQL: uma abordagem comparativa

Myller Claudino [1], Damires Souza [2], Ana Carolina Salgado [3]

[1] mcf@cin.ufpe.br. Universidade Federal de Pernambuco – Centro de Informática – Recife – PE. [2] damires@ifpb.edu.br. Instituto Federal de Educação, Ciência e Tecnologia da Paraíba – Unidade Acadêmica de Informática. [3] acs@cin.ufpe.br. Universidade Federal de Pernambuco – Centro de Informática.

RESUMO

As atuais perspectivas computacionais, vindas sobretudo da Web, têm gerado novas demandas relacionadas ao gerenciamento de dados, principalmente em termos de volume, heterogeneidade e dinamismo. Uma tendência atual para facilitar o gerenciamento de dados na Web é a utilização dos denominados Sistemas NoSQL, que se diferenciam dos sistemas que seguem o Modelo Relacional por possibilitarem a implementação de estruturas mais flexíveis. Contudo, a maioria dos bancos de dados de aplicações existentes encontra-se em estruturas relacionais, e a migração de uma base que segue o Modelo Relacional para uma NoSQL requer grande esforço dos projetistas diante das diferenças existentes. Nesse panorama, este artigo descreve os modelos citados, em termos de conceitos e estruturas, e apresenta um estudo comparativo apontando possíveis mapeamentos conceituais entre eles. Aborda também, de forma comparativa, trabalhos de conversão de dados existentes, e indica desafios e possibilidades para novas pesquisas sobre o tema.

Palavras-chave: Modelo relacional. NoSQL. Mapeamento conceitual. Conversão de dados.

ABSTRACT

Current computational perspectives, especially from the Web, have generated new challenges on data management, particularly in terms of volume, heterogeneity and dynamicity. A recent trend to deal with data management on the Web regards the use of NoSQL systems, which differ from Relational Database Systems by allowing the implementation of more flexible data structures. However, most existing application databases still lie in Relational structures, and the migration of a Relational database to NoSQL ones requires great effort on providing the necessary mappings. In this scenario, this paper describes the mentioned data models, in terms of their concepts and structures, and presents a comparative study indicating possible conceptual mappings between them. In addition, it addresses a comparative study on existing data conversion works, and indicates challenges and possibilities for further research on the topic.

Keywords: *Relational Model. NoSQL. Conceptual mapping. Data conversion.*

1 Introdução

As crescentes necessidades de gerenciamento de dados, especialmente na Web, têm incentivado a utilização de sistemas NoSQL (CURÉ *et al.*, 2012). Sistemas, bancos de dados ou abordagens NoSQL são alusões a uma nova gama de sistemas de bancos de dados que não seguem o Modelo Relacional[1] (HAN *et al.*, 2011; HECHT; JABLONSKI, 2011). O termo NoSQL foi primeiramente usado por Strozzi (1998) em um debate, referindo-se a um sistema de banco de dados relacional que não utilizava a linguagem SQL, mas, com o passar dos anos, passou a fazer referência aos sistemas de uma nova categoria de banco de dados.

A demanda pelo uso de sistemas NoSQL ocorre principalmente quando as aplicações necessitam manter serviços com alto desempenho sobre um grande volume de dados (MPINDA; BUNGAMA; MASCHIETTO, 2015). Isso é comum em aplicações que possuem dados distribuídos em várias partes do mundo, que são acessados por centenas de milhares de pessoas simultaneamente. Nesse cenário, as equipes de desenvolvimento, muitas vezes, optam por criar projetos de bancos de dados já considerando sistemas NoSQL ou, caso o banco já exista (em geral, em um Sistema Relacional), realizam sua conversão.

No entanto, em se tratando da necessidade de conversão dos dados de um banco Relacional para um NoSQL, o processo pode gerar um custo elevado para os desenvolvedores, tanto em termos de aprendizado quanto de tempo e esforço para a implementação. Um exemplo dessa necessidade ocorreu com o sistema do Netflix[2]. No ano de 2012, o serviço de *streaming* de vídeo passou a funcionar com cerca de 290 servidores distribuídos, com mais de um milhão de operações por segundo. Isso foi possível depois que o banco de dados do Netflix, que utilizava o Sistema Relacional MySQL[3], passou a utilizar os sistemas NoSQL Cassandra[4] e SimpleDB[5] (IZRAILEVSKY; TSEITLIN, 2011), após sua conversão.

O processo de converter um banco de dados originalmente projetado segundo o Modelo Relacional para um sistema que atende a um modelo NoSQL requer a análise de fatores como tipos de estruturas, tipos de dados, relacionamentos e restrições, que são determinados pelo modelo de dados original e destino. Esse quadro é definido como um problema de heterogeneidade estrutural e é o foco deste artigo.

Tendo em vista esse escopo, neste artigo não serão consideradas questões relacionadas com o desempenho dos sistemas. Quanto a esse aspecto, existem trabalhos que apresentam comparativos de desempenho entre os sistemas de bancos de dados relacionais e abordagens NoSQL, como os trabalhos de Tudorica e Bucur (2011) e de Martins Filho (2015).

Nesse panorama, este artigo introduz conceitos associados aos modelos de bancos de dados em questão, indica os principais problemas relativos à conversão de bancos relacionais para sistemas NoSQL e aborda possibilidades para essa tarefa ser realizada por meio de mapeamentos conceituais. Além disso, apresenta trabalhos existentes, comparando-os de acordo com alguns critérios estabelecidos. Por fim, indica desafios e possibilidades associados à problemática discutida.

O artigo está estruturado da seguinte forma: a Seção 2 realiza uma introdução aos conceitos pertinentes aos modelos citados; a Seção 3 apresenta um cenário exemplo de projeto de banco de dados; a Seção 4 mostra um comparativo entre o Modelo Relacional e os modelos NoSQL em termos de suas estruturas de dados; a Seção 5 apresenta trabalhos relacionados com a conversão de dados entre os modelos; a Seção 6 indica desafios da área e sugere algumas possibilidades para novas pesquisas; por fim, a Seção 7 tece algumas considerações finais.

2 Principais conceitos

Esta seção introduz alguns conceitos relevantes associados à representação de dados nos modelos Relacional e, em especial, nos modelos NoSQL.

2.1 Projeto de banco de dados relacional

O modelo Entidade Relacionamento (ER) é normalmente usado no projeto conceitual de um banco de dados (CHEN, 1976). Embora o projeto conceitual seja independente de considerações físicas, ele fornece a descrição do banco em alto nível, o que ajuda na sua compreensão e, consequentemente, na sua implementação. Composto por entidades, relacionamentos, atributos (simples, compostos, multivalorados), especializações e outros (CHEN, 1976), ele é convertido para um conjunto de tabelas quando mapeado para um banco de dados Relacional.

