Associação Propagadora Esdeva
UniAcademia
Curso de Sistemas de Informação
Trabalho de Conclusão de Curso – Artigo

MIGRAÇÃO DE BASES DE DADOS RELACIONAIS PARA BANCO DE DADOS NOSQL UTILIZANDO TÉCNICAS DE *ETL*

Cézar Ribeiro Galvão Filho[1]
UniAcademia, Juiz de Fora, MG
Evaldo de Oliveira da Silva[2]
UniAcademia, Juiz de Fora, MG

Linha de Pesquisa: Banco de Dados

RESUMO

Muitos sistemas de informação são desenvolvidos utilizando Sistemas Gerenciadores de Banco de Dados Relacionais (SGBD). Os bancos de dados relacionais permitem gerenciar os dados de forma estruturada por meio de tabelas e formatos que dificultam a escalabilidade dos sistemas de informação para armazenar diferentes tipos e formatos de dados. Conforme o aumento do volume de dados, faz-se necessário o uso de bases de dados mais robustas, tais como as baseadas em NoSQL. Este trabalho tem como objetivo estabelecer um processo para migração de base de dados relacional para NoSQL. Apresenta as vantagens e desvantagens envolvidas nas etapas do processo. Um exemplo de migração é apresentado para implementar o processo proposto.

Palavras-chave: Banco de dados. ETL. NoSQL.

1 INTRODUÇÃO

Com a evolução das aplicações e com o grande volume de dados coletados pelos sistemas de informação, é necessário que haja a implementação de bases de dados que permitam mais velocidade na manipulação de dados e maior capacidade de armazenamento. Assim as bases de dados NoSQL (*not only SQL*) surgem como uma opção aos bancos de dados relacionais.

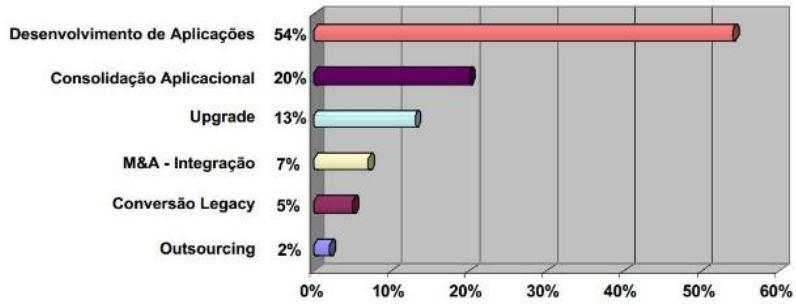
---

[1] Discente do Curso de Sistemas de Informação da UniAcademia. Endereço: Rua Luz Interior, 30. Celular: (32 9 9162-9149). E-mail: cezarribeirogalvao@gmail.com
[2] Docente do Curso de Sistemas de Informação da UniAcademia. Orientador(a).

Conforme Oliveira (2017), as empresas precisaram se adaptar para otimizar e melhorar processos, o que implica na coleta, tratamento e análise de uma grande quantidade de dados. Segundo Manyika et al.(2011) estima-se que os dados cresçam cerca de 40% ao ano.

Assim, ainda existem muitas aplicações que são criadas utilizando como base modelos relacionais, e então, ao atingir determinado ponto de maturidade é necessário fazer sua migração para outro modelo usando o NoSQL, por exemplo. A Figura 1 apresenta alguns números percentuais que motivam este tipo de migração, de acordo com Howard (2007). Outro motivo citado por Diana e Gerosa (2012) é a escalabilidade, pois os bancos de dados NoSQL conseguem lidar com um grande volume de dados de forma escalável, de forma a oferecer uma boa performance de acesso as informações.

Essa escalabilidade, segundo Diana e Gerosa (2012), vem da possibilidade de escalar os bancos de dados NoSQL horizontalmente apenas adicionando uma quantidade maior de servidores, enquanto os bancos de dados relacionais perdem algumas de suas principais funcionalidades ao escalar horizontalmente. Como alternativa pode-se escalar verticalmente, utilizando um hardware mais poderoso, porém isso não é uma solução definitiva devido aos custos envolvidos, e o limite da



capacidade desse hardware em atender grandes volumes de dados.

Figura 1. Motivos que levam a migração das bases de dados (Howard, 2007).

A partir da necessidade de escalar os sistemas de banco de dados relacionais, é necessário planejar, analisar riscos, propor tecnologias para os dados sejam migrados para tecnologias que possam escalar os dados.

Como objetivo geral este trabalho apresenta um processo de migração de dados relacionais para NoSQL de forma a tornar escalável diferentes coleções de

dados relacionais. Para isso, os seguintes objetivos específicos devem ser atingidos: estudo e entendimento dos conceitos e particularidades dos bancos de dados NoSQL, estudo das técnicas e regras utilizadas para migração, como é feita a construção de rotinas ETL (*Extract*, *Transform and Control*), e por fim, construção de rotinas ETL para migração de banco de dados SQL para NoSQL.

Esse trabalho está baseado em uma metodologia bibliográfica com aplicação de um exemplo de carácter exploratório. Assim, conforme Silva e Menezes (2005), a pesquisa exploratória envolve diversas atividades, tal como a pesquisa bibliográfica, que tem por objetivo o levantamento de referencial teórico existente na literatura, através da consulta de artigos, livros e revistas da área.

O trabalho está organizado da forma que segue. A seção 2 apresenta os conceitos teóricos base desse trabalho, tal como as características e tipos de bancos de dados NoSQL, trabalhos relacionados, técnicas de migração de banco de dados e construção de rotinas. A seção 3 descreve as regras para migração que são abordadas na literatura. A Seção 4 apresenta um um exemplo de migração de um banco de dados relacional para um NoSQL. Finalmente, a Seção 5 faz as considerações finais e descreve os trabalhos futuros.

2 REFERENCIAL TEÓRICO

Apresentamos nessa seção os conceitos teóricos em que o trabalho está baseado, tal como banco de dados NoSQL, técnicas de migração que são utilizadas, construção de rotinas ETL e uso de ferramentas, bem como alguns trabalhos relacionados.

2.1. Banco de dados NoSQL

Segundo Fowler (2015) os bancos de dados NoSQL são os que podem ser utilizados distribuídos em hardware comum, e que precisam de um rigoroso esquema para os registros, também não utilizam o modelo matemático em que os bancos relacionais são baseados.

Para Kabakus e Kara (2017) os bancos NoSQL são baseados no princípio BASE (*Basically Available, Soft-state, Eventually consistent*), cujo objetivo é obter um melhor desempenho, escalabilidade e disponibilidade. Já os bancos de dados relacionais, segundo os autores, são baseados no modelo ACID (*Atomicity, Consistency, Isolation, Durability*) para que seja mantida a consistência e integridade

dos dados. Segundo Fowler (2015), nos últimos anos vários bancos de dados NoSQL também tem adicionado suporte ao modelo ACID em seus sistemas.

