Desenvolvimento de um software para migração de um banco de dados relacional Firebird, para o não relacional MongoDB

Mauro A. Murari, Guilherme Bernardino da Cunha

Curso de Bacharelado em Sistemas de Informação – Universidade Federal de Santa Maria, Centro de Educação Superior Norte – RS(UFSM/CESNORS)
Caixa Postal 54, CEP: 98400-000, Frederico Westphalen – RS – Brasil

mauro_murari@hotmail.com, guilherme@ufsm.br

Resumo. *Este artigo apresenta o desenvolvimento de um software para realizar a migração de um banco de dados relacional Firebird® para um não relacional MongoDB®. Foram realizadas seis etapas, sendo elas: levantamento de requisitos, analise base de dados Firebird®, modelagem base de dados MongoDB®, migração dos dados, validação do processo de migração e, otimização dos dados após a migração. Também foram realizados processos para comparar o tempo de busca na manipulação e o tamanho final de cada base de dados. Com os resultados dos comparativos nota-se que os dados migrados para o MongoDB® apresentaram um melhor desempenho que os dados no Firebird®.*

Palavras-chaves: *Banco de dados, migração, Firebird®, MongoDB®.*

Abstract. *This paper presents the development of a software for a migration from a relational database Firebird® for a non-relational MongoDB®. Six stages were carried out, namely: requirements gathering, analysis Firebird® database, MongoDB® database modeling, data migration, the migration process validation and optimization of data after migration. Procedures were also performed to compare the search time in the handling and the final size of each database. With the results of the comparative is noted that the data migrated to MongoDB® performed better that the data in Firebird®.*

Keywords: *Databases, migration, Firebird®, MongoDB®.*

1. Introdução

Com a crescente diversificação de formas de obter dados, documentos e informações por meio de softwares, as consultas nos bancos de dados relacionais em grandes escalas, estão se tornando bastante complexas, gerando um baixo desempenho e insatisfação na qualidade do serviço.

Segundo Leite (2010), "O uso de bancos de dados relacionais têm apresentado certos fatores limitantes, principalmente devido a sua natureza estruturada que não permite muita flexibilidade ao realizar a estruturação desses dados. Assim, diversas

soluções estão aparecendo no mercado com o objetivo de suprir essas deficiências, principalmente no que se refere a questões de escalabilidade e disponibilidade do sistema. Entre estas soluções destacam-se os bancos de dados NoSQL".

Destaca-se o fato que nos estudos realizados para o desenvolvimento deste trabalho não foi identificada nenhuma ferramenta que faz este processo de migração entre os bancos de dados Firebird® e o MongoDB®.

Neste contexto, este artigo apresenta as etapas para o desenvolvimento de um software que realize a migração de um banco dados relacional Firebird® para o banco de dados não relacional MongoDB®. As etapas realizadas foram: levantamento de requisitos, análise da base de dados Firebird®, modelagem da base de dados MongoDB®, migração dos dados, validação do processo de migração e, finalmente, a otimização dos dados.

A primeira etapa envolveu o levantamento de quais as informações que deveriam ser migradas. Após isto foi dado início ao processo de análise do banco de dados relacional Firebird® a fim de identificar quais tabelas e dados serão migrados.

A terceira etapa, foi realizar um estudo sobre o funcionamento e as regras do banco de dados MongoDB® para o desenvolvimento do software. Com base neste estudo foi possível definir a modelagem da base de dados MongoDB que receberá a migração.

Na quarta etapa, chamada migração dos dados, foi realizada uma busca de todas as informações da base de dados Firebird® e replicados para o MongoDB®. Foram migrados dados referentes a internações, médicos, SAME, convênios, usuário, procedimentos e motivos. Este processo manteve a estrutura da base de dados de origem.

Na quinta etapa, com o objetivo de garantir que as informações tenham sido migradas corretamente, ou seja, sem perda de dados, foram realizadas validações de quantidade de registros e conteúdo de cada tabela migrada.

Terminado o processo de validação, foi realizado o processo de otimização da base de dados com o objetivo de melhorar as buscas de dados e remover campos que não são utilizados na base de dados.

A sexta etapa teve o intuito de gerar comparativos para validar o software e demonstrar os principais benefícios da migração de um banco de dados relacional Firebird® para um não relacional MongoDB®. Nesta etapa fica evidente a diferença desempenho entre os dois bancos de dados, no que diz respeito a tempo de busca de informações e ao tamanho final do banco de dados.

Este artigo está organizado da seguinte forma: a seção 2 apresenta o referencial teórico estudado para fundamentação deste trabalho; a seção 3 envolve o estado da arte, comparando o sistema implementado com outros sistemas desenvolvidos para migração de dados; a seção 4 apresenta a solução implementada, detalhando a metodologia e as tecnologias empregadas; a seção 5 aborda os resultados alcançados e validações realizadas e, ao final do artigo, apresentam-se as considerações finais e referências utilizadas.

2. Referencial Teórico

Nesta seção será apresentado um breve referencial teórico sobre as áreas envolvidas no desenvolvimento do software proposto. Foram estudados conceitos de Big Data, bancos de dados relacionais, bancos de dados NoSQL, Python e migração e otimização de dados.

2.1. Big Data

Big Data pode ser definido como um grande conjunto de dados, que necessita de ferramentas específicas para armazenar, recuperar e manipular os dados e documentos (VIEIRA et. al. 2012).

Na última década o volume de dados cresceu exponencialmente em virtude da grande difusão de computadores e da adesão cada vez maior de empresas e usuários a geração de informações. Segundo a IBM, nos últimos dois anos foram gerados cerca de noventa por cento de todas as informações existentes no mundo. Este número só é possível pelo uso de redes sociais, dispositivos móveis, GPS (*Global Positioning System*) e inúmeras outras formas de obter e gerar informação (IBM, 2014).

Atualmente é fácil visualizar o cenário de como isso vem acontecendo, pois todos os dias milhões de e-mails são trocados, novas fotos são armazenadas e milhares de novos vídeos aparecem. A maioria das empresas e alguns setores do governo estão armazenando e analisando informações de seus bancos de dados, em busca de melhoria no atendimento e no serviço prestado aos clientes, usuários e a população em geral.

Em um futuro não muito distante, aparelhos como televisão, geladeira, máquina de lavar, cafeteira e micro-ondas, estarão conectados à internet, disponibilizando dados e informações (OLIVEIRA, 2013).

Big Data se baseia em cinco "V" sendo eles: Volume, Velocidade, Variedade, Veracidade e Valor (GONSOWSKI, 2012):

- Velocidade **–** Indiferente do volume da base de dados, o tempo de resposta para armazenagem e obtenção de dados é superior quando comparado a outros tipos de banco de dados;
- Volume – As ferramentas utilizadas em Big Data devem ser capazes de suportar volumes gigantescos de dados e documentos que estão em constante aumento;
- Variedade **–** É possível localizar informação em formas estruturadas, como banco de dados MongoDB® e Firebird®, porém existem diversas outras formas de armazenar dados, como arquivos de texto, imagens, vídeos, áudio, entre outros;
- Veracidade **–** Aponta a importância de que todos os dados armazenados estejam corretos e coerentes;
- Valor **–** Todos os itens destacados anteriormente não têm importância se não gerarem informações relevantes para a empresa.

As ferramentas de Big Data, além de serem capazes de gerenciar grandes volumes de dados, também devem ser capazes de garantir disponibilidade, replicação e escalabilidade.

2.2. Banco de dados Relacionais

O modelo de dados relacional foi o primeiro modelo existente, criado por volta de 1970 por Edgar Frank Codd. Segundo este modelo, todos os dados são armazenados em tabelas, cada coluna da tabela é conhecida como um atributo e todas as linhas são chamadas de tuplas. Cada atributo tem um domínio, que define o tipo e/ou quantos caracteres o atributo suportará (COSTA, 2011). A figura 1 apresenta a disposição de uma tabela em um banco de dados relacional.