1 http://nosql-database.org/m
2 www.netflix.com
3 www.mysql.com
4 cassandra.apache.org
5 aws.amazon.com/pt/simpledb

Em um Modelo Relacional (CODD, 1970), tabelas são compostas por linhas (tuplas) e colunas (atributos). As colunas são atômicas e pertencem a um tipo de dados simples. Os relacionamentos são implementados por meio de chaves estrangeiras (FKs). Cada tupla é identificada unicamente por meio de uma chave primária (PK). A linguagem SQL, uma linguagem declarativa, é usada para formulação de consultas que envolvem tabelas.

Um exemplo de projeto de banco de dados no Modelo Relacional será apresentado na Seção 4.1.

2.2 Modelos NoSQL

Sistemas NoSQL são uma categoria de bancos de dados que não seguem os princípios do Modelo Relacional (HAN *et al.*, 2011). A nomenclatura NoSQL não diz respeito a um modelo de dados específico. A expressão em si concerne ao agrupamento de alguns modelos de dados que se distanciam da abordagem relacional. Suas características abrangem, em geral, as seguintes (HAN *et al.*, 2011): são implementados como código-aberto, os dados são distribuídos e horizontalmente escaláveis, e apresentam flexibilidade ou ausência de esquema.

Os sistemas NoSQL foram desenvolvidos para solucionar problemas pontuais de manipulação de dados. Eles precisam dar suporte a esquemas flexíveis ou ausentes, permitindo o armazenamento de dados semiestruturados ou não estruturados. Os dados gerenciados são volumosos e devem estar disponíveis às aplicações em tempo ágil (INDRAWAN-SANTIAGO, 2012). Em alguns casos, esses problemas são ou foram específicos de uma instituição. Como ilustração, apresenta-se o caso do sistema Dynamo (DECANDIA *et al.*, 2007), que foi criado pela Amazon[6] devido à necessidade de escalar seus dados de modo a atender dezenas de milhares de usuários, por meio de servidores espalhados em todo o mundo.

Contudo, após seu desenvolvimento específico (voltado a uma determinada aplicação), muitos deles são compartilhados com a comunidade. Assim, outros grupos de desenvolvedores podem contribuir e utilizá-los para atender suas necessidades. Um exemplo é o caso do Netflix (IZRAILEVSKY; TSEITLIN, 2011) que, devido à demanda por serviços mais escaláveis, passou a utilizar o Cassandra. O Cassandra foi originalmente desenvolvido pelo Facebook[7] para uso interno como solução de persistência de dados (LAKSHMAN; MALIK, 2010; IZRAILEVSKY; TSEITLIN, 2011).

No entanto, por buscarem solucionar problemas pontuais, essas implementações de bancos de dados acabam se distanciando umas das outras, o que gera diferentes estruturas de sistemas NoSQL. Essas estruturas estão associadas aos modelos, categorizados em Chave-valor, Colunas, Documentos e Grafo. Os modelos e suas principais características são discutidos a seguir.

2.2.1 Modelo Chave-valor

Dentre os sistemas de bancos de dados conhecidos como NoSQL, o Modelo Chave-valor é o que apresenta representação mais simples. Sua estrutura constitui-se de uma lista de pares de elementos compostos por uma chave e um valor (DECANDIA *et al.*, 2007; ISTVAN *et al.*, 2013). Esse modelo pode ser comparado à estrutura de dados chamada Tabela Hash (MAURER; LEWIS, 1975), na qual valores são associados às chaves de busca que permitem um rápido acesso ao seu conteúdo.

Alguns sistemas dessa categoria permitem – além dos tipos simples de dados, como numerais e cadeias de caracteres – a utilização de listas e conjuntos de valores de tipos simples. Eles costumam dar suporte a bases com grandes volumes de dados. Em contrapartida, esse modelo de banco de dados não costuma permitir que consultas sejam realizadas sobre os seus dados, mas apenas sobre as chaves de busca. Assim, todo o acesso é feito por meio das chaves de busca, e apenas com a chave é possível ter acesso ao valor.

O Modelo Chave-valor também não agrupa os dados por instâncias, diferentemente do realizado no Modelo Relacional por meio de tabelas. No modelo Chave-valor, todos os dados estão armazenados em uma estrutura composta apenas por duas colunas. Essa característica impossibilita a definição de esquemas de dados, sendo possível a introdução de algum metadado apenas por meio da nomenclatura explícita das chaves. Também não existe a possibilidade de realização de consultas mais complexas, como subconsultas. É possível realizar apenas uma consulta por vez e, baseando-se em seu resultado, realizar uma segunda. Essa perspectiva torna esse modelo de dados mais simples, o que diminui os tempos de

6 www.amazon.com

7 www.facebook.com

resposta, permitindo que a capacidade de armazenamento de suas bases de dados seja uma das maiores dos sistemas enquadrados na categoria NoSQL.

Além disso, o modelo não dá suporte a relacionamentos; não existe referência de chaves e, consequentemente, não há integridade referencial. Somando-se a isso, não existe uma linguagem de consulta, o que ocasiona algumas limitações na capacidade de busca.

2.2.2 Modelo em Colunas

Conceitualmente, o Modelo em Colunas é o que mais se assemelha ao Modelo Relacional, pois também é organizado por meio de linhas e colunas. Por outro lado, essa abordagem é projetada para tratar os dados de maneira não normalizada, geralmente não privilegiando a consistência das informações (PERERA, 2012).

O modelo orientado a colunas, como o próprio nome sugere, faz uma inversão na organização de seus dados (PERERA, 2012). Nessa abordagem, os atributos que compõem uma instância são organizados em colunas, e as linhas passam a conter as ocorrências de determinado atributo para cada instância de dados. Dessa forma, as linhas não mais armazenam uma tupla, mas sim um conjunto de atributos de mesmo tipo, enquanto o conjunto de atributos de uma coluna contém a informação de uma instância por completo. Isso permite que consultas que façam análises em subconjuntos de dados possam ser mais eficientes, porém a obtenção de instâncias inteiras passa a ser mais custosa. Essa diferença ocorre pelo esforço necessário para se percorrer as colunas das informações.

Além da inversão na orientação dos dados, o modelo em colunas também apresenta nomenclaturas diferentes para alguns conceitos. Nesse modelo, as instâncias de uma entidade fazem parte da mesma família de colunas.

