Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Análise de Desempenho no Processamento de Dados para Cálculo de Resultado Gerencial de Instituição Financeira em Base de Dados NoSQL Colunar

Felipe de Moura Rezende dos Santos

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Orientadora
Prof. ª Dr. ª Maristela Terto de Holanda

Brasília
2019

Ficha catalográfica elaborada automaticamente,
com os dados fornecidos pelo(a) autor(a)

dAN532a

de Moura Rezende dos Santos, Felipe
Análise de Desempenho no Processamento de Dados para Cálculo de Resultado Gerencial de Instituição Financeira em Base de Dados NoSQL Colunar / Felipe de Moura Rezende dos Santos; orientador Maristela Terto de Holanda. -- Brasília, 2019.
58 p.

Dissertação (Mestrado - Mestrado Profissional em Computação Aplicada) -- Universidade de Brasília, 2019.

1. NoSQL. 2. NoSQL Colunar. 3. Hadoop. 4. HBase. 5. Pig. I. Terto de Holanda, Maristela, orient. II. Título.

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Análise de Desempenho no Processamento de Dados para Cálculo de Resultado Gerencial de Instituição Financeira em Base de Dados NoSQL Colunar

Felipe de Moura Rezende dos Santos

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Prof. ª Dr. ª Maristela Terto de Holanda (Orientadora)
CIC/UnB

Prof. Dr. Marcelo Ladeira
CIC/UnB

Prof. ª Dr. ª Nádia Kozievitch
Universidade Tecnológica Federal do Paraná (UTFPR)

Prof. ª Dr. ª Aletéia Patrícia Favacho de Araújo
Coordenadora do Programa de Pós-graduação em Computação Aplicada

Brasília, 11 de dezembro de 2019

Dedicatória

Dedico este trabalho a todos os nobres colegas da turma do Programa de Pós graduação em Computação Aplicada - PPCA de 2017. Formamos uma família que a todo momento demonstrava companheirismo e auxílio para o que precisasse.

Dedico também aos meus filhos: Bianca Elena, Théo e Caio. Gostaria de demonstrar que, mesmo vindo de baixo, é possível chegar aonde se sonha. Ou até mesmo chegar mais alto do que jamais se imaginou chegar. Sonhos são feitos para sonhar, mas se não fizer o seu papel, eles permanecerão como sonhos. Corra atrás e se dedique. EU AMO MUITO VOCÊS!!!

Agradecimentos

Primeiramente, sou muito grato pela pessoa que tenho ao meu lado. Jéssica, minha esposa, que me ajudou a todo momento durante esses longos anos entre "Aluno Especial" e "Aluno Regular". Quando me inscrevi para cursar a minha última matéria como aluno especial, meu filho estava prestes a nascer, mesmo assim minha esposa me deu todo apoio e suporte para cursar a matéria, e depois, com ele ainda recém nascido, elaborar meu pré-projeto. Ingressamos (não posso considerar que entrei sozinho) com o pequeno completando 6 meses, foram noites sem dormir (não pelo pequeno que era um bom menino, mas pelos estudos). Final do primeiro semestre de aulas e não é que engravidamos novamente? Pois é, previsão de nascimento para julho de 2018, mesma data para a qualificação da minha dissertação e término das aulas do tronco específico. Mas em todo momento ela estava lá, me dando suporte, brigando para que eu arrumasse um tempo para estar ao lado do meu filho. Sou muito grato pela pessoa que tenho ao meu lado.

Meus pais não podem ficar de fora desta seção de agradecimento, pois sempre me ensinaram que o peixe não vem sozinho à nossa mesa. Temos que correr atrás de tudo o que queremos e sonhamos; confesso que nunca me imaginei cursando um mestrado, porém vi, durante este período como aluno especial, que era algo possível e acabou se tornando um objetivo de vida. Obrigado Carlos Antônio e Fernanda por todo ensinamento que me deram durante minha infância, vocês são meus exemplos, meus espelhos e meus amores.

Carla e Caroline são pessoas que tenho comigo a todo momento de minha vida, minhas irmãs. Carla, que foi meu exemplo no sentido de "sair de casa" e morar longe, saiu aos 20 anos para morar em outro país, a fim de estudar e buscar novos conhecimentos. Nunca mais voltou. Caroline, minha irmã caçula, minha afilhada, menina que resolveu entrar para a mesma área de tecnologia, fazendo faculdade de Engenharia da Computação, tenho muito orgulho do que você pode conquistar nessa vida, busque sempre o melhor, busque sempre seus desejos e seus sonhos, eles não vêm sozinhos, corra atrás.

Aos meus avós paternos (in memoriam) e maternos, que sempre me apoiaram e fizeram tudo o que estava aos seus alcances por mim. Sempre me trataram com imenso carinho e respeito. Vô Carlos e Dona Nega, a saudade é grande nesse momento de vitória, mas sei que permanecem olhando por todos nós e sempre me ajudando nas melhores escolhas

para a vida. "Seu" Fernando e Vó Tereza, todo carinho do mundo por vocês. Faziam de tudo para minha felicidade e sempre me defendiam da minha mãe (rs). A vocês ficará o meu eterno carinho e admiração. Muito obrigado!!!

Minha orientadora, Maristela, foi uma pessoa que, quando estive como aluno especial em sua disciplina, pude admirar tamanho conhecimento no assunto que eu pretendia escrever na dissertação. Consegui que fosse minha orientadora, aplicando meu pré projeto. Mesmo se afastando do Brasil para cursar o seu pós doutorado, permaneceu com contatos por meio de Skype e e-mail com as correções da dissertação e recomendações de pesquisa. Muito obrigado pelo carinho, dedicação e paciência.

Não posso deixar de agradecer a minha instituição de trabalho, que permitiu minha dispensa em todas as sextas feiras para que pudesse estar presente nas aulas, durante todo este ano de ensino. E quando apresentei minha proposta de estudo, solicitando apoio para a execução do trabalho, me concederam um mês de trabalho isolado como pesquisador para aplicar meus estudos e documentar os resultados obtidos. Obrigado Márvio, diretor na época do pré projeto; Daniel, atual diretor, Roseane, Alfredo, Marcelo, Fabiano e Cristiano, gerentes executivos. Ao Fabiano guardo um agradecimento especial, pois foi para quem apresentei meu pré projeto para aprovação da dispensa para estudo e que, toda vez que se encontrava comigo nos corredores, me questionava sobre como estavam os estudos. Um simples questionamento que demonstrava um carinho/preocupação que ficaram guardados na memória. A Roseane, o meu agradecimento pelo carinho e entendimento da minha real necessidade de utilizar o afastamento permitido pelo banco para conclusão do trabalho de dissertação acadêmica do mestrado, haja vista ser bolsista da instituição.

Por último, mas o mais especial, tenho que agradecer a Deus por tudo o que tenho conquistado em minha vida, não tive nada de mão beijada, mas ele me ajudou, sempre mostrando o melhor caminho a ser seguido, sempre me tirando dos caminhos que não levariam ao sucesso. Nunca me faltou saúde para que pudesse praticar tudo o que mencionei neste agradecimento e tudo o que já vivi. Sou muito grato por tudo o que já me proporcionou e por tudo o que ainda tenho a alcançar e conquistar. Ainda estou na luta por sucesso e felicidade.

Resumo

Big Data trouxe a necessidade de análise de grandes volumes de dados. Como nova tendência para armazenamento e processamento de informações em *Big Data*, foram apresentados os sistemas gerenciadores de banco de dados NoSQL (*Not Only SQL*). Mantendo contrato com a Apache/Hadoop, a instituição financeira Beta detém como estrutura de armazenamento e processamento de *Big Data* o ecossistema *Hadoop*, tendo o HBase, que é o sistema gerenciador de banco de dados NoSQL no qual a estrutura é baseada em armazenamento em colunas. Este trabalho tem como objetivo analisar o desempenho de processamento com a alteração de uma estrutura de base de dados, atualmente em sistemas gerenciadores de banco de dados relacional, para um sistema distribuído NoSQL colunar. Desta forma, era esperada uma redução no tempo de geração e disponibilização de valores para acompanhamento do resultado gerencial da instituição financeira Beta. Para uma análise da atividade, foram demonstrados os resultados de desempenho de cargas dos dados no Apache HBase e o tempo de processamento destes dados. Como resultado, segundo comparação com o processamento atual dos cálculos dos componentes de resultados gerenciais atualmente realizados na instituição, demonstrou uma melhora considerável.

Palavras-chave: NoSQL. NoSQL Colunar. Hadoop. HBase. Big Data.

Abstract

Big Data has brought the need for analyzing large quantities of data. In order to meet these requirements Not Only SQL (NoSQL) systems were developed. The financial institute Beta uses the Apache/Hadoop ecosystem to store and analyze its data. HBase is a NoSQL Columar Storage System. This paper analyzes a switch from a relational DBMS, to a distributed columnar NoSQL system. This should lead to a reduction in the generation time and availability of values to monitor the management results of the financial institution Beta. For an activity analysis, data load performance results was demonstrated in Apache HBase and the processing time of this data. As a result, compared to the current processing of management result component calculations currently performed at the institution, it has shown a considerable improvement.

Keywords: NoSQL. NoSQL Column Store. Hadoop. HBase. Big Data.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

API *Application Programming Interface.*

DRE Demonstrativo de Resultado do Exercício.

ETL *Extract, Transformation and Load.*

GFS *Google File System.*

HDFS *Hadoop Distributed File System.*

NoSQL *Not Only SQL.*

OLAP *Online Analytical Process.*

SGBD Sistema Gerenciador de Banco de Dados.

SGBDR Sistema Gerenciador de Banco de Dados Relacional.

SQL *Structured Query Language.*

Capítulo 1

Introdução

Um sistema contábil abrange os registros dos eventos financeiros e econômicos da organização. Ele tem a finalidade de organizar e resumir informações que possam ser consultadas a qualquer momento, além do perfil econômico de um determinado período [1].

A contabilidade gerencial é um processo de identificação, mensuração, análise e comunicação de informações financeiras, utilizado pela administração para planejamento e controle de uma empresa, para assegurar o uso apropriado de seus recursos. Além de auxiliar na tomada de decisões das empresas a contabilidade gerencial serve como base para a controladoria, principalmente na formação do painel de controle [2]. A contabilidade gerencial fornece informações necessárias para a administração e para o desenvolvimento de uma empresa.

Um grande diferencial da contabilidade gerencial é a apresentação de seus relatórios. Enquanto a contabilidade financeira deve seguir a legislação para a emissão de seus demonstrativos, a contabilidade gerencial não precisa seguir estas normas, e sim atender as necessidades dos usuários.

A boa gestão de uma empresa (e consequentemente sua lucratividade) depende de um plano financeiro com valores e metas bem definidos. Para que esse plano financeiro seja executado com o máximo de rigor e eficiência, o empresário poderá buscar os recursos da contabilidade gerencial. Essas informações fornecidas são transmitidas ao cliente por meio de relatórios, que detalham onde a empresa está desperdiçando dinheiro, quais as fontes de prejuízo, bem como as fontes de lucro.

Podem-se destacar como principais benefícios da contabilidade gerencial [3]:

- Disponibilizar informações que ajudam a administrar diversas áreas, utilizando melhor os recursos da organização;
- Subsidiar a organização para que ela tenha vantagem competitiva e crescimento sustentável;

- Aumentar a eficiência e eficácia de todas as funções de gestão empresarial;
- Ajudar a dar foco no objetivo, com as tomadas de decisão, formações de preços, etc.;
- Contribuir com a previsão dos controles financeiros;
- Ajudar a evitar excessos e desperdícios;
- Colaborar para a comunicação entre todos os níveis de gestão;
- Auxiliar no controle do custo de produção, aumentando o percentual de lucro;
- Oportunizar alternativas de redução de custos e despesas;
- Orientar as estratégias do processo decisório;
- Geração de indicadores do negócio e de seus departamentos.

Com o crescimento da geração de dados estruturados e não estruturados nos sistemas atuais, assim como também na Internet, com as redes sociais, Internet das Coisas, e outros, tem-se um massivo volume de dados que precisa ser armazenado e consultado. Para a gestão deste grande volume de dados foram criados os sistemas gerenciadores de banco de dados NoSQL (*Not only SQL*). Os NoSQL foram criados para trabalhar em sua maioria como sistemas abertos, distribuídos e que suportam o processamento de grandes volumes de dados.

Primeiramente, faz-se necessário informar que, de acordo com políticas de segurança da empresa onde o responsável por este estudo trabalha, não é possível identificá-la pelo nome de registro. Por isso, ela será aqui denominada **Instituição Financeira Beta**.

As operações financeiras são divididas em três tipos específicos. Operações de aplicação são operações em que o cliente aplica dinheiro na instituição; operações de serviço são situações bancárias realizadas na instituição; e operações de crédito são operações em que a instituição financeira concede crédito ao cliente.

Até o presente momento, somente 26 sistemas da instituição se encontram em ambiente corporativo para apreciação desses resultados gerenciais. Esses 26 sistemas administram somente operações de crédito na instituição. Os dados gerenciais da empresa Beta foram considerados como massivos volumes de dados, uma vez que em seus 26 sistemas são gerados diariamente 800 MB de quantidade de dados para cada sistema. Para ter seus resultados gerenciais calculados, a contabilidade gerencial manipula um grande volume de dados, uma vez que para se ter os cálculos de resultado gerencial, existe a necessidade de utilização dos dados referentes às operações dos clientes da instituição.

Os NoSQL podem ser classificados, segundo Nayak *et al.* [4], como: baseado em grafo, baseado em documento, baseado em chave-valor ou colunar. Uma das formas de gerenciar

esse grande volume de dados é utilizando um banco de dados NoSQL, que para evidenciação de resultados, em sua versão colunar segundo [5] [6] [7] [8], detém bons desempenhos para elaboração de *Data Warehouse* na visualização dos resultados em diversas granularidades e dimensões.

1.1 Definição do Problema

A Controladoria de uma instituição financeira é institucionalmente responsável por apurar o resultado gerencial de todas as operações realizadas em suas unidades, o que demanda um grande volume de dados. A estrutura de armazenamento de dados utilizada na instituição financeira Beta tem sido a base de dados relacional, contudo, com o crescente volume de informações a serem tratadas e analisadas esse Sistema Gerenciador de Banco de Dados Relacional (SGBDR) tem se mostrado insatisfatório quanto ao tempo de processamento para gerar tais resultados.

Atualmente, na Instituição Financeira Beta, esse trabalho de geração de valores para a contabilidade gerencial abrange mais de 26 sistemas com aproximadamente 40 milhões de registros em cada um. O modelo do banco de dados é relacional e há uma sequência de passos preestabelecidos para executar o cálculo dos componentes que formam o resultado gerencial das operações. Todos os dados são estruturados e contêm valores numéricos. Este processo tem execução nos finais de semana e tem um tempo de execução do processo de aproximadamente 30 horas.

No caso, os dados são fornecidos por sistemas legados que em seguida são disponibilizados para processamento. Devido aos problemas de armazenamento e desempenho, é realizado um agrupamento dos valores para que haja menos dados a serem processados. O sistema de resultado gerencial, propriamente, só tem processamento aos finais de semana e o processo de fechamento dos resultados é realizado no primeiro final de semana após o encerramento do mês de referência, cujos valores são gerados para o Demonstrativo de Resultado do Exercício (DRE). Por sua vez, o DRE indica o resultado, apresentando exclusivamente dados voltados ao desempenho da empresa no intervalo, ou seja, seu resultado - receitas, despesas e custos [9].