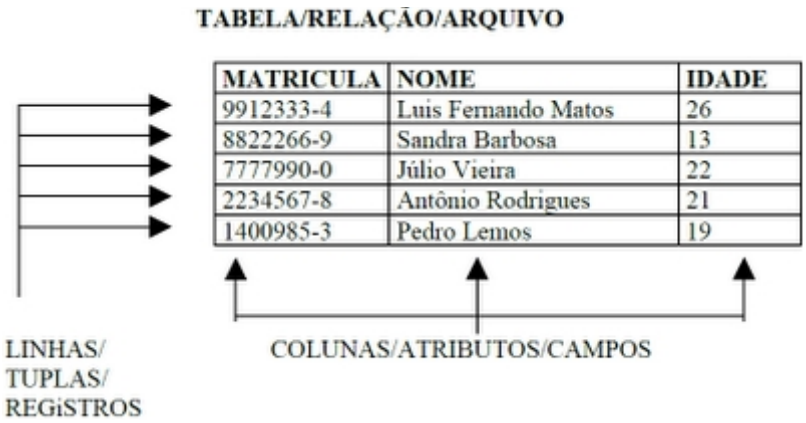

Figura 1 – Exemplo de tabela banco de dados relacional (FREITAS, 2014)

Na grande maioria dos modelos existem chaves primárias para identificação única das tuplas. Existe, também, a possibilidade de relacionar uma tabela com a outra por meio do uso de chaves estrangeiras, criando um relacionamento entre duas tuplas de tabelas distintas, a fim de garantir a integridade da base de dados. A chave primária de uma tabela envolve uma ou mais colunas cujos valores distinguem uma linha da outra na base de dados. Para as chaves estrangeiras também podem ser utilizadas uma ou mais colunas que devem ser vinculadas a chaves primarias de outras tabelas. Este tipo de chave permite a implementação de relacionamento entre as tabelas (VASCONCELLOS, 2014).

Bancos de dados relacionais são baseados na arquitetura ACID, que significa em português, atomicidade, consistência, isolamento e durabilidade (NASCIMENTO, C., 2011):

- Atomicidade: neste aspecto todas as transações são tratadas como uma tarefa única, ou seja, ou todo o trabalho é finalizado ou nada é feito;
- Consistência: o banco de dados deve tratar os dados a cada tarefa solicitada, com o objetivo de garantir que as chaves, tanto primárias como estrangeiras vão ser respeitadas;
- Isolamento: o isolamento entre uma transação e outra permite com que vários usuários acessem a mesma tupla ao mesmo tempo;
- Durabilidade: garantir que os dados estarão sempre acessíveis.

Os bancos de dados relacionais utilizam como padrão a linguagem SQL (*Structured Query Language*), que foi criada pela IBM em 1970. O SQL utiliza conceitos de DDL (*Data Definition Language*) e DML (*Data Manipulation Language*). A DDL é usada para especificar relações, domínios, regras de integridade, entre outros aspectos; já com a DML é possível realizar operações de manipulação tais como *Select, Insert, Update* e *Delete* (COSTA, 2011).

Segundo Leite (2010), "O uso de bancos de dados relacionais têm apresentado certos fatores limitantes, principalmente devido a sua natureza estruturada que não permite muita flexibilidade ao realizar a estruturação desses dados. Assim, diversas soluções estão aparecendo no mercado com o objetivo de suprir essas deficiências, principalmente no que se refere a questões de escalabilidade e disponibilidade do sistema". Entre estas soluções destacam-se os bancos de dados NoSQL, que serão estudados na próxima seção.

2.3. Banco de dados NoSQL

O termo NoSQL vem do inglês *Not Only SQL*. Os bancos de dados NoSQL romperam uma grande história de bancos de dados relacionais, baseados em SQL e em ACID (VIEIRA et. al., 2012).

Os bancos de dados NoSQL começaram a se tornar populares em 2009, surgindo como solução para questões como escalabilidade e processamento de grandes volumes de dados. Segundo Rêgo (2013), "Big Data é a comprovação prática de que o enorme volume de dados gerados diariamente excede a capacidade das tecnologias atuais, geralmente baseadas em bancos de dados relacionais".

O surgimento de bancos de dados NoSQL deve-se, principalmente, à grande dificuldade relacionada à distribuição dos dados em diversos servidores quando utilizado um banco de dados relacional (LEITE, 2010).

Os bancos de dados NoSQL são subdivididos de acordo com a forma como armazenam e gerenciam os dados, podendo ser *Document Store*, *Graph Store*, *Wide Columns Store*, *Key/Value Store* e *Column Oriented Store:* (NASCIMENTO, J., 2014)

- *Document Store* – São baseados em JSON (*JavaScript Object Notation*) ou XML (*Extensible Markup Language*), podendo ser localizados pelo seu ID único ou por qualquer registro que exista no documento. Como exemplos citam-se o MongoDB e o CouchDB;
- *Graph Store* **–** Este modelo é o de maior complexidade, pois armazena os dados na forma de objetos e a busca é feita nestes objetos. Citam-se, como exemplos, o InfoGrid e o Neo4J;
- *Wide Columns Store* **–** Este modelo suporta diversas colunas e linhas, tais como os bancos de dados *BigTable* e Cassandra;
- *Key/Value Store* **–** Este modelo é o mais simples, possuindo uma chave e um valor para essa chave, este modelo é o que suporta maior volume de dados. Destacam-se como exemplos desta categoria o SimpleDB e o Tokyo Cabinet;

- *Column Oriented Store* – O armazenamento é feito por colunas e não por linhas, como normalmente acontece em outros bancos de dados. Exemplos destes bancos de dados são o LucidDB e o MonetDB.

As empresas de grande porte começaram a criar suas próprias soluções para problemas relacionados ao grande volume de dados e documentos, começando então, a publicar artigos científicos descrevendo estas soluções encontradas para gerenciamento de dados distribuídos em grande escala, sem utilizar o termo NoSQL (DIANA; GEROSAL, 2010).

As empresas de grande porte começaram a criar suas próprias soluções para problemas relacionados à grande volume de dados e documentos, começando então, a publicar artigos científicos descrevendo estas soluções encontradas para gerenciamento de dados distribuídos em grande escala, sem utilizar o termo NoSQL (DIANA; GEROSAL, 2010).

2.4. Migração e otimização de dados

A migração de dados é de grande importância para projetos de software, sendo estes, muitas vezes, elaboradas sem muita atenção. O processo de migração de dados pode atrasar e até mesmo causar o fracasso de um projeto de software. O processo de migração de dados deve ser aplicado e validado com devido cuidado, para evitar a perda de informações ou gerar inconsistências nas mesmas (GUEDES, 2014).

A figura 2 apresenta como o processo de migração deve ser aplicado para que todos os as tarefas sejam executadas, a fim de atingir os objetivos de migração e otimização dos dados.