Em uma família de colunas, é possível a existência de atributos não atômicos, por meio da representação de listas de valores. Além disso, nas famílias de colunas, as instâncias podem apresentar quantidades de atributos diferentes, não sendo necessário reservar espaços de armazenamento para valores nulos.

Apesar do Modelo em Colunas facilitar a execução de consultas em um subconjunto de dados, os sistemas dessa abordagem não permitem consultas com a junção de famílias de colunas. Nesses casos, é preferível que informações que necessitem ser resgatadas unidas estejam na mesma família de colunas. Isso se deve ao fato de esse modelo de dados não dar suporte ao uso de chaves estrangeiras, não sendo possível fazer referências de uma instância para outra.

Não é necessariamente uma regra do modelo, mas o Cassandra, um dos exemplos de sistemas de bancos de dados dessa categoria, não possuía, até a versão 1.2, garantias de unicidade para suas chaves primárias. Ainda sobre esse sistema, existe uma linguagem de consulta específica, chamada CQL (*Cassandra Query Language*) (LAKSHMAN; MALIK, 2010), que ajuda na formulação de consultas.

2.2.3 Modelo em Documentos

Assim como o Modelo Chave-valor, o baseado em Documentos também faz uso de associações entre pares de chaves e valores; porém, nesse último modelo, os dados não são dispostos em uma única estrutura de dados, mas sim agrupados em documentos, que podem seguir a codificação XML ou JSON (MCMURTRY *et al.*, 2013), sendo esta última a mais usada.

Um documento é uma coleção de chaves e valores que estão relacionados a uma instância de dados. Os vários documentos pertencentes a um mesmo conceito do domínio são armazenados em uma coleção de documentos, do mesmo modo como as tuplas são armazenadas em tabelas, no Modelo Relacional. Porém, assim como em outros sistemas NoSQL, o Modelo em Documentos é flexível, permitindo que documentos de uma mesma coleção de dados possam apresentar campos distintos uns dos outros.

As implementações dos sistemas NoSQL podem apresentar características diferentes umas das outras, ainda que pertençam a um mesmo modelo de dados. Para este trabalho, foi tomado como exemplo de Modelo em Documentos o do sistema MongoDB[8]. Para este sistema, a chave de uma instância – nesse caso, o documento – é chamada de *ObjectId* ou simplesmente *oid*, um tipo de dados BSON de 12 bytes. A chave pode ser definida no momento da persistência do documento, ou pode ter seu valor gerado aleatoriamente pelo próprio banco. É importante mencionar que, no MongoDB, não é possível definir atributos com propriedades de unicidade, como ocorre com as chaves primárias do Modelo Relacional.

8 www.mongodb.org

Diferentemente dos dois modelos NoSQL abordados anteriormente, o Modelo em Documentos permite que consultas mais complexas, envolvendo documentos de coleções distintas, ocorram. Para essas consultas, é necessário que um documento possua uma DBRef (*Database Reference*) do documento relacionado. Uma DBRef é uma estrutura que armazena as informações da coleção de documentos e o *ObjectId* do documento referenciado.

Apesar de permitirem referências, as DBRef não garantem integridade referencial, pois não é possível assegurar que a referência terá o mesmo valor do *ObjectId*. Além disso, esse tipo de consulta não está disponível na linguagem de consulta do sistema, não podendo ser feita a requisição diretamente sobre a base de dados. Ela apenas é possível por meio de aplicações com acesso ao banco de dados, fazendo uso da sua API. Essas consultas, no entanto, são mais custosas, em termos de desempenho, em comparação com consultas realizadas considerando o aninhamento de mais dados em um determinado documento (dados relacionados são armazenados em um mesmo documento).

2.2.4 Modelo em Grafo

Dentre os modelos classificados como NoSQL, o Modelo em Grafo é o que mais se distancia dos demais. Enquanto as outras abordagens têm seu foco no armazenamento dos dados, esse modelo tem como destaque principal os relacionamentos que ocorrem entre as entidades de sua base (MCMURTRY *et al.*, 2013).

Com base na teoria de Grafos, os bancos de dados que seguem essa abordagem possuem três tipos de elementos (MCMURTRY *et al.*, 2013): nós, arestas e propriedades. Os nós correspondem às instâncias de dados, as arestas se referem aos relacionamentos mantidos entre as instâncias, enquanto as propriedades dizem respeito aos valores de dados contidos nas instâncias, os quais podem ser, por exemplo, booleanos, inteiros, caracteres e conjuntos de valores. O sistema Neo4j[9], um exemplo de NoSQL que implementa esse modelo, permite a definição de chaves de identificação.

Dessa forma, o banco de dados permite que as instâncias de dados (nós) possam estabelecer relacionamentos umas com as outras por meio das arestas, que, assim como os nós, também podem conter propriedades que as descrevem. Além das propriedades, os nós e arestas podem conter rótulos (espécie de nomenclatura) que os classificam em grupos mais específicos. Nos nós, esses rótulos (tipo de metadado) podem ser usados para classificar as instâncias; já nas arestas, eles servem para determinar o tipo de relacionamento que está ocorrendo.

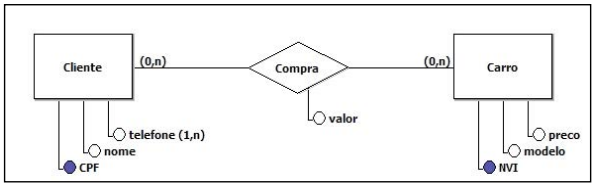
Uma aresta possui um nó de entrada e um nó de saída. Diferentemente dos demais modelos, este, além de dar suporte a referências, também garante a integridade referencial, assegurando que o nó de entrada sempre faça referência ao nó de saída. As chaves de acesso aos nós são definidas automaticamente pelo sistema. No entanto, é possível estabelecer restrições de unicidade para outras propriedades do nó.

3 Cenário exemplo

Nesta seção, é apresentado um cenário exemplo de projeto de banco de dados em um dado domínio. Por meio desse cenário, serão ilustrados os conceitos anteriormente descritos, assim como será possível comparar a representação dos dados em cada modelo.

O exemplo diz respeito a uma aplicação que permite a compra de carros. Dessa forma, os requisitos de dados envolvem carros, clientes associados e a ação da compra. O modelo conceitual ER, com as entidades relevantes para o exemplo citado, é mostrado na Figura 1. Nela estão representadas as entidades Cliente e Carro e o relacionamento Compra. O Cliente possui os seguintes atributos: CPF (identificador), nome (simples) e telefone (multivalorado). A entidade Carro possui os atributos: NVI (identificador), modelo (simples) e preço (simples). O relacionamento de compra entre o cliente e o carro possui um atributo Valor e tem cardinalidade máxima N:N.

