Uma abordagem para migração de Banco de dados relacional para NoSQL Orientado a documentos

Tainam Spagnollo Garbin[1], **Denio Duarte**[1], **Geomar A. Schreiner**[1], **Samuel da Silva Feitosa**[1]

[1]Universidade Federal da Fronteira Sul (UFFS)
Campus Chapecó
Chapecó – SC – Brazil

tainamgbn@gmail.com, {duarte,gschreiner,samuel.feitosa}@uffs.edu.br

Abstract. *Migrating data from a relational model to a document-oriented model is a complicated process that involves multiple stages. This paper proposes a data denormalization methodology to migrate data from PostgreSQL to MongoDB. Our experiments have shown that our approach effectively migrates data between those different models, and the number of foreign keys significantly affects migration time.*

Resumo. *A migração de dados de um modelo de banco de dados relacional para um modelo orientado a documentos é complexa e envolve uma série de passos. Este artigo propõe uma metodologia baseada na desnormalização dos dados para realizar a migração de dados do PostgreSQL para o MongoDB. A migração envolve várias etapas, incluindo validação dos dados migrados. Os experimentos revelam que a abordagem é efetiva e a quantidade de chaves estrangeiras impacta significativamente o tempo de migração.*

1. Introdução

A constante popularização da utilização de serviços online trouxe consigo um expressivo aumento no volume de armazenamento das informações, bem como a necessidade de métodos mais eficazes para manipular essas informações. Muitas empresas demandam por bancos de dados (BDs) capazes de gerenciar esses grandes volumes de dados de forma eficaz e com um alto desempenho.

Os BDs relacionais (BDRs) tradicionais são utilizados há décadas para o armazenamento e manipulação de dados [Sadalage and Fowler 2013]. Uma das principais características deste modelo de dados é que a manipulação dos dados tem conformidade com as propriedades ACID (atomicidade, consistência, isolamento e durabilidade). Porém, manter a conformidade com estas propriedades em ambientes com grandes volumes de dados e em que alta disponibilidade é necessária se torna inviável. Novos modelos de BDs surgiram para preencher esta lacuna, os quais são focados, geralmente, em alta disponibilidade e baixo custo de operação [Schreiner et al. 2020].

Esses novos modelos de BDs são popularmente conhecidos como NoSQL. Esses BDs podem ser classificados segundo o modelo de dados como chave-valor, documentos, colunar e orientado a grafos [Sadalage and Fowler 2013]. Porém, apesar das vantagens apresentadas pelos BDs NoSQL a falta de padronização entre os modelos de dados e a

curva de aprendizado afetam, consequentemente, a sua adoção como principal meio de armazenamento de dados complexos.

A tarefa de migrar uma base de dados armazenada em um BDR para um BD NoSQL é complexa e onerosa. Entende-se migração por um processo de transferência de dados de uma plataforma/formato para outra plataforma/formato. Ou seja, é um processo de trazer dados de sistemas de origem para um sistema de destino [Sarmah 2018]. A migração de dados é um processo de várias etapas. Esse processo envolve analisar os dados do sistema de origem, mapear dados do sistema de origem para o novo sistema, projetar os programas de conversão, construir e testar os programas de conversão que conduzem a migração [Sarmah 2018].

Este trabalho é baseado na proposta de [Karnitis and Arnicans 2015] e apresenta uma abordagem que visa a migração de dados armazenados em BDR para BD NoSQL orientados a documentos. A abordagem é baseada em uma série de passos automatizados que analisa todas as tabelas do BDR e gera o mapeamento para estruturas de coleções de documentos utilizando o conceito de agregados, considerando as cardinalidades entre as tabelas para gerar a agregação. Para avaliar a solução proposta foram executados experimentos que migram dados armazenados no PostgreSQL para o MongoDB. Os experimentos demonstram que a solução migra as informações em sua totalidade com um desempenho não proibitivo.

O restante deste artigo está organizado da seguinte forma: a próxima seção apresenta brevemente as formas normais, o conceito de normalização e a estrutura do modelo de dados orientado a colunas. A Seção 3 apresenta uma discussão dos trabalhos relacionados. Em seguida, a Seção 4 apresenta a proposta desenvolvida e a Seção 5 os experimentos. A Seção 6 conclui este trabalho.