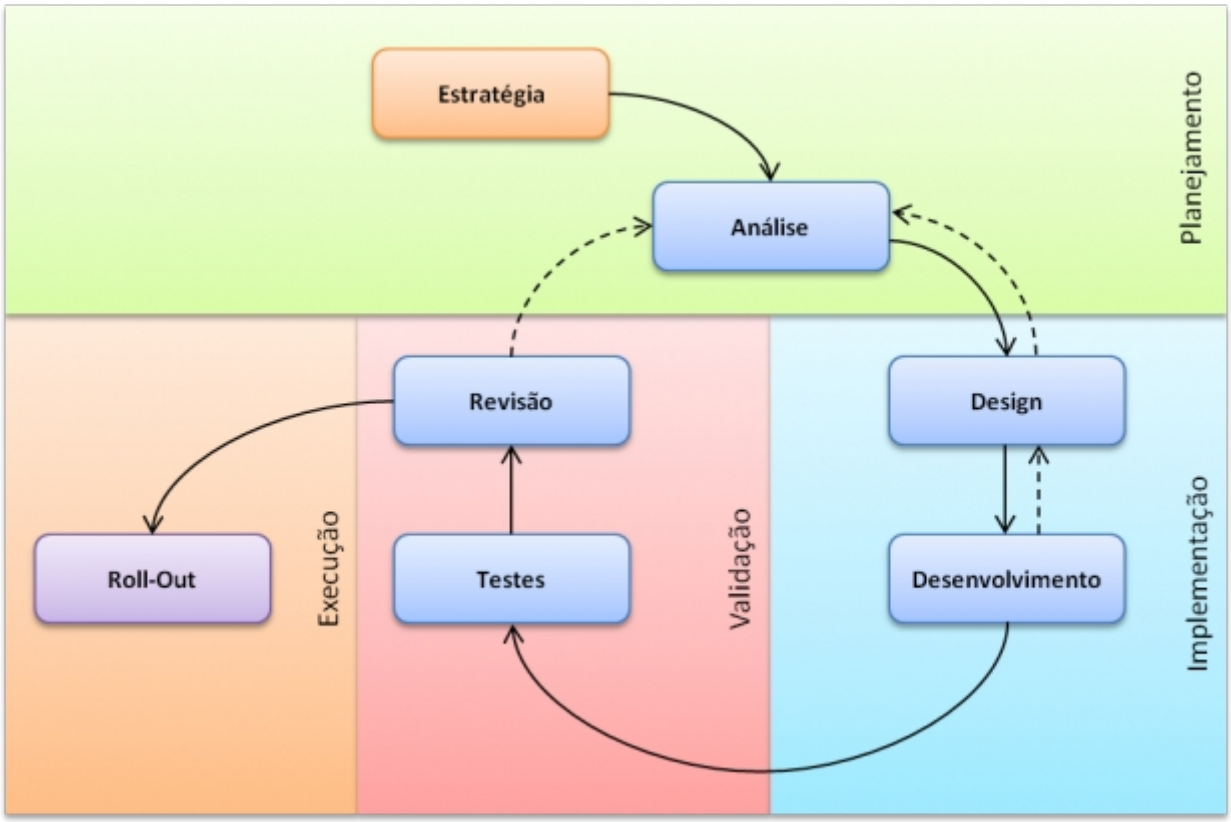

Figura 2 – Processo de migração de dados (GUEDES, 2014)

De acordo com a Figura 2, na etapa de planejamento deve ser criada a estratégia, que tem como objetivo identificar quais as informações serão migradas em uma visão mais ampla, sendo que, este processo normalmente inclui todos os *stakeholders* do projeto. Após a criação da estratégia, a base de dados deve ser analisada, à procura de informações requisitadas pelos *stakeholders* (GUEDES, 2014).

No processo de implementação do software de migração, a base de dados de origem e destino deve ser modelada através de diagramas. Posterior a isto, deve ser desenvolvido o software que migrará os dados de uma base de dados para outra (GUEDES, 2014).

No processo de validação devem ser feitos testes que para responder questões como, quantos registros foram migrados, se os dados foram para as colunas corretas e até mesmo se os dados estão em formatação correta. Tendo essas respostas em mão é feita uma revisão que dirá se é preciso refazer a análise, o design, o desenvolvimento e os testes (GUEDES, 2014).

A otimização de bancos de dados tem grande importância neste trabalho tendo-se em vista a complexidade da base de dados Firebird do Hospital de Frederico Westphalen-, que apresenta muitas informações duplicadas, o tamanho de seus campos está acima do que é realmente utilizado e o tempo de busca das informações apresentando uma demora elevada.

Uma das formas de deixar o banco de dados mais leve é utilizar o menor espaço possível de disco; isso pode oferecer grandes melhorias em virtude de uma leitura e

escrita mais rápida no banco de dados. Para tanto, podem ser utilizados tipos de dados menores, como, por exemplo, em vez de utilizar o tipo *INT*, utilizar *MEDIUMINT*, o que reduz o espaço armazenamento.

2.5. Python

O sistema foi implementado por meio da linguagem de programação Python. Python é uma linguagem de programação criada por Guido van Rossum em 1991. A linguagem tinha como principal objetivo a produtividade e legibilidade. Seu código fonte é aberto e esta funciona em qualquer plataforma (PYSCIENCE Brasil, 2014).

Os principais pontos fortes da linguagem são a fácil manutenção, pouco uso de caracteres especiais e o uso de identação para delimitar os blocos de código. A linguagem Python possui suporte para diversos paradigmas de programação, destacando-se a orientação a objetos e a programação procedimental (PyScience Brasil, 2014).

Python é uma linguagem de programação fácil de aprender e que possui diversos recursos, funções e classes prontas que facilitam o desenvolvimento, deixando somente a parte lógica para o desenvolvedor.

Existem diversas extensões que podem ser adicionadas ao Python com o objetivo de adaptar a linguagem para cada necessidade. Neste trabalho foram utilizadas duas bibliotecas externas, sendo elas a Pymongo e FDB.

O Pymongo é um *driver* que faz a conexão entre a aplicação Python e o MongoD; foi criado com o objetivo de facilitar o uso do MongoDB em softwares escritos em Python (PYMONGO, 2014).

O FDB foi desenvolvido pela empresa Firebird, é o substituto do KInterbasDB, funciona também como um *driver* para conectar o Python às bases de dados relacionais Firebird (FIREBIRD, 2014).

3. Estado da arte

Nesta seção são apresentados alguns softwares que utilizam técnicas de migração de dados de um banco de dados para outro. Existem algumas ferramentas que fazem o processo de migração de dados, porém antes de iniciar o processo são necessárias configurações de acesso, consultas escritas em SQL (*Structured Query Language*), tratamento de relacionamentos e ajuste manual dos dados a serem migrados, tornando assim o processo bastante complexo. Estes softwares fazem na maioria das vezes, somente o processo de migração dos dados, deixando de lado, a otimização dos dados.

Nas pesquisas realizadas não foram encontradas ferramentas que realizam o processo de migração de dados entre uma base de dados relacional Firebird e uma base de dados NoSQL MongoDB que são o foco deste trabalho

3.1. MySQL Workbench – Oracle

O MySQL Workbench é uma ferramenta escrita em linguagem C, de código livre e multiplataforma. A empresa proprietária do MySQL Workbench é a Oracle, o software hoje se encontra na versão 6.0.

Em 2012 a Oracle anunciou uma nova ferramenta para migrar dados do *Microsoft SQL Server* para o MySQL usando o MySQL Workbench, assim como é possível migrar dados de bases do *PostgreSQL* e *SyBase* (NARAYANASWAMY, 2012).

A ferramenta garante questões de segurança como login e senha para autenticar nos dois bancos de dados. É possível ainda fazer todo o modelo ER(Entidade-Relacionamento) do banco de dados que está sendo migrado através do MySQL Workbench.

O software oferece como principais vantagens a migração de dados, mineração de dados e a possibilidade de gerar diagramas ER, porém tem como desvantagens não realizar processo de otimização de dados e não migrar dados dos bancos de dados MongoDB® e Firebird®.

3.2. Pentaho Data Integration

O PDI (*Pentaho Data Integration*) é um software de código aberto, escrito em linguagem java. Começou a ser desenvolvido em 2004 pela empresa *Pentaho Corporation*.

O PDI além de ser um software para extrair, transformar e carregar dados, também possibilita a geração de relatórios, análise OLAP (*On-line Analytical Processing*) e mineração de dados (PENTAHO, 2014).

Para realizar as configurações, conexões e regras de negócio é apresentada para o usuário uma interface gráfica, com a possibilidade de usar as operações de arrastar e soltar, apenas ajustando as propriedades de cada objeto.

O software oferece como principais vantagens a migração de dados e o suporte aos bancos de dados MongoDB® e Firebird®, porém tem como desvantagens não gerar diagramas ER (Entidade-Relacionamento), não minerar dados e não realizar o processo de otimização dos dados.