Figura 1 – Modelo conceitual exemplo.



Fonte: Elaborada pelos autores.

Na Seção 4, o exemplo, em nível conceitual, será mapeado para estruturas de acordo com o modelo Relacional e as diversas abordagens NoSQL.

9 http://neo4j.com/

4 Comparando o Modelo Relacional e as Abordagens NoSQL

A seguir são ilustradas as diferenças estruturais entre o modelo de dados Relacional e as abordagens NoSQL. Para isso, será usado o exemplo apresentado na Seção 3.

Após as comparações, serão indicados possíveis mapeamentos conceituais entre os elementos das estruturas em questão. Para a definição dos mapeamentos, foi selecionado um sistema específico para cada modelo NoSQL, visto que sistemas NoSQL podem ter diferenças entre si, mesmo quando implementam o mesmo modelo de dados. A escolha desses sistemas levou em conta ferramentas de código aberto em destaque no mercado[10] que disponibilizam documentação.

Ao final da seção, serão realizadas algumas considerações acerca dos mapeamentos conceituais entre os modelos.

4.1 Cenário no Modelo Relacional

Ao transformar o modelo conceitual, do exemplo mostrado na Figura 1, para um modelo Relacional, são obtidas as tabelas e seus esquemas, conforme mostra a Figura 2. A notação empregada é a sugerida por Heuser (2006).

Figura 2 – Esquema Relacional do Exemplo.

Cliente (CPF CHAR(11), nome VARCHAR2(30))
Telefone (CPF CHAR(11), numero VARCHAR2(9),
CPF referencia Cliente)
Carro (NVI CHAR(17), modelo VARCHAR2,
preço FLOAT(5,2))
Compra (CPF CHAR(11), NVI CHAR(17),
valor FLOAT(5,2),
CPF referencia Cliente,
NVI referencia Carro)

Fonte: Elaborada pelos autores.

No exemplo, as entidades Cliente e Carro são representadas por tabelas que contêm os atributos pertencentes a cada entidade. Na tabela Cliente estão os atributos Nome e CPF. Este último possui valores únicos para cada registro, sendo assim utilizado como chave primária (PK). O atributo multivalorado Telefone, do Modelo Conceitual, passa a ser representado por uma tabela, permitindo a introdução de diversos valores.

A tabela Carro possui os atributos NVI (PK), Modelo e Preço. No Modelo Relacional, o relacionamento de compra de carro também foi transformado em tabela, possuindo uma chave primária, composta por CPF e NVI, além do campo valor. NVI e CPF são, individualmente, chaves estrangeiras (FK) que apontam para Carro e Cliente, respectivamente.

4.2 Chave-valor x Relacional

Para representar o cenário exemplo na abordagem Chave-valor, é preciso levar em consideração as características (e limitações) do modelo, comparado ao Modelo Relacional. Ambos os modelos funcionam em estrutura tabular, mas o Chave-Valor é formado por uma única estrutura, composta por duas colunas: uma correspondente à chave e outra, ao valor associado. O campo de chave corresponderia à chave primária no Modelo Relacional, também apresentando a propriedade de unicidade. Porém, o valor da chave no Modelo Chave-valor é definido pelo usuário, e, nessa chave, normalmente são incluídas informações de metadados. Essa possibilidade é ilustrada na primeira coluna do Quadro 1, um exemplo de instanciação dos dados do cenário exemplo (Seção 3) no Modelo Chave-Valor.

Quadro 1 – Exemplo no Modelo Chave-valor.

Chave	Valor
CPF1	88456707830
nome1	Webber
telefone1	999999999
NVI1	9BWHE21JX24060960
modelo1	Gol 2013
preco1	28.000,00
CPF1:NVI1	26.500,00
CPF2	76052657278
nome2	Sidartha
telefone2	[999572110, 988081046]
88456707830: 9BWHE21JX24060960	25.500,00

Fonte: Elaborado pelos autores.

10 http://db-engines.com/en/ranking

Como pode ser visto no Quadro 1, devido à ausência de esquema, é necessário exibir a estrutura com valores de dados. Nesta base de dados, existem duas instâncias de clientes, uma de carro e a realização de uma compra, não havendo distinção explícita entre as entidades. Para essa representação, cada atributo ou conjunto de atributos é armazenado em uma linha de dados, sendo referido à sua correta instância pela nomenclatura da chave. Por exemplo, as informações da instância do cliente Webber podem ser buscadas com a referência 1 sobre os atributos de cliente – CPF1, nome1 e telefone1 são as chaves para os atributos CPF, nome e telefone do cliente Webber, que é o primeiro cliente da base de dados.

Além disso, os valores dos nomes dos clientes, do tipo caractere, estão na mesma coluna que os preços dos carros, que são do tipo numérico. A ausência de esquema e a simplicidade da estrutura permitem que esses diferentes tipos de dados estejam próximos.

A diferença entre os modelos de dados também é expressa nos relacionamentos entre as entidades. O Modelo Chave-valor não permite relacionamentos, mas, para muitos casos, é necessário manter as associações entre as entidades. Para isso, foi inserida uma linha de dados relacionando o CPF do cliente ao NVI do carro. Dessa forma, é possível manter o registro de que carro foi vendido para qual cliente, informando o valor da venda.

Os mapeamentos desses e dos demais conceitos não representados no exemplo, mas que foram identificados como importantes para a conversão de bancos de dados entre o Modelo Relacional e o Chave-valor, com base no sistema Redis[11], são apresentados no Quadro 4.

4.3 Colunas x Relacional

Como comentado, o Modelo em Colunas inverte a organização dos dados, ordenando primeiramente as colunas e depois as linhas. Como ilustração, apresenta-se um exemplo na Figura 3. Nesta, constam o armazenamento orientado a linha (utilizado pelo Modelo Relacional) e o armazenamento orientado a coluna (empregado no Modelo em Colunas). Enquanto no primeiro os atributos referentes a uma tupla de dados são organizados de modo próximo (CPF, nome, telefone), no segundo os atributos de uma mesma categoria são priorizados.

11 redis.io

Para analisar os valores de determinado atributo, o banco de dados efetua uma consulta semelhante à obtenção dos atributos de uma linha no Modelo Relacional, enquanto o resgate de uma instância seria equivalente a consultar, no Modelo Relacional, se em cada linha de dados existe uma coluna com determinado valor.

Figura 3 – Exemplo de Armazenamento no Modelo em Colunas e no Relacional.

Armazenamento orientado a linha
88456707830, Webber, 999999999; 76052657278, Sidartha, 999572110
Armazenamento orientado a coluna
88456707830, 76052657278; Webber, Sidartha; 999999999, 99957211

Fonte: Elaborada pelos autores.