O resultado desse processamento é disponibilizado em dois sistemas evidenciadores de resultado, com formato e granularidade distintos. Um sistema apresenta os resultados agrupados por prefixo (cada unidade da instituição contém um número de prefixo) e cada componente de resultado presente em uma linha distinta. E o outro sistema demonstra os resultados por cliente e operação, com cada componente de resultado presente em um atributo da tabela de resultado gerencial. Vale lembrar que esse procedimento é executado

para todas as operações de todos os clientes da instituição (aproximadamente 65,2 milhões de clientes).

Diante do exposto, surgem alguns questionamentos, os quais esse trabalho pretende investigar. Seria possível usar NoSQL Colunar para armazenar os dados objeto da contabilidade gerencial? Essa alteração traria uma melhora considerável no tempo de processamento dos resultados gerenciais das operações? Haveria uma redução no tempo de apuração dos valores em questão?

1.2 Justificativa

A hipótese dessa dissertação é que com a alteração do banco de dados para NoSQL Colunar, seria possível propor a alteração do sistema de resultado gerencial para processamentos diários de acompanhamentos dos resultados das operações de crédito contratadas até o dia anterior, tendo em vista que diminuiria o tempo para processamento dos dados. Sendo possível também incluir uma demonstração desses valores processados em uma base de dados para consultas e validação dos *stakeholders*.

O desenvolvimento deste projeto se torna relevante para a atuação da Controladoria ao se ter em vista o impacto na redução do tempo que a hipótese levantada traria na apresentação das informações ali administradas, visto que as demandas desses dados têm prazo curto para entrega, já que interferem diretamente nos resultados contábeis, servindo de base para muitas tomadas de decisão e definição de condutas.

1.3 Objetivo

O objetivo geral desta dissertação é analisar o processamento de resultados gerenciais da instituição financeira beta utilizando um banco de dados NoSQL colunar, assim como também, analisar o desempenho do processo de apuração de resultado gerencial com a alteração da estrutura de dados. Devido à diferença de estruturas entre as operações de crédito, operações de aplicação e de serviços, este trabalho utilizará somente operações de crédito.

1.3.1 Objetivo Específico

Para alcançar o objetivo geral, foram definidos os seguintes objetivos específicos:

- Levantar a granularidade necessária para atender os evidenciadores disponibilizados dos resultados gerenciais;

- Definir um processo de ETL (Extração, Transformação e Carga - *Extract, Transformation e Load*) para importar dados do modelo relacional para o modelo NoSQL colunar de banco de dados;
- Desenvolver um programa para realizar os processamentos de cálculos dos componentes de resultado gerencial;
- Realizar o armazenamento dos valores de resultado gerencial em ambiente *Hadoop/Hive*;
- Comparar o desempenho em termos de tempo de processamento das soluções propostas.

1.4 Estrutura do Documento

Este documento está dividido da seguinte forma:

- O Capítulo 2 traz o referencial teórico necessário para a pesquisa, com um breve conceito de banco de dados relacional e banco de dados NoSQL. Faz uma descrição sobre *Data Warehouse* e sobre o ecossistema *Hadoop*. Além disso, apresentação sobre os trabalhos relacionados sobre este tema.
- No Capítulo 3 é descrita a proposta de solução para o problema apresentado nesta dissertação.
- O Capítulo 4 descreve a análise de resultados obtida com o processamento da metodologia proposta para o trabalho.
- A conclusão e os trabalhos futuros estão descritos no Capítulo 5.

As figuras ou tabelas de outros autores apresentam a indicação da fonte após a legenda. As tabelas e figuras resultantes dessa pesquisa não apresentam indicação da fonte por serem de autoria própria do autor dessa pesquisa.

Capítulo 2

Referencial Teórico

Este capítulo apresenta conceitos fundamentais para o desenvolvimento desta pesquisa. Na Seção 2.1 tem-se o conceito sobre Banco de Dados Relacionais. Na Seção 2.2 introduz-se o Banco de Dados NoSQL, apresentando conceitos básicos sobre o Banco de Dados baseado em Armazenamento de Documentos, Banco de Dados de Grafos, Banco de Dados de Chave-Valor, sendo dada ênfase ao Armazenamento de Colunas na Subseção 2.2.1, por ser essa a modalidade a ser trabalhada na dissertação. O conceito sobre *Data Warehouse* é apresentado na Seção 2.3. Na Seção 2.4 é descrito o Ecossistema Hadoop, contendo cinco subseções descrevendo estruturas utilizadas para execução do trabalho. Estudos relacionados ao tema proposto na dissertação são apresentados na Seção 2.5.

2.1 Banco de Dados Relacional

Um banco de dados relacional é uma coleção de dados com relacionamentos predefinidos entre si [10]. Esses itens são organizados como um conjunto de tabelas com colunas e linhas. As tabelas são usadas para armazenar informações sobre os objetos a serem representados no banco de dados.

Um esquema de banco de dados representa a configuração lógica da totalidade ou de parte de uma base de dados relacional. Ele pode existir tanto como uma representação visual quanto como um conjunto de fórmulas conhecidas como restrições de integridade que regem um banco de dados [11]. Como parte de um dicionário de dados, um esquema de banco de dados indica como os atributos que compõem o banco de dados se relacionam, incluindo tabelas, exibições, procedimentos armazenados e muito mais.

As restrições de integridade auxiliam a evitar danos em banco de dados, assegurando que mudanças feitas por usuários autorizados não resultem na perda de consistência de dados. As tabelas no modelo relacional podem ter relacionamentos, que são gerados a partir de chaves primárias (PK - *Primary Key*) e chave estrangeira (FK - *Foreing Key*).

A Figura 2.1 apresenta um modelo de relacionamento entre Empregado e Departamento. Como pode ser observado, o Empregado tem uma chave primária *idEmpregado* e uma chave estrangeira *Departamento_idDepartamento*, que cria a associação entre o Empregado e o Departamento.

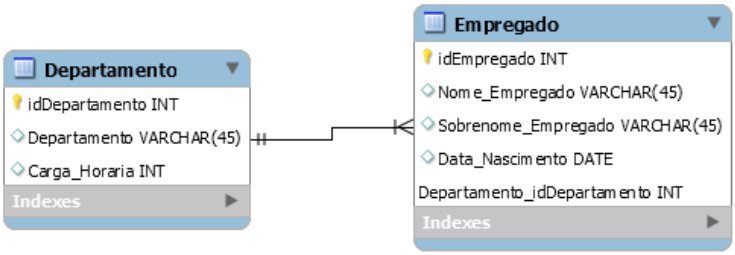


Figura 2.1: Exemplo de relacionamento entre duas tabelas em ambiente relacional.

2.2 Banco de Dados NoSQL

Armazenamento de Big Data é uma combinação de dados estruturados, semi-estruturados e não estruturados em larga escala [12]. Tem-se que o Big Data é caracterizado com 4v [13] [14] [8]: Volume, Veracidade, Velocidade e Variedade. Presente em [15] e [16] uma quinta dimensão, se tratando de análise de Big Data - Valor. Descrevendo que a análise de dados sem gerar valor não oferece contribuição para a organização. Ma-Lin Song *et al.* [17] colocam os 4 V (Volume, Veracidade, Velocidade e Variedade) como características de Big Data, porém acrescentam Valorização, descrevendo como 5 V estas características que acrescentam significativamente a complexidade de resolver problemas relevantes.

Nayak *et al.* [4] descrevem que para o armazenamento de Big Data, foram criados novos sistemas gerenciadores de banco de dados, os NoSQL, que podem ser categorizados em 4 tipos:

- Banco de Dados Baseado em Armazenamento de Documentos:

  Os bancos de dados de documentos usam um modelo orientado a documentos para armazenar dados. Eles armazenam um registro e seus dados associados em uma única estrutura chamada "documento". Cada documento contém vários atributos e valores associados [18]. Os documentos podem ser recuperados com base em valores de atributos usando várias interfaces de programação de aplicativos (API - *Application Programming Interface*) ou linguagens de consulta fornecidas pelo SGBD.

  O armazenamento de documentos é caracterizado pela organização de dados sem esquema. Isso significa que os registros não precisam ter uma estrutura uniforme,

ou seja, registros diferentes podem ter atributos diferentes. Os tipos de valores de atributos individuais podem ser diferentes para cada registro. Os registros também podem ter uma estrutura aninhada, enquanto os atributos podem ter mais de um valor, como um vetor.

Os armazenamentos de documentos geralmente usam formatos padrão, como *JavaScript Object Notation* (JSON) ou *Extensible Markup Language* (XML) para armazenar os registros. Isso permite que os registros sejam processados diretamente nos aplicativos. Documentos individuais são armazenados e recuperados por meio de uma chave. Além disso, os armazenamentos de documentos dependem de índices para facilitar o acesso aos documentos com base em seus atributos [19].

- Banco de Dados de Grafos:

  Bancos de dados de grafos são sistemas de gerenciamento de banco de dados que foram otimizados para armazenar, consultar e atualizar estruturas em grafo. Os grafos consistem em nós e arestas, em que os nós atuam como objetos e as arestas atuam como relacionamento entre os objetos [20].

  Em geral, os bancos de dados de grafos são bastante úteis quando os relacionamentos entre dados são mais importantes do que o próprio dado [21]. Por exemplo, ao representar redes sociais, gerar recomendações (por exemplo, venda cruzada) ou conduzir investigações forenses (por exemplo, detecção de padrões).

  Os bancos de dados de grafos são especializados no gerenciamento eficiente de dados fortemente vinculados [22]. Portanto, aplicativos baseados em dados com muitos relacionamentos são mais adequados para bancos de dados de grafos, já que operações de alto custo, como junções recursivas, podem ser substituídas por percursos eficientes.

- Banco de Dados Chave-Valor:

  Os bancos de dados pertencentes a esse grupo são, essencialmente, tabelas *hash* distribuídas que fornecem pelo menos duas operações: *get (key)* e *put (key, value)* [5]. Um banco de dados de chave-valor mapeia itens de dados para um espaço de chave que é usado para alocar pares de chave/valor para computadores e localizar um valor com eficiência, considerando sua chave. Esses bancos de dados são projetados para escalar para terabytes ou até petabytes, além de milhões de operações simultâneas adicionando computadores na horizontal.

  Um exemplo simples de um armazenamento de chave-valor é um serviço de conteúdo da Web distribuído, em que cada chave representa a URL do elemento e o valor pode ser qualquer tipo de arquivo, desde PDFs e JPEGs até documentos JSON

ou XML. Dessa forma, os *designers* de aplicativos podem aproveitar os nós disponíveis em um *cluster* de máquinas para gerenciar um grande número de solicitações e grandes quantidades de conteúdo da Web [23].

- Banco de Dados Orientados por Coluna ou Colunar:

  Os bancos de dados colunares armazenam dados coluna por coluna, o que torna as operações baseadas em colunas específicas, como funções agregadas, muito rápidas [6]. Por ser o Sistema Gerenciador de Banco de Dados escolhido para solução do problema apresentado no trabalho, este tipo de SGBD está descrito na Subseção 2.2.1.

2.2.1 Armazenamento de Colunas

Os avanços na tecnologia da informação levaram a uma quantidade cada vez maior de dados disponíveis, criando demanda para analisá-los usando algoritmos de uma variedade de abordagens, por exemplos, estatísticas, *clusterização* e previsão. Tais análises são tipicamente executadas no sistema de banco de dados, pois fornecem uma execução otimizada e um escalonamento eficiente no volume de dados usado [24].

Os bancos de dados de Coluna Larga, Coluna de Famílias ou Colunar armazenam dados por colunas e não impõem um esquema rígido aos dados do usuário. Isso significa que algumas linhas podem ou não ter colunas de um determinado tipo. Uma das principais diferenças entre eles é com relação aos valores nulos. Considerando um caso de uso com diferentes tipos de atributos, o banco de dados relacional deverá armazenar cada valor nulo em sua respectiva coluna. Ao contrário disso, o banco de dados colunar salva somente a chave-valor do campo, se for realmente necessário [22].

Enquanto o SGBD Relacional mantém a forma conhecida de apresentação e armazenagem dos dados, disposto entre linhas e colunas, o NoSQL colunar mantém cada coluna separada. Como demonstrado na Figura 2.2 adaptada de [25], cada valor contém uma coordenada de valor composta de sua *Row Key*, nome da coluna, chave da coluna, o nome da família da coluna e *timestamp* (representa um ponto específico na linha do tempo e leva em consideração o fuso horário em questão).

A chave de linha serve para identificar os valores de coluna pertencentes à mesma tupla. O nome da coluna permite identificar o atributo de um valor, podendo ser composto pelo nome da família da coluna e pelo nome da coluna. De fato, a coluna pode ser composta ou simples. Se o nome da coluna for prefixado, isso significa que o nome da coluna consiste no nome da família de colunas (prefixo) e no nome da coluna aninhada. Nesse caso, ele é chamado de atributo composto (pertence a uma família de colunas), caso contrário, é considerado um atributo simples. Finalmente, o *timestamp* permite verificar a coerência

dos dados. Cada valor recebe um registro de data e hora pelo sistema para a proposição de consistência de dados.

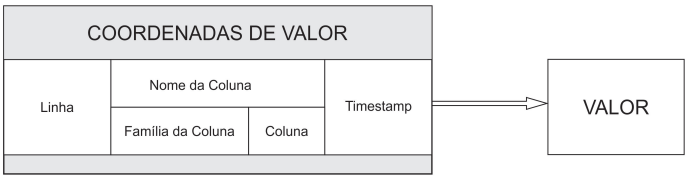


Figura 2.2: Estrutura de composição de um valor em NoSQL Colunar.

Bancos de dados orientados a colunas usam tabelas como modelo de dados, mas não suportam a associação de tabelas. Os bancos de dados orientados por colunas possuem as seguintes características [18]:

- os dados são armazenados por coluna, ou seja, os dados são armazenados separadamente para cada coluna;
- cada coluna de dados é o índice do banco de dados;
- acessam apenas as colunas envolvendo o resultado das consultas para reduzir o I/O do sistema;
- consultam processos simultaneamente, ou seja, cada coluna é tratada por um processo.

As Figuras 2.3 e 2.4 demonstram um exemplo de tabelas em ambiente relacional com seus dados disponibilizados em linhas, como por exemplo na Tabela Departamento podemos ver os itens dispostos em ordem "1, 'Departamento Pessoal', 8; 2, 'Almoxarifado', 8; 3, 'Vendas', 8;" e para a Tabela de Empregado "1, 'Felipe', 'Rezende', '1986-11-18', 2; 2, 'João', 'Bosco', '1965-01-16, 2; 3, 'Fernanda', 'Santos', '1965-02-03', 3; 4, 'Fabiano', 'Silveira', '1986-11-18', 3; 5, 'Carlos', 'Silva', '1963-09-30', 1; 6, 'Roberta', 'Andrade', '1986-11-18', 1;".

idDepartamento	Departamento	Carga_Horaria
1	Departamento Pessoal	8
2	Almoxarifado	8
3	Vendas	8

Figura 2.3: Exemplo de tabela de departamentos em ambiente relacional.

idEmpregado	Nome_Empregado	Sobrenome_empregado	Data_Nascimento	Departamento_idDepartamento
1	Felipe	Rezende	1986-11-18	2
2	João	Bosco	1965-01-16	2
3	Fernanda	Santos	1965-02-03	3
4	Fabiano	Silveira	1986-11-18	3
5	Carlos	Silva	1963-09-30	1
6	Roberta	Andrade	1986-11-18	1

Figura 2.4: Exemplo de tabela de empregados em ambiente relacional.

Transformando esses dados em um banco de dados colunar, os dados ficariam dispostos por coluna, não mais em linhas, sendo assim demonstrado na ordem de "1, 2, 3;

'Departamento Pessoal, 'Almoxarifado', 'Vendas'; 8, 8, 8;" para a tabela de departamentos e para a tabela de funcionário "1, 2, 3, 4, 5, 6; 'Felipe', 'João', 'Fernanda', 'Fabiano', 'Carlos', 'Roberta'; 'Rezende', 'Bosco', 'Santos', 'Silveira', 'Silva', 'Andrade'; '1986-11-18', '1965-01-16', '1965-02-03', '1986-11-18', '1963-09-30', '1986-11-18'; 2, 2, 3, 3, 1, 1;".