2. Referencial Teórico

Esta seção apresenta fundamentos teóricos que embasam este estudo sobre a migração de BDR para NoSQL orientado a documentos. Inicialmente, são apresentados, brevemente, os conceitos de formas normais e (des)normalização; e o modelo de dados orientado a documentos.

Formas Normais: as formas normais são conceitos de BDR para que as tabelas de um banco sejam projetadas, de tal forma em que os dados estejam normalizados. De maneira geral, as formas normais visam eliminar redundância de dados, minimizar erros de atualizações de informação e facilitar a consulta dos dados. As formas normais são regras pautadas na dependência funcional (DF) entre os atributos de uma tabela.

Uma DF é descrita como $\alpha \rightarrow \beta$ e pode ser lida: o conjunto (possivelmente unitário) de atributos β depende funcionalmente do conjunto (possivelmente unitário) de atributos α. Por exemplo, $CPF \rightarrow nome$ indica que o nome depende do valor do CPF, ou seja, todas as vezes que o banco armazenar um valor de CPF o valor armazenado no atributo *nome* deve ser o mesmo. Pode-se entender que *CPF* é uma chave para a tupla <*CPF*, *nome*>

Existem cinco formas normais: $1FN$, $2FN$, $3FN$, $4FN$ e $5FN$. Além das 5 formas normais, a BCNF (*Boyce–Codd normal form*) realiza uma correção sobre a $3FN$. Geralmente, na literatura, apenas três são utilizadas para verificar se um BD está norma-

lizado: $1FN$, $2FN$ e $BCNF$.

É importante ressaltar que as formas normais são hierárquicas, ou seja, uma tabela está na segunda forma normal ($2FN$), por exemplo, se e somente se respeitar a primeira forma normal ($1FN$). A seguir, as três formas normais mais comumente usadas para verificação do BD serão apresentadas.

Primeira Forma Normal ($1FN$): A primeira forma consiste em que todos os atributos de uma tabela não armazenem mais de um valor (multi-valorado). Caso um atributo armazena valores multi-valorados, deve-se decompô-lo até armazenar apenas valores atômicos. A decomposição pode ser feita de duas maneiras: (*i*) criar uma nova tabela ou (*ii*) criar novos atributos, separando os valores.
Segunda Forma Normal ($2FN$): a segunda forma contempla as regras referente a $1FN$ e acrescenta que um atributo não pode depender parcialmente da chave. Caso seja necessário, deve-se criar uma nova tabela com os dados que dependem funcionalmente de parte das chaves da tabela.
Forma Normal de *Boyce-Codd* (*BCNF*): Esta forma vem como uma correção a $3FN$ em que o objetivo é corrigir as anomalias de atualizações que ocorriam na $3FN$ quando houvesse a incidência de atributos não-chave que dependem funcionalmente e diretamente da chave primária. Uma tabela pode estar na $3FN$ e não necessariamente está na $BCNF$. A $BCNF$ restringe uma tabela a possuir dependências funcionais em que os atributos do lado esquerdo sejam sempre chaves na tabela.

Um BD que atenda até a *BCNF* é dito normalizado, ou seja, é livre de redundâncias e possíveis inconsistências.

Desnormalização: segundo [Sanders and Shin 2001], a desnormalização pode ser descrita como um processo de reduzir grau de normalização visando melhorar o desempenho do processamento de consultas. A desnormalização é o processo inverso ao da normalização, ou seja, são acrescidas redundâncias no banco de dados. O principal objetivo da desnormalização é reduzir o número de tabelas físicas que devem ser acessadas para recuperar os dados desejados, reduzindo o número de junções necessárias para derivar uma resposta de consulta.

Pode-se entender a desnormalização como um processo de armazenar o projeto lógico do BD com uma forma normal mais fraca, relaxando o atendimento a $1FN$, $2FN$ e $BCFN$. Manter um projeto sob as formas normais mais fracas garante uma execução mais rápida de consultas e transações mais frequentes da aplicação.

