LUÍS ANTÔNIO PAIN JÚNIOR

Estudo comparativo das plataformas e tecnologias utilizadas em Data Warehouse e Big Data

Monografia apresentada ao PECE – Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo como parte dos requisitos para conclusão do curso de MBA em Tecnologia de Software.

São Paulo
2015

LUÍS ANTÔNIO PAIN JÚNIOR

Estudo comparativo das plataformas e tecnologias utilizadas em Data Warehouse e Big Data

Monografia apresentada ao PECE – Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo como parte dos requisitos para a conclusão do curso de MBA em Tecnologia de Software.

Área de Concentração: Tecnologia de Software

Orientador: Prof. Dr. Jorge Rady de Almeida Junior

São Paulo
2015

DEDICATÓRIA

A minha amada namorada Harumi.

A meus pais Luís e Bárbara.

A meus irmãos Matheus, Yuri e Karina.

A meus cachorros Luke e Maggie.

AGRADECIMENTOS

Agradeço a minha namorada Harumi, que sempre me apoiou e muito ajudou na construção desse trabalho. A sua presença, amor e dedicação.

A meus pais Luís e Bárbara, que me deram todas as oportunidades para que tudo acontecesse.

A meus irmãos Matheus, Yuri e Karina, os quais me fizeram o que hoje sou e que sempre serão meus melhores amigos.

À Avanade pela oportunidade de realização do trabalho e pelo respeito aos horários.

Aos professores do Curso de Tecnologia de Software, por todo ensinamento, críticas e sugestões.

Ao PECE pelo apoio e ensinamentos.

A meu orientador Jorge Rady pela paciência e compreensão.

RESUMO

As empresas utilizam há um bom tempo, "Business Intelligence", Inteligência de Negócio ou simplesmente BI (*Business Intelligence*), que é a transformação dos dados com a finalidade de gerar valores estratégicos, competitivos ou até novas oportunidades. Dados de seus clientes, fornecedores, operações, usuários, sensores e outros são utilizados para fornecer informações de tendências e vantagens competitivas. Desde sua origem, inúmeras estratégias foram concebidas, tecnologias foram criadas e as empresas a utilizam das mais diversas maneiras. Data Warehouse e Big Data são plataformas que exploram o mundo de BI com peculiaridades distintas. O trabalho inicia definindo cada uma das frentes com seus contextos históricos, conceitos teóricos, arquiteturas e técnicas de mineração. Consiste também em explorar suas vastas formas de implementação, as diferenças analíticas, os detalhes de armazenamento, requisitos, performance, qualidade dos dados, focando nas três principais propriedades do Big Data: Volume, Velocidade e Variedade, expondo as tecnologias como NoSQL, Hadoop, MapReduce e Sistemas distribuídos. Finalmente, propõe três cenários distintos para plataformas com diferentes tecnologias e arquiteturas, facilitando a escolha e auxiliando tomadores de decisões ou mesmo profissionais na área de TI.

Palavras-chave: Big Data, Data Warehouse, MapReduce, Data Mining, NoSQL, Hadoop, Hadoop Distributed File System, Modelos dimensionais, Base de dados distribuídas, Sistemas distribuídos Dados estruturados, Dados não estruturados.

ABSTRACT

Companies have been using BI or Business Intelligence for a long time, which is handling the data with the goal of generating strategic values, competitive values or even new oportunities. Data from their clients, suppliers, operations, users, sensors and others are used to provide trend information and competitive advantages. Since the beginning, various strategies were made, tecnologies creates and enterprises use it in different ways. Data Warehouse and Big Data are tecnologies which explore the BI world with their own singularities. This work start by defining each imperative with its respective historical context, theoretical concepts, architectures, and mining techniques. Also include study in the implementation, analytical differences, storage details, requirements, performance, data quality, and focus on the three main characteristics of the Big Data: Volume, Velocity and Variety, exhibiting the new trends like NoSQL, Hadoop, MapReduce and Distributed Systems. Finally, propose a set three platforms with distinct tecnological scenarios with the already presented solutions with a way to choose the best fit scenario given the properties of a company, assisting both decision makers and IT professionals.

Keywords: Big Data, Data Warehouse, MapReduce, Data Mining, NoSQL, Hadoop, Dimensional modeling, Hadoop Distributed File System, Distributed database, Distributed systems, Structured data, Unstructured data.

LISTA DE ILUSTRAÇÕES

Figura 1 - Benefícios da uma plataforma de BI 14
Figura 2 - Processo de extração 17
Figura 3 - Data mart independente e dependente, respectivamente 19
Figura 4 - Modelo genérico de arquitetura de um Data Warehouse 20
Figura 5 - Esquema estrela 24
Figura 6 - Esquema floco de neve 25
Figura 7 – Modelo “Galaxy” 26
Figura 8 - Três componentes principais do Big Data: Volume, Velocidade e Variedade 29
Figura 9 - Modelo 5 Vs proposto por 30
Figura 10 – Modelo do pipeline do processamento de dados do Big Data 32
Figura 11 - Armazenamento e pesquisa de dados no HDFS 34
Figura 12 - Funcionamento do MapReduce 36
Figura 13 - High Performance computing cluster (HPCC) 37
Figura 14 - Hadoop Stack 38
Figura 15 - História dos dados e sistemas 41
Figura 16 - Arquitetura de alto nível do Hadoop 43
Figura 17 - Fluxo das consultas em bases paralelas 45
Figura 18- Arquitetura Shared-Nothing de bases paralelas 46
Figura 19 - Sistemas de bases para estudos analíticos 47
Figura 20 - Exemplo de armazenamento chave-valor 49
Figura 21 - Evolução dos dados 52
Figura 22 - Integração de documentos não estruturados 53
Figura 23 – Análise de dados como serviço, proposto 55
Figura 24 - Índice de linhas utilizado no HDFS 57
Figura 25 - Índice de colunas 58
Figura 26 - Efeito de um index Column Store 59
Figura 27 - Estrutura dos dados utilizada pelo RCFile 62
Figura 28 - Arquitetura Data Warehouse de alto desempenho 63
Figura 29 - Passos para identificação dos cenários de BI 69
Figura 30 - Identificação das plataformas 70
Figura 31 - Arquiteturas do Data Warehouse 71
Figura 32 - Arquitetura geral do Hadoop 73
Figura 33 - Arquitetura do HadoopDB 75

LISTA DE TABELAS

Tabela 1 - Comparação de compatibilidade arquiteturas para Data Warehouse 21
Tabela 2 - Origem e domínios de Big Data 39
Tabela 3 - Comparação entre BI tradicional e Big Data 42
Tabela 4 - Comparação entre os índices 59
Tabela 5 - Diferença entre Stream Data e Dados tradicionais 61
Tabela 6 - Comparação das extensões do Hadoop 66
Tabela 7 - Comparação das extensões do Hadoop 67
Tabela 8 - Potencial das indústrias em Big Data.....77
Tabela 9 - Relevância dos fatores na escolha da plataforma.....78
Tabela 10 - Matriz de seleção de tecnologias 78

LISTA DE ABREVIATURAS E SIGLAS

AaaS ..... *Analytics as a Service*
ACID ..... *Atomic, Consistent, Isolation and Durable*
API ..... *Application Program Interface*
BASE ..... *Basically Available, Soft state and Eventual consistency*
BD ..... *Big Data*
BDAaaS ..... *Big data Analytics as a Service*
BI ..... *Business Intelligence*
DFS ..... *Distributed File Systems*
DW ..... *Data Warehouse*
EDW ..... *Enteprise Data Warehouse*
ETL ..... *Extraction Transform and Load*
GFS ..... *Google File System*
HOLAP ..... *Hybrid Online Analytical Processing*
HPCC ..... *High Performance Cloud Computing*
I/O ..... *input / output*
IDC ..... *Internet Data Center*
JDBC ..... *Java Database Connectivity*
JSON ..... *JavaScript Object Notation*
KPI ..... *Key Performance Indicator*
MOLAP ..... *Multidimentional Online Analytical Processing*
NoSQL ..... *Not only SQL*
ODBC ..... *Open Database Connectivity*
OLAP ..... *Online Analytical Processing*
OLE DB ..... *Object Linking and Embedding, Database*
OLTP ..... *Online Transaction Processing*
PAX ..... *Partition Attributes Across*
RCFile ..... *Record Columnar File*
ROLAP ..... *Relational Online Analytical Processing*
SGBDR ..... *Sistema Gerenciador de Bases de Dados Relacionais*
SQL ..... *Structured Query Language*
SSD ..... *Solid-state drive*

CIO ..... *Chief Information Officer*
ECL ..... *Enterprise Control Language*
VLBD ..... *Very Large Data Base*
BLOB ..... *Binary Large Object*
HDFS ..... *Hadoop Distributed File System*
HDW ..... *High Performance Large Scale Data Warehouse*
IBGE ..... Instituto Brasileiro de Geografia e Estatística
XML ..... *Extensible Markup Language*
XMLA ..... *Extensible Markup Language for Analysis*

SUMÁRIO

RESUMO 5
ABSTRACT 6
LISTA DE ILUSTRAÇÕES 7
LISTA DE TABELAS 8
LISTA DE ABREVIATURAS E SIGLAS 9
SUMÁRIO 11
1. INTRODUÇÃO 13
1.1 Motivações 13
1.2 Objetivo 14
1.3 Justificativas 15
1.4 Estrutura do trabalho 16
2. DATA WAREHOUSE 17
2.1 Arquiteturas 18
2.2 *ETL* (Extração, Transformação e Carregamento) 22
2.3 Modelos dimensionais 23
2.4 Metadados 27
2.5 Ferramentas e técnicas 27
2.6 Conclusão do capítulo 27
3. BIG DATA 29
3.1 Hadoop 33
3.2 MapReduce 35
3.3 Arquitetura 36
3.4 *Data Mining* para *Big Data* 38
3.5 Conclusão do capítulo 39
4. *BIG DATA* E *DATA WAREHOUSE* – AS TECNOLOGIAS ENVOLVIDAS E AS DIFERENÇAS 40
4.1 Sistemas Distribuídos 43
4.2 *NoSQL*– Alternativa às bases convencionais 47
4.3 Dados não estruturados e serviços 51
4.4 Índices de linha, coluna e híbridos 56
4.5 Stream Data 60
4.6 Soluções alternativas 61
4.6.1 *RCFile* – Sistema de *DataWarehouse* 62
4.6.2 *Data Warehouse* de Alto desempenho 63
4.6.3 Solução híbrida 64

4.6.1 Outras soluções 65
4.7 Conclusão do capítulo 68
5. PLATAFORMAS E ARQUITETURAS PARA CENÁRIOS CORPORATIVOS 69
5.1 Data Warehouse 70
5.2 HDFS e MapReduce 72
5.3 Solução híbrida 74
5.4 Escolha das plataformas 75
5.5 Escolhas das tecnologias 78
5.6 Estratégia de seleção e exemplo 79
5.7 Conclusão do capítulo 80
6. CONCLUSÕES 81
6.1 Considerações finais 81
6.2 Contribuições e trabalhos futuros. 82
REFERÊNCIAS 83
APÊNDICE A – EXEMPLO DE CÓDIGO PARA CONTAGEM DE PALAVRAS NUMA COLEÇÃO DE DOCUMENTOS NO MAPREDUCE 90