Row Key	Família de Coluna	Coluna	Valor	Timestamp
1	empregado	nome_empregado	Felipe	123456789
1	empregado	sobrenome_empregado	Rezende	123456790
1	empregado	data_nascimento	1986-11-18	123456791
1	empregado	Departamento_idDepartamento	2	123456792
2	empregado	nome_empregado	João	123456793
2	empregado	sobrenome_empregado	Bosco	123456794
2	empregado	data_nascimento	1965-01-16	123456795
2	empregado	Departamento_idDepartamento	2	123456796
3	empregado	nome_empregado	Fernanda	123456797
3	empregado	sobrenome_empregado	Santos	123456798
3	empregado	data_nascimento	1965-02-03	123456799
3	empregado	Departamento_idDepartamento	3	123456800

Figura 2.5: Exemplo de tabela em NoSQL Colunar.

A Figura 2.5 demonstra um exemplo de carga de dados em ambiente NoSQL colunar, em que a chave primária do banco de dados relacional se transforma em *Row Key*. Definir uma "Família de Coluna" não é obrigatório, porém neste exemplo foi definido que a família de coluna seria composta com o nome da tabela de onde se originam os dados. O campo Coluna contém o nome da coluna e o campo Valor recebe o valor referente àquela coluna. Para finalizar, existe o campo *Timestamp*, que registra o momento exato em que estes dados foram carregados na tabela. Neste exemplo o valor de *Timestamp* é algo fictício.

Este exemplo demonstra de forma prática a diferença entre as disposições de dados entre as duas estruturas de armazenamento de dados. Enquanto a estrutura de banco de dados relacional organiza seus dados em linhas, o modelo de banco de dados colunar armazena seus dados dispostos em colunas.

2.3 *Data Warehouse*

Um *data warehouse* é uma base de dados para um processo analítico online (OLAP) para tomada de decisões [25]. Ele é projetado de acordo com uma modelagem dimensional que tem por objetivo observar fatos por meio de medidas, também chamadas de indicadores, de acordo com as dimensões que representam os eixos de análise. Segundo Khaled *et al.* [26], os bancos de dados relacionais têm demonstrado limitações para armazenar e gerenciar *Big Data* em um *data warehouse*.

Estudos como [13] [27] [28] mostram que a migração de SGBD tradicionais para NoSQL colunar pode ser positiva em se tratando de desempenho de evidenciação de resultados. No geral, a vantagem do modelo de dados NoSQL colunar é a aplicação mais adequada

na agregação e no *data warehouse* [5], pois o modelo orientado a colunas mantém alta performance em análise de dados e processos de *Business Intelligence*.

2.4 Ecossistema Hadoop

O Apache Hadoop é uma estrutura de computação escalável que abstrai os problemas de distribuição de dados, agendamento e tolerância a falhas dos aplicativos [29]. O Hadoop é o núcleo de um ecossistema sobre o qual vários provedores estão construindo plataformas como um serviço. A seguir, são descritos os componentes do Hadoop utilizados neste trabalho.

2.4.1 *Hadoop Distibuted File System* - HDFS

Em 2006, a Google desenvolveu e implementou um sistema de armazenamento distribuído chamado Bigtable [30]. Bigtable usa *Google File System* (GFS) para armazenar *log* e arquivos de dados. GFS é um sistema de arquivos distribuídos que mantém múltiplas réplicas de cada arquivo para melhor disponibilidade e confiabilidade.

Um sistema similar ao GFS [31] [32] é o *Hadoop Distributed File System* (HDFS), um sistema de armazenamento de dados que suporta centenas de nós em um *cluster* e fornece uma capacidade de armazenamento econômica e confiável. Para a camada de consistência, HDFS utiliza o *Zookeeper*. Ele utiliza o protocolo *Zab* para propagar as atualizações do estado incremental do nó primário para os nós de *backup*, o que torna este sistema de armazenamento tolerante a falhas com um mecanismo de réplicas.

2.4.2 Zookeeper

Segundo descrito em [33], *Zookeeper* fornece a seus clientes a abstração de um conjunto de nós de dados (*znodes*), organizados de acordo com um espaço de nomes hierárquico. Os *znodes* nessa hierarquia são objetos de dados que os clientes manipulam como a API do Zookeeper. Espaços de nomes hierárquicos são comumente usados em sistemas de arquivos. É uma maneira desejável de organizar objetos de dados, já que os usuários estão acostumados a essa abstração e permitem uma melhor organização dos metadados de aplicativos.

Protocolo *Zab* [34] possui dois modos de operação: recuperação e transmissão. Quando o serviço é iniciado ou após uma falha de líder, o *Zab* passa para o modo de recuperação. O modo de recuperação termina quando um líder aparece e um quorum de servidores sincroniza seu estado com o líder. Sincronizar seu estado consiste em garantir que o líder e o novo servidor tenham o mesmo estado.

2.4.3 Map/Reduce

MapReduce é um modelo de programação para processamento e geração de grandes conjuntos de dados. Os usuários especificam uma função de *Map* que processa um par de chave/valor para gerar um conjunto de pares de chave/valor intermediários e uma função *Reduce* que opera com todos os valores intermediários associados à mesma chave intermediária [35]. O *MapReduce* fornece tolerância a falhas de granularidade fina para trabalhos grandes, uma falha no meio de uma execução de várias horas não requer a reinicialização do trabalho do zero. Similar ao HDFS, a estrutura do *Hadoop MapReduce* [36] também adota uma arquitetura *master-slave*: um *JobTracker (master)* e um número de *TaskTrackers (slaves)*. O *JobTracker* coordena todos os trabalhos em execução no sistema por uma tarefa de agendamento para rodar os *TaskTrackers*.

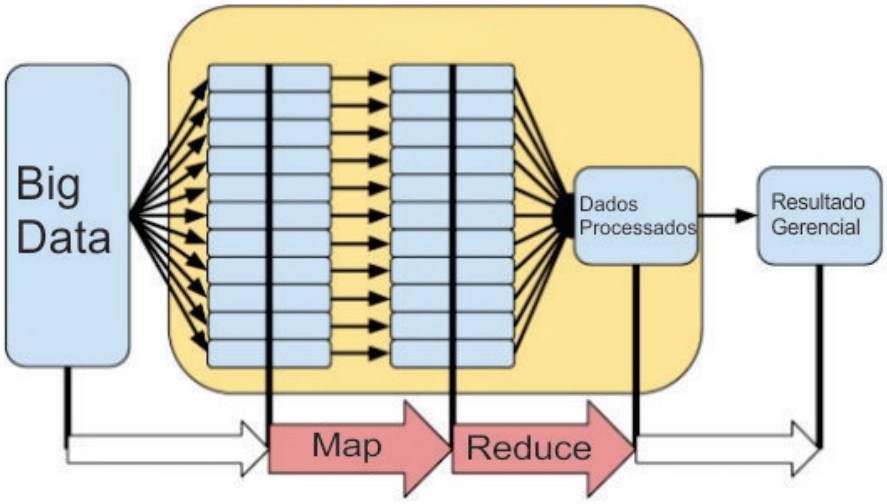


Figura 2.6: Exemplo de estrutura MapReduce.

A Figura 2.6 apresenta um exemplo da estrutura do MapReduce, sendo realizado o mapeamento dos valores a serem processados. Antes do *Reduce* é feita uma ordenação do valores e depois são demonstrados os valores produzidos no MapReduce.

2.4.4 HBase

HBase, um projeto de código aberto da Apache, é um sistema distribuído tolerante à falha e com alta escalabilidade, orientado por colunas, constituindo um banco de dados NoSQL construído sob o HDFS. É utilizado para leituras e escritas de acesso aleatório em base de dados muito grandes [36]. Todos os acessos aos dados são feitos por meio da chave primária e de qualquer verificação dos resultados da tabela HBase em um trabalho *MapReduce*. Verificação paralela, em termos de resultados de trabalho *MapReduce* em tempo de resposta de consulta é mais rápido e tem melhor rendimento geral.

Similar ao HDFS e ao *MapReduce* [36], HBase também adota arquitetura *master-slave*. O *HMaster (master)* é responsável por atribuir regiões a *HRegionServers (slaves)* e pela recuperação de falhas do *HRegionServer*. O *HRegionServer* é responsável por gerenciar as solicitações do cliente para leitura e gravação. O HBase usa o *Zookeeper*, outra estrutura do Hadoop já apresentada neste trabalho, para gerenciar o *cluster* do HBase. Não há suporte para a linguagem de consulta SQL no HBase. No entanto, há um projeto de integração Hive/HBase que permite que instruções SQL escritas no Hive acessem tabelas do HBase.

A Tabela 2.1 [14] apresenta um comparativo entre as características do HDFS e do HBase. O HDFS é um sistema de arquivos distribuído apropriado para armazenar grandes arquivos, e o HBase é um banco de dados não relacional distribuído desenvolvido no topo do HDFS. Em se tratando de armazenamento, o HDFS armazena arquivos grandes (de Gigabytes a Terabytes de tamanho) nos servidores do *Hadoop*, enquanto o HBase coloca internamente em "*StoredFiles*" indexados que existem no HDFS para pesquisas de alta velocidade. O acesso aos dados no HDFS é feito principalmente por meio de MapReduce e o HBase fornece acesso a linhas únicas de bilhões de registros.

Tabela 2.1: Comparativo entre as características do HDFS e HBase.

Propriedades	**HDFS**	**HBase**
Sistema	HDFS é um Sistema de Arquivos Distribuído apropriado para armazenar grandes arquivos.	HBase é um banco de dados não relacional distribuído desenvolvido no topo do HDFS.
Consulta e Performance de pesquisa	O HDFS não é um sistema de arquivos de propósito geral. Não fornece pesquisa rápida de registros em arquivos.	Ele permite pesquisas de registro rápidas (e atualizações) para tabelas grandes.
Armazenamento	O HDFS armazena arquivos grandes (gigabytes a terabytes de tamanho) nos servidores do Hadoop.	O HBase coloca internamente em *"StoredFiles"* indexados que existem no HDFS para pesquisas de alta velocidade.
Processamento	O HDFS é adequado para processamento em lote de operações de alta latência.	O HBase é construído para operações de baixa latência.
Acesso	Os dados são acessados principalmente por meio do MapReduce.	O HBase fornece acesso a linhas únicas de bilhões de registros.
Operações de I/O	O HDFS é projetado para processamento em lote e, portanto, não suporta operações de leitura/gravação aleatórias.	O HBase permite operações de leituras/gravações. Os dados são acessados por meio de comandos shell, APIs do cliente em Java, REST, Avro ou Thrift.

2.4.5 Sqoop

Um método de ETL para carga de dados em ambiente *Big Data* é o *Sqoop*, também presente no *Ecossistema Hadoop* [37]. Sqoop foi projetado com o propósito de transferência de grande volume de dados entre *Apache Hadoop* e estruturas de banco de dados relacionais [38]. O Sqoop realiza cópias rapidamente de sistemas externos, inclusive de *data warehouse* corporativo para o Hadoop. Garante um desempenho rápido ao paralelizar a transferência de dados. Sqoop suporta análise de dados de maneira eficiente [38].

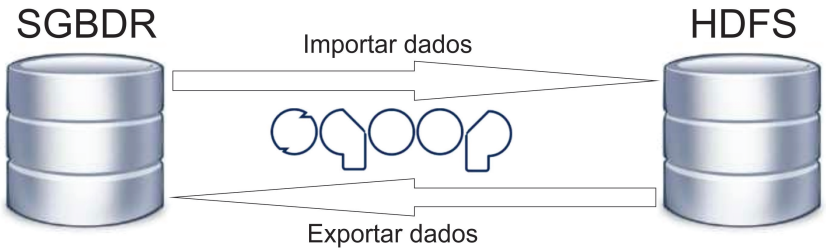


Figura 2.7: Exemplo da funcionalidade de importação e exportação do Sqoop.

A Figura 2.7 demonstra a possibilidade de realização, tanto para a importação como a exportação de dados de uma estrutura SGBDR para o HDFS/HBase. Para o objetivo proposto no trabalho, de importação dos dados da estrutura SGBDR para o *HBase*, existem duas formas distintas de execução. Pode-se realizar a carga por completo da tabela de um ambiente para outro, até mesmo selecionando algumas colunas específicas para a carga, como pode-se realizar a carga por meio de uma consulta em SQL.

2.4.6 Pig

Apache Pig é uma plataforma para analisar grandes conjuntos de dados que consiste em uma linguagem de alto nível para expressar programas de análise de dados, juntamente com infraestrutura para avaliar esses programas. A principal propriedade dos programas Pig é que sua estrutura é passível de paralelização substancial, o que, por sua vez, lhes permite lidar com conjuntos de dados muito grandes [39].

Apache Pig é uma álgebra de consulta simples que permite ao usuário declarar transformação de dados em arquivos ou grupos de arquivos [40]. O Pig Latin suporta um conjunto limitado de operadores de agregação, por exemplo contagem, média, mínimo e máximo. Estes não incluem medidas estatísticas comuns como variância e correlação.

2.4.7 Hive

Apache Hive é um projeto de *data warehouse* de código aberto da Fundação Apache. O *Hive* integrou armazenamento, consulta e gerenciamento de dados para grandes conjuntos de dados. *Hive* é uma aplicação de *data warehouse* construída em cima do *Hadoop MapReduce*. Ele permite que os usuários lidem com os dados armazenados no *Hive*, da mesma forma que os dados são armazenados em um banco de dados regular. O *HiveQL* é uma linguagem semelhante ao SQL, permite armazenamento do usuário e solicita dados no *Hive*. O *Hive* permite que o usuário com experiência no tradicional execute consultas familiares no *MapReduce*.

Hive é uma estrutura para soluções em *data warehouse* com código aberto construído pela Fundação Apache [41] para integrar armazenamento de dados e consultas com a estrutura do *Hadoop MapReduce*, com funcionalidades similares ao SGBDR. Suporta expressões similares ao SQL para consultas, chamado de *Hive Query Language (HiveQL)* [42]. Além do que, *HiveQL* suporta códigos de *MapReduce* personalizados para serem ligados a consultas.

Os dados em *Hive* são organizados em [42] [43]:

- **Tabelas:** São tabelas análogas a tabelas de banco de dados relacionais. Cada tabela tem uma pasta correspondente no HDFS. Os dados da tabela são serializados e armazenados em arquivos dentro do diretório.
- **Partições:** Cada tabela pode ter uma ou mais partições que determinem a distribuição dos dados dentro de sub diretórios na pasta da tabela.
- ***Buckets:*** Os dados em cada partição podem ser divididos por *buckets* baseado no *hash* da coluna na tabela. Cada *bucket* é armazenado em um arquivo dentro do diretório da partição.

2.5 Trabalhos Relacionados

Esta seção apresenta os trabalhos relacionados a esta dissertação. Os trabalhos são apresentados da seguinte maneira: inicialmente, trabalhos de comparativos entre desempenho de SGBDR e NoSQL são apresentados, em seguida trabalhos com NoSQL colunar e por último são descritos trabalhos que utilizam armazenamento de *Data Warehouse* em ambiente *Big Data* utilizando a estrutura *Apache/Hive*.