A partir do Exemplo (Figura 3), uma compreensão dos conceitos presentes no Modelo em Colunas pode ser obtida com a Figura 4. Nela estão representadas as instâncias de dois clientes e um carro. Na primeira linha (Webber), são encontradas as famílias de colunas "Dados Pessoais" e "Dados de Carro". Na família de colunas Dados Pessoais, podem ser vistas as colunas com os atributos do cliente em si, enquanto na família de colunas Dados de Carro, estão presentes as colunas contendo os atributos do carro comprado pelo cliente. Comparando a primeira linha (Webber) com a segunda (Sidartha), pode-se comprovar a flexibilidade do esquema, pois o cliente Sidartha não possui nenhum carro associado. Assim, não existem informações sobre o Carro para a instância Sidartha. Da mesma forma, percebe-se a lista de valores na coluna Telefone.

Figura 4 – Ilustração de família de colunas para o Exemplo.

Webber	Dados Pessoais			Dados de Carro		
	CPF	nome	Telefone	NVI	modelo	preço
	88456707830	Webber	999999999	9BWHE21JX24060960	Gol 2013	28.000,00
Sidartha	Dados Pessoais					
	CPF	nome			Telefone	
	76052657278	Sidartha			[999572110, 988081046]	

Fonte: Elaborada pelos autores.

O Quadro 4 apresenta mapeamentos possíveis entre os conceitos do Modelo Relacional e aqueles do Modelo em Colunas, considerando o sistema Cassandra.

4.4 Documentos x Relacional

Com base no Exemplo (Figura 3), o Quadro 2 ilustra a representação dos dados dos clientes estruturados no Modelo em Documentos, utilizando a notação JSON.

Quadro 2 – Documentos de clientes para o Exemplo.

```
{
    _id: "88456707830",
    Nome: "Webber",
    Telefone: "999999999"
}
{
    _id: "76052657278",
    Nome: "Sidartha",
    Telefones: [
      {telefone: "999957211"},
      {telefone: "988081046"}
    ]
}
```

Fonte: Elaborado pelos autores.

No Quadro 2, são mostradas as instâncias de clientes armazenadas em um documento, assim como as definições de campos. A definição do campo se encontra próxima ao dado em si, como no caso do Nome com o seu valor correspondente (exemplo: Nome: "Webber"). Além disso, percebe-se que, nesse modelo, é possível representar campos com mais de um valor, como o que ocorre com Telefone. Para este, existe um campo chamado Telefones que possui uma lista de campos Telefone.

Nesse sistema, o conceito de chave difere, em certo ponto, do conceito de mesmo nome do Modelo Relacional. Como comentado, a chave é um objeto do tipo *ObjectId*, cuja nomenclatura segue o padrão do sistema, definido como "_id" (Quadro 2). Neste caso, foi usado o valor do CPF como chave do documento, por ser um atributo único.

Existem duas formas de representar os relacionamentos existentes entre as instâncias de dados: por meio de campos incorporados ou por referências DBRef. O Quadro 3 ilustra a representação dos dados conforme as duas possibilidades.

Como pode ser visto no Quadro 3, na primeira possibilidade, o documento do cliente (Webber) possui uma referência para o documento do carro (9BWHE21JX24060960). Já na segunda possibilidade, as informações do carro fazem parte do documento do cliente.

Quadro 3 – Relacionamento por referência e por aninhamento.

Implementação	Documentos

DBRef	{
	_id: "88456707830",
	Nome: "Webber",
	Telefone: "999999999",
	Carro: objectId("9BWHE21JX24060960")
	}
	---------- // ----------
	{
	_id: "9BWHE21JX24060960",
	Modelo: "gol 2015",
	Preço: 28.000,00
	}

Aninhado (*embedded*)	{
	_id: "76052657278",
	Nome: "Sidartha",
	Telefones: [
	{telefone: "999957211"},
	{telefone: "988081046"}
	],
	Carro: {
	NVI: "C021E21JX240LP925",
	Modelo: "fiesta 2012",
	Preço: "15.000,00"
	}
	}

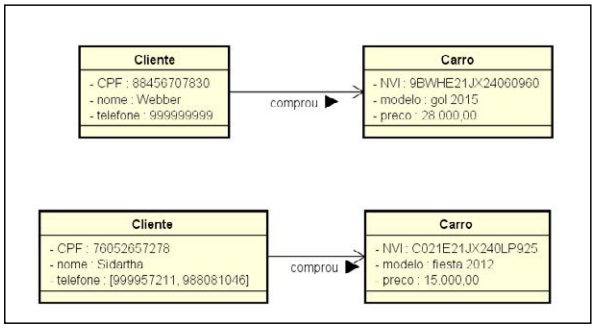
Fonte: Elaborado pelos autores.

Os mapeamentos identificados entre o Modelo Relacional e o Modelo em Documentos (MongoDB) são mostrados no Quadro 4.

4.5 Grafo x Relacional

Para o Exemplo, considerando o Modelo em Grafo, as instâncias de dados são representadas por nós. Cada nó possui um rótulo que o diferencia (classifica) entre clientes e carros. As informações de cada nó, como nome e CPF dos clientes ou modelo e preço dos carros, são armazenadas como propriedades em cada nó. Os relacionamentos das compras dos carros são representados por meio das arestas, que apresentam a palavra "comprou". A Figura 5 ilustra o Exemplo no Modelo em Grafo, usando, para isso, uma representação dos dados por meio de um diagrama de classes UML (FOWLER, 2014).

Figura 5 – Ilustração do Exemplo no Modelo em Grafo.



Fonte: Elaborada pelos autores.

Comparando este modelo ao Modelo Relacional, as arestas funcionam como chaves estrangeiras. O nó do cliente Sidartha está associado ao carro de modelo Fiesta 2012 por meio de uma aresta do tipo comprou. As variações na cardinalidade dos relacionamentos são determinadas apenas pela quantidade de arestas que envolvem determinado nó. Por exemplo, o nó Webber poderia efetuar a compra de vários carros, bastando apenas que houvesse a associação entre o nó do cliente e os dos carros relacionados.

A possibilidade de garantir a unicidade de uma propriedade permite que dados como o CPF possam manter suas características exclusivas.

Os mapeamentos identificados entre os conceitos do Modelo Relacional e do Modelo em Grafo, com base no sistema Neo4j, são apresentados no Quadro 4.

