UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

CENTRO DE INFORMÁTICA

GRADUAÇÃO

BRUNO DE ASSIS PEREIRA

Um Processo para Desenvolvimento de um *Data Warehouse* em um cenário de *Big Data* utilizando a ferramenta *Hadoop*

TRABALHO DE GRADUAÇÃO

Recife
2016

BRUNO DE ASSIS PEREIRA

Um Processo para Desenvolvimento de um *Data Warehouse* em um cenário de *Big Data* utilizando a ferramenta *Hadoop*

Trabalho de conclusão de curso apresentado à disciplina de TG como parte dos requisitos necessários à obtenção do título de Bacharel em Ciências da Computação.

Orientador: Fernando da Fonseca de Souza

Recife
2016

Agradecimentos

Agradeço a todos que me acompanharam durante todo o desenvolvimento deste trabalho e também agradeço a todos que estiveram ao meu lado durante todo o curso de graduação seja me ensinando, trabalhando ou aprendendo comigo. Agradeço principalmente aos meus pais, Marisa Assis e Leonel Pereira, aos meus irmãos, Rodrigo Pereira e Gabriela Khuni, a minha namorada Thais Frota, e aos meus primos e amigos. Agradeço também ao meu orientador Fernando Fonseca por ser um excelente professor ao me guiar não só neste trabalho, mas no decorrer do curso.

Resumo

Dados são a codificação das informações e do conhecimento. Realizar um gerenciamento sobre essa informação envolve um processo de organizar, adquirir, armazenar, recuperar, e gerenciar esses dados. Estes são coletados por meio de diferentes processos e usados para auxiliar tomadas de decisão, de modo que aqueles que estão executando e consumindo os resultados do processo possam compreender toda a informação e atender aos seus diversos requisitos.

Um elemento que evoluiu para dar suporte ao processo de tomada de decisões foi o *Data Warehouse* (DW) (SUN et al., 2014). Por meio da capacidade de coletar, armazenar, e gerenciar dados, aplicando métodos tradicionais e estatísticos de medições para criar relatórios e plataformas de análise, o *Data Warehousing* se tornou um elemento chave no processo de tomada de decisões. Esta última pode se tornar ainda mais segura caso se possa analisar um número maior de dados.

O *Big Data* (MELOROSE; PERROY; CAREAS, 2015), conjunto de dados que se caracteriza pelo seu grande volume gerado em alta velocidade e em diversos tipos e formatos, tem se popularizado principalmente pela seguinte razão: as plataformas tecnológicas que surgiram junto com esse complexo e amplo conjunto de dados, provêm a capacidade de processar vários formatos e estruturas de dados sem ter que se preocupar com as restrições associadas aos sistemas de bancos de dados tradicionais. Como exemplo de ferramentas tem-se o *Hadoop* (JURNEY, 2013) que é uma arquitetura proposta como solução para o processamento de *Big Data* em uma plataforma mais barata com rápida escalabilidade e processamento paralelo.

Construir um *Data Warehouse* que contém *Big Data* possibilitará o desenvolvimento de análises sobre esses dados, e esta atividade trará conhecimento valioso que antes estava escondido ou não estaria disponível tão facilmente.

Neste trabalho serão explorados projetos que realizaram análises sobre *Big Data*. Ao fim dos estudos, a construção de um DW sobre um cenário de *Big Data* utilizando o *Hadoop* será realizada a fim de apresentar por meio da criação de relatórios contendo aplicação de métodos estatísticos, os conceitos que foram extraídos e o potencial que esse tipo de desenvolvimento tem.

Lista de ilustrações

Lista de quadros

Lista de tabelas

Sumário

1 Introdução

A Internet mudou e continua a causar grande impacto sobre o comportamento da sociedade, pois essa grande disseminadora de informações altera a forma como são gerenciados os negócios, a interação com os clientes, o compartilhamento de informações entre pessoas, a determinação do valor de um determinado produto, e, além disso, a Web proveu um novo canal para geração de lucro. Quando se deseja receber informação a respeito de novos produtos, ou até mesmo entender sobre as dificuldades já vivenciadas por outras pessoas ao utilizar serviços e produtos de uma determinada empresa, realiza-se uma busca na Internet.

No entanto, não são somente os consumidores que possuem interesse na informação disponibilizada no mundo das redes. Buscando entender melhor as necessidades de seus clientes, as pequenas e grandes empresas utilizam dos dados gerados pelos seus *stakeholders*[1] para realizar de forma mais precisa o processo de tomada de decisão.

A principal função da aplicação de análises sobre os dados é de realizar tomadas de decisão, e para auxiliar nessa tarefa criou-se o Data Warehouse (DW). O Data Warehousing foi desenvolvido para dar suporte ao processo de tomada de decisões por meio da sua capacidade de coletar, armazenar, e gerenciar dados, aplicando métodos tradicionais e estatísticos de medições para criar relatórios e plataformas de análise.

Atualmente, vive-se a era do *Big Data* na qual grandes volumes de dados de diversos formatos são gerados em alta velocidade. Construir um *Data Warehouse* contém *Big Data* possibilitará o desenvolvimento de análises sobre esses dados, e esta atividade trará conhecimento valioso que antes estava escondido ou não estaria disponível tão facilmente.

Uma das ferramentas mais utilizadas para resolver o processamento de *Big Data* é o *Hadoop* que possui uma rápida escalabilidade e utiliza de processamento paralelo. Além disso, essa ferramenta também faz uso de plataformas mais baratas tornando-a assim mais acessível.

1.1 Motivação

Essa grande disponibilidade dos dados tem levado a um crescimento no interesse sobre os métodos para extrair conhecimento e informação útil dos mesmos – *Data*

[1] pessoas, grupos ou organizações que tem interesse ou se importam com uma organização; pessoas, grupos ou organizações que podem afetar ou podem ser afetados por ações, objetivos e políticas de uma organização.

Science (Ciência dos dados) (PROVOST, 2013). O entendimento sobre as ferramentas que fazem parte dessa área possibilita que se possa fazer parte desse novo mundo rico em oportunidades para realizar grandes descobertas. Conseguir desenvolver uma plataforma com a qual o processo de tomada de decisão se torna ainda mais eficiente traz benefícios para diversas áreas – eduacação, saúde, aviação, desenvolvimento sustentável, construção, dentre outras.

1.2 Objetivos

Este trabalho tem como objetivo principal analisar e compreender mais sobre o processo de análises sobre Big Data, utilizando técnicas de DW.

1.2.1 Objetivos Específicos

Para alcançar o objetivo principal, são definidos os seguintes objetivos específicos:

- Desenvolver um processo de criação de um Data Warehouse que envolva Big Data;
- Construir o DW utilizando o Hadoop;
- Criar relatórios contendo resultados da aplicação de métodos estatísticos, os conceitos que foram extraídos; e
- Destacar o potencial existente na aplicação de técnicas de Data Warehousing sobre Big Data.

1.3 Metodologia

Com o objetivo de proporcionar um maior aprendizado sobre o tema de *Data Warehousing* aplicado a *Big Data* uma metodologia exploratória foi aplicada. Com este tipo de pesquisa, é possível se familiarizar com o fenômeno investigado de modo que o próximo passo da pesquisa possa ser melhor compreendido e com maior precisão (EDUCAÇÃO, 2013).

Para realizar este trabalho, inicialmente foi realizado um estudo sobre o estado da arte de modo a entender sobre o atual cenário *Big Data* e como aplicar conceitos de *Data Warehouse* ao mesmo. Com isso, foi encontrada a ferramenta *Hadoop*. Após realizar estudos sobre a ferramenta, com a utilização da máquina virtual da *Cloudera Distribution for Hadoop* (CDH)[2] e do *framework* de geração de dados da TPC (*Transac-*

[2] http://www.cloudera.com/downloads/cdh/5-8-2.html

tion Processing Performance Council) *Express Big Bench* (TPCx-BB)[3] foi construído um DW para os dados gerados com o *Hive*[4].Com esse DW foi possível extrair informações e apresentá-las de modo a mostrar o potencial de um DW sobre um possível cenário *Big Data*.

A fins de estudo e compreensão das tecnologias, foi utilizado o Hadoop no modo pseudo distribuído (*pseudo distributed mode*), pois este é recomendado para testes pelo fato de não ser necessário utilizar um *cluster* de máquinas e conseguir simular um cluster com apenas uma máquina. Dessa forma, para os objetivos levantados o modo escolhido já será suficiente para compreensão das ferramentas e apresentação de potencial.

1.4 Estrutura do Trabalho

Este trabalho será dividido em cinco (5) capítulos. Além do capítulo de Introdução, tem-se a seguinte estrutura: No Capítulo 2 é apresentado o estado atual das técnicas de *Data Warehousing* sobre *Big Data*, de modo a trazer conceitos e características importantes sobre os dois temas. Além disso, também é apresentado o *Hadoop*, uma das maiores tecnologias de *Big Data*, fazendo uma descrição das principais funcionalidades da ferramenta e dissertando sobre as tecnologias que compõem o ecossistema do *Hadoop*, dando ênfase ao *Hive*.

No Capítulo 3 são analisados trabalhos relacionados, trazendo uma descrição dos mesmos, com os resultados obtidos e técnicas aplicadas.

No Capítulo 4 é apresentado o processo e a especificação das ferramentas utilizadas para o desenvolvimento do DW, e é descrita a implementação realizada.

No Capítulo 5 são apresentadas as contribuições e limitações do trabalho, bem como sugestões de possíveis trabalhos futuros.

Por fim, são listadas as referências bibliográficas utilizadas.

[3] http://www.tpc.org/tpcx-bb/default.asp
[4] http://hive.apache.org/

2 Fundamentação Conceitual

Neste capítulo serão descritos conceitos importantes para a compreensão do trabalho. Dentre estes conceitos, serão abordados temas como *Big Data*, as tecnologias do *Big Data* – o *Hadoop* e o seu ecossistema –, e *Data Warehousing*.

2.1 Data Warehousing

Os primeiros conceitos sobre *Data Warehouse* surgiram com a necessidade de armazenar e analisar os dados do OLTP[1]. A habilidade de reunir transações, produtos, serviços, e localizações durante um período de tempo começou a prover capacidades interessantes para as companhias que nunca foi possível no mundo do OLTP, devido ao *design* desses sistemas e devido às limitações da sua infraestrutura em termos de escalabilidade (MELOROSE; PERROY; CAREAS, 2015).

A definição de Data Warehouse que é aceita como padrão pela indústria afirma que o DW é uma coleção de dados orientada por assunto[2], não volátil[3], integrada[4], e que varia no tempo[5], criada para dar suporte sobre o processo de tomadas de decisão (INMON; STRAUSS; NEUSHLOSS,).

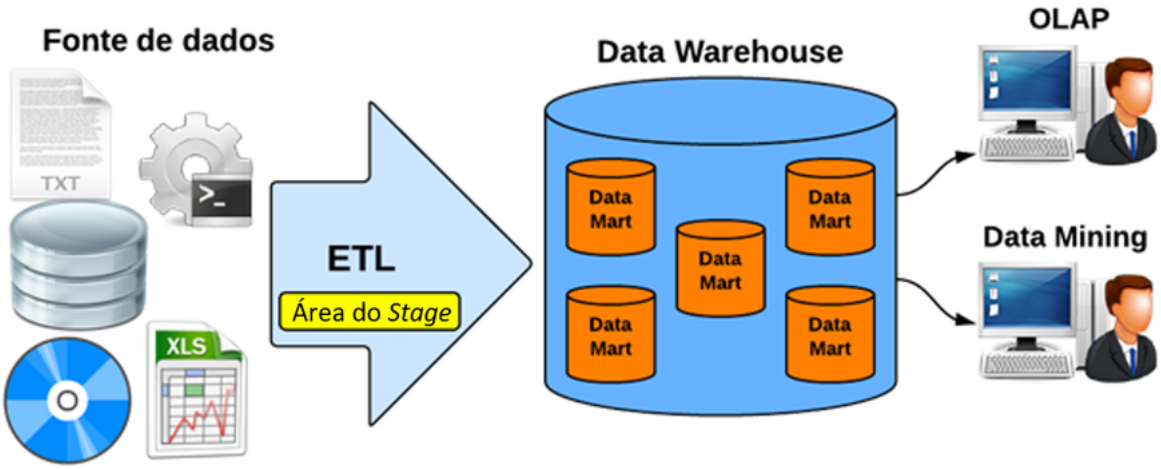
Os elementos básicos que compõem a arquitetura de um *Data Warehouse* (Figura 1) são as seguintes:

- Fonte de dados[6];
- ETL[7];
- *Staging Area*[8];

[1] Processamento de transações em tempo real; sistemas que se encarregam de registrar todas as transações contidas em uma determinada operação organizacional.
[2] Os sistemas transacionais são organizados de acordo com os principais assuntos da empresa em questão.
[3] No ambiente do DW, os dados, antes de serem carregados, são filtrados e limpos. Após esta etapa essas dados sofrem somente operações de consulta e exclusão, sem que possam ser alterados.
[4] A integração dos dados é realizada visando padronizar os dados dos diversos sistemas em uma única representação.
[5] Consiste na manutenção de um histórico de dados em relação ao período de tempo.
[6] Sistemas transacionais da empresa, pode ser composto por diversas formas de dados.
[7] Do Inglês *Extract*, *Trasnform and Load*, é o principal processo de condução dos dados até o armazenamento definitivo no DW. É responsável por todas as tarefas de extração, tratamento e limpeza de dados, e inserção na base do DW.
[8] Área de armazenamento intermediária situada dentro do processo de ETL. Auxilia a transição dos dados das origens para o destino final no DW.

- *Data Warehouse*[9];
- *Data Mart*[10];
- *OLAP*[11]; e
- *Data Mining*[12].

Figura 1 – Estrutura do DW



Fonte: https://corporate.canaltech.com.br/materia/business-intelligence/conhecendo-a-arquitetura-de-data-warehouse-19266/

Com o advento do *Data Warehouse*, as empresas podem atencipar a concorrência no que tange a gestão de informações que fornecem competitividade e inteligência no mercado, além de favorecerem o crescimento e alcance de resultados positivos na organização.

2.2 Data Warehousing e Big Data

As novas tecnologias que surgem estão muitas vezes ligadas à inovação, e essa última mudou a forma que se entra no mundo dos negócios e são providos serviços. Além disso, essas criações inovadoras alteram a forma de avaliação de valor associada a produtos e serviços, e as formas de lucro.

[9] Estrutura de armazenamento das informações decisivas. Apenas os dados com valor para a gestão corporativa estarão no DW.

[10] Estrutura similar ao DW, porém com uma proporção menor de informações.