1. INTRODUÇÃO

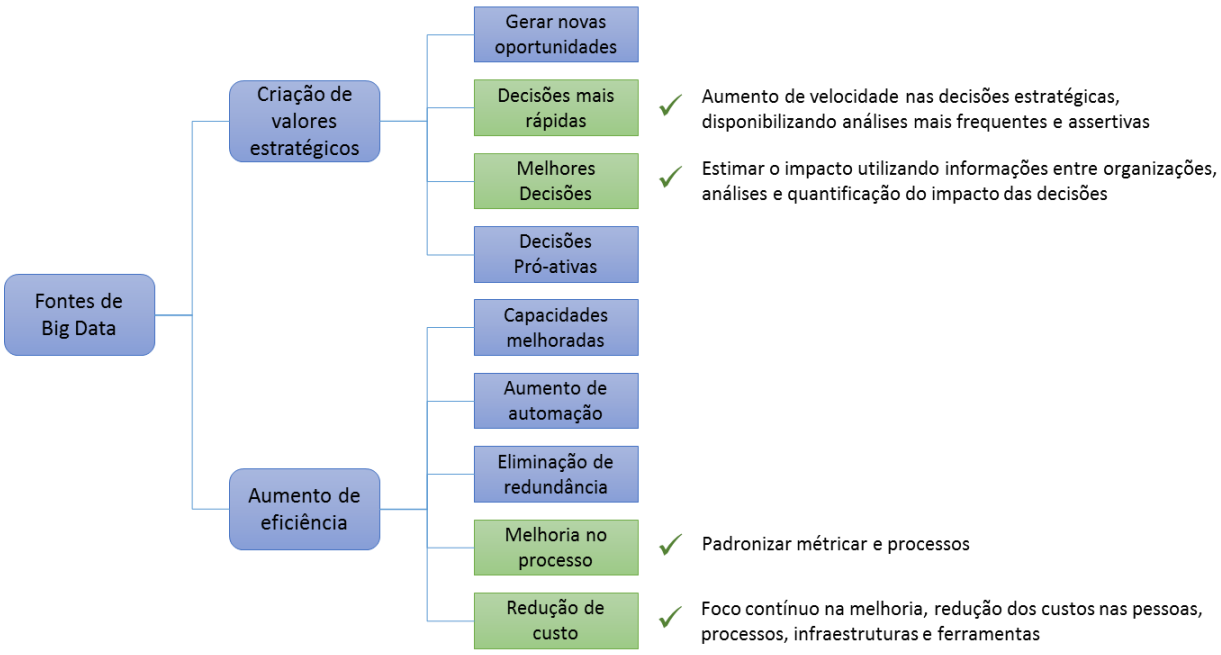
O mercado de banco de dados analíticos atualmente consiste numa capitalização de $180 bilhões com receita de $ 38 bilhões (TEAM TREFIS, 2014), dobrando de tamanho a cada 40 meses (BEDI; JINDAL e GAUTAM, 2014). A IBM estima que todos os dias 2,5 quintilhões de bytes são gerados, sendo que em torno de 90% dos dados foram criados nos últimos 2 anos (LOMOTEY e DETERS, 2014). Essa progressão por sua magnitude ainda é de difícil assimilação, pois as fontes disponíveis nos últimos anos aumentaram consideravelmente como: censo, dados abertos do governo, mídias sociais, notícias, feeders, mercado financeiro, número de dispositivos com sensores, logs, dados na web etc. A necessidade de se gerir essa enorme quantidade e variedade de dados levou à construção de sistemas distribuídos que culminou as atuais bases paralelas, o *Big Data* e outras soluções alternativas. Munido com a constante evolução desses sistemas, determinar a melhor resposta tecnológica se faz de maneira cada vez mais complexa.

Basear estratégias e tomada de decisões em análises de dados tornou-se boa prática, e a procura por sistemas que podem gerenciar, processar e analisar os dados está ficando cada vez mais alta. As empresas estão cientes disso e injetando um enorme capital de investimento em tecnologias para suprir a demanda, mesmo quando a economia sofre forte pressão (ABOUZEID; BAJDA-PAWLIKOWSKI *et al.*, 2009).

1.1 Motivações

O desafio estabelece-se portanto em processar consultas eficientes no grande volume e variedade de dados. As empresas em todas as verticais de mercado (um nicho específico que pode ser comercializado de forma similar; por exemplo: mercado financeiro, indústria automobilística, etc.) estão ficando cada vez mais dependentes da habilidade de acessar, guardar, utilizar e compartilhar as informações de maneira eficiente, com qualidade e em tempo hábil, ajudando a garantir o sucesso do negócio. Dentre as principais motivações de Mohanty, Jagadeesh e Srivatsa (2013), as destacadas em verde são as mais determinantes no que concerne à escolha de uma tecnologia e estão representadas na Figura 1.

Figura 1 - Benefícios da uma plataforma de BI



(MOHANTY, JAGADEESH e SRIVATSA, 2013)

Gerentes, diretores e líderes precisam interpretar os dados e as tecnologias da forma mais clara possível, extraindo deles as informações necessárias para a tomada de decisões de suas empresas, aquelas que o fazem sobressaem-se no mercado e adquirem vantagens em contraoposição aos concorrentes.

1.2 Objetivo

Este trabalho visa discutir, estudar e identificar as duas principais plataformas analíticas (*Big Data* e *Data Warehouse*), além de uma plataforma adicional, híbrida, comparando suas vantagens, desvantagens, cenários de aplicação, similaridades e diferenças. Cada qual podendo ser complementada ou associada com diversas tecnologias, as quais também são apresentadas e discutidas com bases em suas propriedades.

Por fim, propor soluções específicas para cada tipo de cenário conforme as características das empresas e dos dados, formulando e combinando a escolha das diferentes plataformas e tecnologias, definindo arquiteturas únicas.

1.3 Justificativas

Big Data é uma preocupação central em algumas verticais de mercado, como por exemplo em ciências (física, meteorologia, pesquisas biomédicas, genoma), instituições financeiras (bancos, seguros, mercado de capitais), governo (defesa, tesouro). Em contraste, modelos de base de dados relacionais costumam providenciar acesso aos dados de maneira única em sistemas eficientemente acoplados, em que o acesso fácil é dado pela linguagem *SQL (Structured Query Language,* Linguagem de Consulta Estruturada*)*, que é também comum e de fácil acesso em todos os níveis (TEKINER e KEANE, 2013).

Estas duas principais plataformas dominam o mercado de *Business Intelligence (BI*, ou inteligência de negócio*)*, e devido à velocidade do crescimento, as diferentes maneiras que cada vertical trata seus dados, as constantes inovações e a incessante mudança do mercado, dificultam a identificação das melhores opções de ferramentas e da maneira mais eficiente de aplicar suas variações e, sobretudo a existência de abundates tecnologias emergentes cambiáveis entre *Data Warehouse* e variações de *Big Data.*

Gerar valor sobre os dados depende diretamente dos estudos analíticos realizados sobre totalidade do material e, enquanto se discutia a melhor aproximação para a construção de uma *Data Warehouse*, de Kimball ou Inmon, a escolha foi tolhida com a vasta gama de tecnologias e técnicas criadas na tentativa de reger o *Big Data*.

Era necessário um estudo consolidado na área para reduzir o tempo e o esforço gasto de CIOs (*Chief Information Officer*), especialistas e até mesmo usuários na identificação de uma solução para *BI*.

1.4 Estrutura do trabalho

O Capítulo 1 apresenta as motivações, o objetivo, as justificativas e a estrutura do trabalho.

O Capítulo 2 começa com uma introdução ao *Data Warehouse*, segue definindo os conceitos, processos e estratégias.

O Capítulo 3 descreve o conceito de *Big Data*, uma breve história e os principais módulos envolvidos.

O Capítulo 4 estende a definição de *Big Data* comparando com os meios tradicionais de armazenamento e análise, conceituando os fundamentos e técnicas para estruturação dos dados analíticos, quais e como as tecnologias são utilizadas, apresentando exemplos ao longo do contexto.

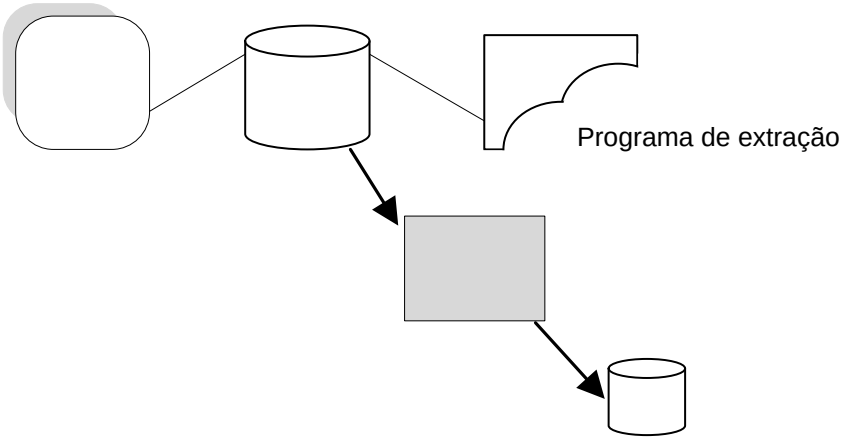
O Capítulo 5 contém proposições de plataformas e tecnologias para solucionar os principais problemas de *BI* e auxilia especialistas na escolha de uma tecnologia, apresenta vantagens e desvantagens do *Data Warehouse* e *Big Data* e finalmente incluí um exemplo prático.

O Capítulo 6 resenha os resultados do capítulo anterior, concluindo com comentários sobre o estudo.

2. DATA WAREHOUSE

O surgimento de tecnologias que possibilitavam escrita e leitura de dados de maneira rápida (diferentemente dos dados em fitas magnéticas), também chamado de *online transaction processing (OLTP)* revolucionou a maneira de como os dados eram tratados. Logo depois do advento dos sistemas *OLTP*, um programa para extrair os dados começou a emergir (INMON, 2005) e o objetivo era utilizar a informação extraída em outro ambiente, de forma que o processamento e a leitura não atrapalhassem as transações *online*.

Figura 2 - Processo de extração



(INMON, 2005)

O conceito surgiu na década de 1980 quando sistemas *OLTP* não conseguiam cumprir tarefas de análise, e se popularizou no início da década de 1990, quando Bill Inmon publicou um guia de como construir um *Data Warehouse.*

Como os dados neste momento estavam em ambientes segregados, foi possível obter o alto desempenho necessário para operações *online*, a concorrência (entre sistemas *OLTP* e tarefas de análise) foi eliminada e não havia mais conflito entre as transações, permitindo concomitantemente que o usuário tivesse controle total sobre a manipulação da informação, dando espaço ao *Online Analytical Processing (OLAP).*

Com a estrutura bem consolidada, o *Data Warehouse* pode se desdobrar em outras variáveis, desde sua formação, estruturação, arquitetura e softwares envolvidos. A

fonte mais comum para o *Data Warehouse* são as próprias bases de dados da empresa, podendo também extrair dados de planilhas, documentos ou até mesmo dados externos, sendo que a fonte deve ser transformada e importada antes de ser utilizada (R. KELLY RAINER, 2012).

De acordo com Inmon (2005, p.52), *Data Warehouse* é “conjunto de dados não voláteis, integrados, que variam no tempo, orientados ao negócio e que dão suporte a entidade gestora e tomadora de decisões”, ou com uma visão mais funcional de KIMBALL e ROSS (2013, p.25): “Um *Data Warehouse* é uma cópia dos dados transacionais especificamente estruturados para consulta e análise”.

2.1 Arquiteturas

A arquitetura de um *Data Warehouse* pode incluir (CHAUDHARY; MURALA e SRIVASTAV, 2011):

- Arquitetura de processo, que consiste nos passos para realizar a extração, transformação e carregamento dos dados;
- Arquitetura de tecnologia, que define a estrutura tecnológica e os protocolos de conectividade (*ODBC, JDBC, OLE DB* etc.);
- Arquitetura de recursos, se refere à variedade de softwares existentes para manter e gerir o *Data Warehouse*;
- Arquitetura de informação, que é a estrutura, passo a passo, da conversão dos arquivos de uma formatação para outra, que permite gerenciar as operações de visualização, exclusão, inclusão e alteração;
- Arquitetura do modelo de dados, que dita como os dados estão distribuídos e organizados.

Os principais tipos de implementação e distribuição de dados utilizados são (HAJMOOSAEI; KASHFI e KAILASAM, 2011):

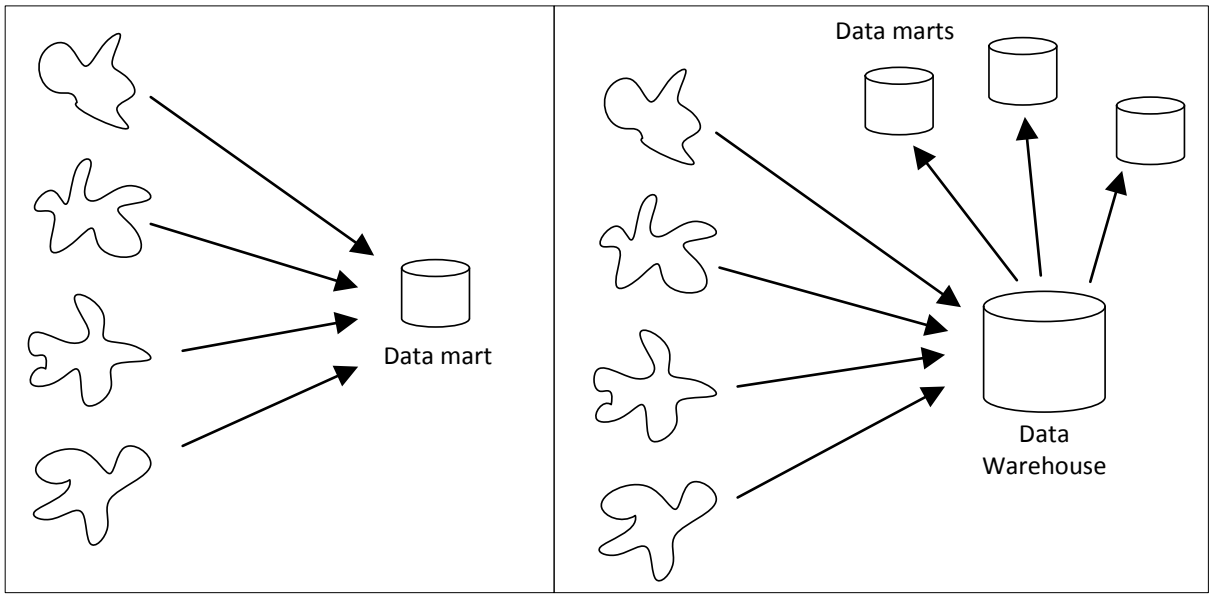
- *Data Warehouse* centralizado: Contém um único banco de dados com todas as informações para uma área de negócio, departamento, divisão ou até mesmo de toda a empresa.
- *Data marts*: São formas simples de *Data Warehouse*, focados numa única área, como vendas, finanças, marketing etc, podendo ser dependentes ou independentes (conforme a Figura 3). A diferença consiste na possibilidade do

último fornecer dados de uma fonte externa sem recorrer a um *Data Warehouse* central.