Yasin N. Silva *et al.* descrevem em [44] estruturas para utilização de SGBD NoSQL em comparação com SQL, utilizando diversas estruturas de NoSQL. Descrevem tratando da importância desse conceito de banco de dados relacional em termos históricos de processamento de dados, dentre outros assuntos, mas descrevem o NoSQL como uma tendência

para tratamento de grande volume de dados. Chegam a uma conclusão de que muitos cenários de aplicativos exigem o processamento de conjuntos de dados muito grandes em uma forma altamente escalonável e distribuída, e que diferentes tipos de sistemas de *Big Data* foram projetados para lidar com esse desafio.

Sourav Mukherjee descreve em [45] que bancos de dados NoSQL foram desenvolvidos nos últimos anos como uma resposta às limitações dos SGBDR e para fornecer desempenho, escalabilidade e flexibilidade essenciais para aplicativos modernos. Apresenta exemplos dos quatro tipos de NoSQL, explicando suas características e quais os SGBD mais utilizados para cada categoria. No final, conclui que a tecnologia está se movendo muito rápido e que análises em tempo real podem ajudar as organizações a se manter atualizadas. O NoSQL permite implantação confiável em banco de dados distribuído que pode atender aos requisitos da organização.

O estudo apresentado por Rashid Zafar *et al.* em [46] também descreve comparativos entre as SGBDR e NoSQL, descrevendo os modelos existentes em NoSQL, com exemplos de SGBD nesta característica. Como conclusão, insere um exemplo de ganho apresentado pela *Netflix* com a migração do sistema *Oracle* para banco de dados colunar. Depois da conversão, a companhia teve uma capacidade de escrita de mais de dez mil linhas por segundo em sua base de dados. Sistemas NoSQL são basicamente utilizados para aplicações com necessidade de alta performance e confiabilidade dos dados, e são executados em vários nós conectados a um *cluster*.

Um comparativo entre o SGBD tradicional, utilizando o MySQL, e o NoSQL com base em documentos, utilizando o MongoDB, é encontrado em [47], que apresenta uma comparação envolvendo quatro operações fundamentais de um banco de dados: Instanciação, Leitura, Escrita e Exclusão. No geral, apesar do ajuste de desempenho no MySQL, o MongoDB ainda apresenta desempenho mais rápido do que o MySQL, desde um pequeno número de linhas até um milhão de linhas. No futuro, o MySQL poderá acompanhar as últimas tendências de Big Data adicionando mais recursos, como nos últimos vinte anos de desenvolvimento [47]. Nos próximos anos, os bancos de dados SQL e NoSQL terão que evoluir para armazenar e organizar dados com mais eficiência.

Em [48], existe a execução da análise de recursos de oitenta soluções NoSQL, tendo em vista os critérios de seleção de tecnologia para sistemas de Big Data, juntamente com uma avaliação cumulativa de casos de uso apropriados e inadequados para cada modelo de dados. Todo momento é mencionado SGBDR em comparação com determinadas características existentes no NoSQL e o porquê seria melhor esta solução. Neste estudo conclui-se que o NoSQL provou ser uma solução viável para o problema de Big Data, principalmente por ter quatro tipos de estruturas diferentes e para cada uma dessas estruturas, diversas opções em código aberto. Depois que o modelo de dados correto é determinado para um

problema de Big Data, é necessário encontrar uma solução que suporte o modelo de dados junto com os recursos desejados conforme os requisitos do sistema de Big Data. O Hadoop fornece maneiras melhores de armazenar e formatar dados em arquivos. Na maioria dos cenários de Big Data estruturado, sabe-se [48] que os formatos de armazenamento baseados em colunas têm um desempenho melhor que os equivalentes baseados em linhas, pois a maioria das consultas requer a recuperação de algumas colunas para várias linhas.

Ameya Nayak *et al.* relatam em [49] que NoSQL é uma alternativa emergente aos bancos de dados relacionais mais amplamente utilizados. Como o nome sugere, ele não substitui completamente o SQL, mas complementa de tal forma que eles podem coexistir. Este trabalho descreve os prós e contras de utilizar NoSQL, assim como as vantagens e desvantagens de cada tipo de banco de dados NoSQL e casos particulares em que cada um poderia ser utilizado. Os usuários devem primeiro considerar vários parâmetros, como linguagem de consulta, interface, disponibilidade, redundância, consistência, custo, suporte e analisar os prós e contras de vários modelos de dados antes de escolher um modelo de dados específico [49].

Como encontrado em diversos estudos, NoSQL não é somente algo que "substitui" o SGBDR, eles devem coexistir. Cada um segue sendo muito útil para requisitos específicos. Para utilização em *Big Data*, atualmente a tendência é a utilização de estruturas NoSQL, com o tipo que atenda a sua questão.

O tamanho dos conjuntos de dados que estão sendo coletados e analisados no setor para *Business Intelligence* está crescendo rapidamente, tornando as soluções de armazenamento tradicionais proibitivamente caras [42]. O Hadoop é uma implementação popular de código aberto, utilizando a estrutura *Map-Reduce* que está sendo usada como uma alternativa para armazenar e processar conjuntos de dados extremamente grandes em hardware comum.

Em [43], Ashish Thusoo *et al.* descrevem como uma alternativa para *Data Warehouse* de grande volume de dados a estrutura *Hive*, presente no ambiente *Hadoop*. Hive é uma estrutura de dados bem conhecida com conceito de banco de dados contendo tabelas, colunas, linhas e partições. Semelhante aos bancos de dados tradicionais, o *Hive* armazena dados em tabelas, em que cada tabela consiste em um número de linhas e cada linha consiste em um número especificado de colunas. Cada coluna tem um tipo associado.

Em [50], Chih-Hung Chang *et al.* descrevem em seu estudo as vantagens da estrutura Hive. Utilizado para Visualização de dados portáteis e múltiplas. Na estrutura Hive, existem padrões bem definidos e usando bancos de dados relacionais. Mantém alta performance, integrado com o Java.

A proposta de contribuição acadêmica deste trabalho está no conceito de realização de análise de desempenho do processamento de resultado gerencial das operações de crédito

de uma instituição financeira, com seus valores sendo gerados diariamente e do processo de captura das informações em um ambiente relacional, realizando a carga em Ambiente de Big Data, utilizando as estruturas existentes no *Apache Hadoop*. Armazenamento dos dados brutos em ambiente *HBase*, preparação dos dados e processamento dos resultados gerenciais utilizando a estrutura *Pig Script* com programas pré-estabelecidos em *Python*, e o armazenamento em ambiente de Data Warehouse no ecossistema *Hadoop/Hive*.

Capítulo 3

Solução Proposta

Neste capítulo é descrita a proposta para consecução dos objetivos estabelecidos no trabalho de dissertação para o mestrado. O capítulo está dividido em quatro seções. A Seção 3.1 descreve como é executado o sistema atual de resultado gerencial da instituição financeira Beta. Na Seção 3.2 é descrita uma proposta de solução para o problema apresentado nesta dissertação, com referenciais teóricos sobre o assunto, e a Seção 3.3 apresenta como foi realizada a implementação da proposta descrita na Seção anterior. O ambiente computacional para a proposta é descrito na Seção 3.4.

3.1 Atual Forma de Processamento

O processo atual de geração de resultado gerencial se inicia em duas atividades totalmente desvinculadas, conforme a Figura 3.1. A primeira atividade realiza em cada um dos processos é a busca das informações necessárias em uma base de dados específica. Para o cálculo de resultado gerencial de operações, é realizada uma busca de informações em um sistema interno que mantém as operações dos clientes da instituição. Alguns componentes de resultado são produzidos em fontes externas, portanto em uma outra atividade desse processo busca-se essas informações na base de dados que detém esses dados armazenados. Sendo assim, realizado o cálculo dos componentes de resultado gerencial de outras fontes.

Após os cálculos iniciais, tem-se a atividade de realização do agrupamento de informações em uma tabela temporária em ambiente relacional, tendo em vista que as duas atividades (cálculo com informações no sistema interno e em fontes externas) realizam apuração de resultados das mesmas operações e que, para realizar o cálculo final de resultado gerencial deve-se realizar um agrupamento dessas informações. Cada operação de crédito contratada pelos clientes da instituição tem uma chave de classificação (informações que a definem como única na instituição). Este agrupamento é realizado colocando todos os componentes de resultado apurados nas duas atividades distintas para a mesma

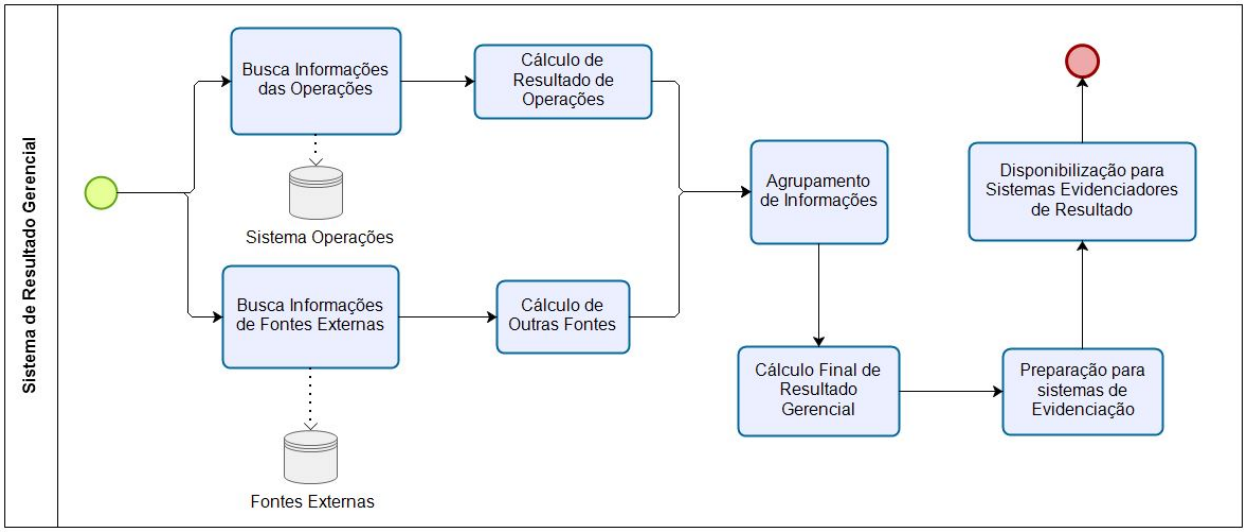


Figura 3.1: Processamento atual do sistema de resultados gerenciais existente na Instituição Financeira Beta.

operação. Então é realizada uma adição de informações dos atributos com campos específicos do sistema evidenciador de resultado gerencial, dessa forma, preparando os valores encontrados para que sejam demonstrados para as áreas interessadas da instituição financeira.

A última atividade a ser realizada é a disponibilização para os sistemas evidenciadores de resultado. Essa geração é realizada somente uma vez ao mês, gerando dois arquivos com granularidades e formatos distintos, para atenderem aos sistemas evidenciadores.

Devido ao grande volume de dados a serem trabalhados e ao tempo de processamento dessas informações, foi definido que a execução desse processo seja realizada aos finais de semana. Os valores são processados novamente, toda semana, repetindo-se o processo do início do mês até o final de semana, até o fechamento do mês de referência, obtendo-se o resultado final a partir do primeiro fim de semana do mês subsequente.

3.2 Proposta de Solução

Um dos grandes problemas encontrados na forma atual de processamento do resultado gerencial da Instituição Financeira, além do tempo de processamento, é a necessidade de esperar o encerramento do mês de referência para se ter o valor do resultado realizado das operações da instituição financeira. A principal vantagem a ser trabalhada com a alteração para um sistema que suporte processamento de grande volume de dados é a possibilidade de executar o processamento dessas informações diariamente, realizando o armazenamento dos valores prévios de resultado com defasagem de um dia - dispensando a espera do encerramento do mês.

Como encontrada na literatura, por exemplo em [51] e [7], uma das melhores opções para o processamento do cálculo de resultados gerenciais em grande volume de dados é a armazenagem em Banco de Dados NoSQL Colunar.

A proposta de solução ao problema levantado no Capítulo 1 é a implementação do armazenamento dos dados com os valores necessários para o cálculo do resultado gerencial em NoSQL colunar. Depois, iniciar a realização do processamento desses dados e o cálculo do resultado gerencial em ambiente *Big Data*. Por último realizar o armazenamento destes resultados obtidos em uma estrutura de *Data Warehouse* existente no ecossistema *Hadoop* - o *Hive*.

A Figura 3.2 demonstra o processamento proposto para o novo formato de realização do cálculo gerencial das operações da instituição. A busca das informações na base de dados das Operações e das Fontes Externas continuaria igual ao processo apresentado anteriormente, com a diferença que não existiria o cálculo prévio desses valores de resultado gerencial. O último passo descrito na figura demonstra o armazenamento em uma estrutura *Data Warehouse* existente no ambiente Hadoop. Sendo assim, podem-se criar diversas bases de dados com dimensões e formatos distintos para atenderem aos sistemas evidenciadores de resultado.

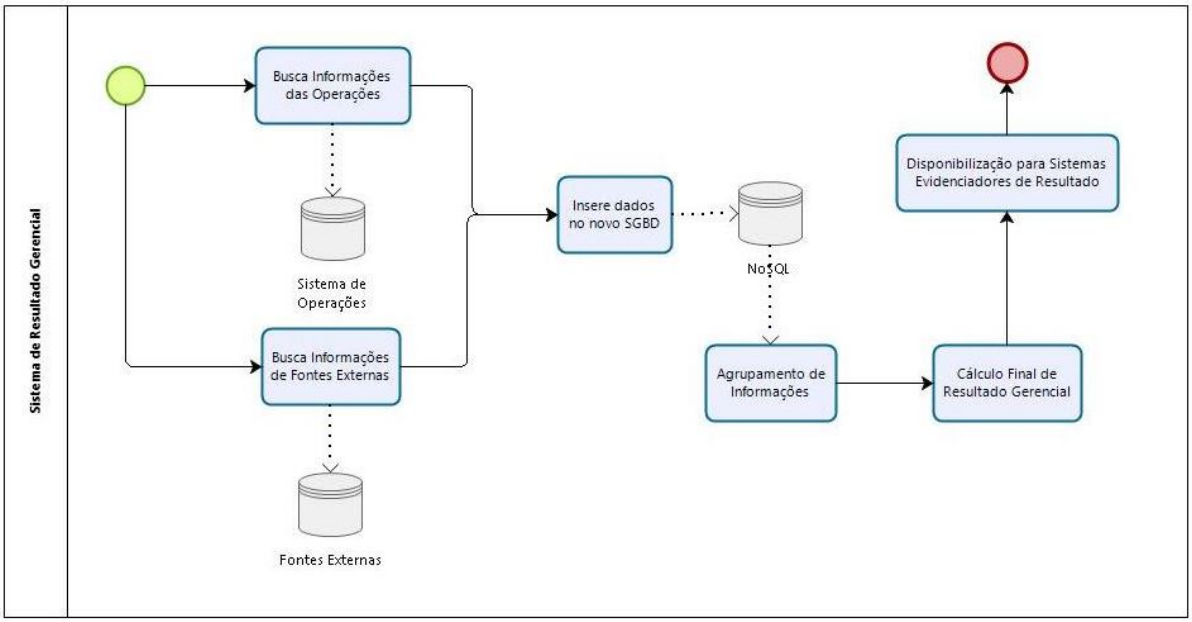


Figura 3.2: Proposta de macro processo de carga de dados em NoSQL.

Por dia, cada sistema gera aproximadamente 800 MB de dados. Atualmente, a instituição Beta tem 26 sistemas, que geram aproximadamente 20,8 GB de dados a serem processados e armazenados diariamente para este projeto de geração de valores de resultado gerencial da instituição financeira. Para o trabalho, foram utilizadas as operações

desses sistemas da instituição Beta (por política da instituição financeira, os nomes dos sistemas não podem ser descritos em trabalhos acadêmicos externos à instituição). Ao se levantar a granularidade necessária para atender aos evidenciadores dos resultados gerenciais, optou-se por manter as mesmas colunas existentes no processamento de cálculo de resultado gerencial existente hoje, na instituição.