A atomicidade diz a respeito a execução de todas as operações em caso de sucesso, e em caso de falha, que nenhuma operação seja feita, ou seja, a base de dados não pode refletir resultados parciais das transações. Um exemplo seria uma transação envolvendo transferência de recursos entre 2 contas bancárias, que envolve várias operações como debitar em uma conta e creditar em outra, onde em caso de erro nenhuma transferência seria feita.

Quanto a consistência temos que as devem ser respeitadas as regras de integridade dos dados, tal como restrições de integridade lógica, unicidade das chaves, entre outros. Assim o banco de dados deve ser levado de um estado consistente a outro, sendo que os dados escritos devem ser válidos conforme as regras e restrições definidas. Um exemplo seria o CPF ser utilizado como chave primária, assim esse registro não pode ser duplicado nem cadastrado com um valor inválido (000.000.000-00).

Já o isolamento trata de que transações paralelas não interfiram umas nas outras, pois várias transações podem acessar de forma simultânea o mesmo registro ou parte dele. Assim o resultado das transações em paralelo deve ser executado de forma sequencial

Finalmente a durabilidade deve garantir que as transações concluídas sejam mantidas no banco de dados mesmo que o mesmo apresente problemas, como travamento ou queda de energia. Isso também garante que os dados estejam disponíveis quando solicitados.

O modelo ACID é previsto nos bancos de dados distribuídos, através do Teorema CAP, criado pelo Dr. Eric Brewer, na ACM (*Association for Computing Machinery)* segundo Reis (2019), descrevendo o comportamento do sistema quando ocorrem requisições de escrita de dados seguida de requisições de consulta.

Assim, conforme o Teorema CAP demonstrado na Figura 2, o sistema de banco de dados deve possuir os 3 seguintes comportamentos: consistência, disponibilidade e tolerância a partição. A consistência garante que sempre seja retornado o dado mais atualizado quando solicitado, a disponibilidade trata a respeito do banco de dados sempre retornar um valor, desde que exista pelo menos

um servidor em execução. E a tolerância a partições fala sobre a tolerância à partição dos dados na rede, e obrigatoriamente ocorre quando se tem um sistema distribuído.

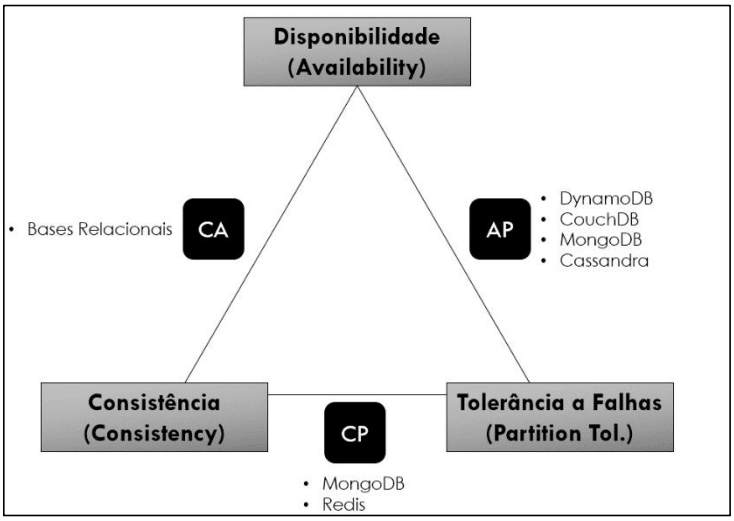


Figura 2: Relação entre os elementos no Teorema CAP (do Autor).

O Teorema de CAP demonstra que somente dois dos três comportamentos podem ser assegurados ao mesmo tempo, sendo assim tem-se as seguintes configurações:

- **CD:** Nesse temos os sistemas de base de dados relacionais, que defendem a consistência e disponibilidade em um sistema não distribuído.
- **CT:** Nesse temos um sistema com banco de dados distribuído, e em caso de falha, é assegurado a consistência ao custo da disponibilidade.
- **DT:** Nesse temos um sistema com banco de dados distribuído, e em caso de falha, é assegurado a disponibilidade ao custo de consistência.

Conforme Rockenbach et al. (2018), o armazenamento primário dos bancos de dados NoSQL pode ser tanto na memória RAM quanto em discos, inclusive os modelos podem ser utilizados em conjunto. A escolha do local de armazenamento deve-se muito ao fato do desempenho, onde conforme Abramova et al. (2014), os sistemas com armazenamento em memória volátil para mapeamento de registros tem um desempenho significativamente melhor, tal fato deve-se que a leitura e gravação de dados em discos rígidos é milhares de vezes mais lenta do que a operação executada em memória do computador, segundo (LAKE e CRIWTHER, 2013).

Os autores Rockenbach et al. (2018) afirmam que os bancos NoSQL podem ser classificados em 4 categorias, conforme suas otimizações, sendo: chave-valor (*key-value*), orietados a documentos (*document*), família de colunas (*column family* ou *columnar*) e banco triplo (*graph database* ou *triple*). Na Figura 3 apresenta uma imagem representativa desses modelos.

Os bancos classificados como chave-valor são aqueles cuja informação armazenada possui sua respectiva chave, de acordo com Rockenbach et al. (2018). Um exemplo é a Amazon, que utiliza o sistema de *key-value* Dynamo para gerenciamento das seguintes funcionalidades: listas de mais vendidos, carrinhos de compras, preferências do consumidor, gerenciamento de produtos, entre outros, conforme DeCandia et al. (2007). Segundo Pokorny (2013) são os bancos NoSQL com as estruturas mais simples dentre os demais, onde a visualização da base de dados pode ser através de uma tabela *hash. G*eralmente são os que possuem armazenamento em memória RAM, devido a sua simplicidade na estrutura. Carlson (2013) também afirma que como é um tipo de banco largamente adotado, ele pode auxiliar outros sistemas de armazenamento como ser utilizado para armazenamento primário. E como desvantagem, Rockenbach et al. (2018) cita que o modelo não permite a recuperação de objetos através de consultas mais complexas, como pesquisas com *join*, por exemplo.

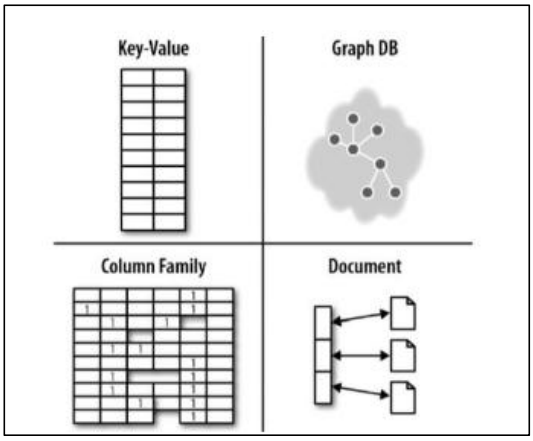


Figura 3. Representação gráfica dos diferentes tipos de bases de dados NoSQL (Robinson et al., 2015).