4.6 Considerações

Com o estudo realizado, foram identificados possíveis mapeamentos conceituais entre o Modelo Relacional e os elementos presentes em cada abordagem NoSQL. O Quadro 4 apresenta esses mapeamentos, de acordo com os sistemas de bancos de dados em destaque para cada modelo.

Nosso estudo lida com a heterogeneidade estrutural dos modelos de dados e alguns aspectos específicos do projeto de banco de dados Relacional. A fim de lidar com essas questões, os mapeamentos levaram em conta os diferentes conceitos existentes, que pertencem aos modelos de dados. Além disso, como uma referência de projeto de banco de dados, foram considerados alguns conceitos não só a partir do modelo Relacional, mas também a partir do modelo conceitual ER. Desse modo, foram considerados conceitos do modelo ER conceitual que, embora não sejam implementados diretamente em um banco de dados relacional, são frutos de elementos existentes no mundo real e podem ser implementados em sistemas NoSQL. Esses conceitos consideram particularmente os atributos compostos e multivalorados, além de especialização. Assim, o Quadro 4 (pág, seguinte) apresenta os conceitos e, para cada um, os correspondentes nos Modelos Relacional, Chave-valor, Colunas, Documentos e Grafo.

Como pode ser visto, a capacidade de representar os conceitos comuns ao Modelo Relacional (e a um projeto de banco de dados Relacional) varia de acordo com a abordagem NoSQL. Sistemas que seguem o Modelo Chave-valor são mais simples e possuem menor robustez para acomodar os conceitos relacionais, enquanto o Modelo em Grafo possui maior capacidade de representá-los.

Além dos desafios quanto à heterogeneidade estrutural entre os modelos de dados, também é preciso levar em consideração as dificuldades de heterogeneidade encontradas entre os sistemas existentes nos respectivos modelos de bancos de dados. Não existe ainda uma padronização em relação a sistemas categorizados segundo um modelo NoSQL, o que permite que sistemas com modelos semelhantes apresentem recursos distintos. Por esse motivo, a conversão de estruturas de dados Relacionais para os modelos NoSQL deve analisar não apenas os modelos de dados em si, mas também os sistemas de bancos de dados de destino.

Quadro 4 – Possíveis Mapeamentos Conceituais entre os Modelos.

Conceito	Relacional	Chave-valor (Redis)	Colunas (Cassandra)	Documentos (MongoDB)	Grafo (Neo4j)
Entidade	Tabela ou conjunto de tabelas	Não existe equivalente	Família de colunas	Coleção de documentos	Rótulo
Instância	Tupla	Um valor ou coleção de valores	Itens de família de colunas	Documento	Nó
Atributo					
Simples	Atributo	Valor	Atributo	Campo	Propriedade
Composto	Dividido em atributos simples ou em tabelas interdependentes	Conjunto de valores	Aninhamento de atributos	Campos incorporados	Conjunto de propriedades ou novo nó
Multivalorado	Dividido em atributos ou em tabelas interdependentes	Conjunto de valores	Aninhamento de atributos	Campos incorporados	Conjunto de propriedades
Nulo	Possui (obrigatório)	Não possui	Não possui	Não possui	Não possui
Restrições					
Identificador de entidade	PK	Chave	Chave primária (não garante unicidade)	*Object identifier* (_id)	Id
Referência	FK	Não possui	Não possui	Possui (DBRefs)	Possui (aresta)
Relacionamentos (1:1, 1:N, M:N)	Referência por FK	Não possui	Aninhamento de atributos	Campos incorporados ou DBRef	Por meio das arestas
Integridade Referencial	Possui	Não possui	Não possui	Não possui	Possui

Fonte: Elaborado pelos autores.

5 Trabalhos relacionados

No panorama da conversão de estruturas de dados do Modelo Relacional para modelos NoSQL, alguns trabalhos vêm sendo desenvolvidos. Considerando que todos os trabalhos analisados têm como banco de origem um Relacional, alguns critérios para avaliação e comparação foram elencados, a saber:

SGBD relacional: considera qual é o sistema usado como origem;

Conversão: indica se o processo de conversão é automático ou depende da intervenção do usuário;

Usuário: analisa o grau de conhecimento exigido do usuário sobre o domínio de dados e a conversão;

Sistemas-destino: determina para quais modelos/sistemas NoSQL o banco-origem pode ser convertido.

Um resumo apresentando os trabalhos avaliados juntamente com os critérios citados é mostrado no Quadro 5.

Quadro 5 – Comparativo entre os trabalhos relacionados.

Trabalho	SGBD origem	Conversão	Usuário	Sistemas Destino
Zhao *et al.* (2014)	MySQL	Automática	Comum	Chave-valor, Coluna, Documento
Portey *et al.* (2015)	MySQL	Automática	Comum	Documento
Karnitis e Arnicans (2015)	Vários Relacionais	Semiautomática	Especialista	Documento
Mpinda *et al.* (2015)	Vários Relacionais	Automática	Comum	Coluna
Freitas (2015)	Vários Relacionais	Semiautomática	Comum	Documento

Fonte: Elaborado pelos autores.

No trabalho de Zhao *et al.* (2014), é apresentado um ambiente de conversão de dados do modelo Relacional para os modelos Chave-valor, Colunas e Documentos. Esse trabalho permite automatizar o processo por meio do estabelecimento de regras de conversão baseadas nas extensões dos dados. Essas extensões – por exemplo, a extensão das informações de telefone de um cliente – dizem respeito à forma como as informações se complementam no Modelo Relacional por meio dos relacionamentos das tabelas. O trabalho aborda um conceito muito comum nos modelos de dados NoSQL – a desnormalização dos dados, em que as informações fragmentadas em diferentes tabelas no Modelo Relacional passam a ser agrupadas dentro de uma mesma estrutura. O modelo proposto fundamenta-se nas referências existentes entre as tabelas relacionais, criando uma estrutura de encadeamento para representar os relacionamentos. Essa abordagem, no entanto, pode desencadear demasiada redundância dos dados e dificultar a identificação das entidades, prejudicando, assim, a execução em alguns modelos, como o de Documentos.

O trabalho de Portey *et al.* (2015) apresenta uma ferramenta desenvolvida para efetuar a conversão de dados. Nessa ferramenta, o usuário fornece como entrada o esquema relacional de um banco de dados MySQL e obtém como saída os dados organizados de forma equivalente, em uma estrutura orientada a Documentos. Nesse caso, a conversão dos dados ocorre de forma automática, não necessitando da intervenção do usuário. Apesar de facilitar e agilizar a realização do procedimento, essa abordagem é limitada, pois não leva em conta a necessidade de conhecimento do domínio de dados. Por exemplo, diferenciar que o Telefone é uma informação do Cliente enquanto Carro é uma entidade à parte pode ser decisivo na hora de compor e popular uma estrutura-destino. O algoritmo de conversão implementado executa uma ação padrão em todos os casos. A ação para conversão, nesse caso, é realizada com a transformação direta das tuplas de dados em documentos. Esse processo mantém a normalização dos dados, o que facilita a conversão, mas, por outro lado, pode diminuir o desempenho das consultas devido à necessidade de junções de documentos.