A Figura 1 (A) apresenta a tabela de vendas normalizada e tem somente os dados que são referentes a venda e o relacionamento com o produto vendido por meio do atributo *ProdutoId*, ou seja, a DF *ProdutoId* $\rightarrow$ *DatVenda Qtde ProdutoId ValorVenda* é respeitada. Já a Figura 1 (B), por outro lado, apresenta a mesma tabela desnormalizada sendo que todas as informações que antes estavam em tabelas secundárias como as informações referentes ao produto estarão presentes somente em uma tabela. A desnormalização procura agilizar as consultas por parte do usuário tendo todos os dados disponíveis em um lugar. Segundo [Bock and Schrage 2002], as tabelas desnormalizadas devem manter as características flexíveis de manutenção de dados das tabelas normalizadas, de modo a evitar anomalias de dados.

Os modelos do BD NoSQL são pautados pelo conceito de desnormalização, pois

IdVenda	DataVenda	Qtde	ProdutoId	ValorVenda
1	2022-10-10	2	1	440.40

(A)

IdVenda	DataVenda	Qtde	ProdutoId	ProdutoNome	ProdutoValor	ValorVenda
1	2022-10-10	2	1	Tênis	220.20	440.40

(B)

Figura 1. Vendas: (A) Normalizada, (B) Desnormalizada

a alta disponibilidade necessita que os dados relacionados estejam agrupados. O modelo orientado a documentos é aquele que mais se apoia na desnormalização para armazenamento eficaz dos dados.

NoSQL Orientado a Documentos: um BD orientado a documentos tem o documento como o principal conceito. Um documento pode ser armazenado e recuperado, nos mais diversos formatos, por exemplo, XML ou JSON. Os documentos são autodescritivos e hierárquicos, a estrutura dos dados pode variar entre valores atômicos (*e.g.* inteiros, strings, etc) e complexos (*e.g.* datas, subdocumentos).

A base de dados é constituída por um conjunto de coleções de documentos e cada coleção possui uma série de documentos [Sadalage and Fowler 2013]. Analogamente a um BDR, de maneira hierárquica, uma coleção de documentos pode ser vista como uma tabela, e um documento como uma linha/tupla de uma tabela (a Figura 2 apresenta as afinidades entre modelo relacional e um NoSQL orientado a documentos). Apesar destes conceitos serem hierarquicamente semelhantes, suas aplicações se diferem. Documentos pertencentes a uma mesma coleção não necessitam possuir uma estrutura semelhante diferente de um BDR onde as linhas das tabelas devem seguir o mesmo esquema. Além disso, em um documento é possível realizar incorporações, por exemplo, uma lista de comentários está embutida a um usuário, o que não é possível em uma tabela dado sua estrutura inflexível.

Em um BD orientado a documentos, geralmente, um documento não possui um esquema associado, ou seja, o usuário é responsável pela estrutura que está sendo armazenada no documento [Sadalage and Fowler 2013]. Desta forma, atributos que não possuem um valor associado tendem a não ser armazenados. Já no SGBDRs, o atributo deve estar no esquema e se não for associado a um valor será marcado como *null* ou vazio.

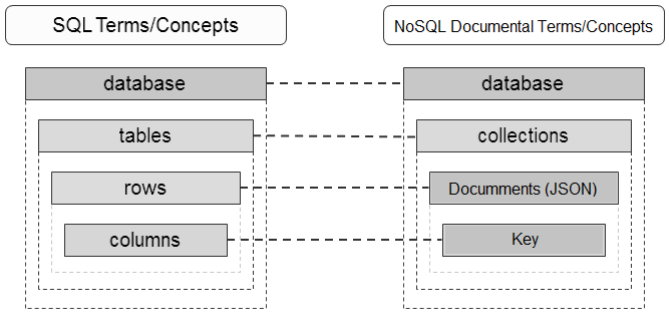

Figura 2. Relacional vs Orientado a Documentos - Fonte: cutt.ly/Kw0Z2jGl

No modelo orientado a documentos, bem como na maior parte dos NoSQL, não é possível realizar a operação de junção. Para evitar junções, a modelagem orientada a documentos considera o conceito de agregação, em que um documento encapsula dados de outros documentos. O conceito de agregação pode ser visto como a desnormalização de uma BDR.