Já os bancos de dados de famílias de colunas, segundo Rockenbach et al. (2018), tem uma estrutura similar as bases de dados relacionais, porém as informações são armazenadas em colunas ao invés de linhas. Assim, de acordo com Nayak et al. (2013) os dados ficam armazenados em arquiteturas que são

distribuídas massivamente, onde cada chave está associada a um ou mais atributos (colunas) e as informações podem ser agregadas rapidamente através de atividades de entrada e saída, ficando assim guardadas. Esse tipo de banco de dados encoraja a duplicidade dos dados (desnormalização), para que haja um ganho na velocidade de leitura, pois não há a necessidade na reconstrução dos mesmos, de acordo com Fowler (2015). Como desvantagem há uma demora maior na alteração dos dados, conforme Nayak et al. (2013). Alguns exemplos de bancos nessa categoria são o Cassandra e a Hbase, ambos da Apache.

Existem bancos de dados orientados a documentos têm as informações armazenadas sobre uma estrutura de árvore, utilizando formatos como XML ou JSON, e ferramentas que utilizam essa estrutura são MongoDB e CouchDB, conforme Rockenbach et al. (2018). A estrutura deste tipo de banco de dados é semelhante aos de chave valor, porém com valor estruturado e com hierarquia. Seu suporte ao processamento e análise de dados é bem variado, fornecendo operações de escrita em tempo constante, com utilização do método *append-only* para armazenamento dos dados. Um exemplo de utilização do MongoDB é demonstrada pelo governo de Chicago, que construiu uma plataforma para gerenciamento de operações inteligentes com base no MongoDB, denominada WindyGrid, de acordo com Goldsmith e Crawford (2014) essa ferramenta foi projetada para trabalhar com o grande volume de dados que é gerado pela cidade, sendo a visualização dos mesmos feita através de um mapa da cidade, que mostra chamadas para serviços de emergência, informações de construções, mensagens públicas de redes sociais e demais dados críticos, conforme Thornton (2013).

Por fim, o banco de dados de grafos ou triplos fazem o armazenamento das informações compostas através de 3 elementos, conforme Fowler (2015), Kabakus e Kara (2017), sendo: propriedade ou relacionamento, sujeito e valor. O foco desse tipo de banco de dados é a criação de relacionamentos entre os dados, onde os dados são armazenados em registros triplos, compostos por sujeito, predicado e objeto, conforme Rockenbach et al. (2018). Rockenbach et al. (2018) cita que através desse tipo de banco é possível a construção de redes complexas de relacionamentos e com inferência para revelação de novos dados. Um projeto atual que busca padronizar a camada de comunicação entre a aplicação e banco é o Apache TinkerPop (APACHE TINKERPOP, 2008).

2.2. Técnicas de Migração de Banco de Dados

Oliveira (2017) menciona a complexidade das tarefas de migração de base de dados, pois além dos passos envolvidos ainda é necessário que os ambientes e bases continuem disponíveis para utilização durante o procedimento, e que o mesmo tenha baixo impacto no funcionamento dos sistemas.

A migração é dividida em fases ou janelas temporais, sendo cada uma constituída por um conjunto de procedimentos com início, meio e fim, conforme Oliveira (2017). São elas:

1. Informacional: visa reunir informações antes do início da migração, como por exemplo o estado da base de dados e os dados que serão migrados. Ao final é feito um levantamento para ver se todos os dados foram migrados.
2. Execução do procedimento: para isso é necessário ter as informações acerca da base de origem e destino. Após isso são extraídos os dados e arquivos da base de origem e eles são duplicados. Um coletor faz o processo de comunicação do software de rede e a replicação entre os servidores de origem e destino. Um processo de extração online pode ser utilizado na extração de todas as DDL e DML feitas desde o início do processo, sendo esse enviado para o destino e servindo como referência para a continuidade dos procedimentos. Algumas ferramentas também trabalham com a exportação dos dados via Ms-Excel, porém esse é um processo que deve ser feito de forma manual. Um exemplo visual simplificado do processo é apresentado na Figura 4.

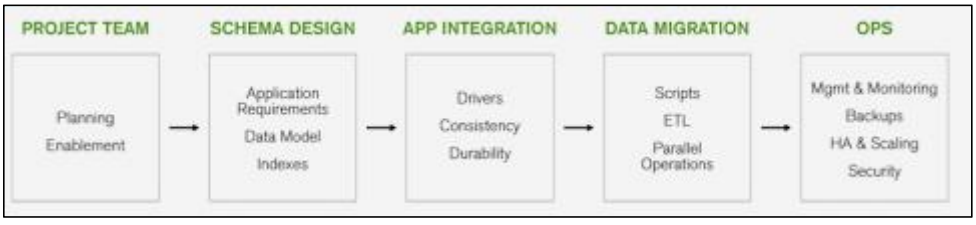
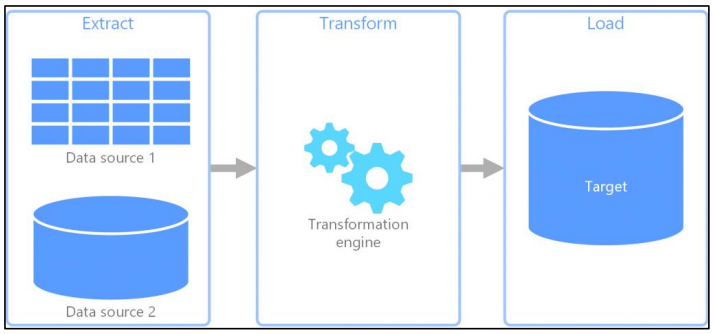


Figura 4. Exemplo simplificado dos passos de migração (Oliveira, 2017).

2.3. Construção de Rotinas ELT (*Extract, Transform and Load*)

Devido a complexidade das tarefas de migração, surge o questionamento de aplicações que possam ser utilizadas para auxiliar no processo de construção deste ETL. Krober (2017) menciona que, para otimizar a construção de rotinas ETL existem algumas ferramentas disponíveis no mercado, tal como o Pentaho, Microsoft SQL Integration, Talent Studio, entre outros. E destaca o uso dessas ferramentas como uma forma de otimizar o desenvolvimento, bem como o uso de scripts personalizados.

Podemos assim definir a rotina ETL confome demonstrado na Figura 5. Onde o procedimento de extração faz a extração de informações relevantes de uma base em seu formato bruto, ou seja, um dado que não foi tratado ou que não teve seu formato validado, convertendo esses dados para o modelo utilizado no negócio, e



por fim fazendo seu carregamento no sistema de destino.

Figura 5. Rotina ETL (Naeem, 2020).

Kimball (2004) afirma que cada projeto possui um cenário diferente, e assim a escolha das ferramentas a serem utilizadas precisam levar isso em consideração. Citamos assim algumas das vantagens do uso de ferramentas de ETL segundo autor: redução de custos no desenvolvimento, facilidade de uso por pessoas não técnicas, integração e geração de metadados, tratamento de exceções e erros por parte das ferramentas, existência de conectores pré configurados para a maior parte dos sistemas de origem e destino, bom desempenho em grandes bases de dados, gerenciamento de balanceamento de carga e escalabilidade.