3.3. SQL Server Integration Services

O *SQL Server Integration Services* é um componente do SGBD *SQL Server* da *Microsoft*, tendo sido integrado ao *SQL Server 2005* na versão 7.0. É um software proprietário, de código fechado e executado somente em sistemas operacionais Windows.

O SSIS (*SQL Server Integration Services*) realiza diversos processos dentro das bases de dados SQLServer, sendo eles, exportar e importar dados e *schemas*, automatizar as manutenções da base de dados e realizar atualizações multidimensionais (MICROSOFT, 2014).

Para o desenvolvedor criar as rotinas, tarefas e modelos de migração de dados, é utilizado um software chamado *Business Intelligence Development Studio* (BIDS) baseado no ambiente *Microsoft Visual Studio.*

O software oferece como principais a vantagens migração de dados, algoritmos para mineração de dados e geração de diagramas ER. Porém tem como desvantagens

não realizar processo de otimização de dados e não migrar dados dos bancos de dados MongoDB® e Firebird®, que são o foco deste trabalho.

3.4. Comparação entre os softwares

Após a apresentação das principais características das ferramentas estudadas, traçou-se um estudo comparativo entre os sistemas encontrados na área de migração de dados e a solução implementada. Esta comparação é apresentada no quadro 1.

Características	Ferramentas			
	Workbench	PDI	SSIS	Solução Implementada
Suporte à MongoDB®				
Suporte à Firebird®				
Migração de Dados				
Migração de Dados de diversos Bancos de Dados				
Otimização de Dados				
Mineração de Dados				
Geração de Diagramas ER				

Quadro 1 – Comparativo entre as ferramentas estudadas

Como visto no Quadro 1, a solução proposta realizará uma atividade que as demais ferramentas estudadas neste artigo não suportam, ou seja, realizar a migração de um banco de dados Firebird® para o banco de dados MongoDB®. As ferramentas estudadas também não realizam o processo de otimização de dados entre os bancos de dados suportados, enquanto o sistema implementado realiza este processo entre o Firebird® e o MongoDB®.

4. Solução implementada

Esta seção apresenta, de uma forma mais detalhada as etapas realizadas durante a implementação do *software* descrito nesse artigo. Também são apresentadas as ferramentas utilizadas e as tecnologias que foram necessárias para o seu desenvolvimento. A implementação da solução foi desenvolvida em 6 etapas, que serão descritas nas subseções seguintes: 1) levantamento de requisitos e análise do banco de dados relacional; 2) Modelagem do banco de dados não-relacional; 3) Migração; 4) Validação da Migração, 5) otimização dos dados e 6) Comparativos.

O processo de levantamento de requisitos foi realizado com os *stakeholders* do projeto, sendo identificados quais dados deveriam ser migrados. Este processo foi realizado através de reunião para identificar as dificuldades de gerar informações da base de dados.

Para utilização do software desenvolvido, em um primeiro momento, devem ser feitas as conexões entre as duas bases de dados, a fim de garantir que o computador possui todos os *drivers* e *softwares* necessários para que a solução implementada possa ser executada. Para realizar a conexão devem ser utilizados os seguintes parâmetros:

Usuário, senha e porta. Para o Firebird® também é preciso informar o caminho da base dados.

Estabelecendo a conexão a mesma é mantida ativa para que possa ser feita a manipulação das informações entre o Firebird® e o MongoDB®. A partir do momento em que todas as manipulações e buscas forem concluídas, e a conexão não for mais necessária, a mesma é fechada.

Com a conexão feita nas duas bases é dado o início ao processo de migração das tabelas que foram previamente selecionadas pelos *stakeholders.* Neste processo é feita uma busca para cada uma das tabelas do Firebird e criado automaticamente um documento semelhante no MongoDB®. Este processo será explicado mais detalhadamente no tópico sobre migração.

Após a migração, para garantir que não houve perda de informações são realizadas duas validações, a primeira por quantidade total de registros e a segunda validando coluna a coluna de cada registro. Estes processos serão explicados de forma mais específica no tópico que trata sobre a validação da migração.

Não havendo identificação de falhas durante o processo de validação da migração, será dado início ao processo de otimização dos dados no MongoDB®. Este processo é dividido em duas tarefas, sendo elas: otimização por chaves e otimização por registros. Estes processos e tarefas são explicados detalhadamente e com exemplos no tópico referente à otimização dos dados.

Para comprovar a melhoria tanto no tamanho do banco quanto no desempenho de busca das informações, serão apresentados comparativos. Estes serão apresentados de duas formas, tempo de busca por tabela e tamanho do banco de dados. Estes processos estão detalhados no tópico comparativos.

Os processos de migração, validação da migração, otimização e comparativo, são realizados para cada tabela existente no banco de dados Firebird®. A figura 3 apresenta a forma como o software desenvolvido neste artigo funciona.

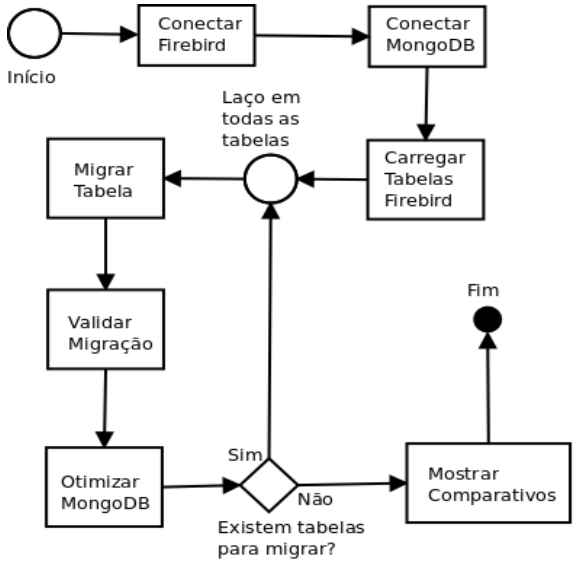

Figura 3 – Fluxograma de funcionamento do sistema (Fonte: do Autor)

4.1. Levantamento de Requisitos e Análise da Base de dados relacional

O primeiro passo para o desenvolvimento do software foi realizar o levantamento de quais dados deveriam ser migrados, juntamente aos responsáveis e *stakeholders*. Após o levantamento foi gerado um diagrama ER (Entidade-Relacionamento) da base de dados Firebird, com o objetivo de identificar as tabelas e colunas que deveriam ser migradas, tendo com base as informações coletadas com o apoio dos *stakeholders*.

Com o término deste processo foi possível obter informações tais como: quantidade de tabelas, tipo de dados, colunas e número de registros das tabelas. Também verificou-se que não existiam relacionamentos entre as tabelas, o que impossibilitou a validação de integridade do processo de validação da migração.

Notou-se, também, durante este levantamento, que a base de dados estava bastante lenta ao realizar consultas de informações, assim como existiam diversos campos que sempre estavam sem conteúdo. Destaca-se também o fato de que as tabelas não possuíam chaves primárias, nem estrangeiras, comprometendo a integridade e o desempenho de um banco de dados relacional. Estes pontos dificultaram bastante o entendimento e análise da base de dados Firebird, pois a mesma também não possuía nenhuma documentação ou dicionário de dados.

A figura 4 apresenta o diagrama ER das tabelas e campos que deveriam ser migrados do Firebird® para o banco de dados MongoDB®.