O trabalho de Karnitis e Arnicans (2015) apresenta um diferencial em relação ao trabalho anteriormente citado. Ele demonstra uma preocupação em conhecer os tipos de informações que estão contidas nas tabelas do banco de origem. Para isso, é requisitado que o usuário – neste caso, um especialista do domínio – classifique as tabelas do esquema de acordo com algumas tipificações preestabelecidas, a saber: Estrutura baseada em Tabela (*Table-like-structure*), Codificador (*Codifier*), Entidade Simples (*Simple entity*), Entidade Complexa (*Complex entity*) e Relacionamento N:N (*N:N-Link*). Dessa forma, a abordagem se torna semiautomática, porém permite que o algoritmo de conversão tenha uma melhor compreensão dos relacionamentos entre as tabelas e suas interdependências existentes. Logo, é possível organizar as informações de maneira mais coerente, permitindo a desnormalização dos dados das entidades.

Mpinda, Bungama e Maschietto (2015) estabelecem um processo de conversão de dados do Modelo Relacional para o modelo NoSQL orientado a colunas, sem restringir o sistema-destino. Para isso, consideram como origem os *scripts* do banco Relacional e levam em conta cinco dimensões presentes no modelo em Colunas, a saber: tabelas, famílias de colunas, chaves, super colunas e colunas simples. No processo de conversão, todas as tabelas relacionais são percorridas e as instâncias de dados são extraídas, sendo convertidas em uma estrutura de objetos de classe. Após isso, são identificados os relacionamentos existentes entre as instâncias dos dados, e direcionado o sentido desse relacionamento. Em seguida, os objetos são analisados, e cada um que tiver objeto-pai será atualizado com os atributos dos objetos- filhos. Esse procedimento ocorre recursivamente até serem concluídos os relacionamentos das tabelas. Ao fim da análise dos relacionamentos, é gerado um conjunto de objetos contendo informações correlatas. Esse processo de conversão desnormaliza os dados estruturados das tabelas, agregando mais informações relacionadas em uma estrutura-destino. Isso é favorável para o modelo em Colunas, pois facilita a análise dos conjuntos de dados. Por outro lado, gera redundância nas informações.

O trabalho de Freitas (2015) também se apresenta como uma abordagem semiautomática. Neste, mais uma vez, é utilizada a estratégia de classificação das tabelas para enriquecer a compreensão do esquema relacional. Porém, neste caso, não se faz necessário o conhecimento de um especialista para realizar a classificação e o refinamento dos dados. O processo proposto permite a conversão de bases de dados relacionais para os diferentes modelos NoSQL.

A implementação do processo, no entanto, precisa analisar particularmente cada fonte de dados-destino para implementar a solução.

6 Desafios e possibilidades

Mesmo com os avanços obtidos, a área de conversão de dados entre bancos Relacionais e NoSQL ainda apresenta desafios a serem enfrentados. Alguns desses desafios estão relacionados com:

A necessidade de assistência do usuário no processo de conversão: apesar das abordagens de conversão possibilitarem a análise dos bancos de origem e proporem soluções, as abordagens mais eficientes exigem uma interação por parte do usuário na compreensão dos esquemas de dados de origem;

Eventuais redundâncias nas bases geradas: a base de dados destino contém as informações da base origem, porém o processo de conversão pode gerar redundâncias nas informações. Se em uma tabela de relacionamentos N:N, por exemplo, forem encontrados atributos além das referências, esses atributos serão duplicados em ambas as instâncias envolvidas no relacionamento. Sendo assim, existe a possibilidade de que, em um dado momento, o valor do atributo seja alterado em uma instância, criando inconsistência com o valor presente em outra instância.

Por meio dos desafios existentes, a conversão de dados entre os modelos também proporciona espaço para novas pesquisas, como:

Automatização na extração do esquema: o processo de conversão de dados envolve a identificação das entidades do esquema relacional; na maioria dos casos, esse procedimento ainda depende da interferência humana. Porém, no escopo da Integração de Dados, existem algumas abordagens que são utilizadas para identificar similaridades entre tabelas de bases de dados diferentes, a partir do uso de alguma referência semântica – por exemplo, uma ontologia de domínio (DOAN; HALEVY, 2005; LIVINGSTON *et al.*, 2015). A inclusão de uma análise semântica sobre a base de dados relacional pode permitir a identificação automática das interdependências que as tabelas possuem. Isso evitaria a necessidade de interação do usuário no processo de conversão;

Conversão de consultas: Assim como a conversão de estruturas de dados, também seria interessante a conversão de consultas. Se, por um lado, a conversão das estruturas auxilia equipes de desenvolvimento a migrarem suas bases de dados do modelo Relacional para modelos NoSQL, por outro lado, a possibilidade de converter *scripts* de consultas auxiliaria as equipes a testarem quão próximo o resultado da conversão ficou da base de dados original. Os desenvolvedores estariam interessados em mecanismos de conversão de consultas para que possam testar a nova forma de persistência com agilidade. Já existem alguns protótipos de ferramentas de conversão de consultas disponíveis[12].

7 Considerações finais

Este artigo apresentou uma revisão acerca do tema "mapeamento de conceitos do Modelo Relacional para Modelos NoSQL". O trabalho foi motivado pela necessidade que algumas equipes de desenvolvimento de grandes sistemas têm de migrar bancos de dados Relacionais para sistemas NoSQL. Isso traz à tona o desafio de permitir processos de conversão que levem em consideração as diferenças estruturais entre eles.

O artigo abordou as diferenças estruturais entre o Modelo Relacional e os modelos NoSQL e apontou possibilidades de mapeamentos conceituais entre eles. Foram também descritos trabalhos relacionados ao tema e suas contribuições para a área. Por fim, desafios que proporcionam espaço para novas pesquisas e possibilidades de soluções foram indicados.

REFERÊNCIAS

CHEN, P. P. The entity-relationship model—toward a unified view of data. **ACM Transactions on Database Systems (TODS)**, v. 1, n. 1, p. 9-36, 1976.

CODD, E. F. A relational model of data for large shared data banks. **Communications of the ACM**, v. 13, n. 6, p. 377-387, 1970.