[11] Do Inglês On-Line Analytical Processing, na arquitetura de um DW se refere às ferramentas com capacidade de análise em múltiplas perspectivas das informações armazenadas.

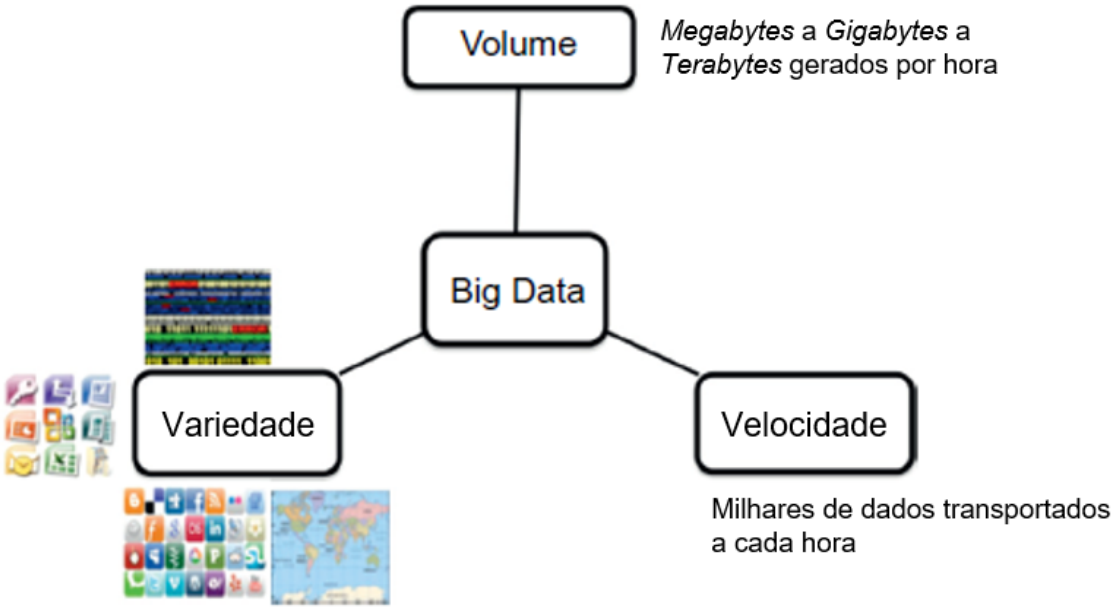
[12] A mineração de dados se refere às ferramentas com capacidade de descoberta de conhecimento relevante dentro do DW.

Essas tendências adicionaram complexidade em termos de processos, e ao mesmo tempo criaram a necessidade de adquirir dados necessários para esses processos, os quais podem prover conhecimentos essenciais para áreas nas quais nunca antes foi possível (MELOROSE; PERROY; CAREAS, 2015).

Com a vasta quantidade de dados agora disponível, as empresas em praticamente todas as áreas estão focadas em explorar essas informações visando obter vantagens competitivas no mercado (PROVOST, 2013).

Os desafios do crescimento de dados e oportunidades são definidos como tridimensionais (MELOROSE; PERROY; CAREAS, 2015). São caracterizados pelo volume[13] crescente, velocidade[14], e variedade[15] - os três V's (Figura 2). A indústria utiliza essa definição como um padrão para classificar *Big Data* (BETTER et al., 2013).

Figura 2 – Os três V's do Big Data



Fonte: Livro "Data Warehousing in the Age of Big Data"

Para conseguir trabalhar com esse tipo de dados, a arquitetura dos *Data Warehouses* precisou sofrer adaptações. A nova geração de DW é mais complexa em se tratando do desenvolvimento de uma arquitetura física, consistindo de várias tecnologias, e vai ser orientada a dados numa perspectiva de integração de todos os dados de

[13] quantidade de dados disponíveis e gerados continuamente.

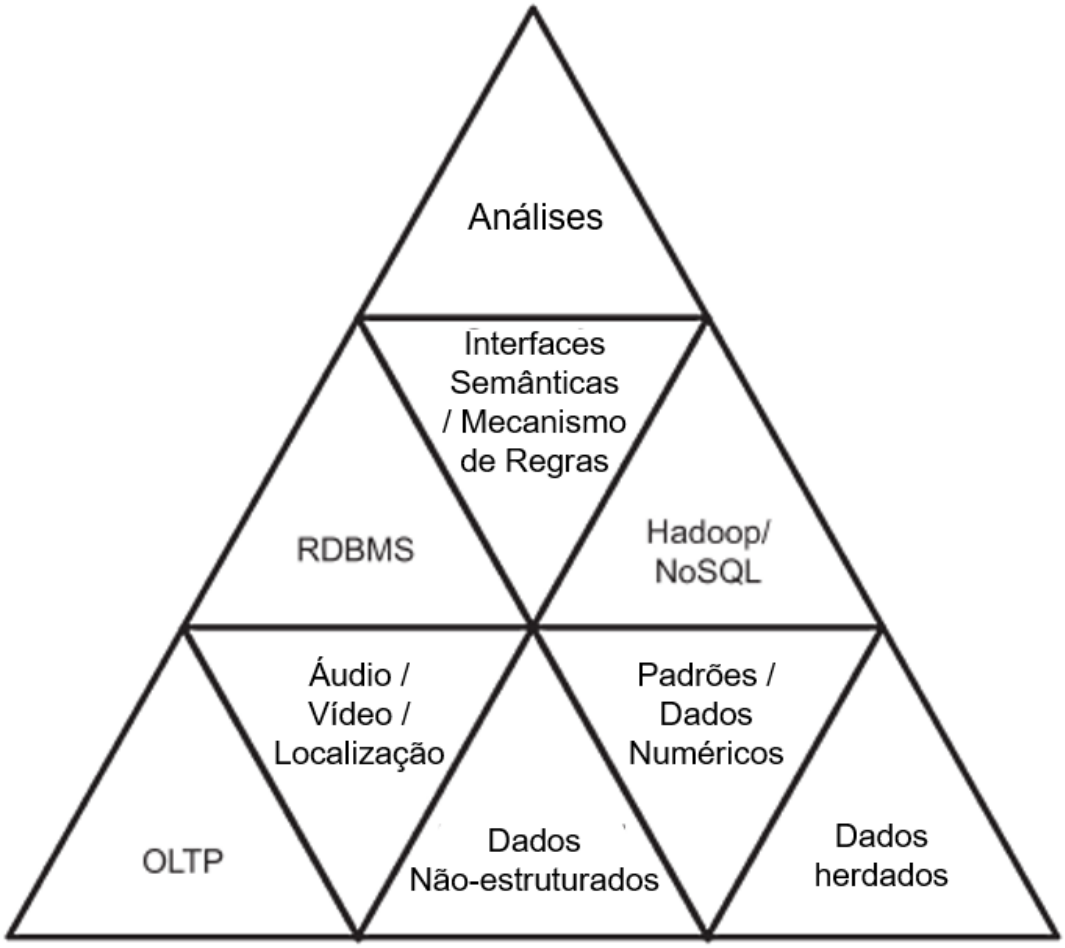
[14] velocidade com que os dados entram e saem, os dados são transmitidos de forma contínua e os conjuntos resultantes são úteis quando a aquisição e os atrasos de processamento levam pouco tempo.

[15] a extensão dos tipos de dados gerados, estes vem em diversos formatos que variam de e-mails a tweets, de rede social a dados de sensores. Não se tem um controle sobre o formato dos dados de entrada ou sobre a estrutura dos dados.

uma empresa que também terão *Big Data* como parte desse conjunto de informações. Além disso, essa geração será extremamente flexível e escalável em se tratando de uma perspectiva de arquitetura de dados (MELOROSE; PERROY; CAREAS, 2015).

Na Figura 3 podemos observar alguns dos componentes da próxima geração de *Data Warehouses*.

Figura 3 – Componentes do DW



Fonte: Livro “Data Warehousing in the Age of Big Data”

A primeira camada representa os dados, a segunda camada representa as tecnologias que vão ser utilizadas para a integração dos dados de vários tipos e fontes de dados, e a camada do topo representa a camada de análise que vai ser utilizada para orientar as necessidades da visualização da próxima geração de *bussiness intelligence*[16] e análises (MELOROSE; PERROY; CAREAS, 2015).

[16] Pode ser descrito como um conjunto de técnicas e ferramentas utilizados para a aquisição e transformação de dados em informação útil e signficativa para realização de análises de negócios.

A próxima geração das atividades referentes a arquitetura e processamento de dados precisará incluir algoritimos como, *Text Mining*[17], *Data Mining*, *Pattern Processing*[18], *Statistical Models*[19] e *Mathematical Models*[20], para resolver problemas específicos de processamento de dados, criação de relatórios e análises a partir do DW sobre cenários com *Big Data* (MELOROSE; PERROY; CAREAS, 2015).

2.3 Big Data

O termo Big Data identifica conjuntos de dados de tipos específicos, principalmente de dados não-estruturados, que povoam a camada de dados das aplicações científicas de computação e da Web. Esses dados têm características específicas em comum:

- Dados em larga escala, que se refere ao tamanho e a distribuição dos repositórios de dados;
- Problemas de escalabilidade, que se refere às capacidades das aplicações ao executar repositórios de dados enormes e de larga escala (i.e., *big data*, em resumo) para escalar rapidamente sobre entradas de tamanho crescente;
- Suporte avançado aos processos de ETL[21] a partir de dados de baixo nível, dados brutos, para torná-los uma informação de certa forma estruturada; e
- Criação e desenvolvimento sobre análises fáceis e interpretáveis sobre repositórios de *Big Data* de modo a obter inteligência e extrair conhecimento útil destes (CUZZOCREA; SACCÀ; ULLMAN, 2013).

Big Data também é o território no qual os tradicionais bancos de dados relacionais e sistemas de arquivos têm suas capacidades de processamento excedidas por altos volumes de transação, pela velocidade de resposta, e pela quantidade e ou variedade de dados. Essa gigantesca quantidade de dados cobre uma variedade de situações, todas elas com a palavra “mais” em comum – mais variedade, mais quantidade, mais usuários, mais velocidade, mais complexidade (BETTER et al., 2013).

[17] Processamento de texto baseado em regras de negócio definidas pelo usuário a fim de extrair dados que podem ser usados na classificação do texto para futuras explorações sobre os dados.

[18] Ramo do aprendizado de máquina que foca no reconhecimento de padrões e regularidades nos dados.

[19] Uma classe do modelo matemático, que inclui um conjunto de suposições sobre a geração de amostras de dados.

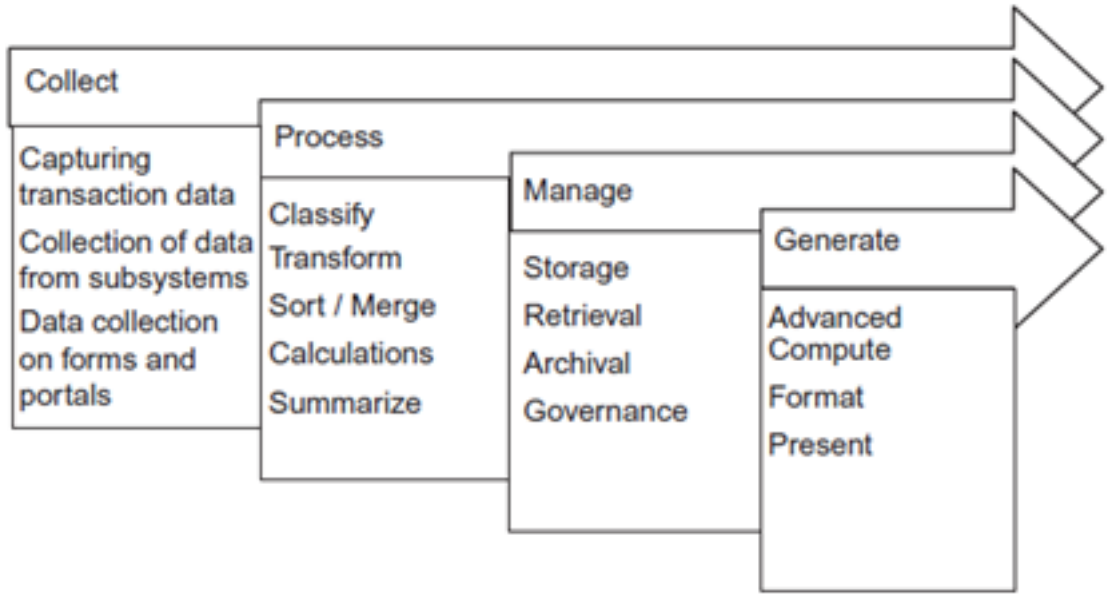
[20] Descrição de um sistema que utiliza conceitos e linguagens matemáticas.

[21] processo de extração de dados dos sistemas fonte para trazê-los para o data warehouse; Extrair, Transformar, e carregar (Load) dados.

A forma como se lida com o processamento de *Big Data* se dá por meio de processamento e armazenamento distribuídos, redes neurais, arquiteturas multiprocessadoras, e conceitos de orientação a objetos, combinados com técnicas de processamento de dados da Internet.

O processamento de dados pode ser definido como coleta, processamento, e gerenciamento de dados resultando em geração de informação para os usuários finais (Figura 4).