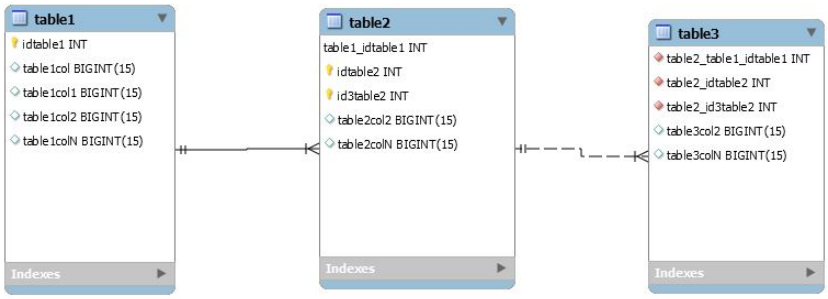


Figura 3.3: Tabelas em ambiente relacional.

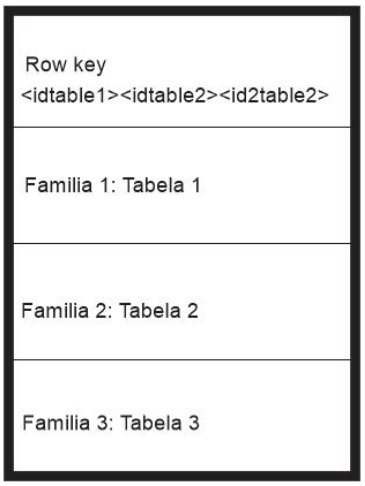


Figura 3.4: Tabelas criada no HBase.

No sistema de armazenamento atual da empresa, os dados estão distribuídos em três tabelas relacionais, cada uma com um grupo de informações distintas. Atendendo à granularidade necessária para o processo de geração de valores de resultado gerencial das operações financeiras da instituição, somente foram capturados os atributos específicos de cada tabela que já existem no processo atual de geração desses resultados, sendo treze os atributos da primeira tabela, vinte e sete os atributos da segunda tabela e vinte e oito os atributos da última tabela.

Para armazenamento das informações em HBase, foi criada somente uma tabela, porém, como descrito na Seção 2.4.4, criaram-se três famílias de colunas, cada uma recebendo

valores de uma das três tabelas em ambiente relacional. A Figura 3.3 exemplifica as três tabelas em ambiente relacional, e a Figura 3.4, como foi criada a tabela em *HBase*. Como pode ser observado, a *Row Key* da nova tabela no *HBase* é uma composição das três chaves do modelo relacional.

A Tabela 1, presente na Figura 3.3, é como se fosse uma "tabela mãe", ela contém o número do contrato de crédito e todas as informações descritivas dessa operação. A Tabela 2 é uma tabela que contém a descrição específica daquela operação; na instituição alguns tipos de contratos permitem a contratação de mais de uma operação. A Tabela 3 mantém as mesmas características da Tabela 2, exceto que seu conteúdo contém os saldos referentes à operação. Dessas três tabelas, a Tabela 1 só contém uma chave primária, esta chave também está presente nas Tabelas 2 e 3. Na Tabela 2 e Tabela 3 existem 2 campos como chave primária a mais, que definem qual é a operação para aquele contrato de crédito.

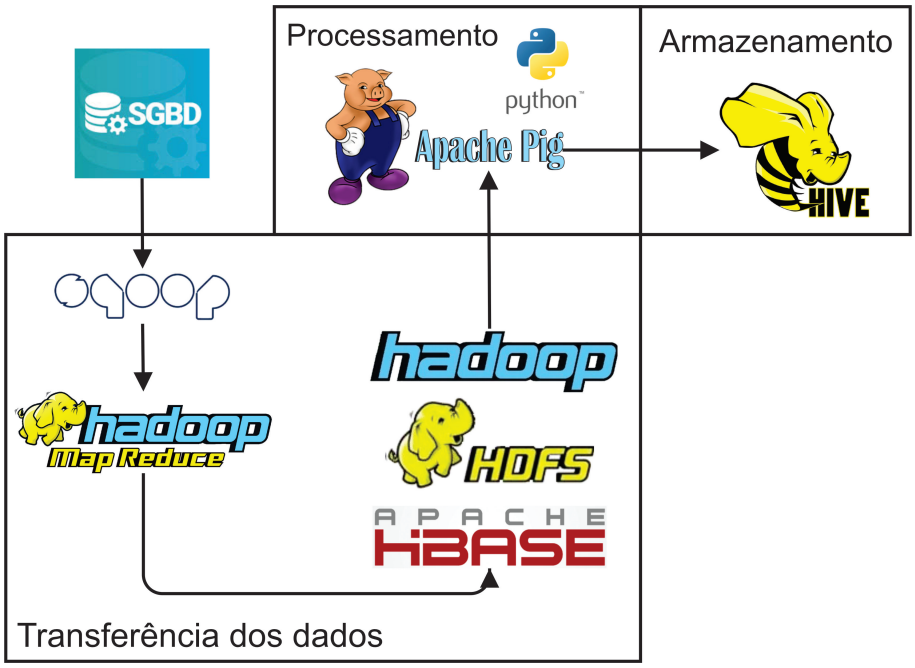


Figura 3.5: Solução proposta para o problema.

A Figura 3.5 apresenta a arquitetura abstrata para a proposta. A proposta é dividida em três principais módulos:

- Transferência de dados: para se executar o processamento das informações em ambiente *Big Data* faz-se necessário o processo de ETL nos dados para o ambiente *Hadoop*. Este processo é realizado com o *Sqoop*. Foi criado um arquivo *shell script* para a realização de cada processamento. Todos os *scripts* desta dissertação podem ser acessados no github em [52]. Diariamente foi executado um processo de *Sqoop*

incremental, em que se realiza a atualização de todas as operações que sofreram alteração em seu estado de condução (se a operação foi liquidada, deixou de estar em estado normal ou foi contratada neste dia). Uma atualização dos saldos para cálculo do resultado gerencial é realizada diariamente, sobrescrevendo os valores já existentes.

- Processamento: com os dados armazenados em ambiente *Hadoop*, pode-se processar os cálculos de resultado gerencial. Para este procedimento, existe a estrutura *Pig Latin*, já descrita neste trabalho. Também foi criado um *script* que deverá ser executado diariamente após a realização dos processos de *Sqoop* com incremento dos valores atualizados. Algumas metodologias de cálculo não podem ser executadas em *Pig Latin*, e para isso foram criados dois programas em *Python* para estas execuções.
- Armazenamento: depois de todo o processamento das informações descritas no item anterior, os dados de resultado gerencial das operações de crédito da instituição financeira devem ser armazenados em ambiente *Hive*, para processos em *Data Warehouse*. Para otimização de desempenho, as tabelas são particionadas por ano e mês, tendo em vista ser o agrupamento necessário para evidenciar os resultados gerenciais da instituição.

3.3 Implementação da Solução Proposta

Nesta seção são apresentados detalhes da implementação da proposta, desde a criação do banco de dados Colunar até o cálculo dos resultados gerenciais das operações.

As informações disponíveis em ambiente relacional, necessárias para o cálculo gerencial estão presentes em três tabelas distintas, sendo duas contendo informações descritivas sobre as operações. Como regra de definição de número de contrato da operação, existe um contrato "Pai", em que se mantém as informações descritivas sobre este contrato na *Tabela 1*. Cada contrato "Pai" pode ter diversos contratos "Filhos", e suas descrições estão presentes na *Tabela 2*. A *Tabela 3* contém os saldos atualizados de todos os contratos "Filhos". A Figura 3.6 apresenta uma demonstração do relacionamento entre as três tabelas descritas acima.

Inicialmente, realizou-se a criação de uma tabela em ambiente HBase, contendo três famílias de colunas, conforme apresentada na Figura 3.4. Cada família de coluna, deverá receber os dados necessários de uma das tabelas de ambiente relacional. Para a realização das cargas de dados, foi implementado o *shell script*, apresentado na Figura 3.7. A Figura 3.7 apresenta o *scrip* desenvolvido para a execução do processo *Sqoop*, realizando a conexão no ambiente relacional e a extração dos dados por meio de uma consulta SQL.

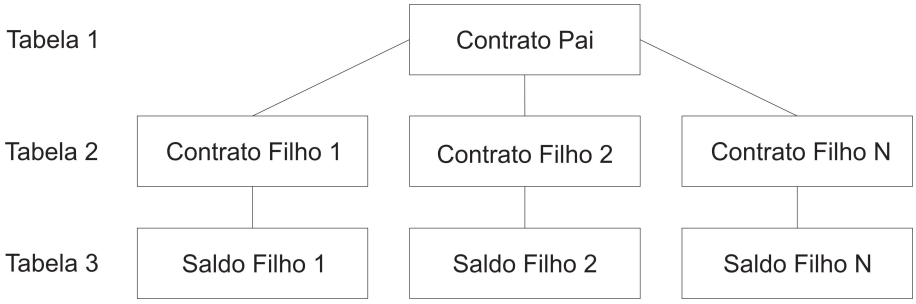


Figura 3.6: Demonstração do relacionamento entre as tabelas responsáveis pelos valores para cálculo do resultado gerencial.

A carga dos dados é feita na estrutura HBase em uma família de coluna específica. O processo se inicia com o comando *import*, haja vista ser a ação desejada para o momento. Em seguida se insere o caminho de onde serão consultados os dados em ambiente relacional, utilizando o parâmetro *connect*. Para realizar a conexão no servidor do ambiente relacional, inserem-se os parâmetros *username* e *password*, informando o usuário e senha para acesso ao servidor. Como demonstrado na Figura 3.4, para cada Tabela em ambiente relacional, foi criada uma família de colunas específica. Portanto, no parâmetro *column-family* deve-se preencher a família de coluna específica da fonte de dados em ambiente relacional.

```
sqoop import
--connect jdbc:db2://{ENDEREÇO DO SERVIDOR DB2}/{NOME DO BD}
--username {USUÁRIO} -password {SENHA}
--query "SELECT COLUNA_1, COLUNA_2, COLUNA_3, COLUNA_N
FROM SCHEMA.TABELA_1
WHERE \$CONDITIONS"
--hbase-table '{SCHEMA HBASE}:{TABELA HBASE}'
--column-family '{FAMÍLIA DE COLUNAS}'
--hbase-row-key COLUNA_1
--split-by COLUNA_1 -m 128
```

Figura 3.7: Processo para carga de dados utilizando a estrutura Sqoop.

Como descrito na Seção 2.4.5, existem duas maneiras de acesso à base de dados. Podemos utilizar o parâmetro *table*, que faz a inserção de toda a tabela em ambiente relacional para o ambiente *Hadoop*, ou podemos utilizar o parâmetro *query*, que realiza uma consulta específica na base de dados. Sendo possível, inclusive, a utilização de *Join*, filtros e outros tratamentos necessários. Para este trabalho, foi utilizado o parâmetro *query* para a execução do tratamento de concatenação das chaves primárias das três tabelas, descrito logo abaixo. A cláusula *Where*, presente no parâmetro *query*, foi utilizada para que sejam inseridas somente as operações que sofreram alteração em seu estado de condução, ou seja, as operações que foram encerradas, liquidadas, ou contratadas até o dia útil anterior ao processo.

Para inserção do *Row key*, que é um campo único, e neste caso temos três chaves primárias distintas, realizamos o *Sqoop* com a opção de *Query* na hora de busca na base em ambiente relacional. Desta forma realizamos um *Join* entre a Tabela 1 e a Tabela 2 como chave de batimento a chave primária presente nas duas tabelas, e trazendo as duas chaves primárias da Tabela 2. Com isso faz-se uma concatenação destes três campos, gerando um valor único para cada operação de crédito da instituição financeira. A necessidade desse *Join* só existe para o processo de carga de dados da Tabela 1, para que seja alterada para a mesma granularidade das Tabelas 2 e 3. Nas outras duas Tabelas, como já possuem as três chaves primárias, só é realizada a concatenação destes três campos formando o item que será inserido como *Row key* no processo de carga ao *HBase*. Porém para cada uma dessas tabelas, foi criado um processo em *Sqoop* para a realização das cargas em sua família de dados correspondente. A partir daí se inserem os valores para carga em ambiente *Big Data*. O parâmetro *hbase-table* contém o nome da tabela criada em ambiente *HBase* anteriormente ao processo de carga dos dados. Para determinar qual família de coluna receberá os valores desse processo de *Sqoop*, utiliza-se o parâmetro *column-family*, informando o nome da família de colunas que receberá os valores. Outro parâmetro necessário para a execução deste processo é o *hbase-row-key*, em que se deve fornecer qual o campo será utilizado como a chave das linhas no *HBase*, necessidade descrita na Seção 2.2.1, quando descrevemos as características do modelo de banco de dados *NoSQL* colunar.

É possível inserir campos específicos para que seja realizada a divisão do processamento de *MapReduce*, assim como em quantos nós ele será dividido. Neste processo utilizou-se como campo de divisão a chave primária existente nas três tabelas do ambiente relacional, aqui denominada como COLUNA_1, e fez-se a divisão em 128 nós.

O cálculo dos resultados gerenciais é realizado em ambiente *Big Data* utilizando *Pig Script* [1] e metodologias de cálculos em *Python*, e os resultados armazenados em *Hive* para evidenciação em painéis específicos.

Uma das diferenças nesse novo processamento é a ausência de cálculos de resultados gerenciais em três fases distintas, como no processo anterior. Neste novo processo, é realizado o carregamento dos dados brutos em ambiente *Hadoop* e armazenado no HBase. Depois disso é que se processam os cálculos do resultado gerencial das operações, ainda em ambiente *Big Data*. Os resultados são armazenados em um *Data Warehouse* para que sejam evidenciados em painéis de divulgação.

O primeiro passo no processo para realização dos cálculos de resultado gerencial utilizando o *Pig Script* é a leitura da base de dados em HBase. Conforme a Figura 3.8, é realizada a leitura das três famílias de colunas da tabela em *HBase*, cada uma gerando

[1] https://github.com/feliperezende86/PPCA

um campo específico do tipo *map*. *Map* é um tipo de dado existente no ambiente *Pig Script* que contém campos do tipo {CHAVE}#{VALOR} como estutura e que consegue realizar o armazenamento das informações no formato do ambiente *NoSQL* Colunar.

```
{VARIÁVEL} =    LOAD 'hbase://{SCHEMA HBASE}:{TABELA HBASE}'
USING org.apache.pig.backend.hadoop.hbase.HBaseStorage(
'{FAMILIA 1}:*, {FAMILIA 2}:*, {FAMILIA 3}:*')
AS ({CAMPO 1}:map[], {CAMPO 2}:map[], {CAMPO 3}:map[]);
```

Figura 3.8: Processo de leitura da tabela em HBase.

O próximo passo para a realização dos cálculos de resultado gerencial é descrito na Figura 3.9, em que é realizada a criação de um conjunto de dados estruturados com relação à base existente em ambiente *HBase*. Para isto, utiliza-se o comando *foreach* {VARIÁVEL} *generate* passando o tipo de dado da coluna, separando por $ e depois informando o nome do campo # e o nome da coluna. Deve-se também renomear a coluna que receberá esse valor. Faz-se isso para todas as colunas que deseja utilizar, existentes na variável com os dados da base de dados lida no processo anterior.

```
{VARIÁVEL_2} = foreach {VARIÁVEL} generate (tipo de dado)${CAMPO 1}#'{NOME DA COLUNA}' AS COLUNA_1,
(tipo de dado)${CAMPO 1}#'{NOME DA COLUNA}' AS COLUNA_2, (tipo de dado)${CAMPO 1}#'{NOME DA COLUNA}' AS COLUNA_3,
(tipo de dado)${CAMPO 2}#'{NOME DA COLUNA}' AS COLUNA_4, (tipo de dado)${CAMPO 2}#'{NOME DA COLUNA}' AS COLUNA_5,
(tipo de dado)${CAMPO 3}#'{NOME DA COLUNA}' AS COLUNA_6, (tipo de dado)${CAMPO 3}#'{NOME DA COLUNA}' AS COLUNA_7;
```

Figura 3.9: Processo de geração de um conjunto de dados estruturado.