No modelo relacional, as tabelas são normalizadas e, os relacionamentos, são feitos por atributos que referenciam outras tabelas (chaves estrangeiras). Assim, ao exportar um BDR deve-se considerar a desnormalização do BDR de origem para não criar um banco de dados orientados a documento que imita exatamente um BDR, utilizado chaves estrangeiras (*ObjectID*) para realizar a junção de dados.

3. Trabalhos Relacionados

O trabalho de [Karnitis and Arnicans 2015] apresenta um método para migrar dados de um BDR para um BD Orientado a Documentos. O método proposto utiliza uma abordagem de desnormalização da estrutura de dados relacional para melhorar o desempenho das consultas NoSQL, utilizando uma abordagem *top-down* para o design de esquemas, começando com o entendimento dos esquemas de BDR.

O trabalho de [Chen et al. 2022] apresenta um método geral para o design de esquemas para BD NoSQL, levando em consideração o perfil de carga de trabalho. O método proposto é baseado em um modelo de grafo de caminho de consulta (QPG), que representa a estrutura de dados de um BD NoSQL. É também utilizada uma abordagem *top-down* para o design de esquemas, começando com o entendimento das necessidades do negócio.

O trabalho de [de Lima and dos Santos Mello 2015] apresenta uma abordagem lógica para o design de esquemas para BD NoSQL document-oriented. A abordagem proposta é baseada em um modelo de transformação de consultas, que mapeia consultas SQL para consultas NoSQL. O método também utiliza uma abordagem *top-down* para o design de esquemas, começando com o entendimento das necessidades do negócio.

O trabalho de [Zhao et al. 2014] apresenta um modelo para converter esquemas de BDR para NoSQL. O modelo proposto é baseado em uma análise de correspondência de atributos entre os dois tipos de BD. O método também utiliza uma abordagem *top-down* para o design de esquemas, começando com o entendimento dos esquemas de BDR.

Já a ferramenta DINo [Frozza et al. 2018] realiza a migração de dados armazenados em um BDR para BDs NoSQL baseados em chave (orientado a colunas, orientado a documentos ou chave-valor). Apesar da ferramenta ser flexível, cabe ao usuário criar sua estratégia de mapeamento, a ferramenta apenas realiza a migração baseada no mapeamento que o usuário define.

A abordagem proposta neste artigo difere das citadas por ter a mínima interferência do usuário, apenas considerando uma estratégia de desnormalização como proposto em [Karnitis and Arnicans 2015].

4. Abordagem Proposta

A abordagem proposta neste trabalho visa a migração de dados armazenados em um SGBDR para um BD NoSQL orientado a documentos. A fim de realizar os testes e

validação da abordagem, a mesma foi desenvolvida considerando como SGBDR principal o PostgreSQL, e como NoSQL o MongoDB. A migração entre modelos de dados é complexa, e deve considerar as peculiaridades de cada modelo de dados, bem como as incompatibilidades entre os tipos básicos.

Na migração proposta, considera-se que toda tabela será candidata a ser mapeada para uma coleção de documentos. Obviamente, esta abordagem acarretaria problemas, pois leva a um limiar onde teremos um mapeamento 1:1 entre tabelas e coleções de documentos. Este mapeamento não é ideal, pois BDs NoSQL, em geral, não possuem suporte a operações de junção, o que impede a combinação direta entre dados em diferentes coleções. Desta forma, neste trabalho, é utilizado o conceito de agregados que permite que documentos que possuem informações complementares possam ser armazenados de forma associada, ou seja, os dados são armazenados de maneira combinada (agregada) eliminando a necessidade de uma junção posterior.