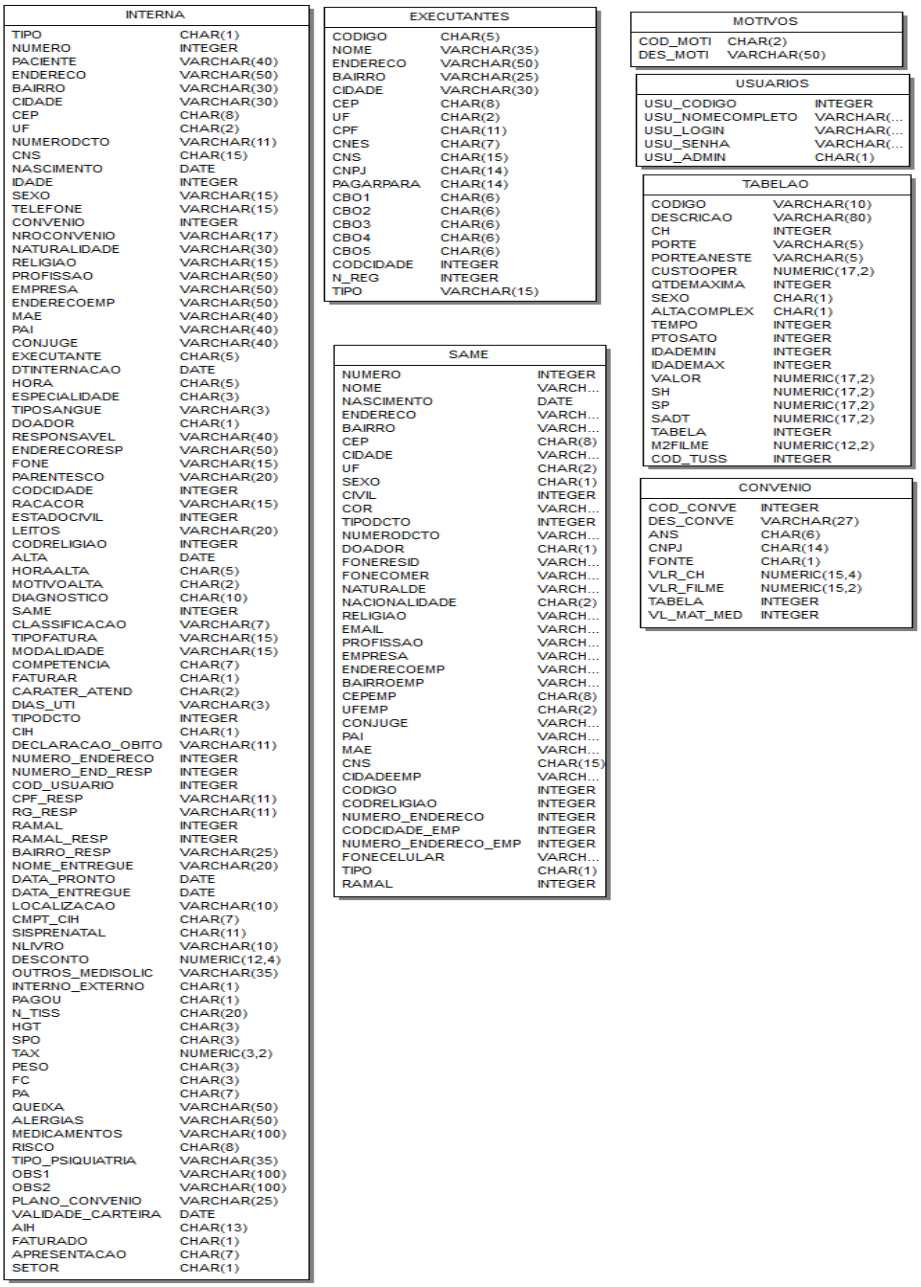

Figura 4 – Diagrama ER das tabelas a serem migradas

4.2. Modelagem da Base de dados não-relacional

Após a análise do banco de dados Firebird® notou-se que, em virtude da complexidade da base de dados, seria muito difícil descartar campos e tabelas do processo de migração. Outro ponto importante era contornar alterações na estrutura da base de dados do hospital, sem ter que alterar o software implementado.

Tendo isso em vista, a modelagem do banco de dados MongoDB®, seguiu da mesma forma do Firebird, respeitando as mesmas tabelas e campos. Porém, no processo de otimização do banco de dados MongoDB®, as colunas e tabelas são removidas se as mesmas não possuírem informações.

Com base nisto, não foram gerados diagramas de estrutura do banco de dados MongoDB®, pois conforme as tabelas do Firebird® foram alimentadas e alteradas, as coleções de documentos do MongoDB® sofreram as mesmas alterações, fazendo com que, a cada nova migração, o diagrama tenha que ser alterado.

4.3. Configuração do Software

O *software* implementado possui dependências, sendo elas do Firebird®, MongoDB®, Python, Pymongo e FDB. Estas ferramentas foram apresentadas e explicadas no tópico 2.5 – Python.

O computador que executará o *sofware* deve ter instalado todas as ferramentas citadas no tópico Python, usando como o guia de instalação de cada ferramenta, respeitando o seu sistema operacional e a arquitetura a arquitetura do mesmo.

Para realizar a conexão com o MongoDB®, o mesmo deve ter sido instalado na porta 27017 do computador em questão. Já para o Firebird®, deve ser apenas colocada a base de dados na mesma pasta do *software*, o nome do arquivo deve ser Dados.fdb.

Para executar o software, o mesmo deve ser aberto através do terminal do Python. Assim que for executado serão apresentadas as mensagens de erro de conexão se tiver alguma configuração incorreta no computador. Se não houver falha é dado início ao processo de migração dos dados.

4.4. Migração

A migração é o primeiro processo realizado pelo software, que visa buscar todas as informações da base de dados Firebird® e transferi-las para o MongoDB®.

A migração está estruturada de forma que, se houver qualquer alteração na estrutura da base de dados do hospital, a migração pode replicar ainda assim mesmo as informações. Por exemplo, caso seja criada uma nova coluna na tabela INTERNA, o processo de migração será capaz de copiar esta nova coluna também.

Na figura 5 é mostrada a mensagem de início e fim do processo de migração da tabela SAME da base de dados Firebird®.

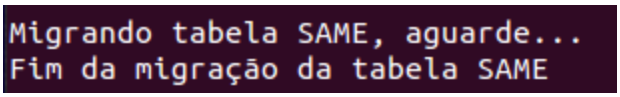

Figura 5 – Migração da tabela SAME

A migração é feita com base nas tabelas levantadas no tópico 4.2. Para cada tabela, antes do início do processo de transferir os dados, são feitos outros dois processos. Primeiramente é criada a coleção respectiva no MongoDB®, ou seja, no momento de migrar a tabela SAME, é criada a coleção SAME na base de dados MongoDB®. Após isso, é realizada uma busca de todas as colunas existentes na tabela em questão do Firebird e armazenadas em uma variável. Isso possibilita, que ao alimentar o MongoDB®, todas as colunas sejam migradas e mantidas com o mesmo nome.

Feito isso, é dado o início ao processo de transferir os dados de uma base de dados para outra. Neste momento é feita uma busca na tabela em questão do Firebird buscando todas as colunas de todas as linhas da tabela. Após carregadas estas informações, é feito um comando de repetição *for*, para que possam ser percorridos todos os registros da tabela. Para cada registro é usado o nome do campo para a chave e o conteúdo para o valor da coleção no MongoDB®.

Os tipos de dados não são mantidos neste processo em virtude da tipagem de dados ser automática na base de dados MongoDB®. No processo de otimização dos dados são feitas diversas alterações na tipagem, sendo que muitas vezes os dados estão gravados e/ou foram migrados como texto, mas o conteúdo armazenado é inteiro.

Este processo tem um desempenho bastante relativo à quantidade de colunas e volume de dados de cada tabela. Em tabelas como a INTERNA, que possui muitas colunas e linhas, o processo demora em virtude da busca no banco de dados Firebird.

4.5. Validação da Migração

O processo de validação tem o objetivo de garantir que todo o conteúdo da tabela foi migrado. É importante garantir que todos os dados foram migrados corretamente para que se tenha certeza de que o processo de otimização seja realizado em toda a base de dados, podendo assim gerar os comparativos com exatidão, pois serão comparados o mesmo volume e quantidade dos dados. A validação é dada em dois passos, sendo eles: validação por quantidade de registros e validação por conteúdo.