CURÉ, O.; KERDJOUDJ, F.; LE DUC, C.; LAMOLLE, M.; FAYE, D. On the potential integration of an ontology-based data access approach in NoSQL stores. In: INTERNATIONAL CONFERENCE ON EMERGING INTELLIGENT DATA AND WEB TECHNOLOGIES (EIDWT), 3., 2012, Bucareste. **Proceedings...** Bucareste: IEEE, 2012. p.166-173.

DECANDIA, G.; HASTORUN, D.; JAMPANI, M.; KAKULAPATI, G.; LAKSHMAN, A.; PILCHIN, A.; SIVASUBRAMANIAN, S.; VOSSHALL, P.;

12 http://www.querymongo.com

VOGELS, W. Dynamo: Amazon's highly available key-value store. **ACM SIGOPS Operating Systems Review**, v. 41, n. 6, p. 205-220, 2007.

DOAN, A.; HALEVY, A. Y. Semantic integration research in the database community: a brief survey. **AI magazine**, v. 26, n. 1, p. 83-94, 2005.

FOWLER, M. **UML Essencial**: um breve guia para linguagem padrão. 3. ed. Porto Alegre: Bookman, 2005.

FREITAS, C. M. **Mapeamentos Conceituais entre Modelo Relacional e Estruturas NoSQL**: um estudo de caso com documentos. 2015. 105 f. Dissertação (Mestrado em Ciência da Computação) – Centro de Informática, Universidade Federal de Pernambuco, Recife, 2015.

HAN, J. HAIHONG, E.; LE, G.; DU, J. Survey on NoSQL database. In: INTERNATIONAL CONFERENCE ON PERVASIVE COMPUTING AND APPLICATIONS (ICPCA), 6., 2011, Port Elizabeth, South Africa. **Proceedings...** Port Elizabeth: IEEE, 2011. p. 363-366.

HECHT, R.; JABLONSKI, S. **NoSQL evaluation**: a use case oriented survey. In: INTERNATIONAL CONFERENCE ON CLOUD AND SERVICE COMPUTING, 2011, Hong Kong. **Proceedings...** Hong Kong: IEEE, 2011. p. 336-341.

HEUSER, C. A. **Projeto de banco de dados**. v. 5. Porto Alegre: Sagra Luzzatto, 2006. 254 p.

INDRAWAN-SANTIAGO, M. Database research: are we at a crossroad? Reflection on NoSQL. In: INTERNATIONAL CONFERENCE ON NETWORK-BASED INFORMATION SYSTEMS (NBiS), 15., 2012, Melbourne, Australia. **Proceedings...** Melbourne: IEEE, 2012. p. 45-51.

ISTVAN, Z.; ALONSO, G.; BLOTT, M.; VISSERS, K. A flexible hash table design for 10Gbps key-value stores on FPGAs. In: INTERNATIONAL CONFERENCE ON FIELD PROGRAMMABLE LOGIC AND APPLICATIONS (FPL), 23., 2013, Porto, Portugal. **Proceedings...** Porto: IEEE, 2013. p. 1-8.

IZRAILEVSKY, Y.; TSEITLIN, A. The netflix simian army. **The Netflix Tech Blog**. 2011. Disponível em: <http://techblog.netflix.com/2011/07/netflix-simian-army.html>. Acesso em: 12 nov. 2015.

KARNITIS, G.; ARNICANS, G. Migration of relational database to document-oriented database: structure denormalization and data transformation. In: INTERNATIONAL CONFERENCE ON COMPUTATIONAL INTELLIGENCE, COMMUNICATION SYSTEMS AND NETWORKS (CICSyN), 7., 2015, Riga, Lativia. **Proceedings...** Riga: IEEE, 2015. p. 113-118.

LAKSHMAN, A.; MALIK, P. Cassandra: a decentralized structured storage system. **ACM SIGOPS Operating Systems Review**, v. 44, n. 2, p. 35-40, 2010.

LIVINGSTON, K. M.; BADA, M.; BAUMGARTNER, W. A.; HUNTER, L. E. KaBOB: ontology-based semantic integration of biomedical databases. **BMC bioinformatics**, v. 16, n. 1, p. 126, 2015.

MARTINS FILHO, M. A. P. **SQL X NoSQL**: análise de desempenho do uso do MongoDB em relação ao uso do PostgreSQL. 2015. 53 f. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) – Centro de Informática, Universidade Federal de Pernambuco, Recife, 2015.

MAURER, W. D.; LEWIS, T. G. Hash table methods. **ACM Computing Surveys (CSUR)**, v. 7, n. 1, p. 5-19, 1975.

MPINDA, A.; BUNGAMA, P.; MASCHIETTO, L. From relational database to column-oriented NoSQL database: migration process. **International Journal of Engineering Research & Technology**, v. 4, n. 5, p. 399-403, 2015.

PERERA, S. **IBM developerWorks**: **Considerações sobre o Banco de Dados Apache Cassandra**. 2012. Disponível em: <http://www.ibm.com/developerworks/br/library/os-apache-cassandra/>. Acesso em: 15 mai. 2014.

PORTEY, M.; DIGRASE, M.; DESHMUKH, G.; NERKAR, M. Database Migration from Structured Database to non-Structured Database. In: INTERNATIONAL CONFERENCE ON RECENT TRENDS & ADVANCEMENTS IN ENGINEERING TECHNOLOGY (ICRTAET), 4., 2015, Nashik, India. Proceedings... Nashik: SNJB, 2015.

MCMURTRY, D.; OAKLEY, A.; SHARP, J.; SUBRAMIAN, M.; ZHANG, H. **Data Access for Highly-Scalable Solutions**: Using SQL, NoSQL, and Polyglot Persistence. [S.l.]: Microsoft, 2013.

STROZZI, C. **NoSQL**: a non-SQL RDBMS. 1998. Disponível em: <http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/nosql/Home%20Page>. Acesso em: 15 mar. 2015.

TUDORICA, B. G.; BUCUR, C. A comparison between several NoSQL databases with comments and notes. In: ROEDUNET INTERNATIONAL

CONFERENCE (RoEduNet), 10., 2011, Iasi, Romênia. **Proceedings...** Iasi: IEEE, 2011. p. 1-5.

ZHAO, G.; LIN, Q.; LI, L.; LI, Z. **Schema Conversion Model of SQL Database to NoSQL**. In: INTERNATIONAL CONFERENCE ON P2P, PARALLEL, GRID, CLOUD AND INTERNET COMPUTING (3PGCIC), 9., 2014, Guangdong, China. **Proceedings...** Guangdong: IEEE, 2014. p. 355-362.