- Homogênea ou Heterogênea: São implementações e envolve apenas uma fonte ou múltiplas fontes respectivamente. Um *Data Warehouse* homogêneo contém apenas uma base, enquanto o heterogêneo, engloba diferentes Sistemas de Gerenciamento de Banco de Dados (HAJMOOSAEI; KASHFI e KAILASAM, 2011).

A Figura 3 ilustra, no lado esquerdo, como os *Data Marts* independentes são alimentados por cada área de negócio. Ao lado direito, cada área tem seus dados extraídos para o *Data Warehouse* centralizado, que por sua vez, exporta esses dados para os diferentes *Data Marts* dependentes.

Figura 3 – Data mart independente e dependente, respectivamente

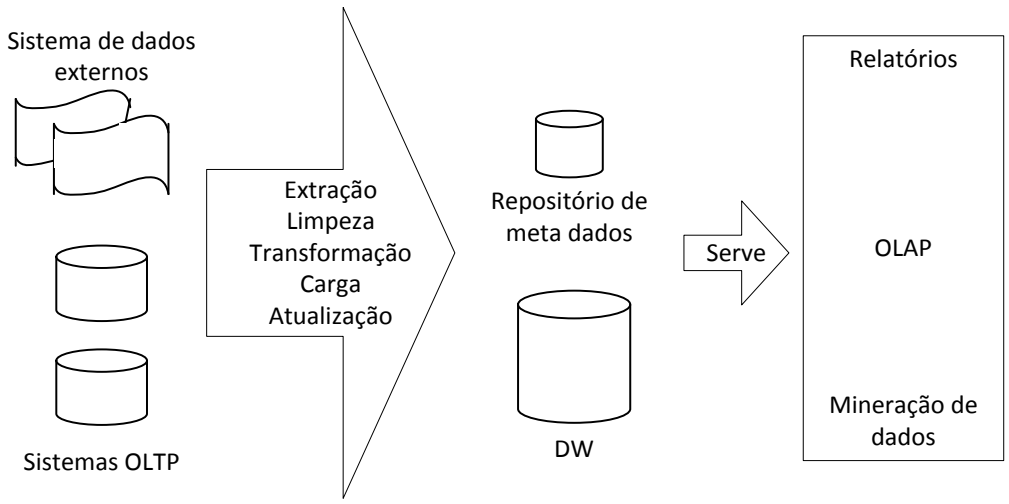


(INMON, 2005)

Pela Figura 4, observa-se um modelo de 3 camadas composto pela fonte de dados, na camada mais baixa, geralmente contendo banco de dados relacionais e ferramentas para a extração, limpeza e transformação dos dados, a camada intermediária composta por um servidor *OLAP*, podendo ser *ROLAP, MOLAP* ou *HOLAP* (o primeiro sendo relacional, o segundo multidimensional e o terceiro, híbrido)

e o repositórios de metadados e a última camada contendo as ferramentas de análises, relatórios e *Data Mining*.

Figura 4 - Modelo genérico de arquitetura de um Data Warehouse



(CHAUDHARY, MURALA e SRIVASTAV, 2011)

Hajmoosaei, Kashfi e Kailasam (2011) propõem uma lista de 18 parâmetros para comparação e escolha de arquitetura num *Data Warehouse*:

1. Independência de local
2. Alta eficiência
3. Implementação em curto prazo
4. Retorno do investimento em curto prazo
5. Baixo risco
6. Flexibilidade nas mudanças locais e globais
7. Baixo custo de implementação
8. Baixo custo de gerenciamento e manutenção
9. Baixo custo de comunicação para consultas locais
10. Ter visão integrada (*data integrity*)
11. Alta tolerância contra falhas
12. Sem ajuste para modelo de dados / meta dado
13. Baixo tráfego para consultas globais
14. Sem necessidade para comunicação de alta velocidade, estável e segura
15. Sem redundância

16. Sem restrição para espaço em disco
17. Sem conflitos para variáveis globais e locais
18. Baixa distância geográfica para os dados de sistemas de armazenamento operacionais locais

O nível de compatibilidade de cada parâmetro com a arquitetura, visualizado na Tabela 1, é dado de acordo com os seguintes critérios:

- Alto – Significa que o parâmetro é altamente compatível com a arquitetura;
- Médio – O parâmetro tem uma compatibilidade média;
- Baixo – O parâmetro é pouco ou nada compatível com a arquitetura.

Tabela 1 - Comparação de compatibilidade arquiteturas para Data Warehouse

Arquitetura / Parâmetro	Data Warehouse Centralizado	Data Mart Dependente	Data Mart Independente	Data Warehouse homogêneo distribuído	Data Warehouse heterogêneo distribuído
1	Alto	Baixo	Alto	Médio	Alto
2	Médio	Alto	Alto	Alto	Alto
3	Baixo	Baixo	Médio	Médio	Alto
4	Baixo	Baixo	Médio	Médio	Alto
5	Baixo	Baixo	Alto	Médio	Alto
6	Baixo	Baixo	Alto	Baixo	Médio
7	Baixo	Baixo	Médio	Médio	Alto
8	Baixo	Baixo	Alto	Alto	Alto
9	Baixo	Alto	Alto	Alto	Alto
10	Alto	Alto	Alto	Alto	Alto
11	Baixo	Alto	Alto	Alto	Alto
12	-	Alto	Baixo	Baixo	Baixo
13	Alto	Alto	Alto	Baixo	Baixo
14	Alto	Alto	Alto	Baixo	Baixo
15	Alto	Médio	Baixo	Baixo	Baixo
16	Baixo	Baixo	Alto	Alto	Alto
17	Baixo	Alto	Alto	Baixo	Baixo
18	Baixo	Baixo	Baixo	Alto	Alto
Compatibilidade com o plano ótimo	32%	47%	75%	52%	70%

(HAJMOOSAEI; KASHFI e KAILASAM, 2011)

A Tabela 1 auxilia na identificação de uma arquitetura ótima, dados os parâmetros propostos.

Dois estilos se sobressaem para construção. O estilo de INMON (2005) que segue uma abordagem *top-down,* diretamente ligada com o *Data Warehouse* Centralizado na Tabela 1, em que abordagem utiliza um projeto de modelos de dados normalizado centralizado e, KIMBAL e ROSS (2013), que conduz uma implantação *bottom-up*, em que os *Data Marts* são criados primeiramente, dando suporte a cada processo específico (coluna "*Data Mart* Independente" na Tabela 1), podendo ser integrado no futuro, criando uma visão mais geral com *Data Marts* dependentes.

2.2 *ETL* (Extração, Transformação e Carregamento)

O *ETL* (*Extraction, transformation, load* em inglês) é um processo complexo e exige muito planejamento. Essa é a atividade mais demorada e ao mesmo tempo a que exige mais atenção num *Data Warehouse*. Consequentemente, uma das principais diferenças ao que se refere à estruturação dos dados para as duas plataformas.
Em torno do ano 2000 e 2001, ficou cada vez mais claro que *ETL* é um processo de extrema importância para as tarefas de integração, já que é custoso, de trabalho intensivo e de missão crítica, fazendo com que se torne um fator imprescindível para o sucesso de um projeto *Data Warehouse*. Quanto às partes individuais do processo, ele pode ser dividido em (VASSILIADIS e SIMITSIS, 2009):

- Extração
  - A extração é o passo conceitualmente mais simples, objetiva a identificação do grupo correto de dados na fonte a qual deverão ser submetidos aos processos seguintes. Tal como o restante do fluxo geral, a extração também acontece nos períodos em que as atividades da empresa encontram-se reduzidas, normalmente após o horário de expediente (horário comercial). Duas restrições técnicas podem dificultar a tarefa:
    - A fonte deve sofrer pouca sobrecarga durante a extração, já que as outras atividades administrativas também podem ocorrer durante esse período e;
    - Por razões técnicas e políticas, os administradores são relutantes em aceitar intervenções grandes nas configurações do sistema e

consequentemente, deve ocorrer o mínimo de interferência com as configurações da fonte.

- Transformação
  - No geral, a transformação e as tarefas de limpeza lidam com os conflitos e problemas, podem ser distinguidos em dois níveis (LENZERINI, 2002):
    - Esquema: Onde ocorrem os conflitos de nome, que pode ser o mesmo para objetos diferentes (homônimos) ou nomes diferentes para o mesmo objeto (sinônimos)
    - Instância: O problema mais típico nesse caso ocorre quando há registros duplicados. Além disso, pode haver dificuldades em integrar dados com diferentes granularidades, como registros agrupados semanalmente e outros mensalmente.
  - VASSILIADIS e SIMITSIS (2009) vai além e mostra que existem outros fatores como “máscara” utilizada para representar valores (ex. sexo: ‘M’, ‘Masculino’, ‘1’).

- Carga
  - Atividade em que o desempenho tem mais influência, consiste em exportar os dados para a base que vai suportar o *DW*.

2.3 Modelos dimensionais

De um modo geral, um modelo é uma abstração e reflexão do mundo real e nos dá a habilidade de visualizar o imperceptível. A modelagem de dados é muito similar à definição anterior (BALLARD *et al.*, 1998). Viqarunnisa, Laksmiwati e Azizah (2011) dividem a modelagem em dois grupos: específico de domínio e genérico. A modelagem genérica possui nível elevado de abstração, enquanto a específica é desenvolvida com um baixo nível de abstração, moldando-se para um domínio particular.

Dois modelos principais são discutidos por Ballard et al. (1998), o modelo multidimensional e entidade relacionamento. Muito se foi discutido em relação à nomeação do melhor modelo, em determinadas condições sobre técnicas especificas,

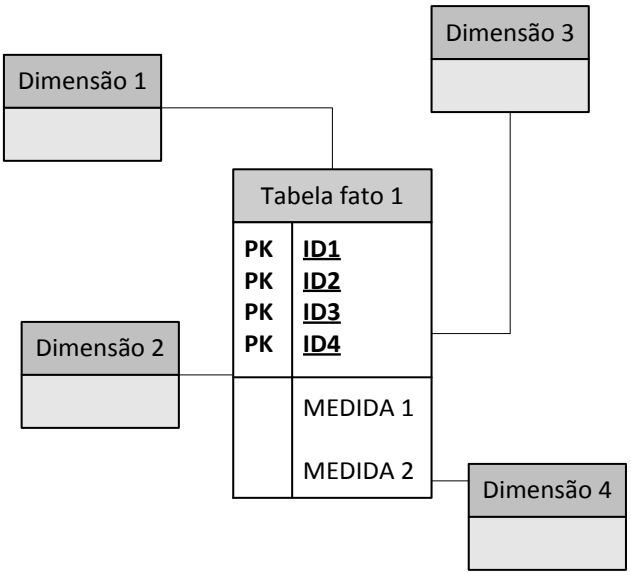
porém nunca se chegou a um concenso e não há uma resposta definitiva (ZHIJUAN; HONGCHANG e XUEFANG, 2012). O que existe são algumas diretrizes que podem ser seguidas para definir uma solução favorável devido a circunstâncias particulares num ambiente característico (BALLARD *et al.*, 1998).