No processo de validação por quantidade é feita uma contagem de linhas na tabela do Firebird® e na respectiva coleção de dados do MongoDB®, garantindo que todas as linhas de informações do Firebird® foram migradas para a nova base de dados. Este processo é bastante rápido, pois retorna apenas uma linha e uma coluna da consulta no Firebird, enquanto no MongoDB® este dado já está armazenado na coleção de dados, não sendo necessária qualquer busca no banco.

Quando houver falha neste processo será apresentada a seguinte mensagem “Falhou – Validação de Quantidade de Registros”, logo abaixo da mensagem é mostrada a quantidade de registros no Firebird e a quantidade de registros no MongoDB®. Esta mensagem é mostrada a fim de que seja revisto o processo de migração para identificar o que causou a falha na migração. Quando este processo não apresentar diferenças, é mostrada a mensagem “OK – Validação de Quantidade de Registros”.

Após concluída a validação por quantidade de registros é dado início ao processo de validação por conteúdo. Este processo valida em cada linha de dados, todas as colunas, ou seja, analisa todos os campos da tabela, checando se os dados que estão no Firebird também estão no MongoDB®.

Este processo busca primeiramente todas as colunas da tabela na base de dados Firebird, para garantir que todos os campos da tabela serão comparados. Feito isso realiza-se uma busca de todas as informações na tabela na base de dados Firebird®. Esta busca retornará todos os dados armazenados naquela tabela da base de dados do hospital. Quando o processo de busca terminar, executa-se um comando de repetição

for em todas as linhas da tabela. Para cada linha, verifica-se o conteúdo de cada coluna e compara-se com o mesmo registro, com a mesma chave da mesma coleção de dados do MongoDB®, a fim de comparar se o conteúdo das duas bases de dados é idêntico.

Caso seja identificada uma falha neste processo será apresentada a seguinte mensagem: “Falhou – Validação de Dados”, logo abaixo será mostrado o conteúdo que não foi encontrado na base de dados MongoDB®. Essa mensagem permite que seja identificada a linha em que ocorreu a diferença, podendo assim ser analisado o problema e corrigido, se necessário no processo de migração. Se não forem identificadas falhas no processo de validação por conteúdo será apresentada a mensagem “OK – Validação de Dados”.

Este segundo processo é bastante demorado, pois além do fato de ter que buscar todas as informações de uma determinada tabela do Firebird®, ainda é preciso fazer um comparativo de todos os campos armazenados na base de dados MongoDB®.

Na figura 6 são mostradas as mensagens apresentadas durante o processo de validação da migração da tabela SAME.

```
Validando tabela SAME, aguarde...
    OK - Validação de Quantidade de Registros
    OK - Validação de Dados
Fim da validação da tabela SAME
```

Figura 6 – Validação da migração da tabela SAME

Sempre que for identificada alguma falha no processo de migração o software mostrará as mensagens e terminará a sua execução, pois se houver falhas os processos de otimização e comparativos serão falhos, já que não tratarão as mesmas informações nas duas bases de dados.

4.6. Otimização dos Dados

O processo de otimização dos dados visa remover colunas sem conteúdo, o com conteúdo idêntico em todos os registros, assim como melhorar a tipagem dos dados a fim de garantir que se está sendo usada a tipagem mais apropriada para o dado. Este processo é executado somente na base de dados MongoDB®.

Este processo tem como principal objetivo ajustar os dados, a fim de garantir que as informações estejam disponíveis e que, quando forem solicitadas, as mesmas sejam acessíveis de forma rápida e com fácil identificação dos conteúdos.

O principal diferencial do software implementado está no processo de otimizar as informações migradas entre as duas bases de dados. Foram implementados os dois processos de otimização de dados, sendo eles: otimização por chaves e otimização por registros.

O primeiro processo é chamado de otimização por chaves e tem como objetivo a remoção de chaves que contenham o mesmo valor em todos os registros da coleção de dados do MongoDB®. Para isso é feita uma busca em toda a coleção de dados do MongoDB® e, após o término da busca, executa-se um comando *for* para percorrer todos os registros, comparando uma determinada chave de valores. Caso todos os

registros da coleção contenham o mesmo valor a chave é removida, pois está armazenando a mesma informação, tornando a mesma desnecessária para comparativos ou consultas.

Após o término deste processo é apresentada a seguinte mensagem: “Remoção de chaves com mesmo valor em todas as linhas: Foram removidas 8 chaves”, no caso o valor 8 variará de acordo com a quantidade de chaves removidas. Logo após a mensagem serão mostradas todas as chaves removidas também, por exemplo: “Chaves removidas: DECLARA_NASC3, DECLARA_NASC2”.

Durante este mesmo processo também é checada a possibilidade de alterar a tipagem de dados da chave para *booleano*, tendo em vista que muitos campos estão armazenados como “S” para sim (*True*) e “N” para não (*False*) na base de dados Firebird. Este processo valida em todos os registros se uma determinada chave possui somente dois valores. Caso for identificada essa situação o software alterará o primeiro valor para verdadeiro e o segundo valor para falso. Feito isso todos os registros daquela chave terão sua tipagem alterada para *booleano*, fazendo a base de dados ficar mais leve, mais flexível e mais rápida quando comparada ao valor de texto para Sim/Não.

Quando acontecer esta situação, será apresentada a seguinte mensagem: “Chaves com apenas dois valores, atualizadas para *booleano*: Foram alteradas X chaves”, o valor de X variará com a quantidade de chaves que foram alteradas. Logo abaixo a esta mensagem são mostradas todas as chaves que foram alteradas, por exemplo: “Chaves alteradas: EMAIL”.

O segundo processo de otimização de dados é chamado de otimização por registros, que trata separadamente cada linha da coleção de dados. Enquanto no primeiro processo eram feitas alterações em todas as linhas, neste as alterações são específicas e pontuais para cada linha de informação.

Em um primeiro momento são buscadas todas as linhas da coleção de dados. Após, analisa-se cada chave de dados da linha realizando a seguinte validação: caso o conteúdo seja nulo a chave é removida e posteriormente é mostrada a seguinte mensagem: “Remoção de chaves sem informação: Foram removidas X chaves”, o valor X variará de acordo com a quantidade de chaves removidas por estarem com conteúdos nulos. Caso o conteúdo da chave for diferente de nulo, é feita a tentativa de alterar o tipo de dados de texto para numérico.

Primeiramente é feita uma validação para identificar qual o tipo de numérico é utilizado: *Integer* ou *Float.* Após identificado o tipo do dado, realiza-se uma atualização no conteúdo da chave para a mesma ficar armazenada de acordo com o novo tipo. Ao concluir a operação é apresentada a seguinte mensagem: “Chaves atualizadas de Texto para Numérico: Foram alteradas X chaves”, o valor de X sofrerá variações conforme a quantidade de chaves que tenham sido atualizadas na coleção de dados. Este processo é muito importante, pois devido ao fato da migração ser feita sem manter a tipagem dos dados isso garante que as informações sejam alteradas para o tipo de origem e/ou otimizadas para uma tipagem mais específica para cada dado.

As mensagens apresentadas durante o processo de otimização permitem que o usuário visualize as colunas que não são usadas na base de dados Firebird, podendo

assim ser feita posteriormente uma otimização também na base de dados do hospital, a fim de aumentar o desempenho e diminuir a complexidade da mesma.