Figura 4 – Processamento de dados tradicional

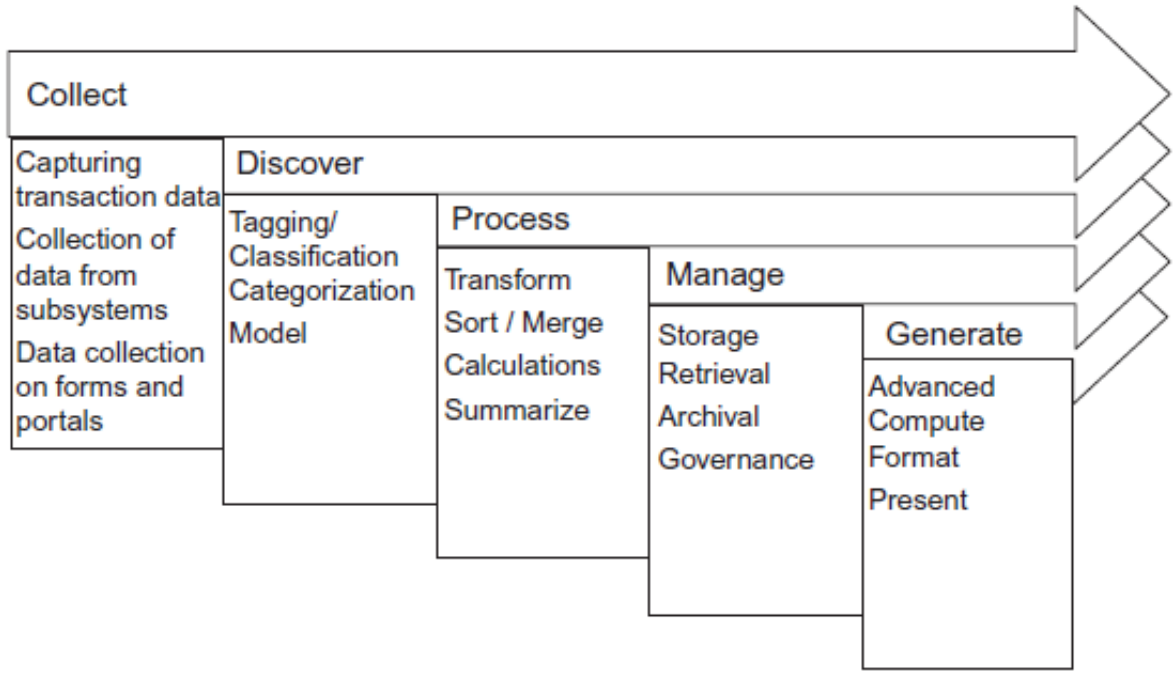


Fonte: Livro “Data Warehousing in the Age of Big Data”

No entanto, o processamento de dados em se tratando de *Big Data* ocorre de forma diferente (Figura 5). O primeiro estágio é o de coleta de dados. Nesse estágio, os dados são recebidos a partir de diferentes fontes e carregados para um sistema de arquivos chamado de *landing zone*[22] ou *landing area*. Os dados são distribuídos em subdiretórios de acordo com o seu tipo. No estágio seguinte, os dados são carregados com a aplicação de metadados (esse é o momento em que pela primeira vez se aplica uma estrutura para os dados) e estes são preparados para a transformação. Na etapa de transformação, os dados são modificados por meio da aplicação das regras de negócio e do processamento do conteúdo. Por fim, o conjunto de dados resultantes pode ser extraído para processamentos futuros que incluem análises, relatórios operacionais, integração com *data warehouse*, e visualizações (MELOROSE; PERROY; CAREAS, 2015).

[22] centro onde os dados vão ficar armazenados; sistema de arquivos.

Figura 5 – Processamento de dados *Big Data*



Fonte: Livro “Data Warehousing in the Age of Big Data”

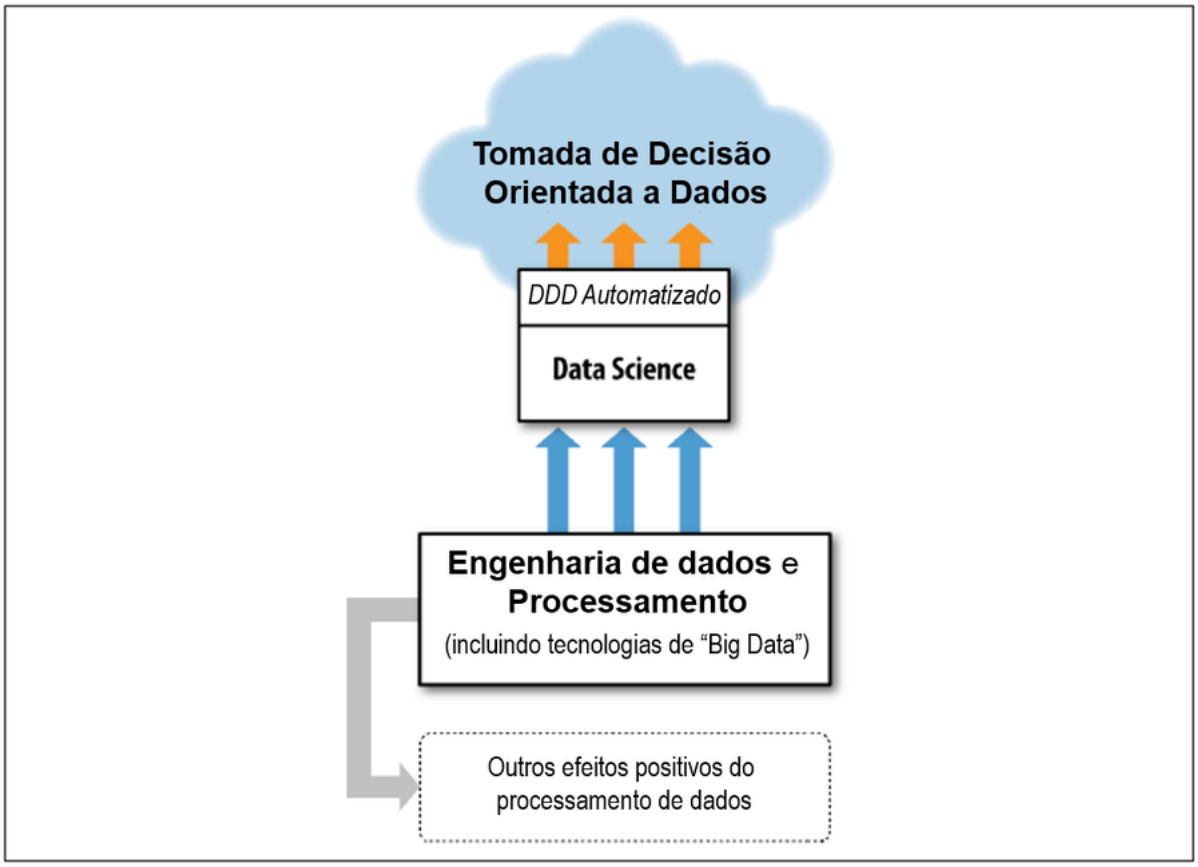
2.3.1 Tecnologias de *Big Data*

Os sistemas de processamento de dados tradicionais não têm capacidade para realizar tal tarefa sobre esses conjuntos de dados que são muito grandes, por esse motivo são necessárias novas tecnologias. Para realizar o processamento de *Big Data*, tecnologias como *Hadoop*[23], *Hive*[24], *HBase*[25], *MongoDB*[26] começaram a ser amplamente utilizadas (MELOROSE; PERROY; CAREAS, 2015).

Assim como as tecnologias tradicionais, as tecnologias de *Big Data* são utilizadas para várias tarefas, incluindo a engenharia de dados[27]. Mas, o campo em que as tecnologias de *Big Data* mais conhecidas são utilizadas é o de processamento de dados para dar suporte à técnicas de mineração de dados e outras atividades de *Data Science*, como pode ser visto na Figura 6 (PROVOST, 2013).

[23] http://hadoop.apache.org/
[24] https://hive.apache.org/
[25] https://hbase.apache.org/
[26] https://www.mongodb.com/
[27] área da Engenharia dedicada a processar e tratar dados para aplicações que utilizarão Big Data.

Figura 6 – Data Science no contexto de vários processos relacionados aos dados de uma organização



Fonte: Livro "Data Science for Business"

2.3.2 Hadoop

Uma plataforma confiável e escalável para armazenamento e análises. É executada em máquinas relativamente baratas e é *open source*[28], esse é o *Hadoop* (WHITE, 2010). Essa arquitetura proposta como solução para o processamento de *Big Data*, na sua primeira geração, consistia de um sistema distribuído de arquivos – *Hadoop Distributed File System* (HDFS) –, e de um *framework MapReduce*[29] junto com uma interface de coordernação e uma interface para ler e escrever no HDFS (MELOROSE; PERROY; CAREAS, 2015). Essa estrutura fornecia fundamentalmente um sistema de processamento em lotes.

No entanto, desde a sua origem, o *Hadoop* evoluiu além desse tipo de processamento.

O termo "*Hadoop*" algumas vezes é usado para se referir a um grande ecossistema de projetos, não somente o HDFS e o *MapReduce*. Muitos deles são parte

[28] software que possui o seu código fonte aberto e acessível, podendo ser adaptado para diferentes fins.

[29] um framework para processamento distribuído de grandes conjuntos de dados em clusters.

da *Apache Software Foundation*[30], que provê suporte para a comunidade de projetos *open-source*.

Alguns desses projetos são:

- AVRO[31];
- Cassandra[32];
- Chuckwa[33];
- Hbase[34];
- Hive[35];
- Mahout[36];
- Pig[37]; e
- ZooKeeper[38].

2.3.3 HDFS

Quando um conjunto de dados cresce além da capacidade de armazenamento de uma única máquina, se torna necessário particioná-lo em certo número de máquinas separadas. Sistemas de arquivo que gerenciam o armazenamento por meio de uma rede de computadores são chamados de sistemas de arquivos distribuídos.(WHITE, 2010)

O *Hadoop Distributed File System* (HDFS) é um sistema de arquivos distribuídos, escalável e altamente tolerante a falhas, desenvolvido para ser executado em *hardware* de baixo custo (MELOROSE; PERROY; CAREAS, 2015).

Algumas das características do HDFS são:

- Rendundâcia – o *hardware* pode vir a falhar e os recursos de infra-estrutura podem se esgotar para os processos, mas a redundância presente no HDFS pode lidar com essas situações;

[30] https://www.apache.org/
[31] um sistema de serialização de dados.
[32] um banco de dados *multimaster* e escalável.
[33] um sistema de coleção de dados para gerenciar grandes sistemas distribuídos.
[34] um banco de dados distribuído e escalável que suporta o armazenamento de dados estruturados para grandes tabelas.
[35] uma infraestrutura de *data warehouse* que provê a agregação de dados e consultas *ad hoc*.
[36] uma biblioteca de mineração de dados e aprendizado de máquina escalável.
[37] uma linguagem de fluxo de dados de alto nível e um *framework* de execução para computação em paralelo.
[38] um serviço de coodernação de alta performance para aplicações distribuídas.

- Escalabilidade – a escalabilidade linear na camada de armazenamento é necessária para utilizar o processamento paralelo da melhor maneira possível;
- Tolerância a falhas – a habilidade de automaticamente se recuperar da falha e completar o processamento do dado;
- Compatibilidade entre diferentes plataformas – a habilidade de integração entre várias plataformas; e
- Computação e armazenamento em um mesmo ambiente – os dados e a computação são colocados na mesma arquitetura removendo I/O[39] redundantes e acesso excessivo ao disco.

Um *cluster*[40] HDFS possui dois tipos de nós operando no padrão *master-worker* (mestre-trabalhador): um *namenode* (o mestre) e vários *datanodes* (trabalhadores).

O *namenode* gerencia o *namespace*[41] do sistema de arquivos (WHITE, 2010) por meio de operações de abrir, fechar, mover, nomear e renomear arquivos e diretórios (MELOROSE; PERROY; CAREAS, 2015). Além disso, o *namenode* também sabe quais são os *datanodes* que contém cada bloco de um determinado arquivo. Por sua vez, os *datanodes* armazenam e recuperam blocos quando requisitado, e eles, periodicamente, também reportam para o *namenode* uma lista com os blocos que eles estão armazenando.

Sem o *namenode*, o sistema de arquivos não poderia ser mais utilizado. Se a máquina que está executando o *namenode* parasse de funcionar, todos os arquivos presentes no sistema de arquivos seriam perdidos uma vez que não se teria como saber como reconstruir os arquivos que estão particionados pelos *datanodes*. Por esse motivo, o *Hadoop* utiliza de *backups* ou de *namenodes* secundários para evitar esse tipo de problema.

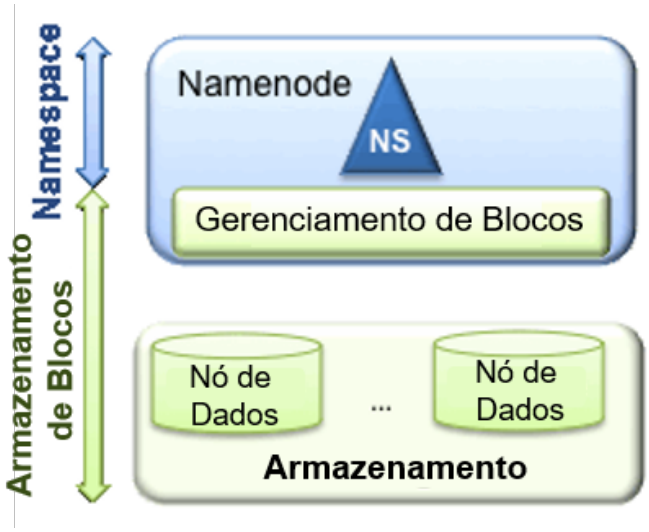
Na Figura 7 podemos observar a estrutura de um *cluster* HDFS com os seus nós divididos nas camadas de *namespace* e *block storage* (armazenamento de bloco).

[39] operações de entrada ou saída de dados por meio de algum código ou programa, para algum outro programa ou hardware.

[40] rede de computadores que trabalham em conjunto para que estes possam ser vistos como um sistema único.

[41] espaço que mantém a árvore do sistema de arquivos e os metadados para todos os arquivos e diretórios pertencentes a árvore.

Figura 7 – Estrutura do HDFS



Fonte: https://hadoop.apache.org/

2.3.4 MapReduce

MapReduce é um modelo de programação utilizado para processar conjuntos de dados extremamente grandes (MELOROSE; PERROY; CAREAS, 2015). Esse modelo funciona quebrando o processamento em duas fases: a fase de mapeamento e a fase de redução. Cada fase tem pares chave-valor como entrada e saída. Os tipos de cada um deles podem ser definidos pelo programador. O programador também especifica duas funções: a função Map e a função Reduce (WHITE, 2010).

A função *Map* recebe um conjunto de dados e o converte em um outro conjunto de dados, no qual elementos individuais são quebrados em tuplas.

A função *Reduce* utiliza a saída de um *Map* como sua entrada e combina essas tuplas de dados em um conjunto de tuplas menor.

2.3.5 Hive

Hive é um *framework* para aplicar técnicas de *data warehousing* que realiza o gerenciamento dos dados armazenados no HDFS e provê uma linguagem de consulta baseada em SQL[42].

O *Hive* surgiu a partir da necessidade de gerenciar e aprender sobre os grandes volumes de dados que o *Facebook*[43] estava produzindo todos os dias na sua rede social. Depois de tentar alguns sistemas diferentes, o time escolheu o *Hadoop* para o

[42] *Structured Query Language* (Linguagem de Consulta Estruturada); linguagem de pesquisa declarativa padrão para bancos de dados relacionais.

[43] https://www.facebook.com/

armazenamento e processamento, já que ele tem baixo custo e atende aos requisitos de escalabilidade.

Esse framework foi criado para que analistas com fortes habilidades em SQL pudessem executar consultas em grandes volumes de dados. Hoje, *Hive* é um projeto de sucesso da *Apache* usado por várias empresas como uma plataforma de processamento de dados escalável (WHITE, 2010).