Algumas das ferramentas ETL são citadas abaixo:

- ***Pentaho:*** é um software de código aberto desenvolvido em Java em 2004, sendo considerado uma das melhores ferramentas para inteligência

empresarial pelo InfoWord (2011). Seu objetivo é o de fazer análises de big data, trabalhando nativamente com bancos NoSQL e Hadoop.

- **Talend**: lançado em 2006 e possui conectores para boa parte das aplicações existentes no mercado, consegue gerenciar todo o ciclo de vida como solução ETL.
- **Microsoft Integration Services**: componente de banco de dados da Microsoft que pode ser utilizada para migração de vários tipos de dados, trabalhando com hierárquia de componentes.

2.4. Trabalhos Correlatos

No trabalho de Neto et al. (2015) é apresentado um estudo relacionado a migração de dados, onde os autores compactam os resultados em uma tabela de requisitos para realizar a migração. Outro trabalho é o de Antaño et al. (2014), que apresenta algumas métricas usadas na migração de bases de dados relacionais para NoSQL.

Uma visão de 15 sistemas NoSQL é apresentado por Deka (2014), mencionando sistemas como Cassandra, BigTable, Pnuts, Redis, entre outros. Apesar de falar sobre vários detalhes de cada um dos sistemas, o trabalho do autor não oferece um estudo comparativo entre os mesmos.

O trabalho dos autores Rockenbach et al. (2018) apresenta um extenso estudo comparativo acerca dos tipos de bases de dados NoSQL, passando por suas características mercadológicas, de projeto e manutenção. Tal estudo auxilia na escolha por parte dos gestores para qual tipo de banco mais adequado para fazer a migração de um banco de dados relacional ou adotar, pois é necessário avaliar cada um conforme as características do projeto, devido a grande diferença de categoria entre as bases de dados.

Zhang et al. (2015) tem por foco em sua pesquisa apresentar uma visão estruturada de vários sistemas NoSQL (MemepiC, RAMCloud, Redis, Memcached, MemC3 e TxCache), abordando o gerenciamento e processamento dos dados em memória através da descrição de qual carga de dados é mais adequada a cada sistema, tratamento para grandes conjuntos de dados maiores do que a memória disponível, uso de consultas personalizadas em linguagem nativa e finalmente aborda estratégias para construção de índices, controle de concorrência e tolerância a falhas.

3 REGRAS PARA MIGRAÇÃO DE BANCO DE DADOS SQL PARA NOSQL

A migração pode ser feita de 3 formas, conforme Oliveira (2017), migração online com sincronização contínua, migração offline através de ferramenta e migração offline através de scripts. A Figura 6 apresenta essas opções.

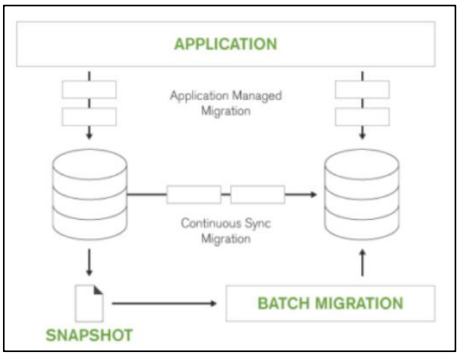


Figura 6. Opções para migração de dados SQL para NoSQL (Oliveira, 2017).

3.1. Características

A migração online com sincronização contínua é a migração ocorrer em paralelo a execução da aplicação, tem seu fluxo demonstrado na Figura 7, e segundo Oliveira (2017) a aplicação nunca interrompe o acesso a base de dados, sendo que ao final da migração somente seu direcionamento é alterado, com perda de serviço miníma. Nesse processo, apesar do modelo estar sempre utilizando a base de dados, é necessário realizar algumas paradas para validação e também para verificar se os dados não estão sendo criados e duplicados. A vantagem desse procedimento é que os novos dados que vão entrando no sistema já podem ir sendo direcionados para a nova base de dados, porém em caso de falha ou qualquer problema o sistema pode ficar comprometido por tempo indeterminado.

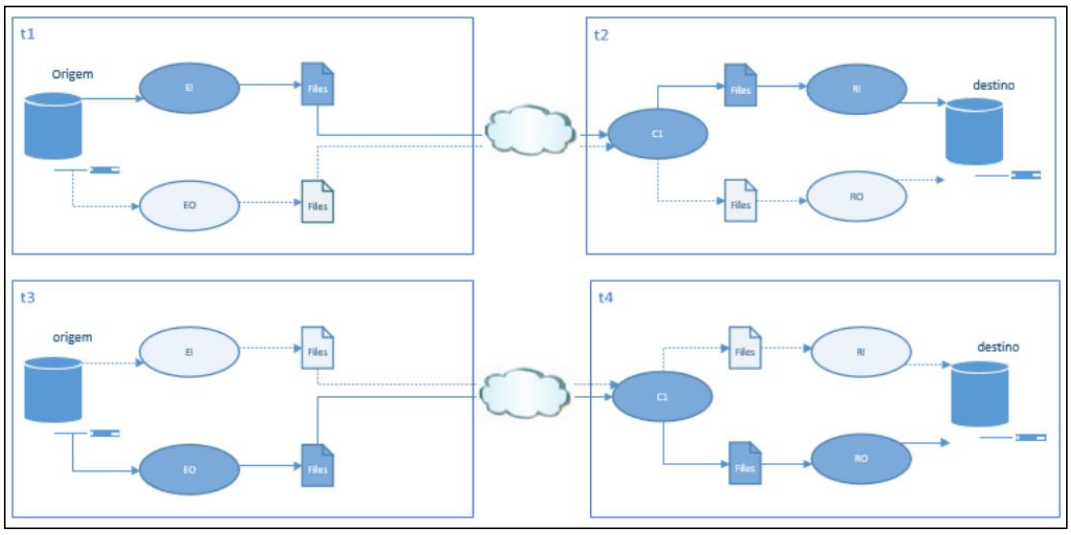


Figura 7. Método de migração online (Oliveira, 2017).

Já o método de migração offline através de ferramenta, utilizada nesse trabalho e demonstrada na Figura 8, é baseada no modelo online, tendo como diferença de que a migração é feita todo em um único carregamento, tendo controle manual em cada passo e sendo realizada através de um software de migração. Como vantagens dessa técnica temos que o sistema não ficará comprometido caso ocorra alguma falha no processo, é possível controlar e validar cada etapa de forma independente, bem como testar todo o novo ambiente antes de fazer a configuração do sistema. Porém o sistema pode ficar indisponível enquanto o processo não for concluído.

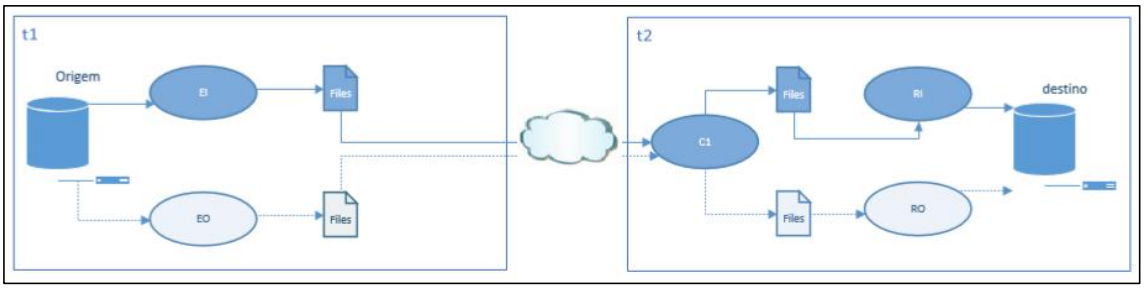


Figura 8. Método de migração através de ferramenta (Oliveira, 2017).