Na ampliação da escolha do modelo, temos ainda o multidimensional, o qual abrange diferentes esquemas (CHAUDHARY; MURALA e SRIVASTAV, 2011), (INMON, 2005), conforme detalhado a seguir:

O esquema estrela consiste em uma ou mais “tabelas fato” (uma tabela que armazena os valores detalhados de medidas) referenciando as demais “tabelas dimensões” (que contém atributos dos registros da “tabela fato”). Recomendado para ambientes onde as consultas são simples por natureza (DWHWORLD, 2010). A Figura 5 apresenta um exemplo do esquema.

- Esquema Estrela

Figura 5 - Esquema estrela

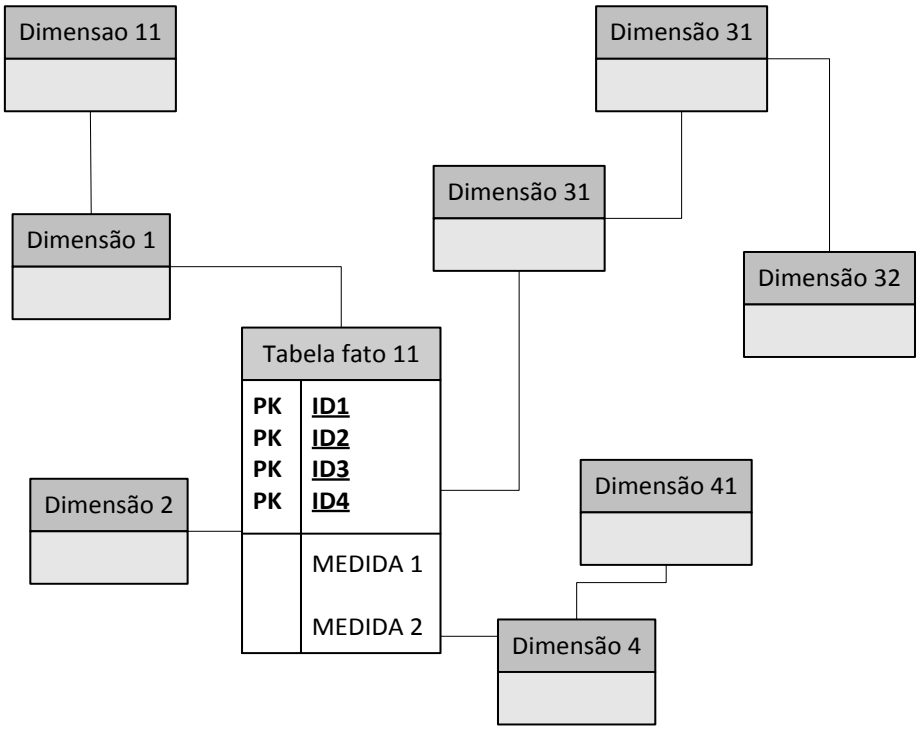


(INMON, 2005)

Da mesma maneira que um esquema estrela, o esquema floco de neve tem uma “tabela fato” centralizada, envolvido por “tabelas dimensões”, porem estas são normalizadas (ou seja, reduzir a redundância de dados e as chances dos dados se tornarem inconsistentes), formando o efeito “floco de neve”, conforme a Figura 6.

- Esquema Floco de neve (*Snowflake*)

Figura 6 - Esquema floco de neve

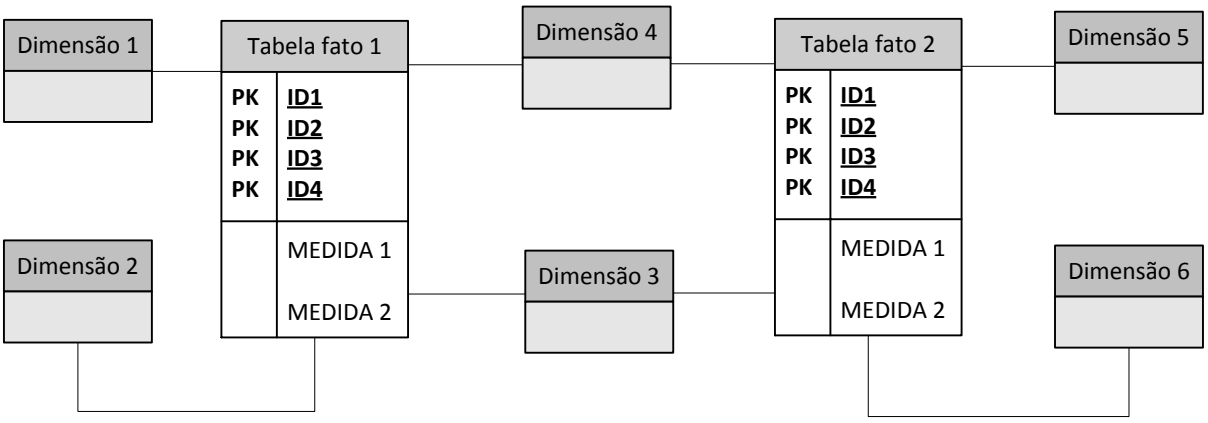


(INMON, 2005)

Composto de múltiplas “tabelas fato” com dimensões em comum, o esquema “*galaxy*” é geralmente composto pela junção de diferentes *Data Marts*, conforme pode ser observado na Figura 7.

- Esquema *Galaxy*

Figura 7 – Modelo "Galaxy"



(INMON, 2005)

Um modelo dimensional é um ótimo início para o processo de construção de um *Data Warehouse*, no entanto, existe um trabalho significativo que precisa ser realizado antes de se iniciar o desenvolvimento. Transformações devem ocorrer para base da criação desse modelo (MOHAMMAD RIFAIEA, 2008):

- A remoção de dados puramente específicos para a operação da aplicação;
- A adição de elementos de uma chave de dimensão de tempo para a estrutura, se já não existir;
- A adição dos dados derivados adequados;
- Transformação do relacionamento entre dados para artefato de dados. Esses artefatos são uma maneira de capturar relações atuais para entidades que mudam suas dependências ao longo do tempo;
- Acomodar diferentes níveis de granularidade encontrados no *Data Warehouse*;
- Realizar a junção dos dados similares de diferentes locais;
- Múltiplas ocorrências da mesma entidade devem ser combinadas, criando apenas um registro na base. Isso reduz o trabalho de indexação na hora de buscar múltiplas ocorrências da mesma entidade, e pode reduzir significativamente o custo da sumarização e relatórios de informação;
- A separação dos atributos de acordo com as características de estabilidade. Isso significa agrupar as propriedades baseado na propensão de mudança.

Ao verificar todos os modelos verifica-se que a escolha afeta diretamente a legibilidade dos dados e as transformações que devem ocorrer, servindo como base toda e qualquer integração futura de fontes adicionais. Um modelo simples, como o esquema estrela, pode ser escolhido para facilitar consultas e leituras, porém pode implicar dificuldades na adaptação dos dados.

2.4 Metadados

A definição mais comum sobre metadados é que representam são dados sobre dados, ou seja, contêm informações que possibilita o entendimento do que os dados significam. Em *Data Warehouse*, os metadados definem os objetos, as entidades e agem como um diretório, ajudando a localizar o conteúdo dos dados. O Metadado e o *Data Warehouse* são Base de dados ou instâncias diferentes, podendo ou não residir em hardwares distintos (MORASCHI, 2013).

2.5 Ferramentas e técnicas

As ferramentas podem ser divididas em duas categorias principais: *front-end* e *back-end* (CHAUDHARY, MURALA e SRIVASTAV, 2011). As ferramentas *back-end* são responsáveis pelos processos de extração, transformação, limpeza e armazenamento dos dados. As ferramentas *front-end* (mais comumente chamadas de OLAP) são responsáveis pela visualização dos dados e são distribuídas por diferentes fornecedores.

Dependendo da te categorias: OLAP Multidimensional (MOLAP) e OLAP Relacional (ROLAP) (ZHIJUAN, HONGCHANG e XUEFANG, 2012).

2.6 Conclusão do capítulo

O *Data Warehouse*, estabelecido há três décadas, pode ser implementado de diversas maneiras, que devem se adequar as características do negócio. Apesar das diferentes arquiteturas e modelos, como os processos são bem definidos, é simples realizar um planejamento assertivo e ter um ambiente pronto para análise a partir de seis meses. Desde sua concepção, as ferramentas e ténicas evoluíram baseadas em experiências

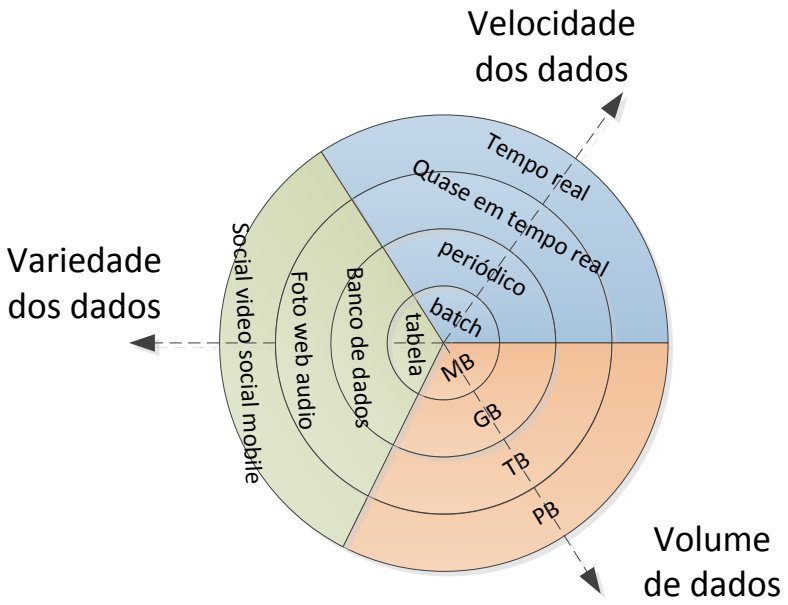
e casos de sucesso. Algumas tecnologias envolvidas no processo serão discutidas no capítulo 5.

3. BIG DATA

O volume de dados vem crescendo consideravelmente nos últimos anos com as introduções das redes sociais e dados gerados automaticamente por registros, sensores, transações online, aplicações e arquivos de multimídia como vídeos, áudios e imagens (SAGIROGLU e SINANC, 2013).

Por esse motivo, *Big Data* requer um modelo revolucionário sobre a maneira tradicional de análise, que é caracterizado pelos seus 3 principais componentes, conforme mostra a Figura 8 que demonstra como cada dimensão pode crescer independentemente.

Figura 8 - Três componentes principais do Big Data: Volume, Velocidade e Variedade

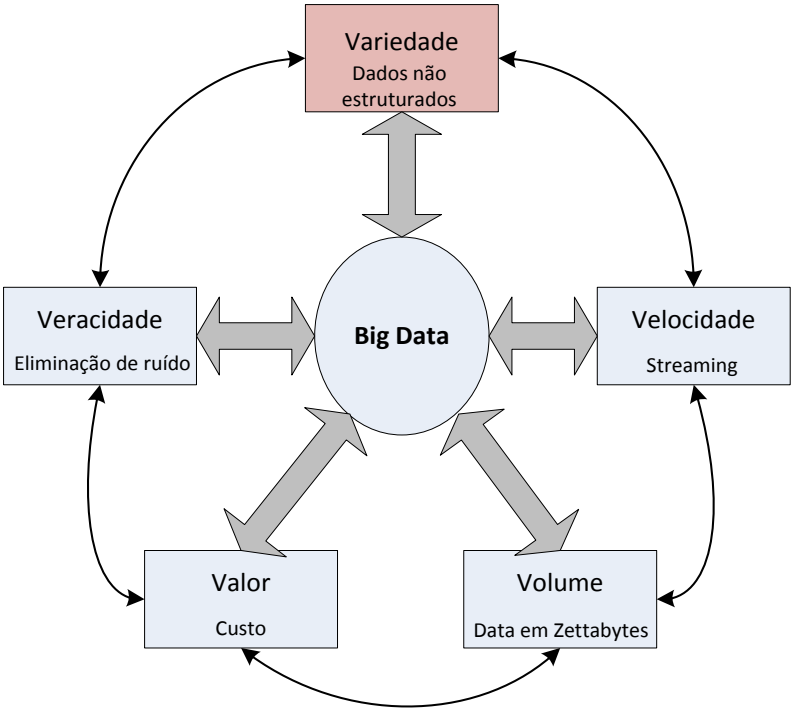


(ZHANG e HUANG, 2013)

As primeiras descrições das características do *Big Data* apareceram em 2001, quando Laney (2001) identificou 3 atributos chaves para a medição de dados: alta variedade, alta velocidade e alto volume, porém anteriormente, COX e ELSWORTH (1997) utilizaram o termo pioneriamente quando descreveram que a visualização fornecia um desafio interessante para os sistemas de computadores: Os grandes *Data Sets* (conjunto de dados) ultrapassam as capacidades de memória, do disco local e até disco remoto.