Os principais objetivos do projeto *Hive* são:

- Construir um sistema para gerenciar e consultar dados usando técnicas estruturadas no *Hadoop*;
- Usar *MapReduce* nativo para execução nas camadas do HDFS e do *Hadoop*;
- Usar o HDFS para realizar o armazenamento dos dados do *Hive*;
- Armazenar metadados chave em um RDBMS[44];
- Estender interfaces SQL, uma ferramenta familiar de *data warehousing* em uso pelas empresa;
- Alta extensibilidade: tipos e funções definidas pelo usuário, formatos, e *scripts*;
- Aumentar a escalabilidade e o desempenho do *Hadoop*; e
- Interoperabilidade com outras plataformas.

Como mencionado anteriormente, essa ferramenta dá suporte a consultas expressadas em uma linguagem declarativa semelhante a SQL – *HiveQL*[45] –, que são compiladas em *MapReduce jobs* executadas no *Hadoop*. Além disso, o *Hive* também inclui um catálogo do sistema, a *metastore*, que contém esquemas e estatísticas, e é usado na exploração de dados e na otimização de consultas.

2.3.5.1 Arquitetura do Hive

A arquitetura do sistema *Hive* pode ser vista na Figura 8. Os principais componentes dessa arquitetura são:

- *Metastore* – armazena o catálogo do sistema e metadados sobre as tabelas, colunas e partições;

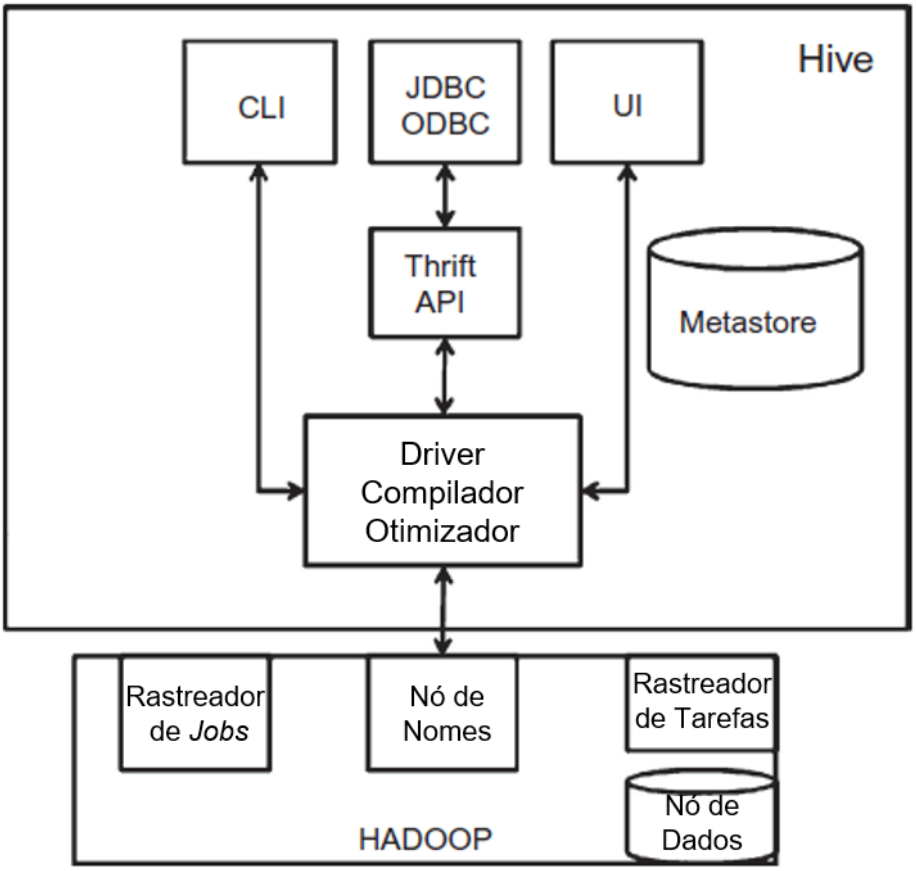
[44] Relational Database Management System; é um sistema de gerenciamento de banco de dados que é baseado num modelo relacional.

[45] https://cwiki.apache.org/confluence/display/Hive/LanguageManual

- *Driver* – mantém detalhes da sessão, identificadores dos processos, e estatísticas, e gerencia o ciclo de vida de uma declaração *HiveQL* enquanto ela se transporta pelo *Hive*;
- Compilador de consultas – compila *HiveQL* em tarefas de *Map* e *Reduce*;
- Mecanismo de execução – processa e executa as tarefas produzidas pelo compilador em ordem de dependência. O mecanismo de execução gerencia todas as interações entre o compilador e o *Hadoop*;
- *Thrift Server* – provê uma interface *thrift*[46], um servidor JDBC/ODBC[47], e uma API rica para integrar *Hive* com outras aplicações;
- CLI e *Web* UI – duas interfaces para o cliente. A interface de linha de comando (CLI) permite execuções em linha de comando e a interface *web* é um console de gerenciamento; e
- Interfaces – interfaces de extensibilidade que incluem as interfaces *SerDe*[48] e *ObjectInspector*[49], UDF[50], e UDAF[51] que permitem que o usuário defina suas próprias funções personalizadas.

[46] Camada de comunicação entre processos.
[47] http://docs.oracle.com/javase/7/docs/technotes/guides/jdbc/bridge.html
[48] https://cwiki.apache.org/confluence/display/Hive/SerDe
[49] https://hive.apache.org/javadocs/r0.10.0/api/org/apache/hadoop/hive/serde2/objectinspector/ObjectInspector.html
[50] https://cwiki.apache.org/confluence/display/Hive/LanguageManual+UDF
[51] função de agregação definida pelo usuário

Figura 8 – Estrutura do Hive



Fonte: Livro “Data Warehousing in the Age of Big Data”

2.3.6 Avro

Apache Avro é um sistema de serialização de dados de linguagem neutra. Esse sistema foi criado com o intuito de resolver a maior desvantagem do *Hadoop*: a falta de portabilidade de linguagens. Ter um formato de dados que pode ser processado por várias linguagens (C[52], C++[53], C#[54], Java[55], JavaScript[56], Perl[57], PHP[58], Python[59] e Ruby[60]) facilita o compartilhamento dos conjuntos de dados entre usuários (WHITE, 2010).

O Avro provê estruturas de dados ricas; um formato binário de dados rápido e compacto; e chamadas a procedimentos remotos (RPC[61] – *Remote Procedure Calls*)

[52] https://www.tutorialspoint.com/cprogramming/
[53] http://www.cplusplus.com/
[54] https://msdn.microsoft.com/en-us/library/67ef8sbd.aspx
[55] https://www.oracle.com/java/index.html
[56] https://www.javascript.com/
[57] https://www.perl.org/
[58] http://php.net/
[59] https://www.python.org/
[60] https://www.ruby-lang.org/pt/
[61] Tecnologia utilizada para a implementação do modelo cliente-servidor de computação distribuída.

(APACHE,).

2.3.7 Cassandra

Apache Cassandra é um sistema de banco de dados que provê escalabilidade linear[62] e alta disponibilidade sem comprometer o desempenho. Além disso, também é tolerante a falhas em *hardwares* de baixo custo (APACHE,).

Normalmente, refere-se ao Cassandra como um sistema de banco de dados de arquitetura híbrida, uma vez que este combina um modelo de dados orientado a colunas do *BigTable*[63] com o *MapReduce* do *Hadoop*, e implementa os padrões do *Dynamo*[64] (MELOROSE; PERROY; CAREAS, 2015).

2.3.8 Chuckwa

Apache Chuckwa é um sistema de coleção de dados de código aberto utilizado para monitorar grandes sistemas distribuídos. Esse sistema é construído sobre o HDFS e o framework MapReduce, e ele herda a escalabilidade e robustez do Hadoop (APACHE,).

O Chuckwa também inclui um conjunto de ferramentas flexível utilizado para exibir, monitorar, e analisar resultados para extrair o máximo de informações úteis para o usuário dos dados disponíveis coletados (MELOROSE; PERROY; CAREAS, 2015).

2.3.9 HBase

Apache HBase é um sistema de banco de dados distribuído e escalável do Hadoop. Normalmente utilizado quando é necessário realizar acesso randômico de leitura ou escrita em tempo real ao Big Data (APACHE,).

O HBase também é orientado à colunas e é construído sobre o HDFS (WHITE, 2010). Além disso, esse banco de dados também provê escalabilidade e desempenho ilimitado, e possui certas características de um sistema de banco de dados ACID[65]. O Apache HBase é classificado como um banco de dados *NoSQL* devido a sua arquitetura e o seu *design* serem alinhados com *Base* (*Being Available and Same Everywhere* – estar disponível e ser o mesmo em qualquer lugar) (MELOROSE; PERROY; CAREAS, 2015).

Uma chamada de um procedimento remoto é iniciada pelo cliente enviando uma mensagem para um servidor remoto para que este execute um procedimento específico.

62 Ao escalonar a aplicação, esta não apresentará falhas nem perdas de recursos, dessa forma o fator de escalabilidade permanecerá constante (THARAKAN, 2007)

63 https://cloud.google.com/bigtable/

64 aws.amazon.com/dynamodb

65 Atomicidade, Consistência, Isolamento e Durabilidade são um conjunto de propriedades das transações em bancos de dados

2.3.10 Mahout

Apache Mahout é um projeto cujo objetivo é a construção de um ambiente para a criação rápida de aplicações escaláveis de aprendizado de máquina de elevado desempenho (APACHE,).

O Mahout consiste de uma coleção algorítmos escaláveis de aprendizado de máquina que são executados no *Hadoop* (SITTO; PRESSER, 2015).

2.3.11 Pig

Apache Pig aumenta o nível de abstração para o processamento de grandes conjuntos de dados. Com o Pig, as estruturas de dados são mais ricas, tipicamente sendo multivaloradas e aninhadas, e as transformações que o usuário pode aplicar sobre os dados permitem a extração de informações úteis dos mesmos. (WHITE, 2010).

Em tempo de compilação, a atual infraestrutura do Pig consiste de um compilador que produz sequências de programas *MapReduce*. A arquitetura da linguagem do Pig é uma plataforma de linguagem textual chamada de *Pig Latin*, na qual os objetivos de *design* se basearam no requisito de lidar com grandes processamentos de dados com complexidade mínima (MELOROSE; PERROY; CAREAS, 2015).

2.3.12 ZooKeeper

Apache ZooKeeper é uma aplicação utilizada para desenvolver e manter um servidor de código aberto que permita uma coordernação distribuída altamente confiável (APACHE,).

Em um ambiente controlado como o do RDBMS, as tarefas são geradas de forma controlada e a coordenação precisa somente garantir um gerenciamento da rede sem perda de dados e realizar uma verificação sobre o estado dos nós de um sistema distribuído. No caso do *Hadoop*, o menor volumde de dados começa com vários *terabytes* e o dado está distribuído em arquivos pertencentes a diferentes nós. Manter as consultas dos usuários e as tarefas associadas requer um coordenador que seja flexível e escalável, assim como o ZooKeeper (MELOROSE; PERROY; CAREAS, 2015).

2.4 Comentários Finais

Os estudos realizados sobre os conceitos de *Big Data*, as tecnologias do *Big Data*, e *Data Warehousing* trouxeram conhecimentos valiosos para uma maior compreensão sobre o estado da arte. Com esse conhecimento, pode-se entender melhor à respeito de aplicações sobre essa área, e possibilita a apresentação de um processo de desenvolvimento de um DW para *Big Data*.

No próximo Capítulo (3) serão apresentados trabalhos relacionados que realizaram a aplicação de técnincas de análise sobre *Big Data* em diversas áreas e cenários. Com isso, será possível observar algumas das possibilidades de desenvolvimento que este tipo de abordagem traz.

3 Trabalhos Relacionados

Neste capítulo serão abordados trabalhos relacionados que realizaram a aplicação de técnicas de análise sobre *Big Data*. Para cada trabalho, será apresentado o problema encontrado, a solução aplicada e as limitações, conforme descrito no Quadro 1.

Quadro 1 – Critério de Análise e Apresentação

Atividade	**Descrição**
Seleção	• Selecionar trabalhos que apresentem soluções para problemas relacionados a análises sobre *Big Data*; • Os trabalhos selecionados precisam definir o problema abordado, as tecnologias aplicadas para a solução, e possíveis limitações ou trabalhos futuros; • Não existe uma filtragem em relação à temática abordada; • Preferência por trabalhos que tenham utilizado algum dos projetos do Apache ou a Cloudera.
Descrição	• Descrever cada um dos trabalhos relacionados apresentando o problema, a solução e as limitações dos mesmos.

Fonte: Elaborado pelo Autor (2016)

3.1 Trabalho 1 – Análises sobre Imagens Médicas (ZHANG et al., 2016)

3.1.1 Problema

Análises sobre *Big Data* estão chegando à área de saúde. O surgimento das tecnologias da *Internet of Things*[1], como rastreadores para medição dos sinais vitais, aplicativos de saúde para *smartphones*, estão facilitando o processo de coleta contínua de milhões de usuários sobre os seus dados pessoais de saúde. Uma única instituição pode armazenar cerca de dez a cem *terabytes* de dados sobre imagens médicas.

Além do problema do volume ser claro, também é possível encontrar a necessidade da velocidade pois computações sofisticadas são aplicadas sobre imagens que

[1] Internet das Coisas; é a rede de comunicação entre dispositivos, veículos e outros dispositivos conectados.

surgem a altas taxas de velocidade, e a variedade nos dados também surge quando imagens são posteriormente combinadas com registros estruturados de pacientes.

Análises sobre imagens médicas podem depender de uma *pipeline*[2] altamente especializado para comparar e agregar imagens em 3D de forma única.

Esse trabalho busca entender sobre os comportamentos desse tipo de análise a partir de uma perspectiva de sistemas distribuídos.

3.1.2 Solução

A execução do *pipeline* é realizada em paralelo em um *cluster* sobre uma camada de armazenamento distribuída que dá suporte ao compartilhamento de dados entre os nós e entre os estágios do *pipeline*.

Para realizar a execução paralela, foram consideradas duas tecnologias: LSF[3] e *Spark*[4].

Para a camada de armazenamento foi utilizado o IBM *Spectrum Scale*[5].

3.1.3 Limitações

Após realizar experimentos, foi constatado que realizar tarefas menores faria com que menos memória fosse requisitada, pois a memória foi um dos gargalos encontrados. No entanto, para realizar essa redução, seria necessário aplicar alterações sobre os algoritmos de registro de imagens utilizados e essa tarefa é considerada não trivial.