Na figura 7 são mostradas as mensagens apresentadas durante o processo de otimização dos dados da coleção SAME.

```
Otimizando tabela SAME, aguarde...
    Remoção de chaves com mesmo valor em todas as linhas:
        Foram removidas 4 chaves
        Chaves removidas: BAIRROEMP, CIDADEEMP, NUMERO_ENDERECO_EMP, RAMAL
    Chaves com apenas dois valores, atualizadas para booliano:
        Foram alteradas 1 chaves
        Chaves alteradas: EMAIL
    Remoção de chaves sem informação:
        Foram removidas 1006311 chaves
    Chaves atualizadas de Texto para Numérico:
        Foram alteradas 631000 chaves
Fim da otimização da tabela SAME
```

Figura 7 – Otimização da coleção de dados SAME

O processo de otimização é bastante importante, pois ele mostra que na base de dados existem muitos campos que não possuem nenhuma informação, sendo que não há a necessidade de existirem. Outro ponto é o fato de que milhares de chaves são removidas, pois estão com conteúdo nulo, isso mostra que muitos campos estão sendo preenchidos em raros momentos.

Após o processo de otimização pôde-se notar que a base de dados ficou mais rápida e leve. Também notou-se que a base de dados ficou mais simples de trabalhar, pois são mantidos apenas os campos que possuem valores que podem ser utilizados para filtragens e buscas de informações. Isso permite que seja mais simples e rápida a manipulação dos dados e a extração de informações da base de dados.

Na figura 8 é mostrado um comparativo de busca das três maiores tabela do banco de dados do Hospital. Esse comprativo é feito entre a base de dados MongoDB® antes e depois do processo de otimização de dados.

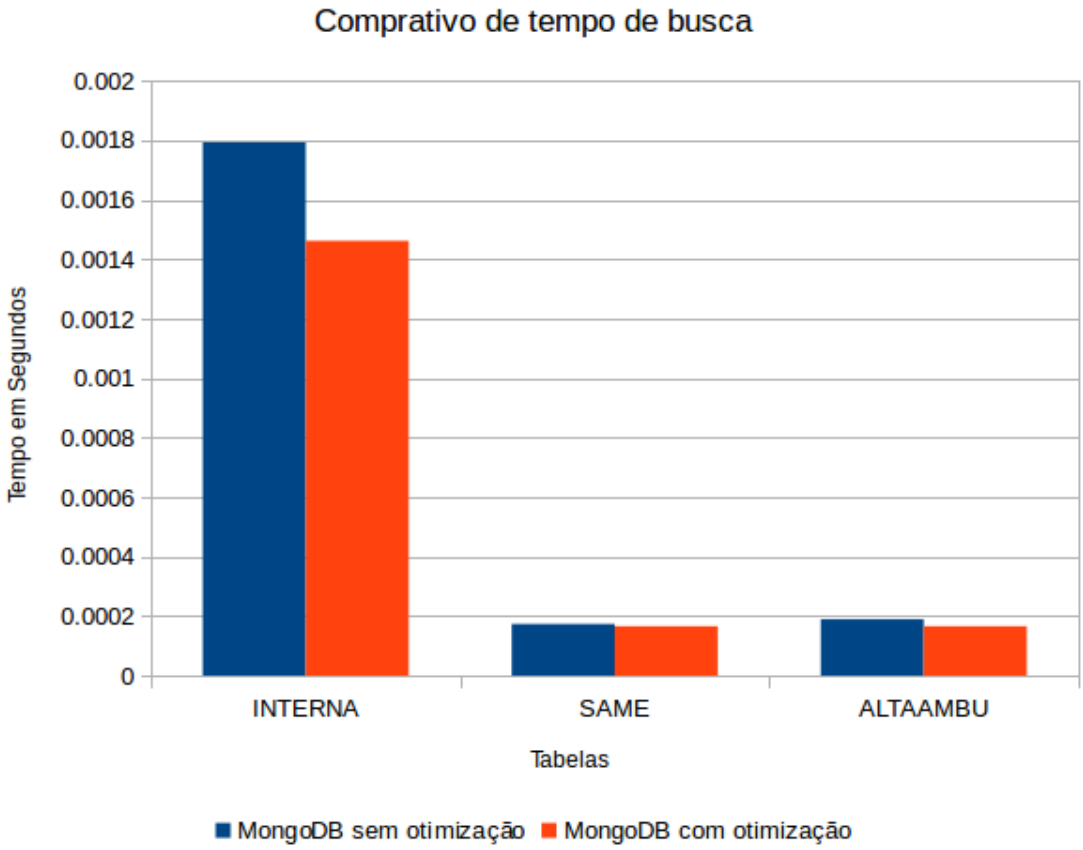

Figura 8 – Comparativo de tempo de busca

Essa figura mostra que houve ganho em desempenho após a otimização, esse ganho varia muito entre o tamanho e os dados da tabela de origem, porém nota-se que quanto maior a tabela, maior o ganho de desempenho.

Porém, o maior ganho de desempenho nota-se do Firebird® para o MongoDB®, onde que a tabela INTERNA, no Firebird®, levava cerca de 58 segundos para executar a busca, enquanto no MongoDB®, não chega a levar meio segundo. Infelizmente não pode ser representado por gráficos pois a diferença foi muito grande, ocultando assim as informações.

4.7. Comparativos

Os comparativos têm como principais objetivos comprovar e validar que os objetivos do software desenvolvido foram atingidos. Os comparativos são apresentados de duas formas distintas, sendo elas: tempo de busca de cada tabela e tamanho da base de dados.

O comparativo por tempo é feito para cada tabela migrada pelo software desenvolvido. Este processo compara o tempo de busca de todas as informações de uma tabela no Firebird, com o tempo de busca de todas as informações da correspondente coleção de dados do MongoDB®. O tempo de busca na base de dados Firebird mostra-se muito mais demorado que a busca na base de dados otimizada do MongoDB®. Um exemplo que pode ser citado é a tabela SAME, o tempo de busca desta tabela levou

cerca de 08.62 segundos no Firebird®, enquanto no MongoDB® não chegou a meio segundo, cerca de 0.000183 segundos.

Na figura 9 são mostradas as mensagens apresentadas durante o processo de realização dos comparativos dos dados da tabela SAME.

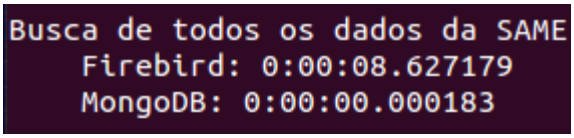

Figura 9 – Comparativo da tabela SAME

O segundo comparativo é relacionado ao tamanho do banco de dados. Neste processo comparou-se o tamanho total armazenado em disco de cada base de dados. Este processo é feito a fim de comprovar que tanto pelo uso do MongoDB®, quanto pela otimização a base de dados se tornou mais leve. O banco de dados Firebird® do Hospital de Frederico Westphalen possui cerca de 150 *megabytes* de dados em disco, apenas com as tabelas utilizadas para o desenvolvimento do software. Enquanto a base de dados MongoDB®, criada após a migração e otimização possui cerca de 4 *megabytes*, ou seja, muito mais leve e rápida.

Na figura 10 são mostrados os dados de comparativo de tamanho da base de dados Firebird® e da base de dados MongoDB®.

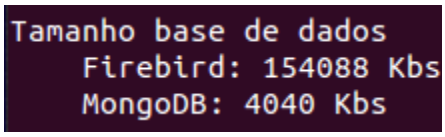

Figura 10 – Comparativo de tamanho de base de dados

Os comparativos comprovam que o software implementado auxilia na solução dos problemas de desempenho, já que a busca de todas as informações ficou muito mais rápida e que a manipulação da base de dados ficou mais simples.

5. Considerações Finais

Durante o processo de migração foi comprovado, com base no desenvolvimento e demonstração do software apresentado, que é possível realizar a migração de uma base de dados relacional para uma base de dados não relacional, sem precisar fazer alterações na base de dados e não perder informações durante este processo. A migração pode ser vista como uma solução para softwares que utilizam uma base de dados relacional e que podem ter um desempenho melhor utilizando outro modelo de banco de dados.

Com relação à otimização de dados foi possível notar que muitas colunas não estão sendo utilizadas na base de dados Firebird® do Hospital de Frederico Westphalen.