Para simplificar o significado de *Big Data*e sua implicação, Lomotey e Deters (2014) define cinco conceitos (5V) do modelo, conforme a Figura 9, ante o modelo 3V.

Figura 9 - Modelo 5 Vs



(LOMOTEY e DETERS, 2014)

Os 5 Vs representam:

- Volume: A quantidade dos dados gerados nas empresas e corporações cresce numa taxa exponencial. As redes sociais (o twitter sozinho com 12 Terabytes de dados diários) estão gerando dados que em breve entrarão na casa de Zettabytes diários (RUPANAGUNTA; ZAKKAM e RAO, 2012).
- Variedade: Os dados são de múltiplas fontes e em formatos heterogêneos, como multimídia, texto, blogs, e-mails, sensores etc., além de não conterem esquema, o que desloca os dados para o lado semiestruturado.
- Velocidade: A taxa com que os dados estão sendo gerados mudou o foco de "*data sets*" (batch) para "*streaming data*".
- Valor: Enquanto os dados estão sendo gerados, coletados e analisados, é importante dizer que os dados têm custos. Os dados podem ser considerados

"*commodities*" (mercadorias) e serem vendidos para terceiros para aumento de receita.

- Veracidade: Existe uma necessidade de se verificar a acurácia dos dados eliminando o ruído com metodologias como sanitização. Esse procedimento é necessário para garantir a qualidade dos dados e a efetividade das análises

Ainda estendendo a definição de *Big Data*, o detalhamento de Demchenko, De Laat e Membrey (2014) dividem 5 partes cujas características e a infraestrutura dos componentes são descritas a seguir:

1. Os 5 conceitos (5 Vs), descritos na Figura 9.

2. Um novo modelo de dados
   - Relacionamento dos dados, proveniência e integridade de referência;
   - Ciclo de vida dos dados e variabilidade / Evolução.

3. Um novo modelo analítico
   - Análise em tempo real, interativa e autoaprendizagem de máquina.

4. Novas ferramentas de infraestrutura
   - Infraestrutura baseada em nuvem para armazenagem, processamento, rede e computação de alta performance;
   - Integração de serviços providos por diferentes fornecedores heterogêneos com múltiplos *stakeholders;*
   - Modelos de segurança para infraestrutura de processamento e armazenamento confiáveis.

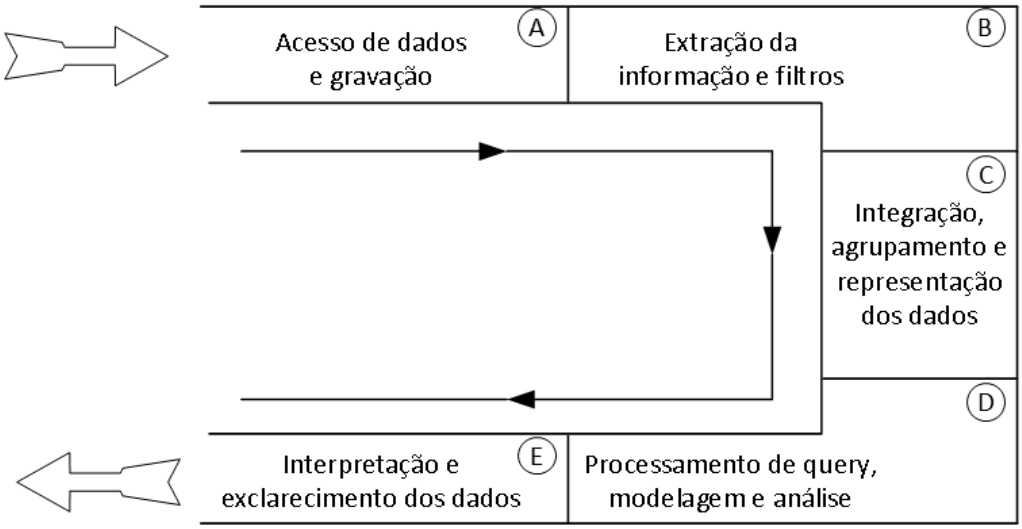
5. Novas fontes de dados e objetivos, que muitas vezes definem os tipos de dados e as estruturas (por exemplo, dados correntes, dados correlacionados etc.)
   - Alta velocidade para captação de uma variedade de sensores e fonte de dados;
   - Entrega de dados para diferentes tipos de visualizações e consumidores;

- Entrada e saída totalmente digitalizados, sensores de rede, controle digital completo.

Adicionalmente, os autores definem o ecossistema de *Big Data* não apenas como uma problemática de banco de dados ou *Hadoop* (explicado na sessão seguinte), que constituem o núcleo da tecnologia, mas o complexo todo de componentes para armazenar, processar, visualizar e entregar os resultados para o seu alvo.

O encadeamento *(pipeline)* dos dados é muito similar ao *DW*, sendo composto pelas fases contidas na Figura 10 (MATHUR, SIHAG, *et al.*, 2014):

Figura 10 – Modelo do pipeline do processamento de dados do Big Data



(MATHUR, SIHAG, et al., 2014)

A. O acesso e a gravação dos dados ocorrem sem tratamento na primeira fase.
B. Como os dados não estão em formato adequado para serem lidos, deverão receber transformação e seleção.
C. Gravar os dados e enviar ao banco de dados não é suficiente, portanto, de maneira similar aos metadados do *Data Warehouse*, essa parte é responsável por integrar a informação.
D. Onde os algoritmos para *Data Mining* são convertidos para *Big Data*.
E. A principal tarefa do *pipeline* acima é fazer análises, que é realizada nessa etapa.

3.1 Hadoop

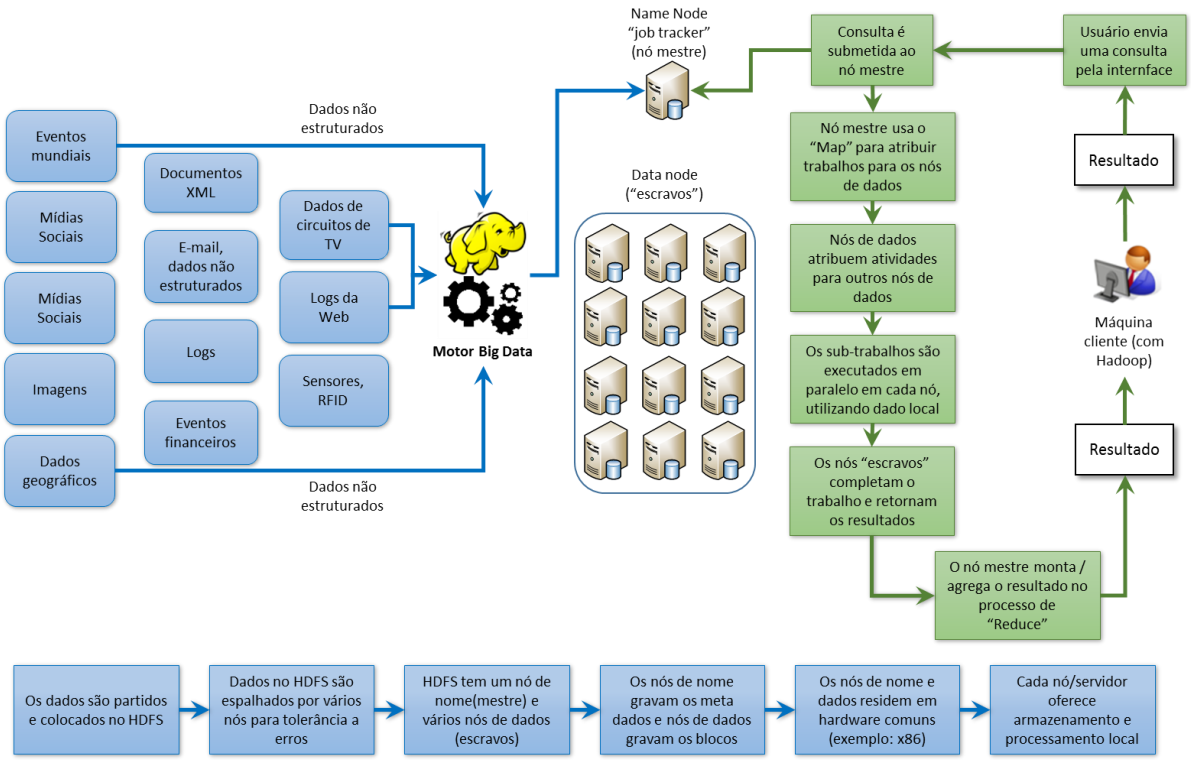
Esse importante componente que está muito presente no *Big Data* é uma biblioteca *open source* (código aberto*)* que inclui um *framework* (abstração de vários códigos provendo funcionalidade genérica) o qual permite o processamento distribuído de *Data Sets* enormes entre *clusters* (aglomerações) de computadores utilizando modelagem de programação simples (CHANDARANA e VIJAYALAKSHMI, 2014). Escrito em Java, ele é executado em um cluster de servidores configurados com um armazenamento direto. Existe uma variedade de opções, desde um simples computador até milhares deles, cada um oferecendo poder de processamento e armazenamento.

A pilha do *Hadoop*, que constitui seus componentes, módulos e projetos relacionados, é listada abaixo (APACHE, 2014):

- *HDFS:* Um sistema de arquivo distribuído de alta tolerância a erros responsável por armazenar os dados no *cluster*;
- *MapReduce:* Uma ferramenta de programação paralela para distribuição nos *clusters* de processamento. É detalhada no capítulo 4.2;
- *Hbase:* Uma base de dados escalável para acesso de leitura/escrita aleatórios.
- *Pig:* Uma ferramenta de programação para criar programas *MapReduce*;
- *Hive:* Uma aplicação de base de dados que fornece uma interface *SQL* e modelo relacional;
- *Sqoop:* Um projeto que transfere dados entre bases relacionais e o *Hadoop*;
- *Avro:* Um sistema de serialização de dados;
- *Oozie:* Um fluxo para atividades dependentes do *Hadoop*;
- *Chukwa:* Um subprojeto do *Hadoop* de monitoração de acumulação de dados em sistemas distribuídos;
- *Flume:* Um sistema distribuído para coletar, agregar e mover grandes quantidades de dados de diferentes fontes no *HDFS* ou outra central de dados;
- *ZooKeeper:* Um serviço centralizado para fornecer sincronização distribuída e serviços de grupo;
- *HiveQL:* uma linguagem próxima do *SQL*, que é compilado em *jobs* (atividades) de *MapReduce.*

Da lista anterior, o *HDFS* (*Hadoop Distributed File System*), o componente de armazenamento e uma das principais partes do *Hadoop*, foi desenvolvido principalmente para suportar a tolerância a erros e fornecer escalabilidade. Os arquivos gravados são divididos em blocos (geralmente de 64 a 128 Mb) e replicados em três ou mais servidores. O *HDFS* também fornece *APIs* (Interface de Programação de Aplicações) para o *MapReduce*, leituras e escritas em paralelo. Capacidade e performance são facilmente escaláveis com a adição de nós. A Figura 11 ilustra o processo básico e, a captura e distribuição de dados no *Hadoop*.

Figura 11 - Armazenamento e pesquisa de dados no HDFS



(ECOMCANADA, 2013)

O processo representado pela Figura 11 para armazenamento e consulta é realizado com os seguintes passos:

1. Partição dos dados e armazenamento no *HDFS;*
2. Distribuição pelos nós;
3. Gravação dos metadados;

As consultas são enviadas ao nó mestre (*job tracker*, em inglês, responsável pelo gerenciamento de todo *cluster*), que por sua vez envia o comando de trabalho aos nós escravos. Os subtrabalhos são executados em paralelo e o resultado retorna ao usuário.

3.2 MapReduce

Em 2004, o Google publicou um artigo (DEAN e GHEMAWAT, 2004) sobre um processo chamado *MapReduce* que utilizava uma arquitetura de distribuição e processamento paralelo. Esse procedimento revolucionou e melhorou substancialmente o desempenho dos métodos convencionais de processar e gerar grandes *data sets*. O projeto teve muito sucesso e originou o *Hadoop* da *Apache.*
O programa recebe um conjunto de chaves e valores e produz na saída outro conjunto de chaves e valores. O programador expressa a computação em duas funções: *Map* e *Reduce*.