Além disso, notou-se a necessidade de reescrever o *pipeline* de modo a utilizar o *Spark* de forma mais eficiente. RDDs[6] precisam ser criados para as imagens para que se possa beneficiar do paralelismo oferecido pelo *Spark*.

3.2 Trabalho 2 – Transformando o cenário do Mercado com o Cloudera

3.2.1 Problema

A EMS (*Experian Marketing Services*) ajuda comerciantes a se conectarem com clientes através da comunicação utilizando uma variedade de canais, guiados por análises avançadas em um extenso banco de dados geográficos, demográficos e de estilo de vida.

---

[2] Um conjunto de elementos de processamento de dados conectados em série, onde a saída de um elemento é entrada do seguinte.

[3] http://www-03.ibm.com/systems/spectrum-computing/products/lsf/

[4] http://spark.apache.org/

[5] http://www-03.ibm.com/systems/storage/spectrum/scale/

[6] *Resilient Distributed Dataset*; Uma coleção de elementos tolerante a falhas que possam ser operados em paralelo.

A EMS construiu o seu negócio sobre um efetivo processo de coleta, análise, e uso dos dados. No entanto, os comerciantes passaram a exigir mais, pois estes queriam observar o comportamento dos seus respectivos usuários e dar respostas em tempo real. Para isso, seria necessário analisar as últimas compras dos usuários, os padrões de buscas online feitas por eles, e as atividades desses consumidores nas redes sociais.

Mas, a grande quantidade de dados disponíveis nesses canais digitais requer uma infraestrutura tecnológica que possa acomodar processamento rápido, armazenamento em larga escala, e análise flexível sobre dados multiestruturados.

3.2.2 Solução

Para definir uma nova infraestrutura foram utilizados dois critérios:

- A infraestrutura deve ter a capacidade de processar tanto dados em tempo real quanto em lote; e
- A infraestrutura deve ter a escalabilidade como uma de suas características para que seja possível acomodar grandes e crescentes volumes de dados.

Tendo esses critérios definidos, a *Cloudera Distribution for Hadoop* (CDH) foi escolhida.

Depois de explorar o *Hadoop*, a EMS criou um mecanismo de acoplamento, chamado de *Cross-Channel Identity Resolution* (CCIR), usado para manter um repositório persistente dos *touchpoints*[7] dos clientes. O CCIR é executado no HBase, que resolve os problemas de persistência, redundância, e tem a habilidade de automaticamente redistribuir os dados. Além disso, o HBase permite tanto o processamento de dados em tempo real quanto em lote.

A Experian realiza o abastecimento de dados no CDH-CCIR usando *scripts* customizados de extração, transformação e carga (ETL) a partir de estruturas internas e de bancos de dados relacionais, incluindo IBM DB2[8], Oracle[9], SQL Server[10]. A Experian também usa o *Hive* e o *Pig* para realizar consultas e análises.

3.2.3 Limitações

Para a aplicação da nova estrutura envolvendo a CDH foi necessário envolver arquitetos especialistas na área. Além disso, também foi necessário um treinamento

[7] Qualquer forma de interação entre um cliente e um negócio, seja ele pessoalmente, através de uma página *web*, um aplicativo ou qualquer outra forma de comunicação.

[8] http://www.ibm.com/analytics/us/en/technology/db2/

[9] https://www.oracle.com/

[10] https://www.microsoft.com/pt-br/server-cloud/products/sql-server/overview.aspx

sobre *Hadoop* e *HBase* antes de começar a desenvolver a aplicação.

Com a infraestrutura tecnológica que a EMS possuía, a empresa não conseguia acomodar um rápido processamento, um armazenamento em larga escala, e realizar análises flexíveis sobre dados multiestruturados.

Após a implementação do *Hadoop* através da CDH, esse trabalho não apresenta limitações ou trabalhos futuros, além disso este trabalho não explora outras possibilidades tecnológicas que poderiam resolver o problema.

3.3 Trabalho 3 – Aprimorando a conectividade de assistências médica com *Big Data*

3.3.1 Problema

Empresas de assistências médicas precisam armazenar os dados sobre saúde por longos períodos. Uma empresa de TI na área de saúde estabeleceu uma política de que se deveria manter um histórico de sete anos dos dados acerca de reclamações e remissões, no entanto os seus sistemas de banco de dados tiveram problemas ao lidar com essa nova política enquanto processava milhares de reclamações todos os dias.

A quantidade de dados era muito grande para que os bancos conseguissem controlá-los. Eles estavam trabalhando acima do limite e estavam sobrecarregados, e isso começou a criar problemas com o processamento de produção em tempo real.

3.3.2 Solução

Utilizando o CDH, a empresa pôde se beneficiar das vantagens oferecidas pelo *Hadoop* que é capaz de separar o *Big Data* dos dados de processamento transacional, e permite um processamento mais suave de informação. Basicamente, ele permite reduzir a sobrecarga presente nos bancos de dados.

Hoje a empresa de TI usa o *Flume*[11] para mover os dados dos seus sistemas para o seu *cluster* CDH todos os dias. A empresa carrega os dados do CDH para um banco de dados da *Oracle* para fins de gerar faturamento, e esse carregamento ocorre uma ou duas vezes por dia utilizando o *Sqoop*[12].

3.3.3 Limitações

Para poder fazer uso das tecnologias da *Cloudera*, foi necessário avaliar a arquitetura inicial do projeto para que se pudesse tirar maior proveito das capacidades

[11] http://flume.apache.org/
[12] http://sqoop.apache.org/

do *Hadoop*. Além disso, também foi necessário realizar um treinamento sobre as ferramentas para que a solução pudesse ser aplicada.

O trabalho não apresenta a possibilidade de utilização de outras tecnologias que poderiam resolver o problema abordado. Devido a necessidade de uma solução rápida, esse trabalho se limita a algumas tecnologias do ecossistema do *Hadoop*, e não busca apresentar tecnologias alternativas.

3.4 Trabalho 4 – Um Sistema Analítico de *Big Data* para analisar a percepção de segurança dos cidadãos (CAMARGO et al., 2016)

3.4.1 Problema

A percepção de segurança (PoS – *Perception of Security*) se refere à avaliação subjetiva dos riscos da população. O PoS é um ponto muito importante para os governantes e os responsáveis pela tomada de decisão ao se planejar a segurança das cidades. Uma prática comum para medir PoS é baseada em pesquisas de opinião, na qual os cidadãos respondem um conjunto de perguntas relacionadas à segurança.

Por um lado, a percepção de segurança depende bastante de muitos aspectos como psicológico, cultural e social, então essa percepção pode mudar dinamicamente quando um desses aspectos muda. Por outro lado, PoS é muito subjetiva já que depende da percepção qualitativa dos indivíduos.

Os governantes precisam ter acesso à informação em tempo real para tomar decisões baseadas em dados atualizados que permitem que os representantes das cidades sejam bem mais precisos. Redes sociais são uma nova forma de comunicação que possuem duas características importantes: informação em tempo real e interatividade.

Nesse trabalho é apresentado um sistema de percepção de segurança que pode ser usado pelos governos a fim de realizar análises sobre dados obtidos a partir de redes sociais com o objetivo de entender, analisar e prevenir eventos de segurança de forma mais sábia.

3.4.2 Solução

A solução é composta por duas camadas. A primeira é a camada de processamento de dados, e a segunda é a camada analítica de dados.

Na primeira camada há cinco componentes:

- *Crawler*[13];

[13] Programa de computador que navega pela *internet* de uma forma metódica e automatizada.

- Componente de pré-processamento;
- Componente de detecção de segurança;
- Componente de *tagging* automático; e
- Componente de indexação relacionada à segurança.

O *Crawler* permite que os dados sejam coletados a partir do *Twitter*[14]. Uma vez que os dados foram coletados, o componente de pré-processamento extrai campos como usuário, texto do *tweet*, local, e latitude/longitude.

No passo seguinte, os dados são filtrados a fim de detectar somente aqueles que estão relacionados à segurança. O componente de detecção de segurança possui um conjunto de palavras definidas por um especialista na área que são decisivas no processo de filtragem. Uma abordagem de aprendizado de máquina é aplicado de modo a obter um conjunto de dados devidamente classificado.

Usando a biblioteca do *Apache Lucene*[15], são aplicadas técnicas de indexação baseadas na recuperação de informação que foram construídas como um mecanismo para indexar o significado dos termos relacionados à segurança.

Armazenou-se todos os *tweets* relacionados à segurança em um sistema de banco de dados *NoSQL*[16] chamado de *MongoDB*[17] e em um conjunto de tabelas *Google Fusion*[18]. O *MongoDB* permite uma busca eficiente sobre diferentes campos JSON[19], por esse motivo torna-se possível realizar consultas por data, texto, usuário, e latitude/longitude em cada *tweet*. As tabelas *Google Fusion* permitem a visualização dos *tweets* em um mapa e construção de um mapa de calor.

Com isso, foi possível aplicar análises utilizando mapas e realizando consultas nas bases de modo a obter resultados a respeito da percepção de segurança dos cidadãos.

3.4.3 Limitações

A quantidade de dados obtida para realizar o trabalho não era considerável, além disso, o escopo do projeto não englobava um grande número de cidades. O trabalho não discorre sobre possíveis problemas com escalabilidade e também não aponta trabalhos futuros.

[14] https://twitter.com/
[15] https://lucene.apache.org
[16] https://aws.amazon.com/pt/nosql/
[17] https://www.mongodb.com/
[18] https://www.google.com/fusiontables
[19] http://www.json.org/

3.5 Trabalho 5 – Aplicando transformações centradas no cliente

3.5.1 Problema

A coleta e o uso de dados de forma eficiente se tornou muito importante para a Nokia[20], uma vez que a empresa busca aprimorar a sua habilidade de entender e prover melhores experiências para os seus usuários com os seus celulares e outros produtos de localização. A companhia utiliza de processamento de dados e análises complexas para construir mapas com modelos de tráfego preditivo e elevação de camadas, para conseguir informações sobre pontos de interesse pelo mundo, para entender sobre a qualidade de seus aparelhos, dentre outras atividades.

Antes da utilização do *Hadoop*, vários grupos da Nokia estavam construindo armazéns para acomodar informações individualmente. Não levou muito tempo para que a empresa percebesse que aquilo poderia proporcionar grande valor caso os seus armazéns viessem a ser integrados, permitindo que os dados globalmente captados pudessem ser cruzados de modo a obter uma versão da verdade única[21] e compreensível.

3.5.2 Solução

Para crescer e dar suporte ao uso extensivo do *Big Data*, a Nokia conta com um ecossistema tecnológico que inclui um *Teradata*[22] *EDW*[23], vários *data marts* da *Oracle* e do *MySQL*[24], tecnologias de visualização, e, o principal, o *Hadoop*.

A Nokia possui mais de 100 *terabytes* (TB) de dados estruturados no *Teradata* e *petabytes* (PB) de dados multiestruturados no *HDFS*. O *cluster* centralizado do *Hadoop* que fica no núcleo da infraestrutura da Empresa contém 0.5 PB de dados.

Os *data warehouses* e *data marts* da Nokia continuamente transmitem dados multiestruturados para um ambiente *Hadoop*, permitindo que todos os funcionários da empresa tenham acesso aos dados. A Compania também faz uso do *Sqoop* para mover os dados do *HDFS* para o *Oracle* ou *Teradata*.

3.5.3 Limitações

Inicialmente, a Nokia precisou implementar uma nova infraestrutura que pudesse ter diariamente um fluxo de *terabytes* de dados não estruturados que surgiam a partir

[20] http://www.nokia.com/

[21] Do inglês *Single version of* truth, é um conceito técnico utilizado para descrever o *data warehousing* ideal onde todos os dados de uma organização está armazenado de forma consistente e não redundante.

[22] http://br.teradata.com/?LangType=1046&LangSelect=true

[23] *Enterprise Data Warehouse*; base de dados unificada que armazena todas as informações de uma empresa e torna as mesmas acessíveis para toda a organização.

[24] https://www.mysql.com/

dos aparelhos telefônicos, dos seus serviços, arquivos de *log*, e outras fontes.

Realizar um processamento complexo de dados na escala de *petabytes* usando bancos de dados relacionais era extremamente caro e iria limitar os tipos de dados que poderiam ser consumidos.

Utilizar o *Hadoop* iria facilitar e reduzir bastante os custos. No entanto, para colocar o novo sistema em produção é necessário entender sobre as ferramentas e plataformas que dão suporte ao processo, e estas, por sua vez, tornam a implementação bastante complexa. Por esse motivo, a Nokia passou a utilizar o CDH, que simplifica bastante todo o processo. No entanto, ainda assim, vários meses se passaram até que a empresa decidisse padronizar o uso do *Hadoop*, pois o número de especialistas na ferramenta era muito limitado.

3.6 Comentários Finais

Após analisar os cinco trabalhos aqui apresentados, pode-se notar o potencial que a análise sobre *Big Data* possui sobre as mais diversas áreas, e se tornar ciente sobre eventuais problemas relacionados ao desenvolvimento de soluções para esse tipo de análise. O Quadro 2 apresenta um sumário sobre os trabalhos analisados.

Quadro 2 – Sumário de Análises

Trabalho	**Tópico**	**Descrição**
Trabalho 1	Problema	• Aplicação de análises sobre imagens médicas
	Tecnologias Utilizadas	• LSF • Spark • IBM Spectrum Scale
	Limitações	• Grande requisição sobre memória; • Alterações sobre algoritmos de registro de imagens são muito complexas; • Reescrever *pipeline* de modo a otimizar o uso do Spark; e • Criar RDDs para se beneficiar do paralelismo oferecido pelo Spark.
Trabalho 2	Problema	• Criação de uma infraestrutura tecnológica que possa acomodar processamento rápido, armazenamento em larga escala, e análise flexível sobre dados multiestruturados
	Tecnologias Utilizadas	• CDH • Hadoop • HBase • IBM DB2 • Oracle • SQL Server • Hive • Pig
	Limitações	• Não apresenta limitações ou trabalhos futuros; • O trabalho não descreve outras possibilidades para solucionar o problema.
Trabalho 3	Problema	• Aprimorar a conectividade de assistências médicas
	Tecnologias Utilizadas	• CDH • Hadoop • Flume • Oracle • Sqoop
	Limitações	• Devido a necessidade de uma solução rápida, esse trabalho se limita a algumas tecnologias do ecossistema do *Hadoop*, e não busca apresentar tecnologias alternativas.
Trabalho 4	Problema	• Analisar a percepção de segurança dos cidadãos
	Tecnologias Utilizadas	• Apache Lucene • MongoDB • Google Fusion
	Limitações	• Quantidade de dados pequena; • O trabalho não discorre sobre possíveis problemas com escalabilidade; e • O trabalho não aponta próximos passos.
Trabalho 5	Problema	• Aprimorar habilidade de entender e prover melhores experiências para os usuários
	Tecnologias Utilizadas	• Teradata EDW • Oracle • MySQL • Hadoop • Sqoop • HDFS
	Limitações	• Utilização do Hadoop não ocorreu de imediato devido à complexidade de uso da ferramenta; • Foi necessário a utilização da CDH para implantar a nova infraestrutura; • Por não ter muitos especialistas na área, o tempo de implantação tornou-se mais longo.