Por fim, a última forma é a migração offline através de *script*, exigindo interação do técnico em todas as etapas, de forma que serão analisados como os dados serão extraídos, armazenados, enviados, carregados, entre outros. De todas as formas é a mais trabalhosa mas que permite maior personalização dos processos, isso pois cada etapa terá que ser tratada e programada de forma individual, sem o auxílio visual de ferramentas.

O processo de migração entre SQL e NoSQL deve considerar ainda as adaptações que serão necessárias, conforme exemplo entre SQL e NoSQL apresentada na Figura 9, onde a tabela passará a ser uma coleção, as chaves estrangeiras serão incorporadas, entre outros. De acordo com Sirqueira e Dalpra (2018) cada coluna será representada como um campo do documento BSON, podendo conter uma lista de valores, um objeto ou um único valor.

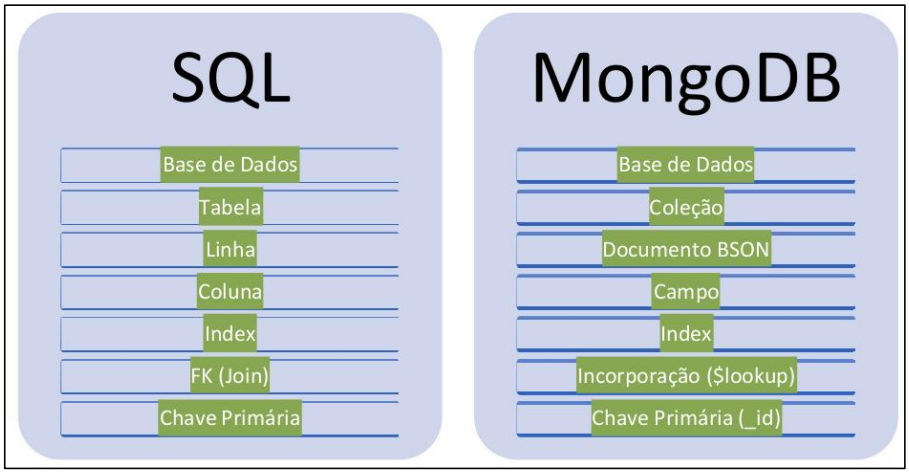


Figura 9. Conceitos entre SQL e NoSQL (Sirqueira e Dalpra, 2018).

Para facilitar a migração, Oliveira (2017) define alguns passos que podem ser aplicados, sendo: planejamento e definição das métricas que serão utilizadas, quantidade de registros e situação atual do banco de dados, mapeamento dos tipos de dados e suas equivalências, restrições, codificações, ensaio, migração e finalmente, monitoramento final.

No planejamento é analisado os requisitos para migração e é onde são analisadas todas as tarefas que devem ser feitas, envolvendo a parte funcional, desenvolvimento, infraestrutura e validação das necessidades do negócio de forma a escolher a melhor técnica de migração a ser utilizada. As métricas serão definidas de forma a avaliar como foi o processo.

A situação inicial da base de dados é importante para poder conferir ao final do processo se todos os dados foram migrados com sucesso. Caso a técnica utilizada seja a online, Oliveira 2017] sugere a utilização da validação temporal, pois como haverá registros sendo adicionados utilizar essa quantidade como métrica não é uma informação precisa.

O mapeamento dos dados deve ser feito de forma a verificar se os dados precisarão ser convertidos no destino, sendo que no caso de não haver uma forma correspondente, Oliveira (2017) cita as seguintes alternativas: conversão do dado para algum tipo que seja aceito pela aplicação de destino antes da migração, conversão durante a migração ou excluído. Essa conversão dos dados não pode ser feita após o processo de migração ser concluído, devendo ser feito ou durante ou antes, conforme Oliveira (2017), que ainda ressalta que caso existam tipos de dados não compatíveis nos dois bancos de dados todo o processo de migração ou alguns dados podem ser perdidos.

Alguns bancos de dados possuem a função de “disparar operações” após alguma ocorrência, conhecidas como “triggers”, que por vezes são utilizadas para auditoria, e para esses casos devem ser desativadas durante o processo de migração, caso durante a migração seja executado alguma alteração na origem, afim de evitar uma auditoria na própria migração. Caso na base NoSQL que será utilizada não existir essa configuração deve-se buscar outra forma de implementação.

Deve-se verificar se o padrão dos dados da origem será a mesma do destino, principalmente em se tratando de bases de dados que armazenam informações de vários idiomas, segundo Oliveira (2017). Caso seja verificado que são codificações diferentes a conversão deve ser feita, similarmente a conversão de tipo de dados apresentada anteriormente.

Antes da migração ser de fato realizada, pode-se realizar ensaios, cujo objetivo é tentar simular da forma mais precisa possível os problemas que podem ocorrer durante o processo, principalmente no caso da migração online. Oliveira (2017) sugere que essa etapa seja feita através de amostragem de dados e que todo ambiente seja testado, e que seja possível voltar ao ambiente original após os testes.

Para o momento da implementação, todo o ambiente deve ser controlado, as interações, execuções e atividades, devendo ser seguidas conforme o planejamento elaborado. Também deve-se avaliar no momento se a migração deve ser abandonada ou não, em caso de falhas, bem como o processo de retrocesso ao ambiente original.

Finalmente, após a implementação ter sido concluída, deve-se validar todo o ciclo, a quantidade de dados que foi migrada conforme a base de dados original, para somente então alterar os dados de conexão nos arquivos. Importante que o processo seja monitorado por algum tempo para verificar se não vai acontecer nenhum problema, principalmente quando se é modificado tipos de dados ou codificação das informações.

4 CONSTRUÇÃO DE ROTINAS ETL PARA MIGRAÇÃO DE DADOS SQL PARA NOSQL

Nessa seção são apresentados os procedimentos que são utilizados para realizar a migração da base SQL para a NoSQL com base na técnica de migração offline através de ferramenta.

4.1. Criação de um banco de dados de exemplo

O banco de dados apresentado na Figura 10 é utilizado como exemplo para a migração. É apresentada uma tabela com 3 (três) relacionamentos e 1 (um) relacionamento que representa a conexão com mais uma tabela. Exemplificado a modelagem de uma empresa na área financeira onde os dados são resultante da execução do processo de empréstimo. Desta forma, tem-se um registro de LoanApplication para as solicitações de empréstimos, que possui uma lista de Proposals para as propostas da financeira e um AcceptedProposal para a proposta aceita. E um registro de Loan para os empréstimos efetivados com um ApprovedPlan para os dados originais do plano de pagamento.