Na atividade do "*Map*", o nó de dados mestres é quebrado em vários subproblemas menores. Um nó processa alguns grupos de problemas sob controle do rastreador de processos (*job tracker*) e guarda o resultado no sistema de arquivos local, onde o passo "*Reduce*" possa acessar.

Um processo intermediário, "*Shuffle"*, redistribui os dados baseado nas chaves produzidas pelo passo *"Map"*. Os dados que pertencem a mesma chave são sempre distribuídos para o mesmo nó. Nem sempre esse passo é necessário.

Na atividade de "*Reduce*" os dados gerados na fase *"Map"* são analisados e trabalhados. Pode haver múltiplas tarefas de "*Reduce*" para paralelizar a agregação e essas tarefas são executadas nos nós de trabalho sob a supervisão do rastreador de processos.

Como um exemplo, suponha que um aplicativo realize contagem de palavras numa coleção de documentos. A função de *Map* realizaria a emissão de cada palavra juntamente com a contagem de suas ocorrências para cada nó processado:

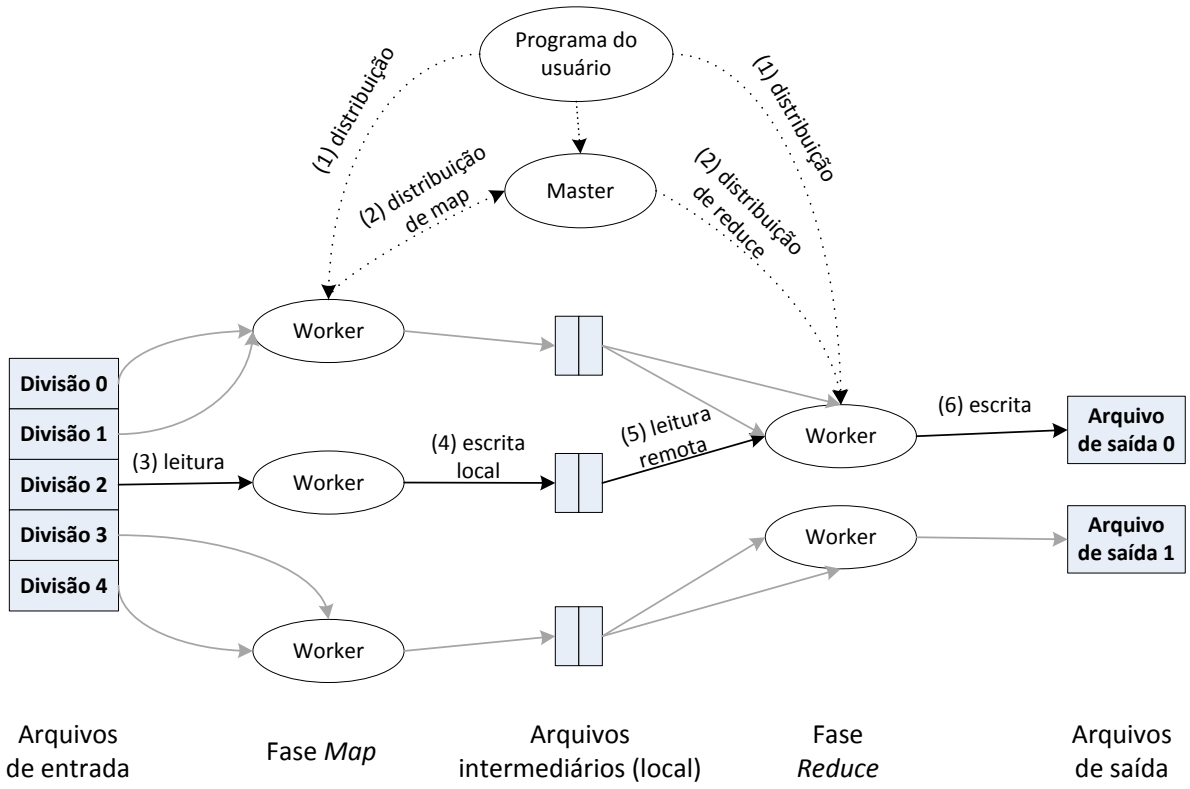
`Map(k1,v1)` → `list(k2,v2)`

Dada a resposta da fase de *Map*, a função *Reduce* é então aplicada para cada grupo em paralelo que, em troca, produz a contagem de cada palavra e sua soma total:

`Reduce(k2, list (v2))` → `list (v2)`

A Figura 12 ilustra o fluxo do MapReduce:

Figura 12 - Funcionamento do MapReduce



(DEAN e GHEMAWAT, 2004)

Juntamente com o *HDFS*, *o MapReduce* é um dos principais componentes do *Hadoop*.

3.3 Arquitetura

A tecnologia que suporta o fenômeno de como armazenar e recuperar os dados de um grande repositório pode ser realizada de diversas maneiras. Duas arquiteturas

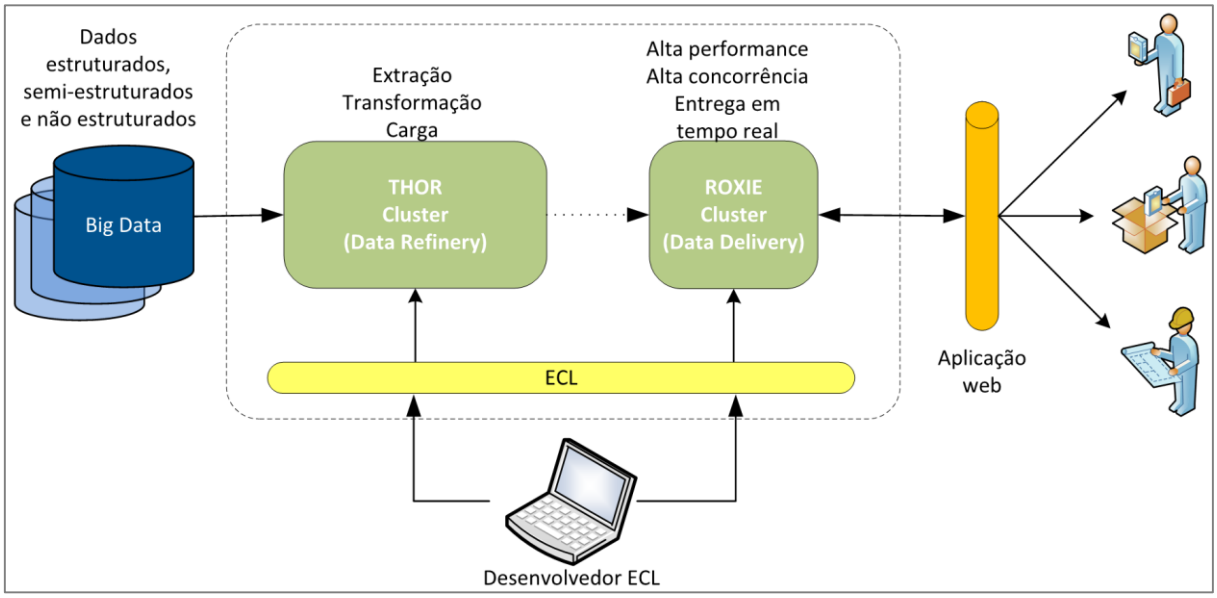
principais são apresentadas por Sagiroglu e Sinanc (2013) como soluções para *Big Data.*

A. Arquitetura *HPCC (High Performance Cloud Computing):*

Uma arquitetura *open source* que pode ser vista na Figura 13, desenvolvida pela *LexisNexis Risk Solutions* que incorpora *clusters* de computadores para alto desempenho e processamento paralelo para aplicações utilizando *Big Data*. Com três componentes principais, a plataforma não exige o uso de ferramentas de terceiros (HPCC, 2014):

- *Thor:* Um motor *ETL* paralelo massivo que permite integração de dados numa escala grande e manipulação de dados em *agrupamentos.*
- *Roxie:* Motor de recuperação de dados de alta performance, baixo tempo de resposta e alta taxa de transferência que permite busca de dados e *consulta* estruturada.
- ECL: A linguagem *Enterprise Control Language* é distribuída entre os nós com sincronização automática dos algoritmos de uso simples e otimizada para consultas e transações.

Figura 13 - High Performance computing cluster (HPCC)

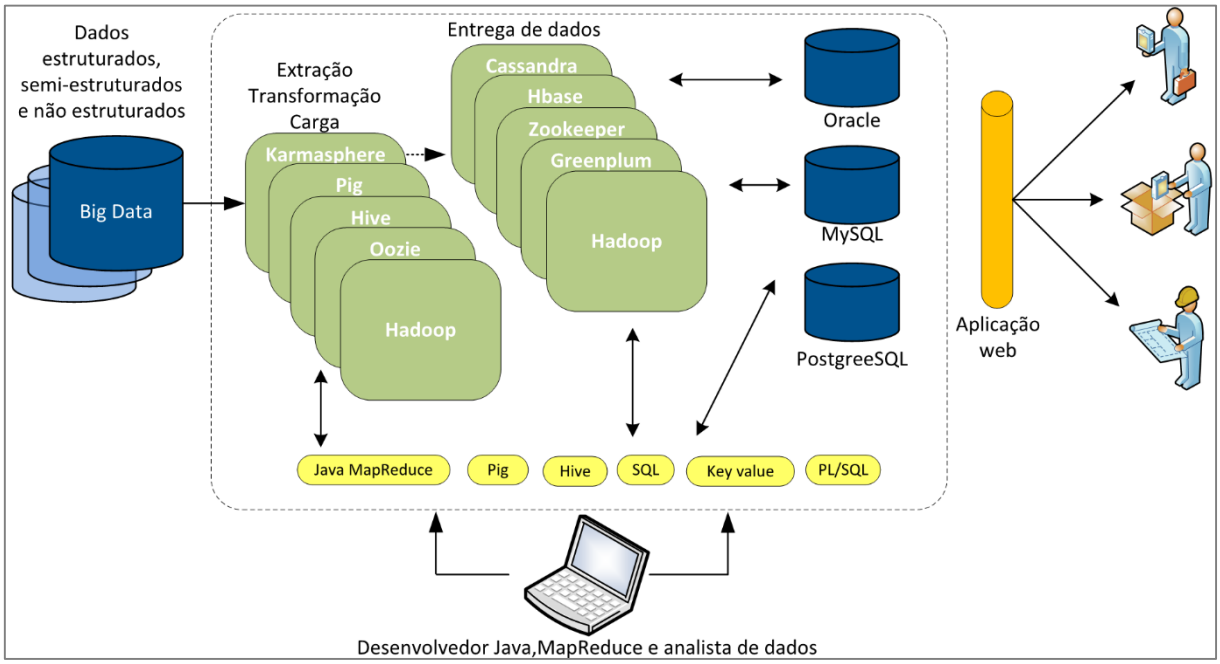


(SAGIROGLU e SINANC, 2013)

B. Arquitetura *Hadoop:*

Com *framework* baseado em Java e plataforma heterogênea e *open source*, ele oferece os itens listados na seção 4.1. Essa arquitetura (SAGIROGLU e SINANC, 2013) está representada na Figura 14, onde é possível observar a pilha do *Hadoop* que realiza a extração da carga e entrega dos dados, sendo suportado pelas bases de dados.

Figura 14 - Hadoop Stack



(SAGIROGLU e SINANC, 2013)

Alguns fornecedores de plataformas utilizadas em *Big Data* (ABOUZEID; BAJDA-PAWLIKOWSKI *et al.*, 2009), (LOSHIN, 2013) são*:* IBM InfoSphere BigInsights, WX2, Kognitio Analytical Platform, ParAccel Analytic Platform, IBM InfoSphere Streams, SAND Analytic Platform, Exasol, Greenplum.

3.4 *Data Mining* para *Big Data*

Técnicas de mineração e recuperação de dados para *Big Data* são basicamente diferentes dos meios convencionais de análise e estatística de amostras pequenas ou simples. Os dados são geralmente desordenados e não confiáveis. Além disso, os dados não estruturados podem ser mais valiosos que as amostras menores, estruturadas (MATHUR; SIHAG *et al.*, 2014).

A mineração de *Big Data* requer dados filtrados, integrados, confiáveis, facilidade no acesso, algoritmos escaláveis e ambientes que são próprios para efetuar a análise. Por outro lado, a mineração também pode ser utilizada para melhorar os padrões de confiança nos dados trabalhados (ZAÏANE, 1999).