Fonte: Elaborado pelo Autor (2016)

Além disso, também se pode observar quais tecnologias foram utilizadas e entender o porque destas terem sido selecionadas.

No próximo capítulo será apresentado um processo para o desenvolvimento de um *Data Warehouse* para um cenário de *Big Data* utilizando a ferramenta *Hadoop*.

4 Arquitetura e Processo de Desenvolvimento do DW

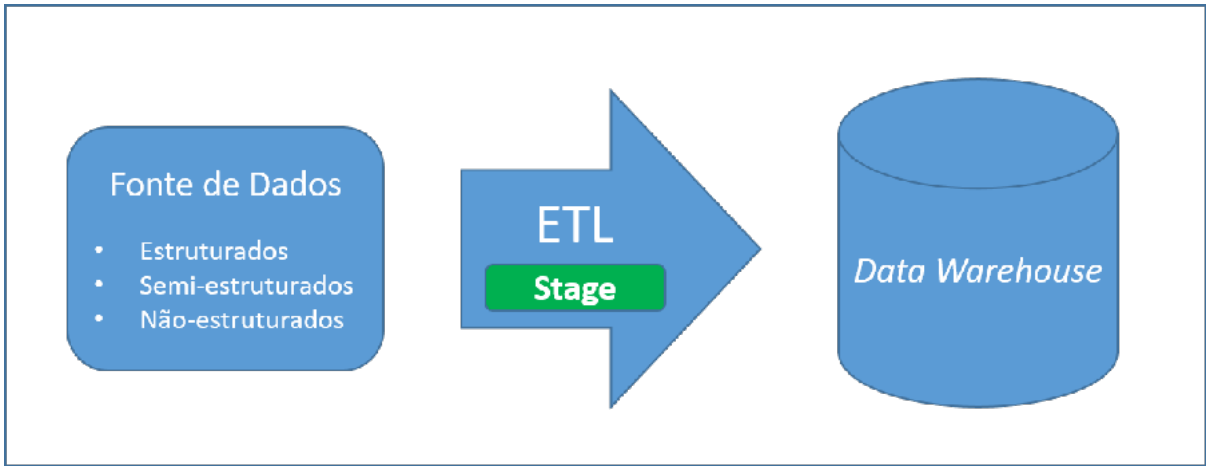
Neste capítulo serão apresentados a arquitetura e o processo de desenvolvimento do DW. Além disso, também será descrita a especificação das ferramentas utilizadas para a construção do *data warehouse*.

4.1 Arquitetura do DW

A arquitetura do DW (Figura 9) desenvolvida neste trabalho consiste dos seguintes elementos:

- Fonte de Dados
- ETL
- *Data* Warehouse

Figura 9 – Arquitetura do Data Warehousing



Fonte: Elaborada pelo Autor (2016)

A fonte de dados abrange todas as informações que serão obtidas a partir da origem. Estes dados são classificados em estruturados, semiestruturados e não-estruturados. Durante o processo de ETL, os dados então são classificados e categorizados de modo a dar significado para os mesmos. Por fim, com as devidas transformações aplicadas e com os dados limpos, é possível carregá-los no DW.

4.2 Especificação das Ferramentas

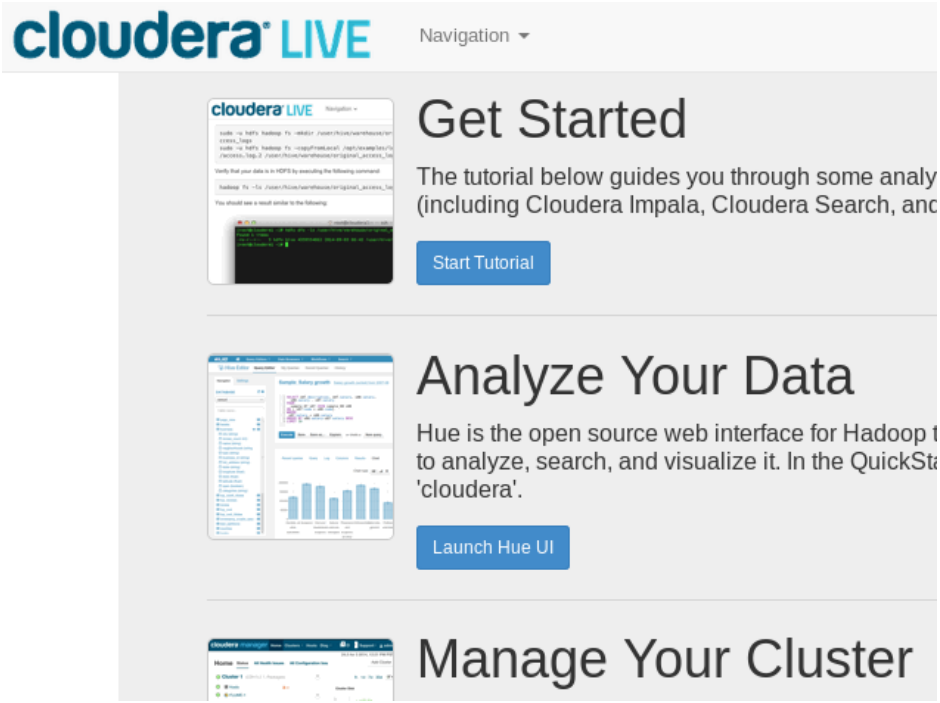
Para a construção do DW foi utilizada a plataforma *Hadoop*, o *framework Hive*, a máquina virtual da *Cloudera Distribution for Hadoop*, e o *framework* de geração de dados da TPCx-BB.

Por ser uma plataforma de código aberto e ser relativamente barata, o *Hadoop* foi selecionado para apresentar o processamento de *Big Data*.

A fim de desenvolver um DW, O *Hive* foi selecionado, pois este *framework* foi criado com o intuito de aplicar técnicas de *data warehousing* para realizar o gerenciamento dos dados armazenados no HDFS. Além disso, o *HiveQL* – linguagem de consulta do *Hive* – possui muitas semelhanças com SQL, o que facilita e auxilia bastante no desenvolvimento e aprendizado.

A CDH foi utilizada pelo fato de ser uma plataforma de distribuição completamente *open source* que integra o *Hadoop* com vários outros projetos incluindo o *Hive*. A máquina virtual da Cloudera já traz todas as tecnologias instaladas e prontas para uso, além de facilitar todo o processo de manutenção e gerenciamento através de suas interfaces (Figura 10). O CDH foi utilizado no modo pseudo distribuído, pois este é recomendado para aprendizado sobre as tecnologias envolvidas pelo fato de não ser necessário utilizar um *cluster* de máquinas e conseguir simular um cluster com apenas uma máquina.

Figura 10 – Interfaces da Cloudera



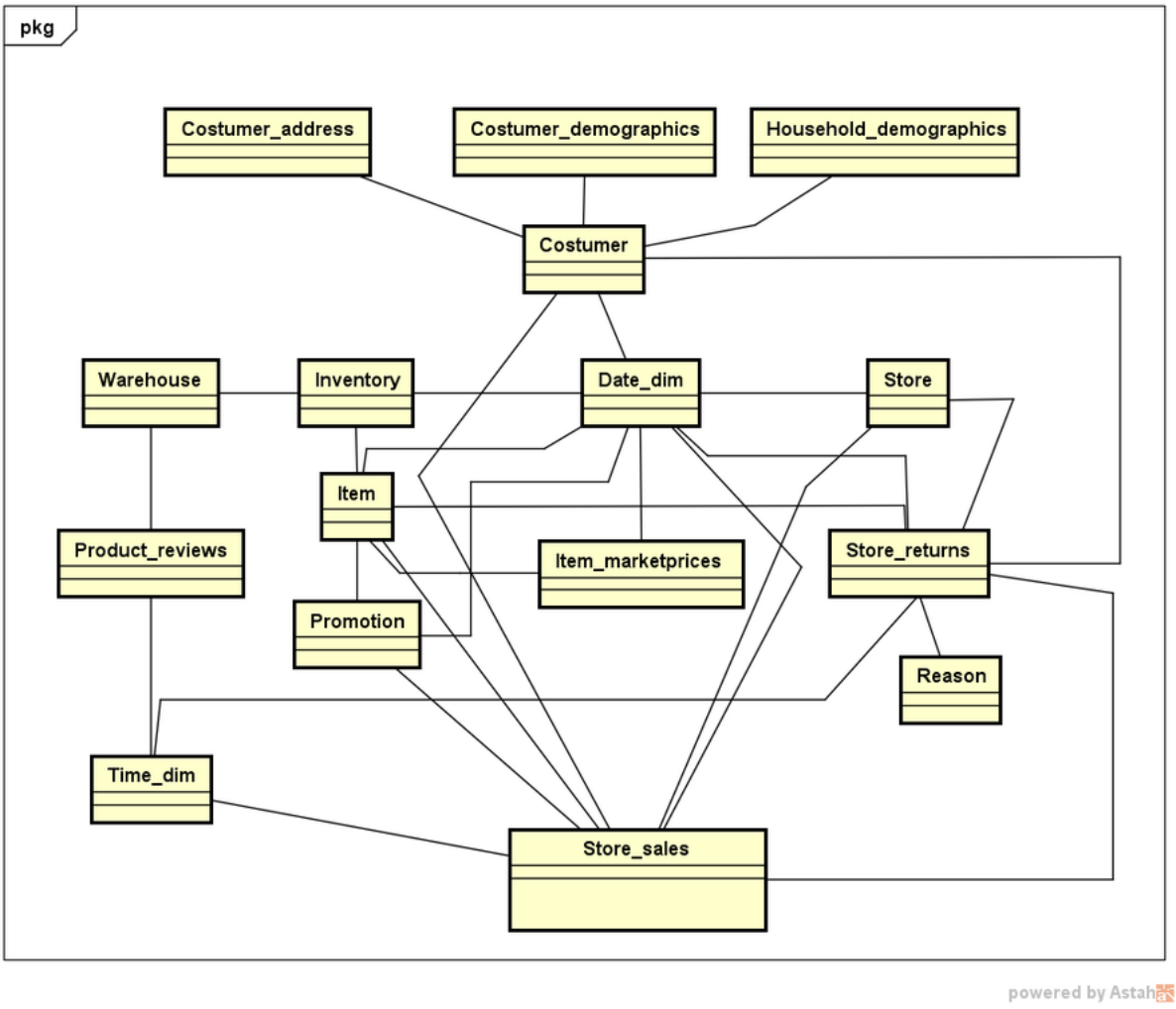
Fonte: Elaborada pelo Autor (2016)

Por fim, o TPCx-BB foi utilizado, pois este define referências de bancos de dados e processamento de transações *Big Data*. O seu *framework* de geração de dados possui parâmetros que permitem o escalonamento de dados e torna possível aplicar testes em diversos cenários. Desse modo, ao utilizar a CDH no modo pseudo distribuído pode-se também definir um volume de dados que seja suficiente para aplicar os testes e compreender sobre o funcionamento do processo sem haver necessidade de se preocupar com limitações de *hardware*.

4.2.1 Desenvolvimento do DW

Para desenvolver o DW, este trabalho utilizou a estrutura das tabelas geradas pelo TPCx-BB. Na Figura 11 pode-se observar um diagrama de classes e ver como as tabelas se relacionam.

Figura 11 – Diagrama de Classes



Fonte: Elaborada pelo Autor (2016)

A seguir, serão descritas as tabelas que compõem o *data warehouse,* o qual foi

construído sobre um modelo de varejo composto por 15 dimensões e 3 fatos.

4.2.1.1 Dimensões

4.2.1.1.1 Data_dim

Tabela 1 – Dimensão de Datas.

Nome	Tipo	Descrição
d_date_sk	BIGINT	Chave começando em 1
d_date_id	CHAR(16)	*String* única de tamanho 16 Conjunto de caractecres: "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
d_date	CHAR(13)	aaaa-MM-dd
d_month_seq	INT	Começa em 0 Conta todos os meses desde a primeira até a última data existente na tabela
d_week_seq	INT	Começa em 0 Conta todas as semanas desde a primeira até a última data existente na tabela
d_quarter_seq	INT	Começa em 0 Conta todos os trimestres desde a primeira até a última data existente na tabela
d_year	INT	Parte da data referente ao ano: aaaa
d_dow	INT	Dia da semana: 1-7, 1 = Segunda
d_moy	INT	Mês do ano: 1-12, 1 = Janeiro
d_dom	INT	Dia do mês: 1-31
d_qoy	INT	Trimestre do ano: 1-4
d_fy_year	INT	Ano financeiro: d_year + $1/2$year
d_fy_quarter_seq	INT	Trimestre financeiro: d_quarter + $1/2$year
d_fy_week_seq	INT	Semana financeira: d_week + $1/2$year
d_day_name	CHAR(9)	Dia da semana como uma *string*: {Segunda, . . . , Domingo}

Nome	Tipo	Descrição
d_quarter_name	CHAR(6)	Trimestre do ano como uma *string*: aaaaT{1..4}: exemplo: 1990T2
d_holiday	CHAR(1)	Feriado (N/Y)
d_weekend	CHAR(1)	Fim de Semana (N/Y)
d_following_holiday	CHAR(1)	Feriado Seguinte (N/Y)
d_first_dom	INT	Primeiro dia do mês no calendário Juliano
d_last_dom	INT	Último dia do mês no calendário Juliano
d_same_day_ly	INT	O dia corrente no calendário Juliano
d_same_day_lq	INT	O dia corrente no calendário Juliano
d_current_day	CHAR(1)	Dia corrente (N/Y)
d_current_week	CHAR(1)	Semana corrente (N/Y)
d_current_month	CHAR(1)	Mês corrente (N/Y)
d_current_quarter	CHAR(1)	Trimestre corrente (N/Y)
d_current_year	CHAR(1)	Ano corrente (N/Y)

4.2.1.1.2 Cliente

Tabela 2 – Dimensão de Clientes.