A decisão de quais tabelas devem ser transformadas em documentos e quais em agregados é complexa e depende da forma com que os dados são acessados. A fim de auxiliar nesta decisão foram propostos três casos-base (C1, C2, C3). Cada caso possui como conceito central as chaves estrangeiras (FKs), por meio do relacionamento estabelecido por elas é definido se um relacionamento será mantido em coleções diferentes ou agregado:

- Caso 1 (C1): Um documento será agregado a outro se e somente se a tabela (T1) for referenciada por somente uma outra tabela (T2) apenas uma vez.
- Caso 2 (C2): Uma tabela (T1) é referenciada por várias outras tabelas (*e.g.*, T2 e T3), a tabela (T1) será uma coleção e terá o atributo referência nas tabelas que faz referência (T2, T3).
- Caso 3 (C3): Todos os documentos que não forem agregados serão considerados coleções (*collections*) e terão um campo que referencia o objeto na tabela qual é citado.

Salienta-se que a chave estrangeira (*FK*) tem um papel essencial na abordagem proposta, pois por ela são definidas quais coleções serão agregadas ou não devido a quantidade de tabelas referenciadas. Para todo o atributo que seja FK, é realizada uma verificação da possibilidade da informação ser mantida como agregado. Essa verificação é o passo da migração mais custoso em tempo e processamento.

O fluxo de trabalho da abordagem é apresentado pela Figura 3. Por meio da Figura é possível visualizar todos os passos executados durante a migração. Cada passo P, na figura, é representado por um numeral nn tendo *Pnn* como o n-ésimo passo.

Inicialmente, o usuário define as seguintes configurações: (*i*) informações de conexão ao PostgreSQL usado para extração dos dados e ao MongoDB, (*ii*) o uso ou não de *ObjectId* para mapear FKs, e (*iii*) inserir ou não campos opcionais como nulos. As configurações são armazenadas no arquivo de configuração ($P01$). Em seguida, são extraídos os nomes das tabelas ordenadas de forma decrescente pelo número de FKs ($P02$). A coleta das tabelas é realizada mediante a uma consulta que ordena os resultados baseados no número de FKs. Quanto mais FKs uma tabela possui, menor é a probabilidade desta ser mapeada para um agregado.

De posse da lista de tabelas, é deflagrado um laço que percorrerá todas as tabelas,