Como mencionado no trabalho, os cálculos necessários para apuração do resultado gerencial foram desenvolvidos utilizando a linguagem *Python*. Foram criados dois programas, o primeiro é onde será realizado o cálculo de volume de operações de crédito na instituição, onde se aplica o saldo da operação dividido pela quantidade de dias corridos do mês de referência. O outro programa realiza o cálculo de "Despesa de Oportunidade". Valor existente no DRE da operação e para realizar o cálculo deve-se informar à função alguns atributos existentes na base de dados a ser tratada e como saída teremos o valor deste componente calculado para o resultado gerencial. Um recurso disponível em *Pig Script* é a utilização de programas criados nessa linguagem, bastando utilizar os comandos da Figura 3.10, onde é descrito o caminho onde está salvo o programa em *Python* e qual o "*Alias*" será chamado.

```
REGISTER '/{caminho no HDFS}/{arquivo.py}'
USING org.apache.pig.scripting.streaming.python.PythonScriptEngine AS {ALIAS}
```

Figura 3.10: Definição de um *Alias* e o caminho onde se encontra o programa em Python.

Com isso, conforme ilustrado na Figura 3.11, novamente é executado o comando *foreach* e *generate*, trazendo todos os atributos já existentes na variável que contém o conjunto

de dado estruturado e chamando a função existente no programa em *Python* ele tem o retorno da função como novos atributos do conjunto de dados estruturados.

```
{RESULTADO} = FOREACH {VARIÁVEL} GENERATE *,
FLATTEN({ALIAS}.{FUNÇÃO(ATRIBUTO)}) as ({ATRIBUTOS DE RESPOSTA});
```

Figura 3.11: Execução e chamado do programa em *Python*.

Para manter armazenado o resultado gerencial calculado neste processo, utilizou-se, como estrutura disponível para *Data Warehouse* existente no ambiente *Hadoop*, o *Hive*. A Figura 3.12 apresenta o código utilizado para a realização da carga do conjunto de dados estruturados na base de dados *Hive* em que já se tem criada uma tabela com as colunas a serem recebidas, e em que, devido ao grande volume de dados apurados diariamente, foi realizada a criação da tabela com um particionamento de *ANO* e *MÊS*.

```
STORE RST INTO '{schema}.{banco de dados HIVE}'
USING org.apache.hive.hcatalog.pig.HCatStorer('ano = aa_ref, mes = mm_ref');
```

Figura 3.12: Processo de carga em ambiente Hive.

3.4 Ambiente Computacional para a Proposta

Como problema apresentado no trabalho, a frequência mensal de processamento do resultado gerencial das operações traz um *déficit* de informações gerenciais de curto prazo. Uma solução proposta para o estudo foi a utilização de um sistema NoSQL Colunar (HDFS/HBase). Como método aplicado, tem-se a criação de um ambiente HDFS/HBase e uma rotina de carga diária das informações utilizando a ferramenta *Sqoop* e o processamento dos dados utilizando *Pig Script* com a carga dos valores produzidos no ambiente *Hive*, todos presentes no ecossistema *Hadoop*. Dessa forma, ter a possibilidade de realização do processo de cálculo dos resultados gerenciais diariamente com um tempo menor de finalização de execução é o resultado esperado pelo trabalho proposto.

Como mencionado anteriormente, por sigilo profissional muitas informações são impossíveis de serem divulgadas em trabalhos externos, mesmo como conteúdo acadêmico. Porém, é possível acessar, por meio de *{link do servidor}:{porta de acesso}/cluster/apps*, a demonstração de algumas informações sobre a configuração do cluster, como demonstra a Figura 3.13, assim como as atividades já executadas, tempo de duração, e as tarefas que estão sendo executadas no cluster naquele momento.

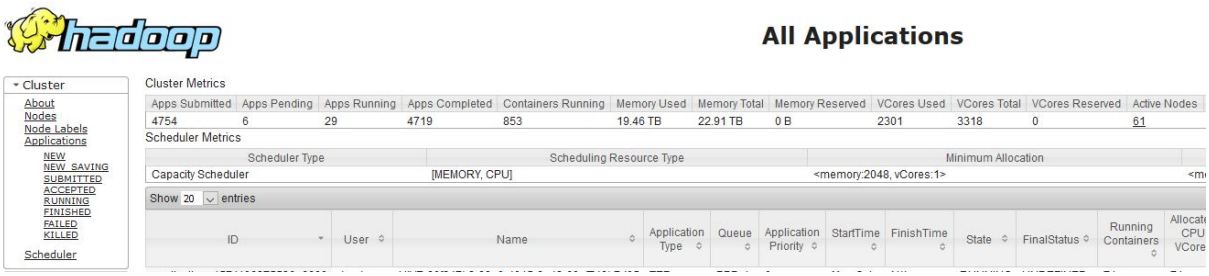


Figura 3.13: Captura de tela de visualização das operações em execução no cluster.

Neste painel, vê-se a configuração de um *cluster* com um total de memória RAM de 22.91 TB, um total de *cores* de 3.318 e 61 *nodes*. Estes são valores disponíveis no painel com data do acesso em 16/10/2019, podendo ser disponibilizados mais recursos a partir do momento que seja necessário.

Para execução em ambiente *Big Data*, a instituição criou máquinas virtuais de acesso remoto, para que seja facilitado o acesso a aplicativos necessários para a execução das tarefas. Estas máquinas virtuais utilizam sistema operacional *Windows 10* e dispõem de processadores *Intel I5* e 16 GB de memória RAM.

Quando se criam comandos em *shell script*, descritos anteriormente, estes são executados por meio do aplicativo MobaXTerm, um aplicativo *free source* que fica disponível para instalação nas máquinas virtuais de usuários que trabalham com Ciência dos Dados. Este aplicativo simula a execução de um terminal *Linux*, e desta forma é possível executar os comandos no *cluster* configurado no servidor.

As versões utilizadas para solução do problema proposto no trabalho são:

- Hadoop - versão 2.7.3.2.6.1.0-129
- HBase - versão 1.1.2.2.6.1.0-129
- Sqoop - versão 1.4.6.2.6.1.0-129
- Hive - versão 1.2.1000.2.6.1.0-129
- Python - versão 3.6.5

Desta forma e neste ambiente foram executados os processos, tanto de captura das informações em ambiente relacional (*Sqoop*), como o processamento das informações gerenciais (processamento em *Pig*) e o armazenamento dos valores apurados para possíveis evidenciações em ambiente de negócios (armazenamento em *Hive*).

Capítulo 4

Análise de Resultados

Este capítulo apresenta uma análise dos resultados da implementação da proposta apresentada no capítulo anterior. A Seção 4.1 contém a descrição da aplicação do projeto desenvolvido e apresenta os resultados obtidos. A Seção 4.2 apresenta a discussão dos resultados apresentados.

4.1 Descrição dos Testes

Durante o mês de julho de 2019, foram executados os processos de captura dos dados do ambiente relacional, necessários para o cálculo gerencial e armazenados em ambiente *Big Data* no *Hadoop - HBase* e também o processamento desses dados utilizando *Pig*, assim como a carga do resultado gerencial em ambiente *Hive*. Os processos foram executados em diversas horas do dia para verificar as características de processamento do ecossistema *Hadoop* configurado na instituição. Os horários de execução foram escolhidos de forma aleatória para que se pudesse analisar se teria alteração de desempenho de acordo com o horário do dia.

Os primeiros testes realizados foram referentes ao processo de transferência dos dados. Para esses testes, foi criado um *script* em *shell script*. Neste *script* foram descritos os três processos de *Sqoop* para envio dos dados do ambiente relacional para o *Hadoop - HBase*, sendo primeiramente os dois processos referentes às tabelas de informações das operações, onde se executa somente uma atualização das operações em que o estado de condução foi alterado no último dia útil anterior, lembrando que o estado de condução identifica as operações que ainda estão vigentes na instituição. O terceiro processo, referente à tabela de saldo, executava a inserção diária dos dados disponíveis em ambiente relacional, sobrescrevendo os valores anteriormente armazenados no *HBase*.

A Tabela 4.1 ilustra os tempos de processamento diário das cargas de dados realizadas pelo *Sqoop*. Todos esses processos de carga permaneceram dentro da média geral de tempo

Tabela 4.1: Tempos de processamento do Sqoop diário.

MOVIMENTO	PROCESSO 1	PROCESSO 2	PROCESSO 3
01/07/2019	00:06:27	00:05:24	01:54:41
02/07/2019	00:15:30	00:15:35	04:49:48
03/07/2019	00:02:43	00:02:50	01:51:00
04/07/2019	00:06:15	00:04:40	01:43:07
05/07/2019	00:02:47	00:02:31	01:45:03
08/07/2019	00:03:15	00:02:52	01:33:23
09/07/2019	00:03:32	00:02:42	03:41:06
10/07/2019	00:04:14	00:02:20	01:48:20
11/07/2019	00:04:35	00:02:38	01:41:57
12/07/2019	00:06:42	00:11:08	01:22:00
15/07/2019	00:03:30	00:02:57	01:58:16
16/07/2019	00:02:50	00:02:29	01:27:37
17/07/2019	00:03:10	00:03:37	01:59:52
18/07/2019	00:04:08	00:03:55	03:17:47
19/07/2019	00:04:55	00:04:06	02:16:10
22/07/2019	00:02:51	00:03:20	01:38:22
23/07/2019	00:03:37	00:02:51	01:45:16
24/07/2019	00:03:23	00:02:42	04:00:34
25/07/2019	00:03:23	00:02:34	01:44:41
26/07/2019	00:03:30	00:02:40	01:42:30
29/07/2019	00:02:59	00:02:29	01:47:08
30/07/2019	00:03:33	00:02:46	01:47:23
31/07/2019	00:04:23	00:03:23	01:49:31

de processamento, sendo para o *Processo 1* um total de 4 minutos e 27 segundos, para o *Processo 2* um total de 4 minutos e 1 segundo e para o *Processo 3* de carga uma média de 2 horas e 8 minutos. Se considerarmos a mediana dos tempos, teríamos 3 minutos e 32 segundos para o *Processo 1*, 2 minutos e 51 segundos para o *Processo 2* e 1 hora 47 minutos e 23 segundos para o *Processo 3*. Um tempo total de 1 hora 53 minutos e 46 segundos para a realização da carga total das três tabelas de ambiente relacional para o *HBase* utilizando o Sqoop. O Gráfico disponível na Figura 4.1 demonstra o processamento dia a dia das realizações do processo de Sqoop, em que o *Processo 1* é referente à carga da *Tabela 1*, o *Processo 2* é referente à carga da *Tabela 2* e o *Processo 3* é referente à carga da *Tabela 3*, lembrando que o Processo 3 executa a carga dos valores dos saldos existentes em todas as operações da instituição financeira.

Em relação ao tempo total para execução do processo, destaca-se o dia 02/07/2019, em que houve uma reestruturação de perfil dos usuários do ambiente Hadoop, determinando as prioridades de execução das tarefas no *cluster*. Houve uma variação no tempo de processamento de 393% para o processo 1, de 446% para o processo 2 e de 238% para o processo 3. Estes valores não foram removidos dos resultados para que fosse possível

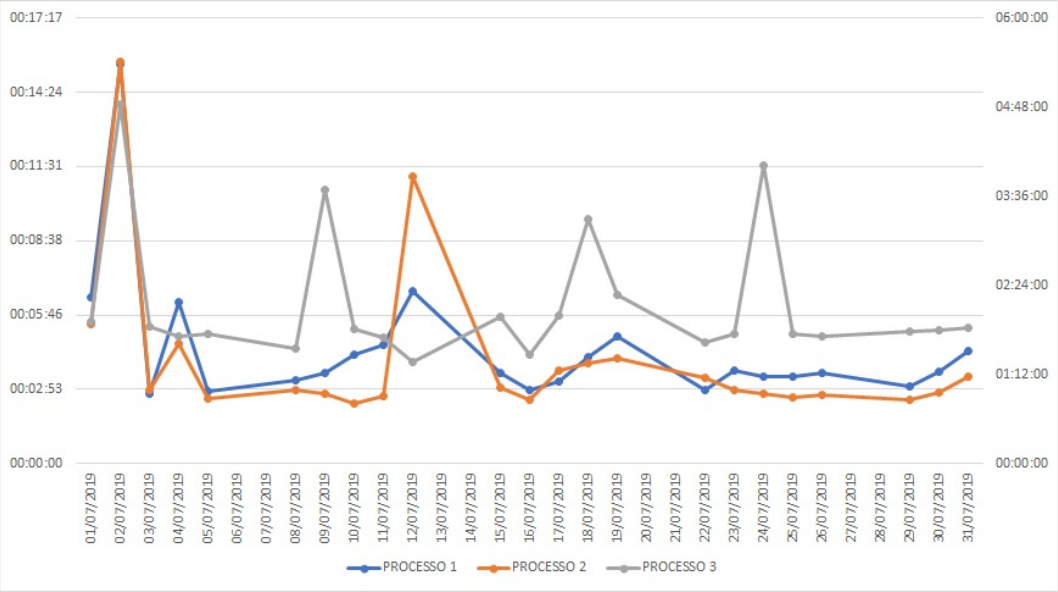


Figura 4.1: Gráfico do tempo de processamento do Sqoop diário.

destacar a importância da correta configuração das características de prioridades dos usuários no *ambiente Hadoop*.

Como apresentado na Figura 3.5, depois de realizar as transferências dos dados, armazenando-os no *HBase*, é passado o comando para o processamento do resultado gerencial, utilizando o *Apache Pig*.

Na Tabela 4.2 estão presentes os tempos de processamento do *Pig Script* contendo, como início e término, os horários de execução da rotina. Como entrada, a quantidade de valores lidos em *HBase* para geração dos valores de resultado gerencial das operações da instituição financeira. Como saída, os valores processados e armazenados em *ambiente Hive*. Há uma grande diferença entre valores de entrada e de saída, haja vista que a *Tabela 1* mantém contratos que não são escopo do trabalho. São contratos de outras linhas da instituição financeira, o objetivo do trabalho é a realização dos cálculos de resultado gerencial somente das operações de crédito da instituição. Como dito anteriormente, utilizaram-se todas as operações de crédito existentes nas três tabelas, portanto, na hora de execução, estas não são processadas e consequentemente não são armazenadas na saída. Para cálculo do tempo total de processamento dos valores de resultado gerencial, faz-se a diferença entre momento do término do processo e o momento de início deste, lembrando que esse processo em *Pig Latin* executa a leitura dos dados armazenados em *HBase*, processa os dados com algumas metodologias aplicadas em *Python* para apuração dos resultados gerenciais e realiza a carga dos valores em ambiente *Hive*. A coluna "Tempo Total"demonstra o tempo de processamento para conclusão do processo descrito como solução do problema levantado no trabalho.

Tabela 4.2: Tempos de processamento do Pig Script diário.