Nome	Tipo	Descrição
c_customer_sk	**BIGINT**	*Serial key*
c_customer_id	**CHAR(16)**	*String* única, tamanho: 16 Conjunto de caracteres: "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
c_current_cdemo_sk	**BIGINT**	Referência aleatória para a tabela *costumer_demographics cd_demo_sk*
c_current_hdemo_sk	**BIGINT**	Referência aleatória para a tabela *household_demographics hd_demo_sk*
c_current_addr_sk	**BIGINT**	Referência aleatória para a tabela *costumer_address ca_address_sk*
c_first_shipto_date_sk	**BIGINT**	Referência aleatória para a tabela *date_dim d_date_sk*
c_first_sales_date_sk	**BIGINT**	Referência aleatória para a tabela *date_dim d_date_sk*

Nome	Tipo	Descrição
c_salutation	**CHAR(10)**	Saudação {Mr., Mrs., Ms., Miss., Sir., Dr.}
c_first_name	**CHAR(20)**	Primeiro nome
c_last_name	**CHAR(30)**	Último nome
c_preferred_cust_flag	**CHAR(1)**	*Flag* para determinar se esse é um cliente "preferencial"
c_birth_day	**INT**	Número aleatório: [1, 31]
c_birth_month	**INT**	Número aleatório: [1, 12]
c_birth_year	**INT**	Número aleatório: [1924, 1992]
c_birth_country	**VARCHAR(20)**	País natal
c_login	**CHAR(13)**	*String* aleatória, tamanho: [1-13]
c_email_address	**CHAR(50)**	Padrão: C_first_name.c_last_name@provedorRandômico.tld
c_last_review_date	**VARCHAR(19)**	Data do ultimo *review* Min: Dia atual – 1 ano Max: Dia atual

4.2.1.1.3 Endereco_cliente

Tabela 3 – Endereço do cliente

Nome	Tipo	Descrição
ca_address_sk	**BIGINT**	*Serial key*
ca_address_id	**CHAR(16)**	*String* única, tamanho: 16 Conjunto de caracteres: “ABCDEFGHIJKLMNOPQRSTUVWXYZ”
ca_street_number	**CHAR(10)**	Nº da rua Valor randômico: [1, 1000]
ca_street_name	**VARCHAR(60)**	Nome da rua
ca_street_type	**CHAR(15)**	Tipo da rua {Street,ST,Avenue,Ave,Boulevard, Blvd,Road,RD,Parkway,Pkwy,Way, Wy,Drive,Dr.,Circle,Cir.,Lane,Ln ,Court,Ct.}
ca_suite_number	**CHAR(10)**	*String* aleatória, tamanho: [1, 10]
ca_city	**VARCHAR(60)**	Cidades
ca_county	**VARCHAR(30)**	Condado
ca_state	**CHAR(2)**	Estado
ca_zip	**CHAR(10)**	Código postal
ca_country	**VARCHAR(20)**	País
ca_gmt_offset	**DECIMAL**	Fuso horário
ca_location_type	**CHAR(20)**	Tipo de Lugar {Single family, Condo, Apartment}

4.2.1.1.4 Demografia_cliente

Tabela 4 – Dados demográficos do cliente

Nome	Tipo	Descrição
cd_demo_sk	**BIGINT**	*Serial key*
cd_gender	**CHAR(1)**	Masculino ou Feminino (M/F)
cd_marital_status	**CHAR(1)**	Estado Civil
cd_education_status	**CHAR(20)**	Escolaridade

Nome	Tipo	Descrição
cd_purchase_estimate	**INT**	Poder de compra
cd_credit_rating	**CHAR(10)**	Avaliação de crédito
cd_dep_count	**INT**	Dependentes
cd_dep_employed_count	**INT**	Dependentes Empregados
cd_dep_college_count	**INT**	Dependentes na faculdade

4.2.1.1.5 Demografia_domicilio

Tabela 5 – Dados demográficos do domicílio

Nome	Tipo	Descrição
hd_demo_sk	BIGINT	*Serial key*
hd_buy_potential	CHAR(15)	Poder de compra
hd_dep_count	INT	Dependentes
hd_vehicle_count	INT	Quantidade de Veículos

4.2.1.1.6 Inventario

Tabela 6 – Inventário dos itens estocados no warehouse

Nome	Tipo	Descrição
inv_date_sk	BIGINT	Referência randômica para tabela *date_dimd_date_sk*
inv_item_sk	BIGINT	Referência randômica para tabela *item i_item_sk*
inv_warehouse_sk	BIGINT	Referência randômica para tabela *warehouse w_warehouse_sk*
inv_quantity_on_hand	INT	Quantidade no inventário

4.2.1.1.7 Item

Tabela 7 – Dimensão Item

Nome	Tipo	Descrição
i_item_sk	BIGINT	*Serial key*
i_item_id	CHAR(16)	*String* única, tamanho: 16 Conjunto de caracteres: "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
i_rec_start_date	CHAR(13)	Data de início de disponibilidade
i_rec_end_date	CHAR(13)	Data de fim de disponibilidade
i_item_desc	VARCHAR(200)	Descrição do item
i_current_price	DECIMAL	Preço atual Número decimal randômico: [0.09, 99.99]
i_wholesale_cost	DECIMAL	Custo total de venda
i_brand_id	INT	Código da marca
i_brand	CHAR(50)	Marca
i_class_id	INT	Código da classe
i_class	CHAR(50)	Classe
i_category_id	INT	Código da categoria
i_category	CHAR(50)	Categoria
i_manufact_id	INT	Código da manufatura
i_manufact	CHAR(50)	Manufatura
i_size	CHAR(20)	Tamanho do item
i_formulation	CHAR(20)	Formulação
i_color	CHAR(20)	Cor
i_units	CHAR(10)	Unidades
i_container	CHAR(10)	Recipiente
i_manager_id	INT	Código do gerente
i_product_name	CHAR(50)	Nome do produto

4.2.1.1.8 Revisoes_Produtos

Tabela 8 – Dimensão com informações sobre revisões dos produtos

Nome	Tipo	Descrição

Nome	Tipo	Descrição
pr_review_sk	BIGINT	*Serial key*
pr_review_date	CHAR(13)	Data do *review*
pr_review_time	CHAR(6)	Referência randômica para a tabela *time_dim t_time_sk*
pr_review_rating	INT	Nota
pr_item_sk	BIGINT	Referência randômica para *ws_item_sk*
pr_user_sk	BIGINT	Referência randômica para *ws_user_sk*
pr_order_sk	BIGINT	Referência randômica para a tabela *web_sales order_id*
pr_review_content	STRING	Descrição da *review*

4.2.1.1.9 Promocao

Tabela 9 – Dimensão com informações sobre as promoções

Nome	Tipo	Descrição
p_promo_sk	BIGINT	*Serial key*
p_promo_id	CHAR(16)	*String* única, tamanho: 16 Conjunto de caracteres: “ABCDEFGHIJKLMNOPQRSTUVWXYZ”
p_start_date_sk	BIGINT	Referência randômica para a tabela *date_dim d_date_sk*
p_end_date_sk	BIGINT	Referência randômica para a tabela *date_dim d_date_sk*
p_item_sk	BIGINT	Referência randômica para a tabela *item i_item_sk*
p_cost	DECIMAL	Custo
p_promo_name	CHAR(50)	Nome da promoção
p_channel_dmail	CHAR(1)	Y/N
p_channel_email	CHAR(1)	Y/N

Nome	Tipo	Descrição
p_channel_catalog	CHAR(1)	Y/N
p_channel_tv	CHAR(1)	Y/N
p_channel_radio	CHAR(1)	Y/N
p_channel_press	CHAR(1)	Y/N
p_channel_event	CHAR(1)	Y/N
p_channel_demo	CHAR(1)	Y/N
p_channel_details	VARCHAR(100)	Detalhes sobre o canal
p_purpose	CHAR(15)	Propósito
p_discount_active	CHAR(1)	Y/N

4.2.1.1.10 Razao

Tabela 10 – Dimensão com informações sobre as razões de retorno dos items

Nome	Tipo	Descrição
r_reason_sk	BIGINT	*Serial key* *String* única, tamanho: 16
r_reason_id	CHAR(16)	Conjunto de caracteres: “ABCDEFGHIJKLMNOPQRSTUVWXYZ”
r_reason_desc	CHAR(100)	Descrição da razão do retorno

4.2.1.1.11 Loja

Tabela 11 – Dimensão de lojas

Nome	Tipo	Descrição
s_store_sk	BIGINT	*Serial key* *String* única, tamanho: 16
s_store_id	CHAR(16)	Conjunto de caracteres: “ABCDEFGHIJKLMNOPQRSTUVWXYZ”

Nome	Tipo	Descrição
s_rec_start_date	CHAR(13)	Referência randômica para a tabela *date_dim d_date_sk*
s_rec_end_date	CHAR(13)	Referência randômica para a tabela *date_dim d_date_sk*
s_closed_date_sk	BIGINT	Data em que a loja está fechada. Referência para a tabela *date_dim d_date_sk*
s_store_name	VARCHAR(50)	Nome da loja
s_number_employees	INT	Número inteiro randômico: [200, 300]
s_floor_space	INT	Número inteiro randômico: [5000000, 10000000]
s_hours	CHAR(20)	Horário de atendimento
s_manager	VARCHAR(40)	Nome do Gerente
s_market_id	INT	Código do Mercado
s_geography_class	VARCHAR(100)	Classe geográfica
s_market_desc	VARCHAR(100)	Descrição do Mercado
s_market_manager	VARCHAR(40)	Gerente do Mercado
s_division_id	INT	Código da Divisão
s_division_name	VARCHAR(50)	Divisão
s_company_id	INT	Código da Empresa
s_company_name	VARCHAR(50)	Empresa
s_street_number	VARCHAR(10)	Endereço igual ao do *warehouse*
s_street_name	VARCHAR(60)	Endereço igual ao do *warehouse*
s_street_type	CHAR(15)	Endereço igual ao do *warehouse*
s_suite_number	CHAR(10)	Endereço igual ao do *warehouse*
s_city	VARCHAR(60)	Endereço igual ao do *warehouse*
s_county	VARCHAR(30)	Endereço igual ao do *warehouse*
s_state	CHAR(2)	Endereço igual ao do *warehouse*
s_zip	CHAR(10)	Endereço igual ao do *warehouse*
s_country	VARCHAR(20)	Endereço igual ao do *warehouse*
s_gmt_offset	DECIMAL	Endereço igual ao do *warehouse*
s_tax_precentage	DECIMAL	Taxa

4.2.1.1.12 Tempo_dim

Tabela 12 – Dimensão de tempo

Nome	Tipo	Descrição
t_time_sk	**BIGINT**	*Serial key* *String* única, tamanho: 16
t_time_id	**CHAR(16)**	Conjunto de caracteres: "ABCDEFGHIJKLMNOPQRSTUVWXYZ" Começa no 0
t_time	**INT**	*t_time* = *t_time_id*
t_hour	**INT**	*t_time_id*/60/60mod 24
t_minute	**INT**	*t_time_id*/60 mod 60
t_second	**INT**	*t_time_id* mod 60
t_am_pm	**CHAR(2)**	Turno {AM,PM}
t_shift	**CHAR(20)**	{*first, second, third*}
t_sub_shift	**CHAR(20)**	{*morning, afternoon, night*}
t_meal_time	**CHAR(20)**	{*breakfast, lunch*}

4.2.1.1.13 Armazem

Tabela 13 – Dimensão de armazéns

Nome	Tipo	descrição
w_warehouse_sk	**BIGINT**	*Serial key* *String* única, tamanho: 16
w_warehouse_id	**CHAR(16)**	Conjunto de caracteres: "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
w_warehouse_name	**VARCHAR(20)**	Nome do *warehouse*
w_warehouse_sq_ft	**INT**	
w_street_number	**CHAR(10)**	Número da rua
w_street_name	**VARCHAR(60)**	Nome da rua
w_street_type	**CHAR(15)**	Tipo da rua
w_suite_number	**CHAR(10)**	
w_city	**VARCHAR(60)**	Cidade

Nome	Tipo	descrição
w_county	**VARCHAR(30)**	Condado
w_state	**CHAR(2)**	Estado
w_zip	**CHAR(10)**	Código Postal
w_country	**VARCHAR(20)**	País
w_gmt_offset	**DECIMAL**	Fuso Horário

4.2.1.1.14 Pagina_Web

Tabela 14 – Dimensão de Página da Web

Nome	Tipo	Descrição
wp_web_page_sk	**BIGINT**	*Serial key*
wp_web_page_id	**CHAR(16)**	*String* única, tamanho: 16 Conjunto de caracteres: “ABCDEFGHIJKLMNOPQRSTUVWXYZ”
wp_rec_start_date	**CHAR(13)**	Referência para tabela *date_dim d_date_sk*
wp_rec_end_date	**CHAR(13)**	Referência para tabela *date_dim d_date_sk*
wp_creation_date_sk	**BIGINT**	Referência para tabela *date_dim d_date_sk*
wp_access_date_sk	**BIGINT**	Referência para tabela *date_dim d_date_sk*
wp_autogen_flag	**CHAR(1)**	
wp_customer_sk	**BIGINT**	Referência para tabela *customer c_customer_sk*
wp_url	**VARCHAR(100)**	URL
wp_type	**CHAR(50)**	Tipo da página {general,order,welcome,ad,feedback,protected,dynamic,review}
wp_char_count	**INT**	Quantidade de caracteres na página
wp_link_count	**INT**	Quantidade de *links* na página
wp_image_count	**INT**	Quantidade de imagens na página
wp_max_ad_count	**INT**	Quantidade máxima de propagandas na página

4.2.1.1.15 Web_site

Tabela 15 – Dimensão de Web Site

Nome	Tipo	Descrição
web_site_sk	**BIGINT**	*Serial key*
		String única, tamanho: 16
web_site_id	**CHAR(16)**	Conjunto de caracteres: "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
web_rec_start_date	**CHAR(13)**	Referência para a tabela *date_dim d_date_sk*
web_rec_end_date	**CHAR(13)**	Referência para a tabela *date_dim d_date_sk*
web_name	**VARCHAR(50)**	Nome do site
web_open_date_sk	**BIGINT**	Referência para a tabela *date_dim d_date_sk*
web_close_date_sk	**BIGINT**	Referência para a tabela *date_dim d_date_sk*
web_class	**VARCHAR(50)**	Classe do site
web_manager	**VARCHAR(40)**	Gerente do site
web_mkt_id	**INT**	Código de Mercado do site
web_mkt_class	**VARCHAR(50)**	Classe do Mercado do site
web_mkt_desc	**VARCHAR(100)**	Descrição do Mercado do site
web_market_manager	**VARCHAR(40)**	Gerente do Mercado do site
web_company_id	**INT**	Código da Empresa do site
web_company_name	**CHAR(50)**	Empresa do site
web_street_number	**CHAR(10)**	Endereço igual ao do *warehouse*
web_street_name	**VARCHAR(60)**	Endereço igual ao do *warehouse*
web_street_type	**CHAR(15)**	Endereço igual ao do *warehouse*
web_suite_number	**CHAR(10)**	Endereço igual ao do *warehouse*
web_city	**VARCHAR(60)**	Endereço igual ao do *warehouse*
web_county	**VARCHAR(30)**	Endereço igual ao do *warehouse*
web_state	**CHAR(2)**	Endereço igual ao do *warehouse*
web_zip	**CHAR(10)**	Endereço igual ao do *warehouse*
web_country	**VARCHAR(20)**	Endereço igual ao do *warehouse*