Outro ponto que se destaca durante o processo de otimização é o fato de que os campos variam bastante, ou seja, em algumas linhas alguns campos são informados, enquanto em outras linhas, outros campos são preenchidos. Isso deixa a base de dados bastante lenta ao se buscar os dados, pois existem diversos campos que não contêm informações. A otimização foi muito importante neste software e mostra que a escolha de uma base de dados não relacional foi correta para o problema em questão.

Durante todo o desenvolvimento do software foram realizados diversos estudos sobre *BigData*, MongoDB® e Python. Estes tópicos haviam sido citados durante o curso, mas não de maneira prática durante toda a graduação em Sistemas de Informação. Durante a II JASI (Jornada Acadêmica de Sistemas de Informação) foi realizado um Minicurso de MongoDB® + Python para demonstrar o funcionamento e criar uma aplicação usando essas duas ferramentas. Estes estudos, bem como o minicurso realizado, comprovam que houve um grande aprendizado no desenvolvimento deste trabalho.

Por fim, destaca-se o fato de que o software ficará disponível para toda a comunidade acadêmica, possibilitando assim, futuras migrações na base de dados do Hospital de Frederico Westphalen conforme a necessidade. Ficará disponível também o código fonte do software, para que possam ser feitas melhorias e possíveis alterações. O software funciona de maneira que qualquer alteração na base de dados Firebird® seja replicada para a base de dados MongoDB®, como seria o caso de remover ou adicionar colunas na base de dados do Hospital de Frederico Westphalen. Ressalta-se a importância de disponibilizar o software para a comunidade acadêmica da UFSM/CESNORS, tendo-se em vista o acordo de cooperação existente entre o Hospital Divina Providência e a UFSM, além dos diferentes projetos de pesquisa e de extensão em desenvolvimento, decorrentes desta cooperação.

Durante o desenvolvimento do software teve-se, como principais dificuldades, a identificação e compreensão do funcionamento da base de dados Firebird do Hospital de Frederico Westphalen, além da dificuldade em implementar o software para que o mesmo funcionasse de forma dinâmica, ou seja, que em caso de alteração da base de dados de origem, o software não precisasse ser alterado.

Ao fazer a análise na base de dados utilizada pelo Hospital de Frederico Westphalen, notou-se que a mesma estava bastante caótica, não possuindo documentação. Além disso, o objetivo da existência e utilidade de algumas tabelas bastante complexo, tais como as tabelas TABELAO e INTERNACANCELA.

Durante o desenvolvimento do verificou-se que, caso ocorresse qualquer alteração na base de dados do Hospital, o software teria que ser capaz de realizar a migração e otimização sem erros e sem perda de informação. Isso dificultou o desenvolvimento do software pois o mesmo foi implementado para que este problema seja contornado automaticamente conforme descrito no tópico de migração.

O software implementado realiza o que foi proposto de forma eficaz, sendo algumas rotinas bastante complexas, porém ainda há muito que pode ser feito para tornar o software mais fácil de utilizar e realizar o mesmo processo entre outros bancos de dados.

O software não possui interface gráfica, o que para usuários com menor conhecimento técnico, constitui-se em uma dificuldade para seu manuseio. Fica como sugestão de trabalho futuro, desenvolver a interface gráfica para configurar as conexões com o banco de dados, quais tabelas devem ser processadas, quais processos realizar e criar relatórios para facilitar a leitura de cada processo realizado pelo software.

Outra melhoria que poderia ser implementada é permitir que sejam feitas outras migrações, utilizando outros bancos de dados, tais como a migração de MySQL® para MongoDB®. A estrutura do software está bastante dinâmica e adicionar suporte a novos bancos de dados seria bastante interessante. Com algumas alterações simples o software é capaz de migrar qualquer base de dados Firebird® para MongoDB®.

Referências

COSTA, Elisângela Rocha. **Banco de Dados Relacionais**. São Paulo: Faculdade de Tecnologia de São Paulo, 2011.

DIANA, Mauricio; GEROSAL, Marco A. **NOSQL na Web 2.0: Um Estudo Comparativo de Bancos Não-Relacionais para Armazenamento de Dados na Web 2.0**, São Paulo, Departamento de Ciência da Computação – Universidade de São Paulo, 2010.

FIREBIRD. **Documentação FDB**. Disponível em: <http://pythonhosted.org//fdb/>. Acesso em 10/05/2014.

FREITAS, Marcel. **Sistema de banco de dados relacional**. Disponível em: <http://marcelmesmo.blogspot.com.br/2011/08/sistema-de-banco-de-dados-relacional.html#.U5rgWfldWSo>. Acesso em 28/05/2014.

GONSOWSKI, Dean. **Analyzing data:** Why a "bigger is better" mentality may be at odds with intelligent information governance. Disponível em: http://www.insidecounsel.com/2012/09/27/analyzing-data-why-a-bigger-is-better-mentality-ma>. Acesso em 26/05/2014.

GUEDES, Anselmo. **Migração de dados:** não deixe essa atividade tornar seu projeto um fracasso. Disponível em: <http://imasters.com.br/desenvolvimento/migracao-de-dados-nao-deixe-essa-atividade-tornar-seu-projeto-um-fracasso>. Acesso em 10/05/2014.

IBM, **Infográfico BigData IBM**. Disponível em: <http://www.ibm.com/midmarket/br/pt/infografico_bigdata.html>. Acesso em 26/05/2014.

LEITE, Gleidson S. **Análise comparativa do teorema CAP entre os bancos de dados NoSQL e bancos de dados relacionais**, Fortaleza, Faculdade Farias Brito Ciencia da Computação, 2010.

MICROSOFT, **SQL Server Integration Services.** Disponível em: <http://msdn.microsoft.com/pt-br/library/ms141026.aspx>. Acesso em 28/05/2014.

NASCIMENTO, Cledilson. **Transações ACID.** 2011, Disponível em: <http://blog.cledilsonweb.com.br/2011/02/transacoes-importancia-do-acid-para-um.html>. Acesso em 27/05/2014.

NASCIMENTO, Jean. **NoSQL – Você realmente sabe do que estamos falando?**, 2014, Disponível em: <http://imasters.com.br/artigo/17043/banco-de-dados/nosql-voce-realmente-sabe-do-que-estamos-falando/>. Acesso em 27/05/2014.

NARAYANASWAMY, Anand. **Do SQL Server ao MySQL: conheça a nova ferramenta de migração da Oracle**. Disponível em: <http://www.infoq.com/br/news/2012/11/migracao-sqlserver-oracle>. Acesso em 28/05/2014.

OLIVEIRA, Débora. **Big Data o desafio de garimpar informações**, Computer World, 2013, Fevereiro, Edição número 554.

PENTAHO, **Data Integration.** Disponível em: <http://community.pentaho.com/projects/data-integration/>. Acesso em 28/05/2014.

PYMONGO. **Documentação Pymongo.** Disponível em: <http://api.mongodb.org/python/current/>. Acesso em 10/05/2014.

PYSCIENCE Brasil. **Python:** O que é? Por que usar? Disponível em: <http://pyscience-brasil.wikidot.com/python:python-oq-e-pq>. Acesso em 10/05/2014.

RÊGO, Bergson L. **Gestão e Governança de Dados: Promovendo dados como ativo de valor nas empresas**. Tijuca/Rio de Janeiro, BRASPORT, 2013.

VASCONCELLOS, Cristhiano B. **Banco de Dados I**. Disponível em: <http://www.profs.iffca.edu.br/~cristhianobv/portal/disciplinas/banco_dados/Apresentacao_bd_7.pdf>. Acesso em16/06/2014.

VIEIRA, Marcos R.; FIGUEIREDO, Josiel M.; LIBERATTI, Gustavo; VIEBRANTZ, Alvaro F. **Bancos de Dados NoSQL: Conceitos, Ferramentas, Linguagens e Estudos de Casos no Contexto de Big Data**. Simpósio Brasileiro de Bancos de Dados. 2012.