MOVIMENTO	INÍCIO	TÉRMINO	ENTRADA	SAÍDA	TEMPO TOTAL
01/07/2019	16:33:17	17:54:50	414.209.740	42.179.745	01:21:33
02/07/2019	13:48:57	20:54:07	414.314.768	42.088.591	07:05:10
03/07/2019	21:36:33	22:14:06	414.226.370	42.105.061	00:37:33
04/07/2019	13:19:31	13:56:22	414.451.067	41.873.887	00:36:51
05/07/2019	09:50:44	10:22:35	414.545.445	42.325.852	00:31:51
08/07/2019	10:48:54	11:22:00	414.671.376	42.269.152	00:33:06
09/07/2019	13:27:46	14:00:05	414.250.293	42.124.466	00:32:19
10/07/2019	10:05:32	10:37:13	414.330.735	42.022.513	00:31:41
11/07/2019	15:38:56	16:08:36	414.407.627	42.101.600	00:29:40
12/07/2019	11:59:10	12:31:52	415.148.678	41.823.157	00:32:42
15/07/2019	13:53:49	14:30:44	415.251.215	41.445.451	00:36:55
16/07/2019	11:05:00	11:39:48	414.527.938	42.032.680	00:34:48
17/07/2019	11:48:43	12:18:11	415.434.880	41.261.948	00:29:28
18/07/2019	14:35:14	15:12:37	415.496.014	41.160.870	00:37:23
19/07/2019	12:16:37	12:51:25	415.594.420	41.034.569	00:34:48
22/07/2019	11:28:08	12:02:21	415.716.101	40.967.883	00:34:13
23/07/2019	13:39:43	14:12:21	415.804.190	40.931.844	00:32:38
24/07/2019	12:05:31	12:41:13	415.865.184	40.820.099	00:35:42
25/07/2019	10:10:41	10:45:02	415.946.027	40.967.426	00:34:21
26/07/2019	10:05:09	10:36:16	416.029.512	40.700.992	00:31:07
29/07/2019	09:18:39	09:51:10	416.126.306	40.686.278	00:32:31
30/07/2019	10:19:56	10:54:50	416.207.746	40.406.500	00:34:54
31/07/2019	08:05:32	08:34:35	416.265.769	40.361.864	00:29:03

Analisando o tempo de processamento, registrou-se como média de quantidade de entrada um total de 415.166.148 linhas e como saída do processamento uma média de 41.464.888 linhas. O tempo médio necessário para a execução da rotina ficou em 52 minutos e 37 segundos, porém tem-se que atentar ao descrito no processo de carga de dados que o tempo de processamento do dia 01/07/2019, assim como do dia 02/07/2019 foi um valor bem superior à média geral de processamento, devido a situações específicas de configuração do *cluster*. Para fins de análise final de desempenho, estamos utilizando a mediana que, para este processo, o valor foi de 34 minutos e 13 segundos. Na Tabela 4.2 existe a coluna "Entrada"e "Saída", que descrevem diariamente o total de operações processadas.

Provavelmente, no primeiro caso, como o processo foi iniciado no final da tarde, acredita-se que já estivesse com as alterações das preferências de execução das tarefas no *cluster*. Para estes casos, houve uma alteração com relação à média de processamento de 243% e de 1269%, respectivamente. Caso desconsidere estes dois dias atípicos e com desempenho justificável, a média de tempo de processamento ficaria descrita em 33 minutos e 30 segundos.

Se realizarmos as somas totais do processo de carga das tabelas e de processamento das informações de resultado gerencial e armazenamento em ambiente *Big Data*, temos um tempo total de 2 horas 27 minutos e 59 segundos para este processo de cálculo de resultados gerenciais com processamentos e divulgação das informações de maneira diária.

4.2 Discussão dos Resultados

Analisando os resultados obtidos no trabalho, foi possível identificar que o tempo médio de processo é de apenas 2 horas 27 minutos e 59 segundo e que o NoSQL para esse tipo de operação poderia trazer redução de tempo de processamento. Sendo assim, gerando um *Data Warehouse* em ambiente *Big Data* com os componentes de resultado gerencial obtidos no processo.

Como descrito no início do trabalho, hoje só existe o processamento no primeiro final de semana subsequente ao encerramento do mês de referência. Como foi mencionado, atualmente o processo de geração de resultado gerencial tem um tempo total de processamento em aproximadamente 30 horas. Alterando o sistema gerenciador de banco de dados para um NoSQL colunar e utilizando a estrutura Hadoop, sendo possível a realização destes processamentos de forma paralela. Com esta alteração, seria possível a realização de processamento diário para que se tivesse acompanhamento dos resultados gerenciais das operações da instituição com apenas um dia de defasagem, melhorando

com isso a possibilidade de gestão dos recursos administrados pela instituição, podendo gerar melhores resultados.

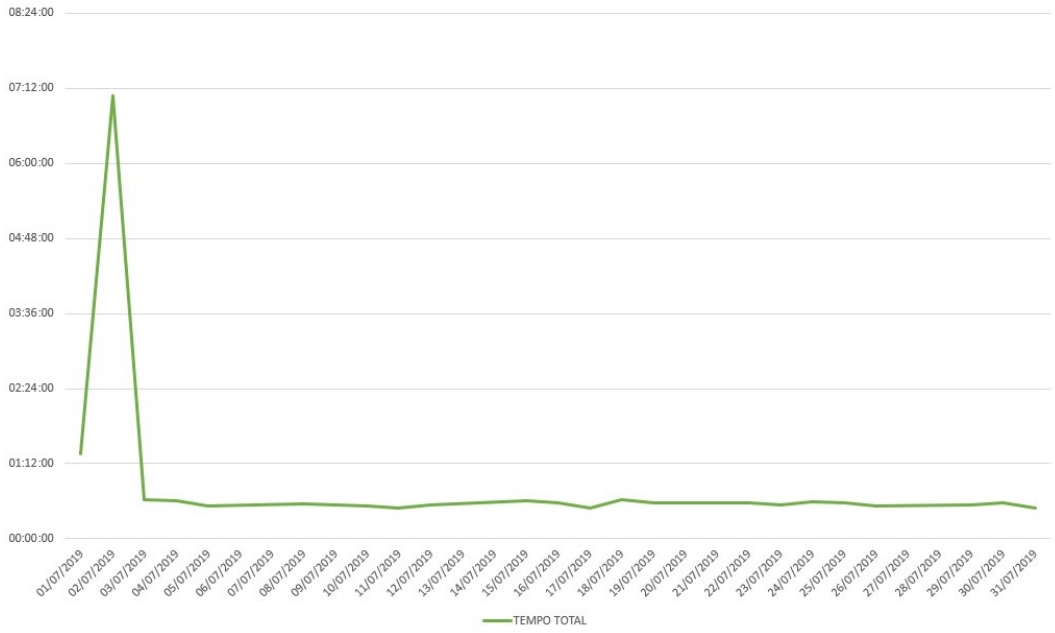


Figura 4.2: Gráfico demonstrando o tempo total de processamento dos resultados gerenciais de todas as operações de crédito da instituição.

Quando se verifica o gráfico com o tempo total de processamento diário das informações na Figura 4.2, vê-se que se manteve uma constante em aproximadamente 30 minutos, excetuando-se os dias iniciais, com o motivo destacado no trabalho.

O processamento foi realizado todos os dias do mês, conseguindo-se dessa forma obter o resultado gerencial das operações exatamente no momento em que se encerra o mês de referência. Todos os valores foram apurados diariamente e o resultado acumulado para o resultado em competência mensal.

Um dos problemas encontrados nos primeiros dias do mês de execução tem referência com a ordem de prioridade de utilização dos recursos do *cluster*. Quando é criado o usuário de acesso ao *cluster*, é possível definir sua ordem de prioridade na execução de suas tarefas. Por questões de alteração de grupo de trabalho responsável por este controle de concorrência, o usuário que executava os processos descritos no trabalho foi inserido em um grupo de baixa prioridade, o que fazia com que todos os processos que entravam na fila para execução tivessem prioridade superior de conclusão, fazendo com que os tempos de resposta tivessem seus valores majorados. A partir do momento que o cadastro de preferência foi regularizado, os tempos de processamento voltaram para a média de 33 minutos de execução.

Tabela 4.3: Tabela demonstrando quantidade de operações e valor de somatório de componente de resultado gerencial por sistema.

SISTEMA	TOTAL DE LINHAS	VALOR (R$)
SISTEMA 1	322.148.817	3.514.330.139,48
SISTEMA 2	249.826.097	269.248.658,09
SISTEMA 3	122.797.777	400.325.604,99
SISTEMA 4	7.355.349	172.694.379,93
SISTEMA 5	6.864.615	21.981.756,63

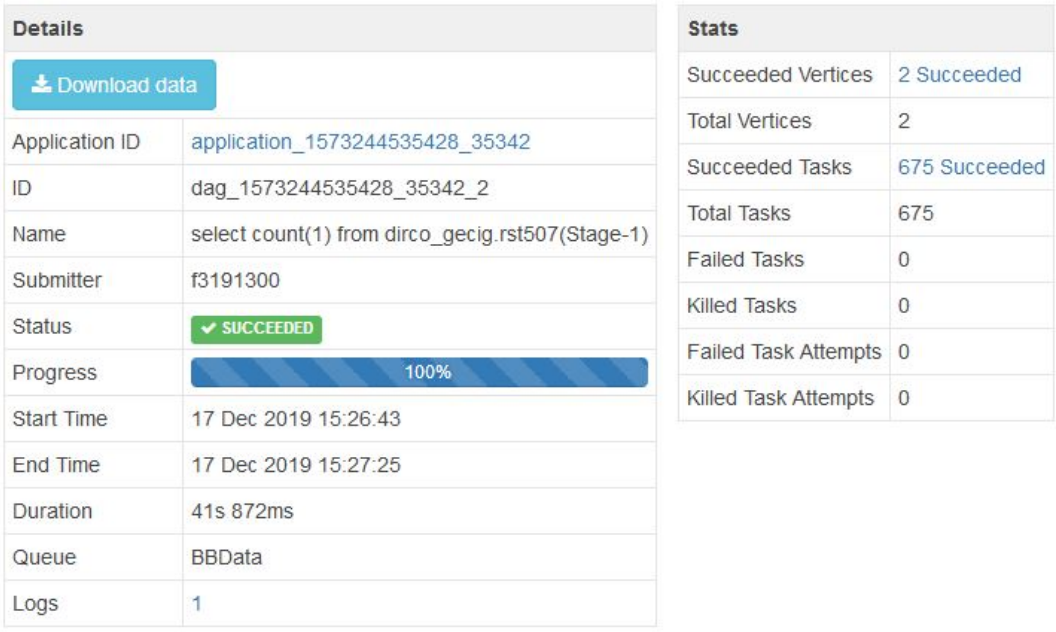
Para analisarmos o desempenho do armazenamento dos dados em um *Data Warehouse* para eventuais consultas e evidenciações em relatórios de resultado gerencial, foi realizada uma consulta na tabela em *Hive* que contém os resultados. Como descrito no trabalho, tanto o sistema como o componente de resultado pesquisado não serão descritos por questão de política de privacidade da instituição. A Figura 4.3 apresenta o resultado da pesquisa realizada na Tabela de resultado do processo realizado em *Pig Script*, com a contagem de quantas linhas existem no sistema e qual o somatório do componente de resultado financeiro para o mês de referência. A Tabela apresenta os cinco primeiros sistemas em ordem de maior quantidade de linhas (operações), sendo o Sistema 1 (contendo 322.148.817 operações e totalizando um valor de 3.514.330.139,48) como o maior valor em quantidade de operações e de valor do componente de resultado gerencial. O Sistema 5 contou com 8.864.615 operações e 21.981.756,63 de resultado gerencial como o último nessa escala de cinco sistemas, lembrando que foram realizados componentes de resultado para os 26 sistemas da instituição.

Outra pesquisa realizada para verificação de desempenho foi com relação à quantidade de linhas existentes na Tabela de resultado do processamento. Para esta pesquisa, teve-se como resultado o total de 951.995.909 linhas, ou seja, foram quase um bilhão de operações processadas ao longo do mês de referência.

A Figura 4.3 contém o tempo total para realizar a consulta contando quantas linhas existem na Tabela de saída do processo de geração de resultado gerencial armazenada em ambiente *Hive*, com um total de 41 segundos. Para realização desta consulta, o Apache Hive realizou o processamento em 675 *maps* paralelamente.

A Figura 4.4 demonstra o *log* contendo o tempo necessário para realizar a consulta, realizando o agrupamento de todos os valores do componente de resultado e contando quantas operações existem no mês de referência da tabela. Neste caso o tempo total ficou em 43 segundos. Para este caso, foram utilizados 686 *maps* para processamento em paralelo.

Como demonstrado neste Capítulo, foi possível realizar o armazenamento dos dados, assim como a realização dos cálculos gerenciais utilizando estrutura de *Big Data*. A rea-



Details

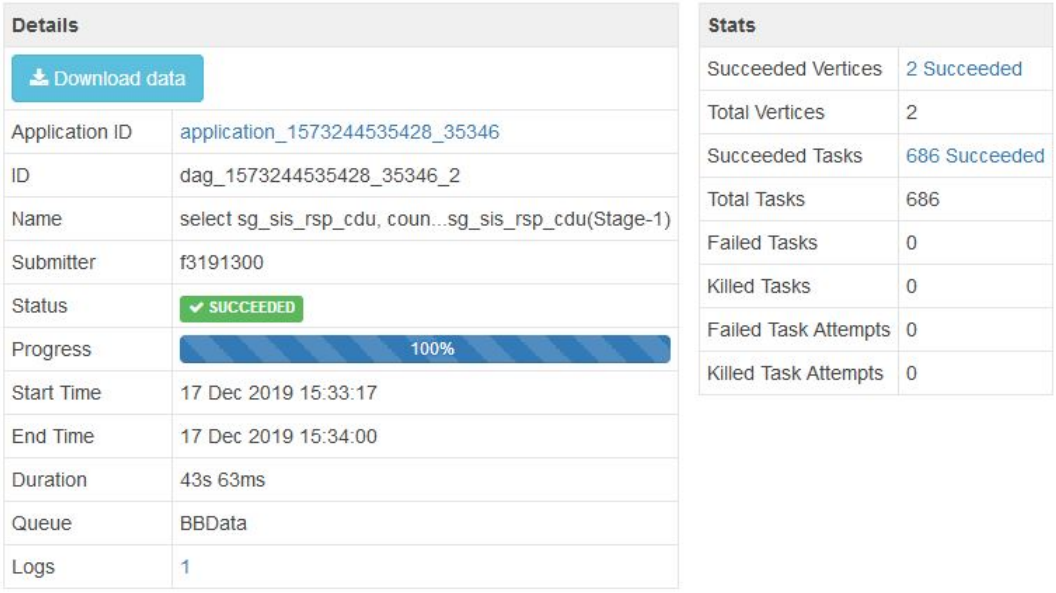
Download data

Application ID	application_1573244535428_35342
ID	dag_1573244535428_35342_2
Name	select count(1) from dirco_gecig.rst507(Stage-1)
Submitter	f3191300
Status	SUCCEEDED
Progress	100%
Start Time	17 Dec 2019 15:26:43
End Time	17 Dec 2019 15:27:25
Duration	41s 872ms
Queue	BBData
Logs	1

Stats

Succeeded Vertices	2 Succeeded
Total Vertices	2
Succeeded Tasks	675 Succeeded
Total Tasks	675
Failed Tasks	0
Killed Tasks	0
Failed Task Attempts	0
Killed Task Attempts	0

Figura 4.3: Log de resultado da consulta de total de linhas na Tabela.