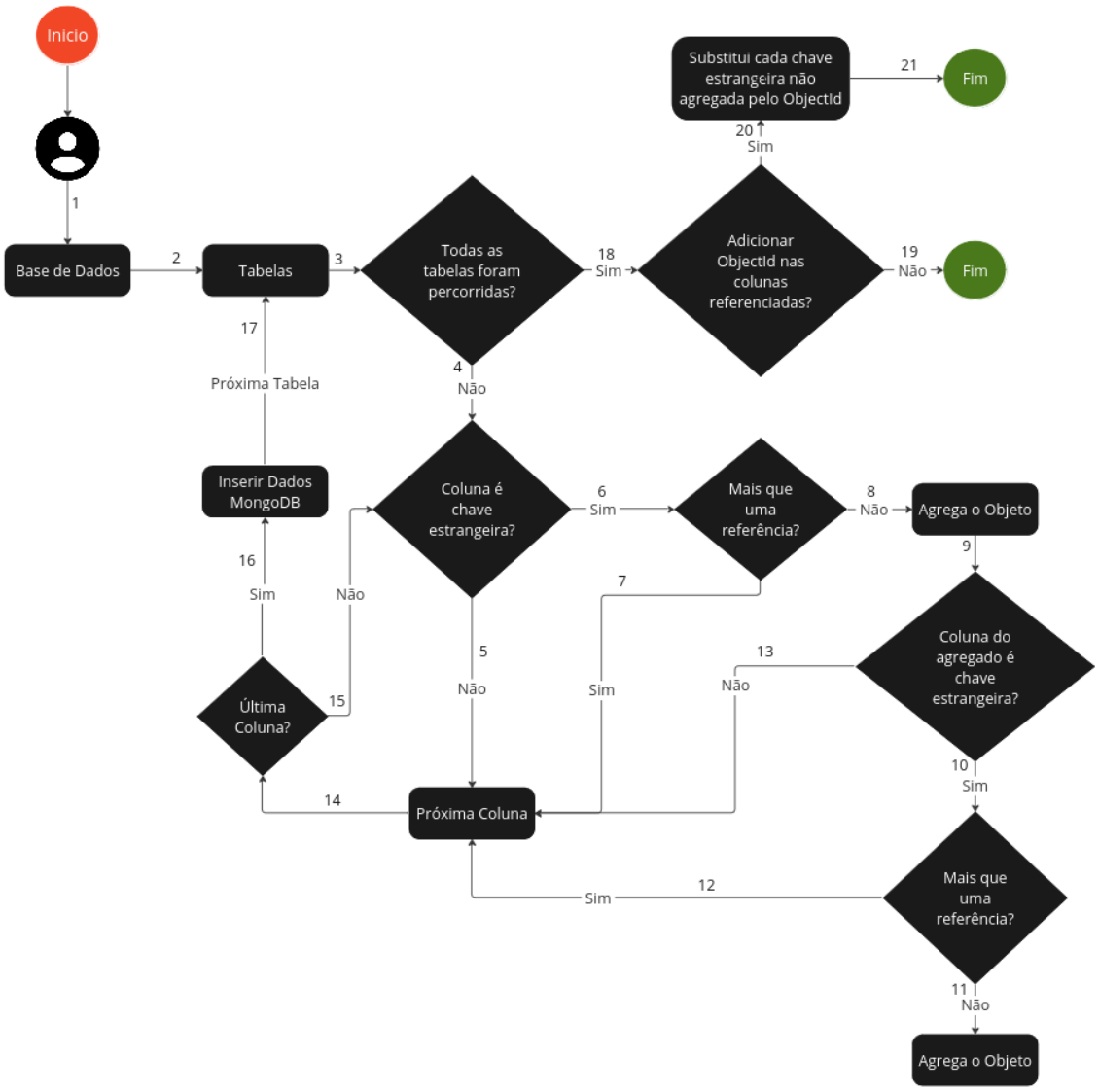

Figura 3. Fluxograma do método proposto

linhas e colunas da base de dados. O laço verifica se a tabela atual já foi percorrida ou agregada ($P03$). Caso a tabela não tenha sido percorrida anteriormente, é feita uma análise da quantidade de FKs e quantidade de tabelas que referenciam a tabela atual. Após isso, serão percorridas todas as linhas e colunas da tabela anteriormente não percorrida ($P04$). Esta análise é realizada para determinar se a tabela atual se enquadra nos casos $C1$ ou $C3$.

Durante o passo $P04$ o foco é identificar apenas as colunas que são FKs, já que estas são a base para o mapeamento. Para cada FK encontrada ($P06$), verifica-se se a chave possui mais de uma referência. Caso a chave possua mais de uma referência ($P07$), a mesma não é candidata a agregação. Se a chave possui apenas uma referência ($P08$), o valor correspondente à chave (na tabela estrangeira) é consultado e agregado ao documento atual.

Após concluída a agregação, é realizada a análise do conteúdo do documento agre-

gado ($P09$). Durante a análise, as colunas do objeto agregado são verificadas para buscar por outras colunas do tipo FK. Para cada FK do agregado ($P10$) é feita a validação de referências. Caso a chave possua apenas uma referência ($P11$) o valor correspondente é agregado, caso contrário, a chave permanece como uma referência e a próxima coluna é analisada ($P12$).

Este processo continua até que a última coluna da tabela atual tenha sido atingida. Quando a análise da última coluna é completada ($P16$) os dados da tabela atual são efetivamente persistidos no MongoDB. Então, o processo de extração é repetido para a próxima tabela ($P17$) até que todas as tabelas tenham sido percorridas.

Por fim, após todas as tabelas terem sido percorridas ($P18$), o usuário opta por adicionar o *ObjectId* nas colunas referenciadas que não foram agregadas. Caso o usuário opte por não utilizar o *ObjectId* ($P19$) a migração é encerrada, e os dados das FKs que não foram mapeadas para agregados permanecem como valores simples no MongoDB. Se o usuário optar por utilizar o *ObjectId* ($P20$) todas as referências de FKs que não foram mapeadas para agregados serão substituídas por referências ao *ObjectId* do documento que referenciam, desta forma, é criada uma referência formal que o dado está referenciando outro documento. A principal vantagem de utilizar esta abordagem é que, quando o documento for consultado, todas suas referências (pelo *ObjectId*) são automaticamente recuperadas em conjunto, combinando todas as informações em um único documento (semelhante a uma operação de JOIN). A desvantagem da abordagem é que, quando os documentos referenciados são grandes, o desempenho na consulta é prejudicado.