Nome	Tipo	Descrição
web_gmt_offset	**DECIMAL**	Endereço igual ao do *warehouse*
web_tax_percentage	**DECIMAL**	Taxa

4.2.1.2 **Fatos**

4.2.1.2.1 Item_precos_mercado

Tabela 16 – Fato Preços de Mercado dos Itens

Nome	Tipo	Descrição
imp_sk	BIGINT	*Serial key*
imp_item_sk	BIGINT	Referência randômica para a tabela *item* *i_item_sk*
imp_competitor	VARCHAR(20)	*String* randômica, tamanho: [1, 20]
imp_competitor_price	DECIMAL	Decimal randômico [0.99, 99.99]
imp_start_date_sk	BIGINT	Referência randômica para a tabela *date_dim* *d_date_sk*
imp_end_date_sk	BIGINT	Referência randômica para a tabela *date_dim* *d_date_sk*

4.2.1.2.2 Retorno_Vendas

Tabela 17 – Fato Retornos de Vendas

Nome	Tipo	descrição
sr_returned_date_sk	**BIGINT**	Referência randômica para a tabela *date_dim* *d_date_sk*

Nome	Tipo	descrição
sr_return_time_sk	**BIGINT**	Referência randômica para a tabela *time_dim t_time_sk*
sr_item_sk	**BIGINT**	Referência randômica para a tabela *item i_item_sk*
sr_customer_sk	**BIGINT**	Referência randômica para a tabela *costumer c_costumer_sk*
sr_cdemo_sk	**BIGINT**	Referência randômica para a tabela *costumer_demographics cd_demo_sk*
sr_hdemo_sk	**BIGINT**	Referência randômica para a tabela *household_demographics hd_hdemo_sk*
sr_addr_sk	**BIGINT**	Referência randômica para a tabela *costumer_address ca_addr_sk*
sr_store_sk	**BIGINT**	Referência randômica para a tabela *store s_store_sk*
sr_reason_sk	**BIGINT**	Referência randômica para a tabela *reason r_reason_sk*
sr_ticket_number	**BIGINT**	Referência para a tabela *store_sales ss_ticket_number*
sr_return_quantity	**INT**	Quantidade de itens retornados
sr_return_amt	**DECIMAL**	Preço * Quantidade
sr_return_tax	**DECIMAL**	*sr_return_*amt * Taxa
sr_return_amt_inc_tax	**DECIMAL**	*sr_return_amt + sr_return_tax*
sr_fee	**DECIMAL**	
sr_return_ship_cost	**DECIMAL**	
sr_refunded_cash	**DECIMAL**	Dinheiro retornado
sr_reversed_charge	**DECIMAL**	
sr_store_credit	**DECIMAL**	

Nome	Tipo	descrição
sr_net_loss	**DECIMAL**	

4.2.1.2.3 Vendas_Loja

Tabela 18 – Fato Vendas da Loja

Nome	Tipo	descrição
ss_sold_date_sk	**BIGINT**	Referência para a tabela *date_dim d_date_sk*
ss_sold_time_sk	**BIGINT**	Referência randômica para a tabela *time_dim t_time_sk*
ss_item_sk	**BIGINT**	Referência randômica para a tabela *item i_item_sk*
ss_customer_sk	**BIGINT**	Referência randômica para a tabela *customer c_customer_sk*
ss_cdemo_sk	**BIGINT**	Referência randômica para a tabela *customer_demographics cd_cdemo_sk*
ss_hdemo_sk	**BIGINT**	Referência randômica para a tabela *household_demographics hd_hdemo_sk*
ss_addr_sk	**BIGINT**	Referência randômica para a tabela *customer_address ca_addr_sk*
ss_store_sk	**BIGINT**	Referência randômica para a tabela *store s_store_sk*
ss_promo_sk	**BIGINT**	Referência randômica para a tabela *promotion p_promo_sk*
ss_ticket_number	**BIGINT**	Número do *ticket*
ss_quantity	**INT**	Quantidade comprada do item

Nome	Tipo	descrição
ss_wholesale_cost	**DECIMAL**	Custo de venda total
ss_list_price	**DECIMAL**	Preço de lista de um único item
ss_sales_price	**DECIMAL**	Preço de venda de um único item
ss_ext_discount_amt	**DECIMAL**	Desconto * Quantidade
ss_ext_sales_price	**DECIMAL**	Custo do item * Quantidade
ss_ext_wholesale_cost	**DECIMAL**	Custo de venda total * Quantidade
ss_ext_list_price	**DECIMAL**	Preço de lista do item * Quantidade
ss_ext_tax	**DECIMAL**	Taxa
ss_coupon_amt	**DECIMAL**	Cupom de desconto
ss_net_paid	**DECIMAL**	Valor líquido pago dos itens * Quantidade
ss_net_paid_inc_tax	**DECIMAL**	Valor líquido pago dos itens incluindo a taxa * Quantidade
ss_net_profit	**DECIMAL**	Lucro sobre o item comprador

4.2.2 Volume de Dados

O volume de dados utilizado foi de 1 gigabyte tendo em vista as limitações da simulação de um *cluster* com apenas uma máquina. Além disso, com o objetivo de entender e apresentar o processo de criação de um DW para um cenário Big Data, este volume de dados é suficiente (MAJDALANY,).

O data generator do TPCx-BB permite que o volume de dados seja escalonado, podendo chegar à casa do terabyte, portanto, com um *cluster* mais potente disponível seria possível aplicar o mesmo processo.

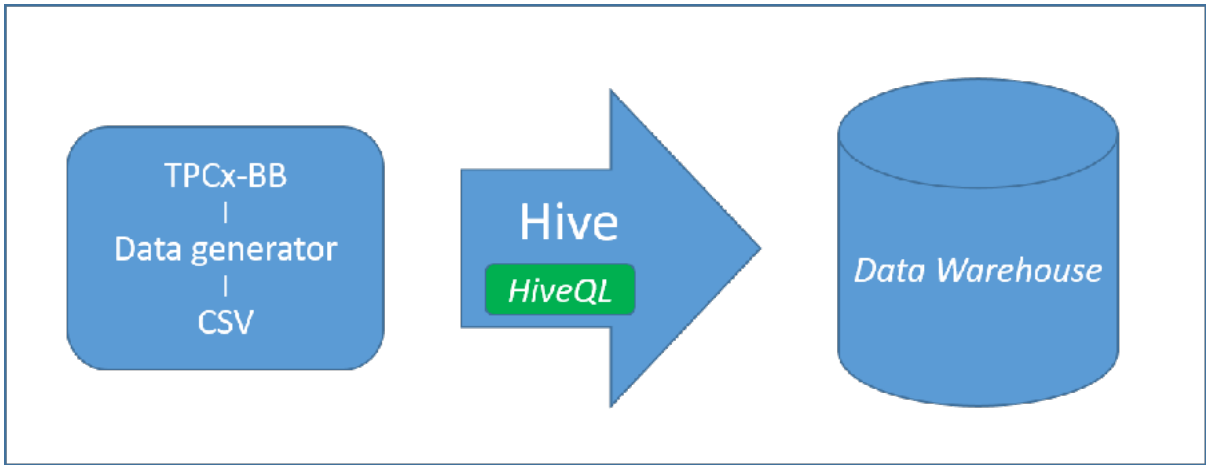
4.2.3 Processo de ETL

Utilizando o *data generator* do TPCx-BB foi possível gerar os dados das respectivas tabelas num formato CSV[1]. O framework já realiza uma das etapas do ETL no que se refere a classificação e categorização dos dados semiestruturados e não-estruturados. Dessa forma, a arquitetura do DW poderia ser vista conforme ilustrada na Figura 12. Os dados gerados pelo framework da TPCx-BB foram utilizados como fonte de dados, e a linguagem HiveQL foi utilizada para carregar os dados no DW. Uma vez

[1] Comma-separated values; arquivos texto com células separadas por delimitadores.

que os dados foram gerados, foi necessário desenvolver os *scripts* para criação das tabelas usando a linguagem de consulta do Hive.

Figura 12 – Arquitetura do DW - TPCx-BB & Hive



Fonte: Elaborada pelo Autor (2016)

Com a *HiveQL* é possível realizar a carga das tabelas diretamente de arquivos CSV ao determinar qual o delimitador que será utilizado. Dessa forma, o processo de carga se dá de maneira simples.

4.3 Comentários Finais

O processo de desenvolvimento de um DW para um cenário *Big Data* descrito neste capítulo utilizou ferramentas que auxiliam e facilitam bastante a evolução do projeto. Tecnologias como a CDH e o *Hive* se mostram bastante úteis ao tornar o desenvolvimento muito mais simples e próximo da realidade dos projetos de BI tradicionais, uma vez que a Cloudera permite transparência para a complexidade existente na instalação e compreensão do *Hadoop* e seu ecossistema, e o *Hive* acomoda usuários de SQL por meio das suas semelhanças.

5 Conclusão

Apesar de já se ter um grande acesso às informações e conseguir realizar processos de tomada de decisão por meio da utilização de técnicas de *Data Warehousing* sobre as tradicionais bases de dados, com o advento do *Big Data* torna-se clara a necessidade das empresas em expandir o conhecimento sobre o mercado e seus *stakeholders*.

Desse modo, a utilização de ferramentas como o *Hadoop* pode ser vista como uma tendência e pode ser um fator muito influente no processo de adaptação das empresas para este novo cenário.

O estudo sobre as tecnologias de *Big Data* é essencial para permitir determinar quais dessas tecnologias serão úteis para solucionar o problema de processamento, armazenamento e gerenciamento dos dados relacionados à empresa.

Neste trabalho, as tecnologias selecionadas trouxeram bastante aprendizado de modo a permitir a compreensão sobre o processamento e armazenamento de *Big Data*. Além disso, o desenvolvimento do DW, com o auxílio da tecnologia *Hive*, possibilitou a imersão nesse novo cenário de dados de modo a apresentar parte do potencial que esse tipo de aplicação possibilita, já que permite trabalhar com informações de diversas fontes, formatos e volumes, cada vez mais presentes na realidade empresarial.

5.1 Limitações

Devido ao desenvolvimento deste trabalho ter sido realizado por meio da simulação de um *cluster* utilizando uma só máquina (modo pseudodistribuído da CDH), algumas limitações de processamento fazem com que não se possa realizar testes completos sobre as tecnologias utilizadas.

Por esse motivo, o volume de dados utilizado – apesar de ser um volume suficiente para compreensão sobre o processo de desenvolvimento de um DW para *Big Data* – não foi escalonado para maiores quantidades. O *framework* de geração de dados da TPCx-BB possui um fator de escalonamento que pode ser definido e possibilita que o volume total de dados alcance a casa do *terabyte*, no entanto essa função não pôde ser devidamente explorada por limitações de *hardware*.

5.2 Trabalhos Futuros

Como foi observado neste trabalho, além do *Hive*, existem diversas tecnologias que compõem o ecossistema do *Hadoop*. O *Spark* é uma delas. Por meio dele também

seria possível realizar o desenvolvimento de um DW para *Big Data*, e assim seria possível analisar uma outra abordagem.

Além de avaliar outras tecnologias, também se pode ter como proposta de trabalho futuro a realização do mesmo processo sobre a base de dados gerada pela TPCx-BB, mas com um maior volume de dados. É possível, ainda, explorar o funcionamento do DW em *cluster* com um número significativo de máquinas a fim de analisar questões de desempenho do sistema.

Aplicar o processo desenvolvido neste trabalho num cenário real seria ideal para obter resultados mais concretos.

Referências

APACHE. *The Apache Software Foundation*. Disponível em: <https://www.apache.org>.

BETTER, B. A. et al. Heading Towards Big Data. p. 220 – 225, 2013.

CAMARGO, J. E. et al. A Big Data Analytics System to Analyze Citizens ' Perception of Security. p. 0 – 4, 2016.

CUZZOCREA, A.; SACCÀ, D.; ULLMAN, J. D. Big data: A research agenda. *Proceedings of the 17th International Database Engineering {&} Applications Symposium on - IDEAS '13*, n. October, p. 198 – 203, 2013. Disponível em: <http://dl.acm.org/citation.cfm?doid=2513591.2527071>.

EDUCAÇÃO, P. *METODOLOGIA CIENT´FICA: TIPOS DE PESQUISA*. 2013. Disponível em: <https://www.portaleducacao.com.br/pedagogia/artigos/50264/metodologia-cientifica-tipos-de-esquisa>.

INMON, W.; STRAUSS, D.; NEUSHLOSS, G. *Data Warehouse 2.0*. Morgan-Kaufmann. Disponível em: <http://gen.lib.rus.ec/book/index.php?md5=4FACB22329B131FFEF49B85C37712225>.

JURNEY, R. *Agile Data Science: Building Data Analytics Applications with Hadoop*. [S.l.: s.n.], 2013. ISSN 1098-6596. ISBN 9781449326265.

MAJDALANY, M. *Transaction Processing Performance Council*. Disponível em: <http://www.tpc.org/>.

MELOROSE, J.; PERROY, R.; CAREAS, S. *Data Warehouse in the Age of Big Data*. [S.l.: s.n.], 2015. ISSN 1098-6596. ISBN 9788578110796.

PROVOST, T. F. F. *Data Science for Business: What you need to know about data mining and data-analytic thinking*. [S.l.]: O'Reilly Media, 2013. ISBN 1449361323,9781449361327.

SITTO, K.; PRESSER, M. *Field Guide to Hadoop*. [S.l.: s.n.], 2015. ISBN 0471462128.

SUN, L. et al. Present situation and prospect of data warehouse architecture under the background of big data. *Proceedings - 2013 International Conference on Information Science and Cloud Computing Companion, ISCC-C 2013*, p. 529 – 535, 2014.

THARAKAN, R. *What is scalability?* 2007. Disponível em: <http://www.royans.net/wp/2007/09/22/what-is-scalability/>.

WHITE, T. *Hadoop : the definitive guide*. 2nd ed. ed. [S.l.]: O'Reilly, 2010. ISBN 9781449389734,1449389732.

ZHANG, R. et al. Big Data for Medical Image Analysis: A Performance Study. *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, p. 1660 – 1664, 2016. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7530064>.