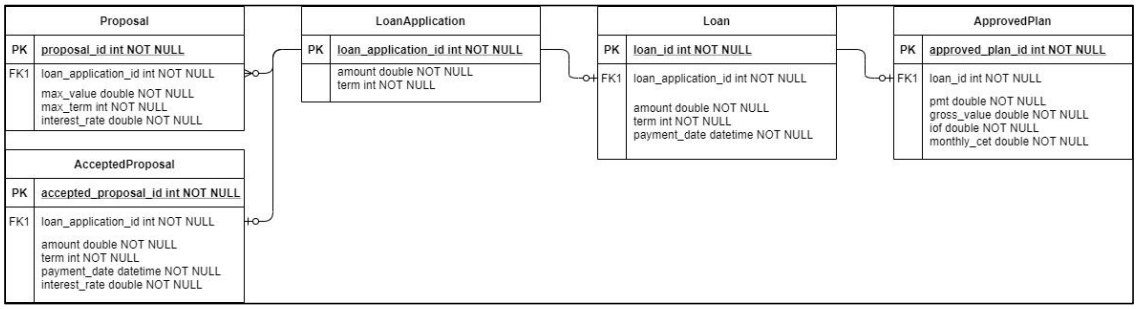


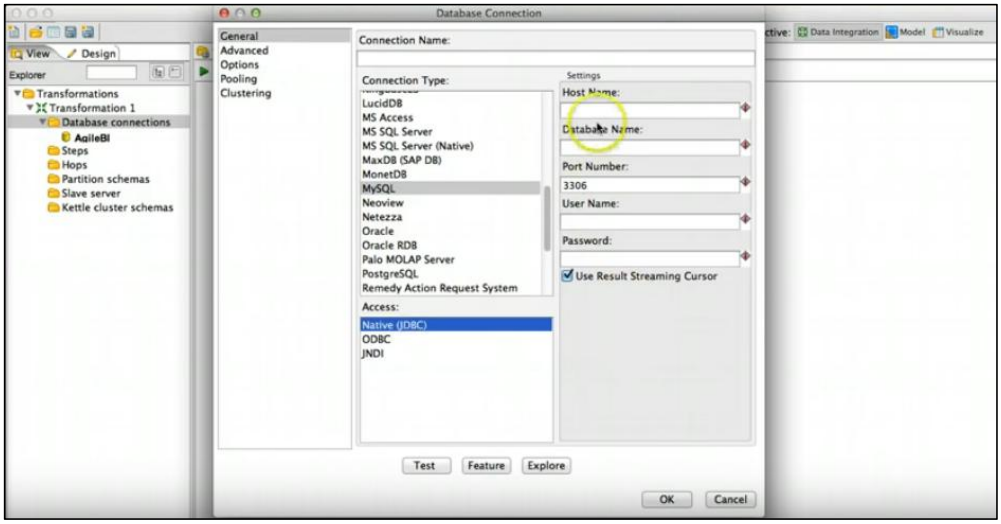
Figura 10. Modelo de banco de dados (do Autor).4.2. Rotinas ETL para migração de dados SQL para NoSQL (do Autor).

4.2. Execução da rotina de migração

A migração foi feita utilizando a ferramenta Pentaho Data Integration (PDI), que contém os processos ETL do Pentaho, esse é um processo independente que pode ser utilizado sem os demais softwares da plataforma, que cobrem tarefas de mineração de dados (data-mining), geração de relatórios (reporting) e análise e manipulação de grandes volumes de dados (OLAP). A ferramenta possui 4 componentes (spoon, carte, kitchen e pan), sendo o Spoon o utilizado para transformações e trabalhos dos ETL's de forma visual.

Na Figura 11 é criado uma nova base de dados e então iniciado o processo de recuperar as informações de outro banco de dados. Na tela apresentada na Figura 12 pode-se escolher de qual banco de dados as informações serão

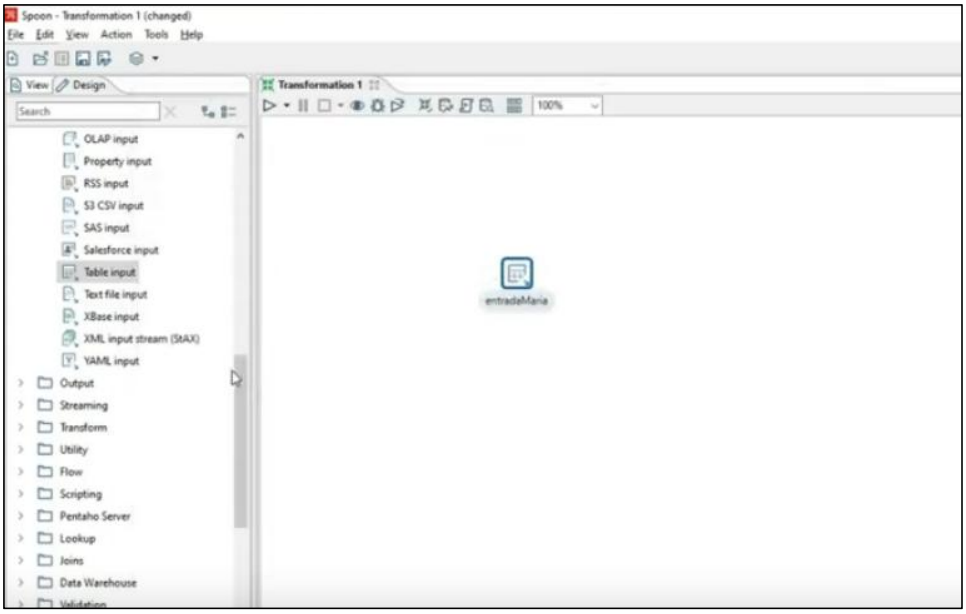
recuperadas e informar as credenciais de acesso, no caso faremos a migração de



uma base MySQL.

Figura 11: Seleção da base de dados para migração (do Autor).

No caso, foi necessário fazer o download do conector independente do MySQL em seu site, e adicionar o arquivo .jar dentro da pasta lib da ferramenta. Em seguida, utilizamos a parte de design da ferramenta para configurar a entrada dos



dados, ou seja, através do MySQL, conforme indicado na Figura 10.

Figura 12: Configuração da entrada de dados (do Autor).

Após isso, selecionamos a saída dos dados, que no caso será MongoDB, e arrastamos a caixa de entrada na saída, criando uma conexão entre as bases de

dados, e fazemos a configuração da base de saída, conforme Figura 13, e conforme base e coleção configurados anteriormente.