Conforme apresentado, o método proposto realiza a migração tabela a tabela. Em cada tabela são analisadas as FKs e caso a chave não possua mais de uma referência os dados são agregados. Na última etapa é fornecida a opção para o usuário substituir referências que não foram agregadas pelo *ObjectId* dos documentos criando uma referência entre os elementos. Para validar a proposta foram realizados um conjunto de experimentos que realizam a migração de dados relacionais para orientados a documentos.

5. Experimentos

Nesta seção são apresentados resultados obtidos com a ferramenta proposta. A ferramenta foi desenvolvida em Python com base na abordagem apresentada anteriormente. O código está disponível gratuitamente no GitHub[1]. Para os experimentos foram selecionadas três bases de dados. Como critério de escolha para as bases foi utilizada a quantidade de dados bem como a quantidade de FKs. As bases selecionadas foram:

- *Rental*: Representa uma locadora fictícia de filmes. Composto por 15 tabelas contendo o total de 31.722 tuplas e 18 FKs.
- *GSJ*: Representa um histórico de realização de chamadas de voz. Esta base de dados é usada para auditoria e resolução de disputas. É necessário que a empresa prestadora do serviço de chamadas de voz guarde os registros das chamadas realizadas por um período de conformidade com a lei vigente do país. A base é composta por 28 tabelas contendo o total de 2.806.907 tuplas e somente 1 FK.
- *DNPRO*: Representa uma empresa que possibilita aos seus clientes realizar o disparo de voz e SMS para alcançar seus clientes. A base é composta por 127 tabelas contendo o total de 5.100.409 tuplas e somente 247 FKs.

[1] `https://github.com/ttainam/PostgreSQL-to-MongoDB-Migration`

O SGBDR origem é o PostgreSQL Versão 14.11 e o SGBD NoSQL destino é o MongoDB versão 6.0. A migração é realizada respeitando os 3 casos (C1, C2 e C3) para gerar as coleções no NoSQL MongoDB. Foram avaliados três parâmetros para verificar a ferramenta: tempo decorrido, completude (todos os dados foram exportados) e corretude (os dados foram exportados corretamente). A completude e a corretude foram verificadas por meio de consultas nos dois BD envolvidos. A completude e a corretude foram respeitadas, pois foi feita uma vericação para ambos os casos.

A Tabela 1 apresenta algumas estatísticas do BDRs origem: total de tuplas, total de tabelas, total de FKs e o tempo decorrido da migração de dados.

Base de Dados	Qtde de Tuplas	Qtde de Tabelas	Qtde de FK	Tempo Decorrido
Rental	31.722	15	18	06:30
GSJ	2.806.907	28	1	05:46
DNPRO	5.100.409	127	247	27:00:00

Tabela 1. Estatísticas dos bancos de dados dos experimentos.

Considerando a execução dos experimentos, é possível verificar que para todos os experimentos os dados foram corretamente exportados, realizando o mapeamento correto para as novas estruturas e mantendo toda a massa de dados (corretude e completude). O BDR Rental teve seus dados exportados em seis minutos e trinta segundos. Para o GSJ o tempo decorrido foi de cinco minutos e 46 segundos. E por fim, para o DB DNPRO teve um tempo decorrido de vinte sete horas, ou seja, muito superior ao tempo dos demais.

Analisando os resultados, pode-se verificar que o maior impacto no tempo decorrido se dá pela quantidade de FKs que a base de dados possui. A base *GSJ*, por exemplo, é oitenta e oito vezes maior em quantidade de tuplas que a base *Rental* e contém treze tabelas a mais, porém contém dezessete FKs a menos. Ao realizar a migração de dados foi constatado que o tempo decorrido foi maior na base *Rental* que na base *GSJ*.