A Tabela 2 lista os principais tópicos de *Big Data* e seus respectivos usos ou aplicações, apresentados para ilustrar a necessidade de análise detalhada desses aspectos:

Tabela 2 - Origem e domínios de Big Data

Origem do Big Data	Principal uso
1. Ciência	(a) Descoberta científica
2. Telecomunicação	(b) Novas tecnologias
3. Indústria	(c) Controle do processo, transporte
4. Negócio e varejo	(d) Serviços pessoais, campanhas, suporte de vendas, planejamento de cadeia logística
5. Cidades, setor púbico	(e) Planejamento urbano, rotas inteligentes, resposta emergencial, propagandas específicas por geolocalização
6. Mídia social e rede	(f) Apoio a saúde, melhora de performance, descobrir necessidades
7. Saúde	

(DEMCHENKO, DE LAAT e MEMBREY, 2014), (SAGIROGLU e SINANC, 2013)

3.5 Conclusão do capítulo

O consenso geral no que se refere a definição de Big Data é que não há um padrão definido de arquitetura, modelos ou ambiente para a tecnologia. Um conjunto de soluções aplicado a um problema relacionado ao excesso de volume, variedade ou velocidade significa tratar os “grandes dados” ou Big Data. Existe uma série de técnicas e tecnologias conhecidas e aplicáveis na maioria dos casos, que assistem a resolução dessas dificuldades.

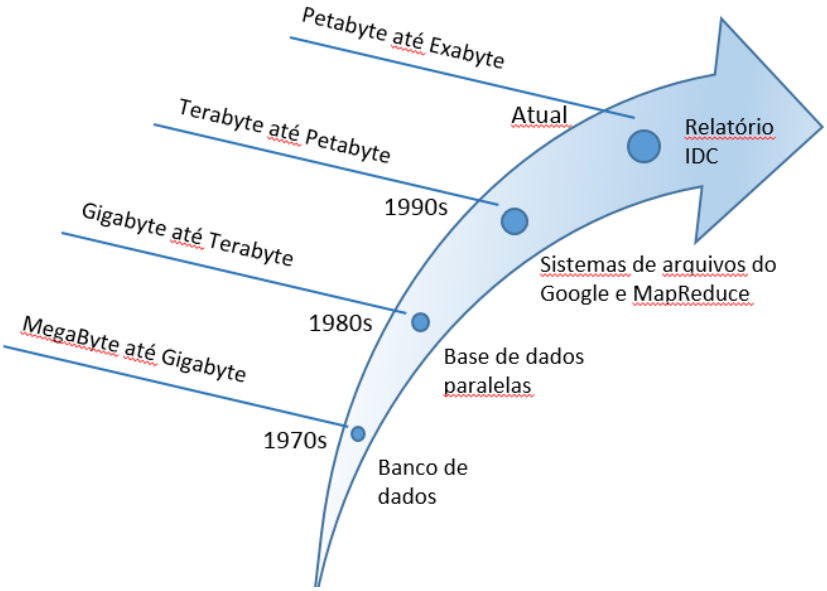
4. *BIG DATA* E *DATA WAREHOUSE* – AS TECNOLOGIAS ENVOLVIDAS E AS DIFERENÇAS

Quando um estudo analítico se transforma em *Big Data*? O tamanho que constitui *Big Data* cresceu de acordo com a Lei de Moore. Em 1975, participantes da primeira conferência *VLDB* (*Very Large Databases* – Banco de dados muito grandes) se preocuparam em lidar com milhares de dados encontrados no censo dos Estados Unidos (SIMONSON e ALSBROOKS, 1975). No contexto, a noção de “grande” eram apenas alguns milhões de dados. Este trabalho analisa soluções para casos com escala de *Data Sets* na web, ordens de magnitude bem maiores.

Em geral, *Big Data* significa dados que não podem ser processados de maneira convencional. Uma planilha cabe na memória, e é relativamente veloz para determinar se os dados estão limpos, se os valores são razoáveis e os resultados podem ser calculados e consultados instantaneamente. Em contraste, um grande conjunto de dados não cabe na memória, realizar uma consulta pode levar muito tempo, existe dificuldade na visualização desses dados e nem sempre são estruturados. Pode ser ainda tão grande que não caberia no disco local, sendo necessário realizar a distribuição em vários nós com armazenamento fisicamente distintos (FISHER; DELINE *et al.*, 2012).

Atingir um consenso sobre a definição de *Big Data* é difícil. Uma das alternativas lógicas seria abranger todas as definições alternativas, cada qual com um foco específico no aspecto de *Big Data* (HU; WEN *et al.*, 2014). A história e evolução dos dados do que era e ainda é *Big Data* pode ser visto na Figura 15. Até 1990 e quase 2000, as bases relacionais não tinham muitos problemas em armazenar os dados, porém uma complicação surgiu quando as empresas perceberam que o formato e tamanho de seus dados não eram mais comportados pelos métodos comuns para a época.

Figura 15 - História dos dados e sistemas



(HU, WEN,* et al.*, 2014)

De acordo com Madeen (2012), *Big Data* é muito grande, rápido e difícil de se processar. Grande na ordem de petabytes e de variadas fontes, rápido a ponto do processamento ter que detectar fraude no ato da venda ou decidir a melhor publicidade para um usuário no momento da sua visita na web e difícil porque as análises tradicionais não são adequadas o suficiente para a análise e os dados são variados.

Grande parte dos sistemas comerciais comportam uma enorme quantidade de dados (Microsoft SQL Server, Oracle Database, Teradata, Vertica), assim como os concorrentes *open source* (Cassandra, HBase, Project Voldermort) (RABL; SADOGHI *et al.*, 2012).

Este capítulo trata das diferenças entre *Big Data* e *Data Warehouse*. Neste trabalho, *Big Data* não significa *MapReduce, Hadoop, SGBDR* (Sistema gerenciador de base de dados) ou *HDFS*, mas sim o conjunto de soluções que algumas vezes propõem uma extensão desses componentes ou conforme definido por Mohanty, Jagadeesh e Srivatsa (2013), uma nova geração de tecnologias e arquiteturas desenhadas para economicamente extrair valores de grandes volumes e variedades de dados, possibilitando captura de alta velocidade, descoberta ou análise, sendo capaz de gerenciar e armazenar as características que constituem o *Big Data*. O escopo de

Data Warehouse, uma tecnologia mais madura e definida, está limitado às análises realizadas sobre um *SGBDR*, contendo essa base mais de um sistema físico ou não.

As propriedades desejadas no tratamento de dados massivos devem conter (HU; WEN *et al.*, 2014):

- Representação dos dados – Uma representação competente ajuda a criar valor sobre a análise.
- Redução de redundância e compressão de dados – Tipicamente existe muita redundânca em grupos de dados brutos. Eliminar as redundâncias e a compressão dos dados aliviam a sobrecarga dos sistemas.
- Gerenciamento do ciclo de vida – Saber ao certo quando manter dados históricos e a frequência de atualização são importantes para manter uma boa gerência.
- Privacidade e segurança dos dados – Sua importância cresceu com a proliferação de dados onlines e a facilidade de acesso.

Os principais atributos por para cada plataforma podem ser vistos no resumo da Tabela 3:

Tabela 3 - Comparação entre BI tradicional e Big Data

	Dados Tradicionais	Big Data
Volume	GB	Constantemente atualizado (TB ou PB atualmente)
Velocidade	Diariamente, de hora em hora …	Mais rapidamente
Estrutura	Estruturado	Semi-estruturado, não estruturado
Fonte	Centralizado	Totalmente distribuído
Integração	Fácil	Difícil
Armazenamento	SGBDR	HDFS, NoSQL, SGBDR
Maturidade	Muito maduro	Novo, em desenvolvimento
Tolerância a falhas	Somente no nível de transação	Muito tolerante
Acesso	Interativo	Por *batch* ou em tempo real

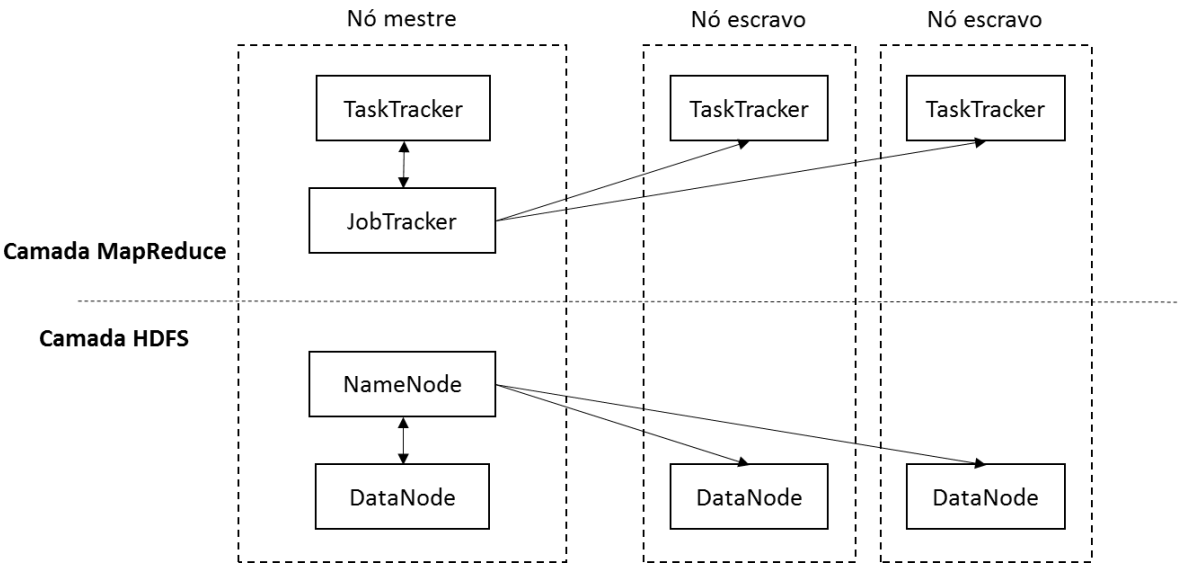
(HU; WEN* et al.*, 2014), (MOHAMMAD RIFAIEA, 2008), (GOSS e VEERAMUTHU, 2013)

As sessões seguintes se dedicam a explicar algumas das tecnologias e soluções que são responsáveis pelas diferenças e similaridades atuais.

4.1 Sistemas Distribuídos

Uma das principais estratégias para análise de *Big Data* é o *Hadoop*. Podendo ser utilizado numa grande variedade de plataformas, utiliza o *HDFS*, um armazenamento distribuído, rápido e tolerante a erros. Foi projetado para armazenar grande quantidade de dados de maneira efetiva e confiável. Seus blocos são divididos igualmente, replicados e essas mesmas configurações podem ser personalizadas. A arquitetura de alto nível do *Hadoop* pode ser verificada na Figura 16.

Figura 16 - Arquitetura de alto nível do Hadoop



(MANIKANDAN e RAVI, 2014)

Para processar os dados, o *Job Tracker* atribui algumas atividades para o *Task Tracker*. O *Name Node* toma as decisões de replicação dos blocos e recebe periodicamente os *Heartbeats* (sinais para demonstrar que um nó está ativo) e *Block Report* (contendo a lista de todos os bloqueios num nó de dados).

O *Hadoop* utiliza o *MapReduce* para distribuir o processamento e funciona com dados estruturados e não estruturados. A popularidade do MapReduce pode ser atribuída devido à sua alta escalabilidade, tolerância a erros, simplicidade e independência de linguagens de programação ou ao sistema de armazenamento (GROLINGER; HAYES *et al.*, 2014).

Já as bases de dados paralelas baseadas em *SGBDR* surgiram pela necessidade de realizar a gestão de um enorme volume de dados (RAY, 2009) e escalam muito bem até dezenas de nós (podendo escalar linearmente em alguns casos). Diferentemente de sistemas construídos com o *MapReduce*, existem poucas bases relacionais conhecidas que contenham centenas de nós e ainda menos contendo milhares de nós. Algumas razões impeditivas para escala de volumes tão grande de nós incluem falhas, que tem aumento de probabilidade com a adição de hardware, além da exigência de homogeneidade entre os subsistemas (ABOUZEID; BAJDA-PAWLIKOWSKI *et al.*, 2009).