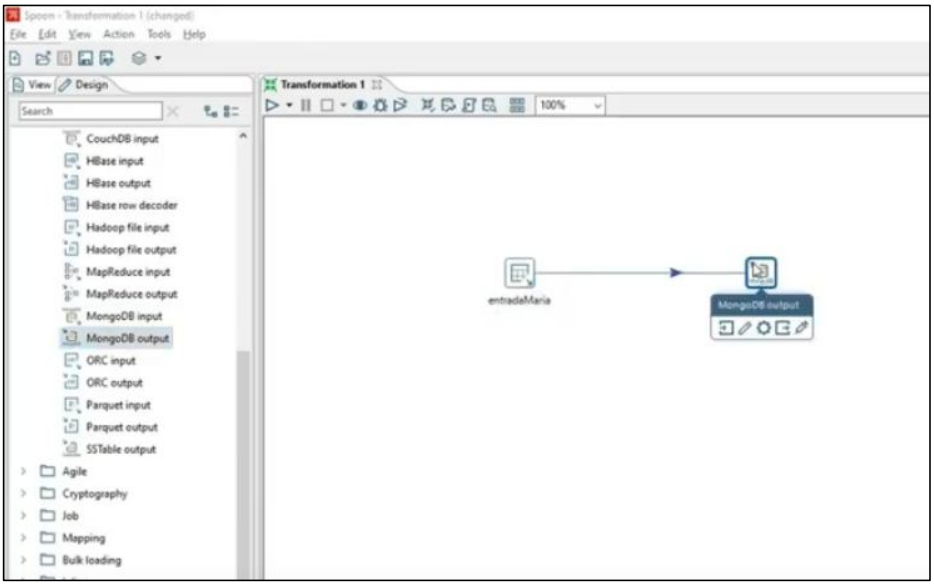


Figura 13: Ligação entre base de entrada e saída (do Autor).

Ao informar os dados do MongoDB, já é carregado as informações de campos vindos do MySQL, o que pode ser visualizado na Figura 14.

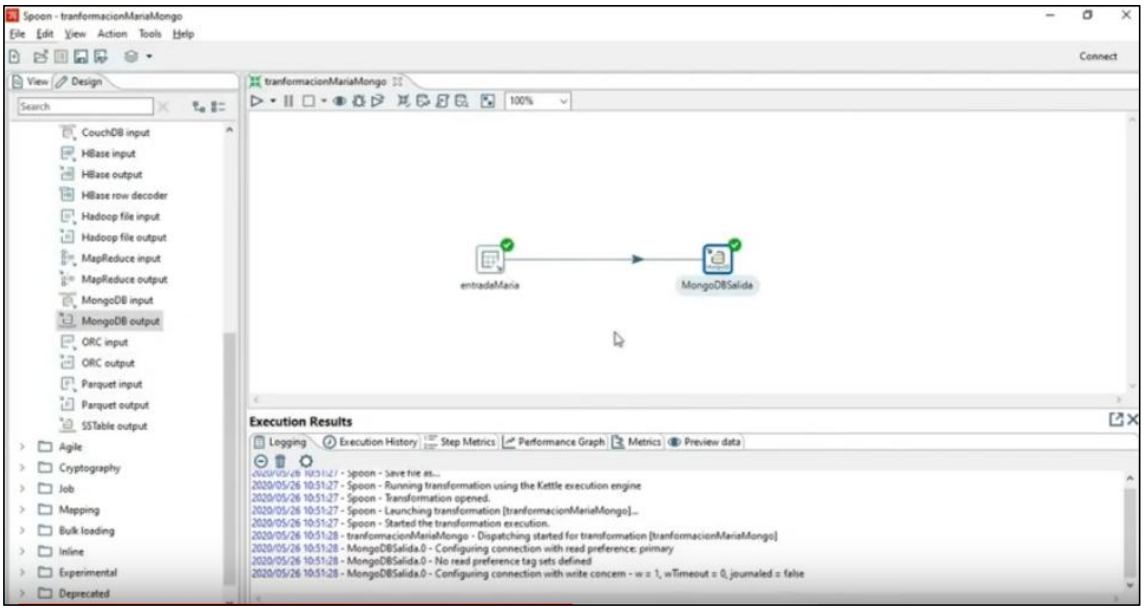


Figura 14: Conexão com sucesso entre entrada e saída (do Autor).

Após as tabelas já estarem sendo lidas pelo MongoDB, iniciamos um “job”, que fará a transformação dos dados, essa tarefa pode ser feita através do modo design da ferramenta também, conforme demonstrado na Figura 15. Também é possível criar um intervalo para que a transformação aconteça e então regras, como por exemplo, caso o dado ainda não exista fazer sua criação, caso já exista fazer sua atualização.

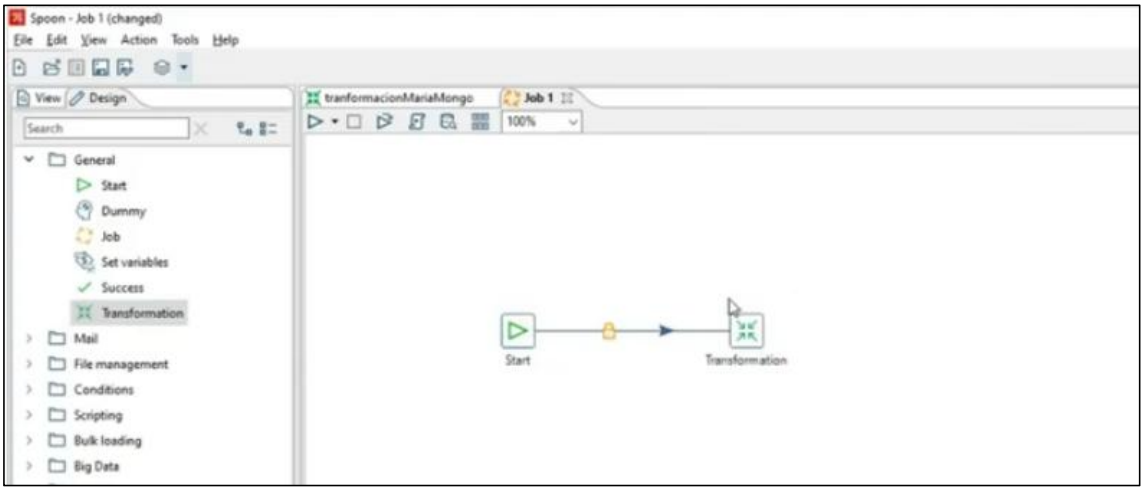


Figura 15: Criação do trabalho de transformação dos dados (do Autor).

Por último, deve ser atualizado as configurações da aplicação com os novos dados de acesso, o que a tornará indisponível por um breve período de tempo. O processo de validação é feito através da conferência da quantidade de registros na base de origem com a quantidade de registros na base de destino.

5 CONSIDERAÇÕES FINAIS

O crescimento das aplicações faz com que, ao se utilizar bases de dados relacionais, as consultas muitas vezes fiquem lentas devido a quantidade de registros, tendo em vista também que todos os registros ficam armazenados em um único local, e para escalar os serviços é necessário a duplicação de toda a base de dados.

Assim, faz-se necessário a migração de uma base de dados relacional para uma NoSQL, cujas principais vantagens são a escalabilidade e a possibilidade de trabalhar com uma grande quantidade de registros. Porém o processo de migração muitas vezes não é tão simples, pois dependendo da quantidade de relações entre as tabelas e da quantidade de dados pode se tornar um processo muito complexo.