A base *DNPRO*, em relação à base *GSJ*, tem 2.293.502 (dois milhões, duzentos e noventa e três mil e quinhentas e duas) tuplas a mais, 99 (noventa e nove) tabelas a mais e 246 (duzentos e quarenta e seis) FKs a mais, porém a diferença de tempo decorrido entre as duas bases é de vinte e seis horas e cinquenta e quatro minutos.

A quantidade de *FK* tem maior impacto no tempo de realização da migração de dados a partir do modelo proposto, isto se dá pelo fato de que para cada FK, é realizada uma série de verificações. Durante o processamento, para cada coluna de cada tabela, são verificadas as FKs. Para cada FK é feita uma verificação de número de ocorrências para definir a qual caso ela pertence. Caso o dado seja agregado, o mesmo é submetido à verificação de suas FKs para possível agregação. Além disso, verifica-se a quantidade de *FK* que apontam para esta tabela. Se somente tiver uma referência, será agregada. Este processo é realizado para cada linha de cada tabela. Desta forma, quanto maior a quantidade de *FK*, mais verificações deverão ser realizadas para determinar os agregados.

6. Conclusão

Este trabalho propôs uma abordagem para a migração de dados relacionais para dados orientados a documentos utilizando os SGBDs PostgreSQL e MongoDB para a implementação.

Foram realizados experimentos com três bases de dados. As bases tinham tamanhos e números de chaves estrangeiras diferentes. Os resultados mostraram que a quantidade de dados presente não é o que gera o maior impacto no tempo decorrido para a realização da migração dos dados. Verificou-se que o maior impacto no tempo decorrido ocorreu devido à quantidade de chaves estrangeiras (FK) que a base de dados possui.

O método proposto pode ser usado para qualquer tamanho de base de dados, porém, uma de suas limitações é que, ao usar bases com grandes quantidades de chaves estrangeiras (FK), o tempo de importação cresce exponencialmente. Desta forma, aconselha-se usar este estudo como base e realizar alterações de acordo com o seu uso. Esta ferramenta foi feita para fins gerais e pode não atender de forma eficiente à sua aplicação.

Como trabalhos futuros se pretende: (i) identificar uma solução ótima para a verificação de chaves estrangeiras e agregação, (ii) adaptar a ferramenta proposta para mais sistemas gerenciadores de banco de dados relacionais e NoSQL, e (iii) criar uma interface em que seja possível do usuário preencher os dados de configuração.

Referências

Bock, D. B. and Schrage, J. F. (2002). Denormalization guidelines for base and transaction tables. *ACM SIGCSE Bulletin*, 34(4):129–133.

Chen, L., Davoudian, A., and Liu, M. (2022). A workload-driven method for designing aggregate-oriented nosql databases. *Data & Knowledge Engineering*, 142:102089.

de Lima, C. and dos Santos Mello, R. (2015). A workload-driven logical design approach for nosql document databases. In *Proceedings of the 17th iiWAS*, pages 1–10.

Frozza, A. A., Schreiner, G., Brüggemann, R., and dos Santos Mello, R. (2018). Dino: uma ferramenta para importação de dados em bancos de dados nosql. In *Anais da XIV ERBD*. SBC.

Karnitis, G. and Arnicans, G. (2015). Migration of relational database to document-oriented database: Structure denormalization and data transformation. In *2015 7th CICN*, pages 113–118. IEEE.

Sadalage, P. J. and Fowler, M. (2013). *NoSQL distilled: a brief guide to the emerging world of polyglot persistence*. Pearson Education.

Sanders, G. L. and Shin, S. (2001). Denormalization effects on performance of rdbms. In *Proceedings of the 34th HICSS*, pages 9–pp. IEEE.

Sarmah, S. S. (2018). Data migration. *Science and Technology*, 8(1):1–10.

Schreiner, G. A., Duarte, D., and dos Santos Mello, R. (2020). Bringing sql databases to key-based nosql databases: a canonical approach. *Computing*, 102(1):221–246.

Zhao, G., Lin, Q., Li, L., and Li, Z. (2014). Schema conversion model of sql database to nosql. In *2014 Ninth 3PGCIC*, pages 355–362. IEEE.