Details

Download data

Application ID	application_1573244535428_35346
ID	dag_1573244535428_35346_2
Name	select sg_sis_rsp_cdu, coun...sg_sis_rsp_cdu(Stage-1)
Submitter	f3191300
Status	SUCCEEDED
Progress	100%
Start Time	17 Dec 2019 15:33:17
End Time	17 Dec 2019 15:34:00
Duration	43s 63ms
Queue	BBData
Logs	1

Stats

Succeeded Vertices	2 Succeeded
Total Vertices	2
Succeeded Tasks	686 Succeeded
Total Tasks	686
Failed Tasks	0
Killed Tasks	0
Failed Task Attempts	0
Killed Task Attempts	0

Figura 4.4: Log de resultado da consulta da Tabela 4.3.

lização de armazenamento foi em *Hive*, para eventuais evidenciações por meio de painéis de resultado. Todos os processos obtiveram um tempo de processamento satisfatório para que fosse possível a implementação do processamento desse grande volume de dados de forma diária.

Capítulo 5

Conclusões

Nesse trabalho foi apresentada a migração de um modelo relacional da instituição Beta para um modelo de dados baseado em NoSQL Colunar. Os resultados atenderam aos objetivos prepostos no trabalho, devido a seu tempo de processamento ser algo satisfatório para processamentos diários desse grande volume de dados para a geração dos resultados gerenciais da instituição.

Pôde-se constatar, com os resultados obtidos com o trabalho, que existe uma maneira mais eficiente de processamento das informações gerenciais para a instituição financeira. Como mencionado, atualmente o processo de geração de resultado gerencial tem um tempo total de processamento de aproximadamente 30 horas. O formato proposto no trabalho teve um tempo total de execução dos cálculos de resultado gerencial e armazenamento dos dados de 2 horas e 28 minutos. O tempo de processamento das informações existentes de todas as operações em situação vigente é algo a se considerar de grande ganho para o trabalho.

Por atender ao trabalho e não ser de grande alteração dos resultados e dos processos da instituição, optou-se por utilizar os mesmos atributos das tabelas detentoras das informações base para os cálculos gerenciais como granularidade para os cálculos gerenciais. Os dados necessários para a realização dos cálculos de resultado gerencial foram carregados em ambiente *Big Data* por meio da estrutura *Sqoop*, presente no ecossistema *Hadoop*. Também podemos constatar, dentro da estrutura do ecossistema *Hadoop*, a presença do *Pig Script*, uma estrutura para processamento de *Big Data* muito eficiente que utiliza o *MapReduce* para otimizar seus desempenhos.

Um dos problemas levantados na pesquisa é o tempo de divulgação das informações processadas, outro ponto de destaque para o trabalho. Haja vista a estrutura de *Data Warehouse* disponível no *Hive*, possibilitando a evidenciação das informações em *Dashboards* tão logo sejam realizados os cálculos de resultados gerenciais. Sempre realizando o acúmulo das informações para o mês de referência.

Dependendo de comandos manuais para execução dos trabalhos, todos os processos foram executados em horário de expediente. Sabe-se que poderia ser realizada a análise do processo em horários de baixa utilização dos recursos do *cluster*, porém não foi possível agendar as rotinas para serem executadas em horário diferente da jornada de trabalho do responsável pelo estudo.

Toda esta estrutura de processamento de grande volume de dados já vem sendo utilizada em outras áreas da diretoria para tratamento de informações de forma mais granular e conseguindo realizar evidenciações em diversos formatos. Alguns processamentos não eram possíveis serem realizados por um total de clientes, tendo em vista o grande volume de dados necessário para esse processamento. Foi realizado o processo de carga dos dados em ambiente *Big Data* e realizado o processamento utilizando o ecossistema *Hadoop*.

Está previsto como trabalho futuro o agendamento destes processos para realização de forma automatizada, sendo assim as execuções devem acontecer durante a madrugada, momento em que se tem uma baixa concorrência para quaisquer tipos de processamentos. Também inserido como trabalho futuro fica a execução do processamento em ambiente *Apache Spark* para verificar o possível ganho de desempenho da geração dos valores de resultado gerencial. Melhora esta considerada a alteração da estrutura do *MapReduce* encontrada no *Apache Spark*. Outro ponto passível de implementação futura é a elaboração dos painéis de evidenciação dos resultados obtidos no processamento, de acordo com os existentes atualmente na instituição financeira. Como o trabalho focou em operações de crédito, seria necessária a ampliação do processo para os outros tipos de operações, para se ter um resultado gerencial por completo dos valores produzidos na instituição.

Referências

[1] Salazar, José Nicolás Albuja e Gideon Carvalho de Benedicto: *Contabilidade financeira*. Cengage Learning Editores, 2004. 1

[2] Nikolay, Rafael e Luiz Fernando Costa Neves: *Contabilidade gerencial como base à controladoria*. Revista Eletrônica do Curso de Ciências Contábeis, 5(9):55–80, 2016. 1

[3] Andrade, Luis Claudio Magnago, Aridelmo José Campanharo Teixeira, Graziela Xavier Fortunato e Valcemiro Nossa: *Determinantes para a utilização de práticas de contabilidade gerencial estratégica: um estudo empírico*. Revista de Administração Mackenzie (Mackenzie Management Review), 14(1), 2011. 1

[4] Nayak, Ameya, Anil Poriya e Dikshay Poojary: *Type of nosql databases and its comparison with relational databases*. International Journal of Applied Information Systems, 5(4):16–19, 2013. 2, 7

[5] Han, Jing, Haihong E, Guan Le e Jian Du: *Survey on nosql database*. Em *2011 6th International Conference on Pervasive Computing and Applications*, páginas 363–366, Oct 2011. 3, 8, 12

[6] Dehdouh, Khaled, Omar Boussaid e Fadila Bentayeb: *Columnar nosql star schema benchmark*. Em Ait Ameur, Yamine, Ladjel Bellatreche e George A. Papadopoulos (editores): *Model and Data Engineering*, páginas 281–288, Cham, 2014. Springer International Publishing, ISBN 978-3-319-11587-0. 3, 9

[7] Su, Fei, Zhenya Wang, Shan Yang, Ke Li, Xin Lu, Yang Wu e Yi Peng: *A survey on big data analytics technologies*. Em *International Conference on 5G for Future Wireless Networks*, páginas 359–370. Springer, 2017. 3, 22

[8] Siddiqa, Aisha, Ahmad Karim e Abdullah Gani: *Big data storage technologies: a survey*. Frontiers of Information Technology & Electronic Engineering, 18(8):1040–1070, 2017. 3, 7

[9] Peruchi Junior, Marcos: *Controladoria: uma análise comparativa entre a perspectiva teórica e a prática em uma empresa do setor metalmecânico*. 2016. 3

[10] Poljak, Robert, Patrizia Poščić e Danijela Jakšić: *Comparative analysis of the selected relational database management systems*. Em *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, páginas 1496–1500. IEEE, 2017. 6

[11] Luo, S., Z. Gao, M. Gubanov, L. L. Perez e C. Jermaine: *Scalable linear algebra on a relational database system*. IEEE Transactions on Knowledge and Data Engineering, páginas 1–1, 2018, ISSN 1041-4347. 6

[12] Bathla, G., R. Rani e H. Aggarwal: *Comparative study of nosql databases for big data storage*. International Journal of Engineering and Technology(UAE), 7(2):83–87, 2018, ISSN 2227524X. 7

[13] Dehdouh, Khaled: *Building olap cubes from columnar nosql data warehouses*. Em *International Conference on Model and Data Engineering*, páginas 166–179. Springer, 2016. 7, 11

[14] Oussous, Ahmed, Fatima Zahra Benjelloun, Ayoub Ait Lahcen e Samir Belfkih: *Big data technologies: A survey*. Journal of King Saud University-Computer and Information Sciences, 2017. 7, 14

[15] Chiang, Roger HL, Varun Grover, Ting Peng Liang e Dongsong Zhang: *Strategic value of big data and business analytics*, 2018. 7

[16] Grover, Varun, Roger HL Chiang, Ting Peng Liang e Dongsong Zhang: *Creating strategic business value from big data analytics: A research framework*. Journal of Management Information Systems, 35(2):388–423, 2018. 7

[17] Song, Ma Lin, Ron Fisher, Jian Lin Wang e Lian Biao Cui: *Environmental performance evaluation with big data: Theories and methods*. Annals of Operations Research, 270(1-2):459–472, 2018. 7

[18] Gessert, Felix, Wolfram Wingerath, Steffen Friedrich e Norbert Ritter: *Nosql database systems: a survey and decision guidance*. Computer Science-Research and Development, 32(3-4):353–365, 2017. 7, 10

[19] Hauger, WK e MS Olivier: *Nosql databases: forensic attribution implications*. SAIEE Africa Research Journal, 109(2):119–132, 2018. 8

[20] Amghar, S., S. Cherdal e S. Mouline: *Which nosql database for iot applications?* Em *2018 International Conference on Selected Topics in Mobile and Wireless Networking (MoWNeT)*, páginas 131–137, June 2018. 8

[21] Moniruzzaman, ABM e Syed Akhter Hossain: *Nosql database: New era of databases for big data analytics-classification, characteristics and comparison*. arXiv preprint arXiv:1307.0191, 2013. 8

[22] Hecht, Robin e Stefan Jablonski: *Nosql evaluation: A use case oriented survey*. Em *2011 International Conference on Cloud and Service Computing*, páginas 336–341. IEEE, 2011. 8, 9

[23] Corbellini, Alejandro, Cristian Mateos, Alejandro Zunino, Daniela Godoy e Silvia Schiaffino: *Persisting big-data: The nosql landscape*. Information Systems, 63:1 – 23, 2017, ISSN 0306-4379. `http://www.sciencedirect.com/science/article/pii/S0306437916303210`. 9

[24] Mattis, Toni, Johannes Henning, Patrick Rein, Robert Hirschfeld e Malte Appeltauer: *Columnar objects: Improving the performance of analytical applications.* Em *2015 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward!)*, páginas 197–210. ACM, 2015. 9

[25] Dehdouh, Khaled, Fadila Bentayeb, Omar Boussaid e Nadia Kabachi: *Using the column oriented nosql model for implementing big data warehouses.* Em *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, página 469. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2015. 9, 11

[26] Dehdouh, Khaled, Fadila Bentayeb, Omar Boussaid e Nadia Kabachi: *Columnar nosql cube: Agregation operator for columnar nosql data warehouse.* Em *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, páginas 3828–3833. IEEE, 2014. 11

[27] Alami, Lamiae, Imad Hafidi e Abdelmotalib Metrane: *Entity resolution in nosql data warehouse.* Em *International Conference on Information Technology and Communication Systems*, páginas 51–59. Springer, 2017. 11

[28] Oditis, Ivo, Zane Bicevska, Janis Bicevskis e Girts Karnitis: *Implementation of nosql-based data warehouses.* BALTIC JOURNAL OF MODERN COMPUTING, 6(1):45–55, 2018. 11

[29] Monteith, J Yates, John D McGregor e John E Ingram: *Hadoop and its evolving ecosystem.* Em *5th International Workshop on Software Ecosystems (IWSECO 2013)*, volume 50. Citeseer, 2013. 12

[30] Chang, Fay, Jeffrey Dean, Sanjay Ghemawat, Wilson C Hsieh, Deborah A Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes e Robert E Gruber: *Bigtable: A distributed storage system for structured data.* ACM Transactions on Computer Systems (TOCS), 26(2):4, 2008. 12

[31] Kambatla, Karthik, Giorgos Kollias, Vipin Kumar e Ananth Grama: *Trends in big data analytics.* Journal of Parallel and Distributed Computing, 74(7):2561–2573, 2014. 12

[32] Ghemawat, Sanjay, Howard Gobioff e Shun Tak Leung: *The google file system.* Em *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, páginas 20–43, Bolton Landing, NY, 2003. 12

[33] Hunt, Patrick, Mahadev Konar, Flavio Paiva Junqueira e Benjamin Reed: *Zookeeper: Wait-free coordination for internet-scale systems.* Em *USENIX annual technical conference*, volume 8. Boston, MA, USA, 2010. 12

[34] Reed, Benjamin e Flavio P Junqueira: *A simple totally ordered broadcast protocol.* Em *proceedings of the 2nd Workshop on Large-Scale Distributed Systems and Middleware*, página 2. ACM New York, NY, USA, 2008. 12

[35] Dean, Jeffrey e Sanjay Ghemawat: *Mapreduce: a flexible data processing tool*. Communications of the ACM, 53(1):72–77, 2010. 13

[36] Vora, Mehul Nalin: *Hadoop-hbase for large-scale data*. Em *Proceedings of 2011 International Conference on Computer Science and Network Technology*, volume 1, páginas 601–605. IEEE, 2011. 13, 14

[37] Vohra, Deepak: *Using apache sqoop*. Em *Pro Docker*, páginas 151–183. Springer, 2016. 15

[38] Aravinth, SS, A Haseenah Begam, S Shanmugapriyaa, S Sowmya e E Arun: *An efficient hadoop frameworks sqoop and ambari for big data processing*. International Journal for Innovative Research in Science and Technology, 1(10):252–255, 2015. 15

[39] Pig, Apache: *Welcome to apache pig*, 2015. 15

[40] Fuad, Ammar, Alva Erwin e Heru Purnomo Ipung: *Processing performance on apache pig, apache hive and mysql cluster*. Em *Proceedings of International Conference on Information, Communication Technology and System (ICTS) 2014*, páginas 297–302. IEEE, 2014. 15

[41] Gruenheid, Anja, Edward Omiecinski e Leo Mark: *Query optimization using column statistics in hive*. Em *Proceedings of the 15th Symposium on International Database Engineering & Applications*, páginas 97–105. ACM, 2011. 16

[42] Thusoo, Ashish, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Suresh Anthony, Hao Liu, Pete Wyckoff e Raghotham Murthy: *Hive: a warehousing solution over a map-reduce framework*. Proceedings of the VLDB Endowment, 2(2):1626–1629, 2009. 16, 18

[43] Thusoo, Ashish, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Ning Zhang, Suresh Antony, Hao Liu e Raghotham Murthy: *Hive-a petabyte scale data warehouse using hadoop*. Em *2010 IEEE 26th international conference on data engineering (ICDE 2010)*, páginas 996–1005. IEEE, 2010. 16, 18

[44] Silva, Yasin N, Isadora Almeida e Michell Queiroz: *Sql: From traditional databases to big data*. Em *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, páginas 413–418. ACM, 2016. 16

[45] Mukherjee, Sourav: *The battle between nosql databases and rdbms*. Available at SSRN 3393986, 2019. 17

[46] Zafar, Rashid, Eiad Yafi, Megat F Zuhairi e Hassan Dao: *Big data: the nosql and rdbms review*. Em *2016 International Conference on Information and Communication Technology (ICICTM)*, páginas 120–126. IEEE, 2016. 17

[47] Li, Yishan e Sathiamoorthy Manoharan: *A performance comparison of sql and nosql databases*. Em *2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, páginas 15–19. IEEE, 2013. 17

[48] Khan, Samiya, Xiufeng Liu, Syed Arshad Ali e Mansaf Alam: *Storage solutions for big data systems: A qualitative study and comparison.* arXiv preprint arXiv:1904.11498, 2019. 17, 18

[49] Nayak, Ameya, Anil Poriya e Dikshay Poojary: *Type of nosql databases and its comparison with relational databases.* International Journal of Applied Information Systems, 5(4):16–19, 2013. 18

[50] Chang, Chih Hung, Fuu Cheng Jiang, Chao Tung Yang e Sheng Cang Chou: *On construction of a big data warehouse accessing platform for campus power usages.* Journal of Parallel and Distributed Computing, 2019. 18

[51] Sridhar, K. T.: *Modern column stores for big data processing.* Em Reddy, P. Krishna, Ashish Sureka, Sharma Chakravarthy e Subhash Bhalla (editores): *Big Data Analytics*, páginas 113–125, Cham, 2017. Springer International Publishing, ISBN 978-3-319-72413-3. 22

[52] `https://github.com/feliperezende86/PPCA.git`, acesso em 12/10/2019. 24