Apesar da dificuldade apresentada, caso haja a necessidade de buscar essa velocidade e disponibilidade, a migração hoje é um dos melhores e por vezes o único caminho a ser seguido devido as limitações de escalabilidade dos bancos de dados relacionais.

Foi apresentado um exemplo de migração de base de dados PostgreSQL para NoSQL, através do uso da ferramenta ETL Pentaho Data Integration (PDI), que auxilia na transformação dos dados.

Como principal vantagem do processo mencionado temos a alta disponibilidade da base de dados migrada, possibilidade de escalabilidade e de trabalhar com uma grande quantidade de dados. Como desvantagem temos que a aplicação tende a ficar alguns minutos offline enquanto é atualizada a configuração do banco de dados, e uma possível complexidade da tarefa de migração.

Como trabalhos futuros, pode ser feito uma análise de performance do processo de migração e explorar as demais técnicas de migração.

ABSTRACT

Many information systems are developed using Relational Database Management Systems (DBMS). Relational databases allow you to manage data in a structured way through tables and formats that hinder the scalability of information systems to store different types and formats of data. As the volume of data increases, it is necessary to use more robust databases, such as based on NoSQL. This work aims to establish a migration process from relational database to NoSQL. It presents as advantages and disadvantages involved in the process steps. A common case study is presented to implement the proposed process.

REFERÊNCIAS

ABRAMOVA, Veronika; BERNARDINO, Jorge.; FURTADO, Pedro. **Which NoSQL Database? A Performance Overview**. Open Journal of Databases: RonPub, 2014. p. 17–24.

ANTAÑO, Manjarrez et al. **Migración de Bases de Datos SQL a NoSQL**. Revista Tlamati Sabiduria 2014. p. 144-148.

APACHE TINKERPOP. **Apache TinkerPop**, 2008. Disponível em <http://tinkerpop.apache.org/>. Acesso em: 15 de out. 2020.

CARLSON, Josiah. L. **Redis in Action**. Manning, Shelter Island, NY, USA, 2013

DECANDIA, Giuseppe et al. **Dynamo: amazon's highly available key-value store**. ACM SIGOPS operating systems review, 2007. p. 205–220.

DIANA, Mauricio de; GEROSA, Marco Aurélio. **NOSQL na Web 2.0: Um Estudo Comparativo de Bancos Não-Relacionais para Armazenamento de Dados na Web 2.0**. 2012. Disponível em <200.17.137.109:8081/novobsi/Members/josino/fundamentos-de-banco-de-dados/2012.1/sbbd_wtd_12.pdf>. Acesso em: 09 de out. 2020.

DEKA, Ganesh Chandra. **A survey of cloud database systems**. IT Professional, 2014. p. 50–57.

OLIVEIRA, Fábio Vieira de. **Migração de bases de dados relacionais para NoSQL - Métodos de Análise**. Instituto Universitário de Lisboa, 2017.

REIS, Fábio dos. **Conceitos de bancos de Dados – O Teorema CAP**. 2019. Disponível em <http://www.bosontreinamentos.com.br/bancos-de-dados/conceitos-de-bancos-de-dados-o-teorema-cap/>. Acesso em: 26 de out. 2020.

FOWLER, Adam. **NoSQL For Dummies**. John Wiley & Sons, 111 River Street, Hoboken, New Jersey, USA, 2015

GOLDSMITH, Stephen; CRAWFORD, Susan. **The Responsive City: Engaging Communities Through Data-Smart Governance**. John Wiley & Sons, 111 River Street, Hoboken, New Jersey, USA, 2014

HOWARD, Phillip. **Como garantir o sucesso nas iniciativas de Migração de Dados**. Informatica. 2007.

INFOWORD. **Best of open source enterprise applications.** 2011. Disponível em <http://www.infoworld.com/%5Bprimary-term-alias-prefix%5D/%5Bprimary-term%5D/best-open-source-enterprise-applications-885&current=9&last=8&auto=y#slideshowTop>. Acesso em: 15 de out. 2020.

KABAKUS, Abdullah T.; KARA, Resul. **A performance evaluation of in-memory databases**. King Saud University - Computer and Information Sciences, 2017. p. 520–525.

KIMBALL, Ralph; CASERTA, Joe. **The data warehouse ETL toolkit: practical techniques for extracting, cleaning, conforming and delivering data**. Indianapolis: Wiley, 2004.

KROBER, Marcel **Um estudo comparativo entre o uso de bases de dados relacionais e não relacionais para datawarehouses**. Univantes, 2017.

LAKE, Peter; CROWTHER, Paul. **Concise guide to databases**. Springer-Verlag London, 111 River Street, Hoboken, New Jersey, USA, 2013.

MANYIKA, James et al. **Big data: The next frontier for innovation, competition, and productivity**. McKinsey Global Institute, 2011.

NAEEM, Tehreem. **Visão geral do processamento de dados ETL**. 2020. Disponível em <https://www.astera.com/pt/tipo/blog/dados-etl/>. Acesso em: 09 de out. 2020.

NAYAK, A.; PORIYA, A.; POOJARY, D. **Type of NOSQL Databases and its Comparison with Relational Databases**. International Journal of Applied Information Systems, 2013. p. 16–19.

NETO, J.R.; Passos E.B. **JExodus: uma ferramenta para migração de dados independente de SGBD**. Teresina: Instituto Federal de Educação e Ciência e Tecnologia do Piauí, 2009.

OLIVEIRA, Fábio; OLIVEIRA, Abílio; ALTURAS, Bráulio. **Migration of Relational Databases to NoSQL** - Methods of Analysis. Mediterranean Journal of Social Sciences, 2018. p. 227-235.

POKORNY, Jaroslav. **NoSQL databases: a step to database scalability in web environment**. International Journal of Web Information Systems, 2013. p. 69–82.

ROBINSON, Ian; WEBBER, Jim; EIFREM, Emil. (2015) **Graph databases: new opportunities for connected data**. 2.ed. O'reilly media, Inc, 2015. 238p.

ROCKENBACH, Dinei et al. **Estudo Comparativo de Bancos de Dados NoSQL**. 2018. Disponível em < https://revistas.setrem.com.br/index.php/reabtic/article/view/286/131>. Acesso em: 15 de out. 2020.

SILVA, Edna Lúcia da; MENEZES, Estera Muszkat. **Metodologia da Pesquisa e Elaboração de Dissertação**. 2005. Disponível em <https://projetos.inf.ufsc.br/arquivos/Metodologia_de_pesquisa_e_elaboracao_de_teses_e_dissertacoes_4ed.pdf>. Acesso em: 23 de out. 2020.

SIRQUEIRA, Tassio; DALPRA, Humberto. **NoSQL e a Importância da Engenharia de Software e da Engenharia de Dados para o Big Data**. Jornadas de Atualização em Informática, 2018. p. 58-98

THORNTON, Sean. **Chicago's windy grid: Taking situational awareness to a new level**. Data Smart City Solutions, 2013.

ZHANG, Hao et al. **In-Memory Big Data Management and Processing: A Survey**. IEEE Transactions on Knowledge and Data Engineering, 2015. p. 1920–1948.