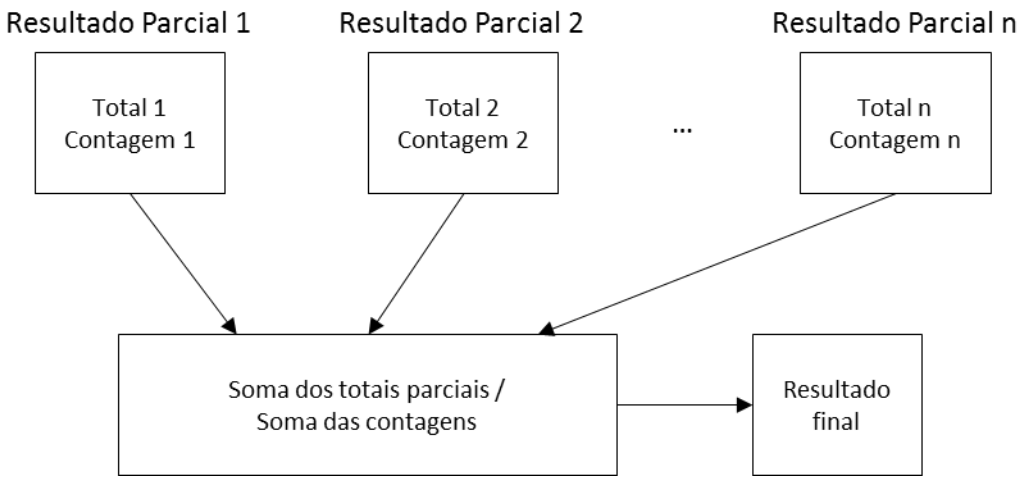
Para um DW, a disponibilidade é sempre de grande preocupação, pois o processamento é distribuído entre os vários nós e depende de todos eles para obter a resposta final. No caso de falha ou problema de comunicação com algum nó, o resultado final se torna incompleto (ANTUNES e FURTADO, 2007). Nas bases paralelas as consultas SQL são automaticamente convertidas e distribuídas entre os nós, enquanto no MapReduce, a programação deve ser feita pelo usuário.

Em consultas de bases paralelas, similarmente ao MapReduce, a tarefa envolve múltiplos nós realizando processamentos simultaneamente, ou seja, o problema inicial é quebrado em várias subatividades e posteriormente os resultados parciais são processados. Por exemplo, uma função de agregação que calcula média de um certo valor será automatizada para calcular a soma e a contagem de seus subgrupos, para, na agregação final, calcular a média. A Figura 17 ilustra o fluxo dos resultados.

Resultados parciais: `Rp1..Rpn = list(SUM(), COUNT())`

Resultado Final: `Rf = SUM(list(SUM())) / SUM(list(COUNT()))`

Figura 17 - Fluxo das consultas em bases paralelas



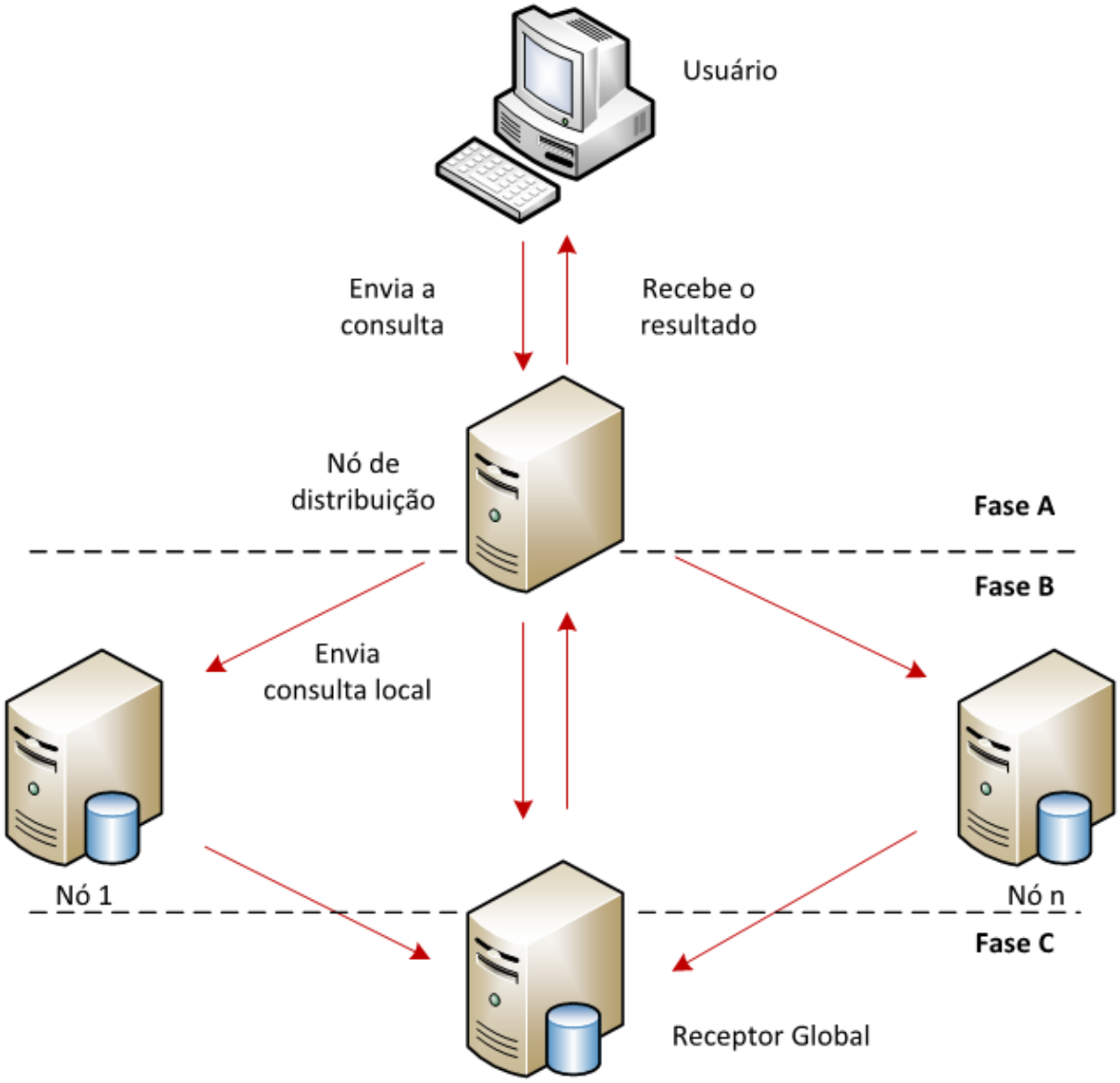
(ANTUNES e FURTADO, 2007)

Existem algumas arquiteturas para se construir e desenvolver base de dados paralelas (RAY, 2009):

- *Shared-memory* – Assim como o nome sugere, uma arquitetura que contém múltiplos processadores porém dividem a mesma memória.
- *Shared-disk* – Contém múltiplos processadores que dividem o disco, ou clusters.
- *Shared-nothing* – Base de dados paralela massiva que não divide disco ou memória.
- *Hierarchical* – É um tipo híbrido entre as três opções anteriores.

Sendo que a arquitetura *shared-nothing* é a mais comum devido à alta escalabilidade, representada pela Figura 18.

Figura 18- Arquitetura Shared-Nothing de bases paralelas



(ANTUNES e FURTADO, 2007)

Nessa configuração, cada nó pode realizar um trabalho independetemente, sem comprometer seus recursos (disco ou processamento, por exemplo), passando os resultados parciais pelo receptor global, que por sua vez fornece o resultado, permitindo um maior desempenho.

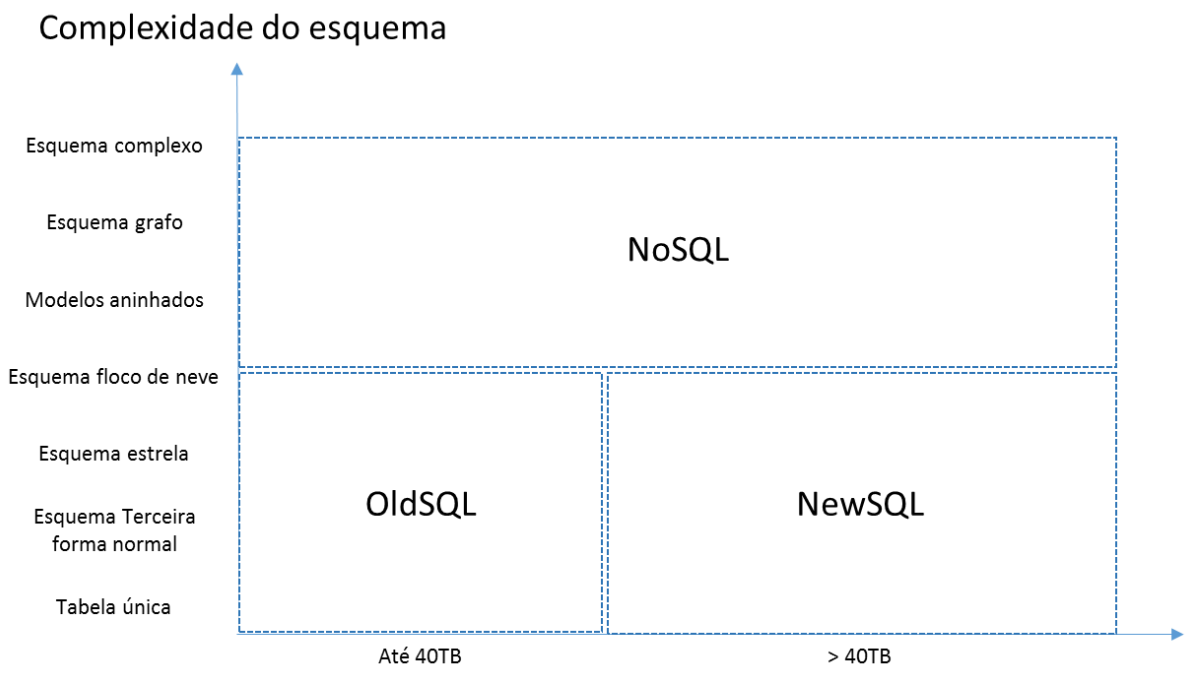
Construir consultas para base de dados paralelas difere do método convencional ao que tange o desempenho e o custo computacional. As consultas devem ser meticulosamente desenvolvidas, ou a estratégia resultará em uma distribuição não

uniforme e a formação de gargalos (RAHMAN, 2013), (ANTUNES e FURTADO, 2007).

4.2 *NoSQL*– Alternativa às bases convencionais

Os sistemas *NoSQL* (*Not only SQL*, não somente SQL) auxiliam diretamente os problemas de estruturação de dados e tamanho ("variedade" e "volume"), sendo que alguns integram a funcionalidade do MapReduce nativamente, eliminando movimentação de dados desnecessárias e facilitando a integração com *Big Data*. É tipicamente executado em hardware comum e com ótima escalabilidade em *clusters* (GUDIVADA; RAO e RAGHAVAN, 2014). O esquema da Figura 19 mostra uma noção geral de como o *NoSQL* é posicionado na análise de dados.

Figura 19 - Sistemas de bases para estudos analíticos



(MOHANTY, JAGADEESH e SRIVATSA, 2013)

NoSQL ou *Not only SQL* fornece um mecanismo para armazenamento e recuperação de dados que é modelado de maneira não convencional. Motivações para esse modelo incluem simplicidade, escalabilidade, disponibilidade (FOWLER e SADALAGE, 2012) além do rápido processamento de consultas para aplicações com necessidades específicas (CARNIEL; DE AGUIAR SA *et al.*, 2012). O banco de dados

foi desenvolvido principalmente seguindo os seguintes aspectos: alta concorrência na escrita e leitura de dados; armazenamento e processamento de consultas eficientes em grandes volumes de dados; alta escalabilidade e disponibilidade; e, redução de custos e gerenciamento operacional (HAN; HAIHONG *et al.*, 2011).

Ao contrário das bases de dados e *DW* tradicionais que se baseiam no paradigma *ACID* (*Atomicity, Consistency, Isolation and Durability*), os modelos do *NoSQL* utilizam o modelo *BASE* (*Basically Available, Soft state, Eventually consistent*). O paradigma *ACID* presente nos *SGBDR* garante que todas as transações sejam consistentes e vistas por todos os usuários simultaneamente, enquanto *BASE* se comporta da seguinte maneira (GUDIVADA; RAO e RAGHAVAN, 2014):

- Disponibilidade básica – Implica que as operações são desconectadas e existe um atraso na sincronização e certa tolerância em inconsistências temporárias.
- Estado flexível – O estado pode mudar sem uma entrada, o que é necessário para manter a consistência eventual.
- Consistência eventual – Se nenhuma atualização for realizada durante tempo suficiente, todos os usuários estarão enxergando o mesmo valor eventualmente.

As bases *NewSQL* (Novo SQL) nada mais são que uma classe de base de dados relacionais que procuram fornecer o mesmo desempenho escalável dos sistemas *NoSQL* para sistema *OLTP*, ainda mantendo as garantias que o paradigma *ACID* fornece, e também comumente oferecendo acesso a linguagem SQL e modelo de dados relacional.

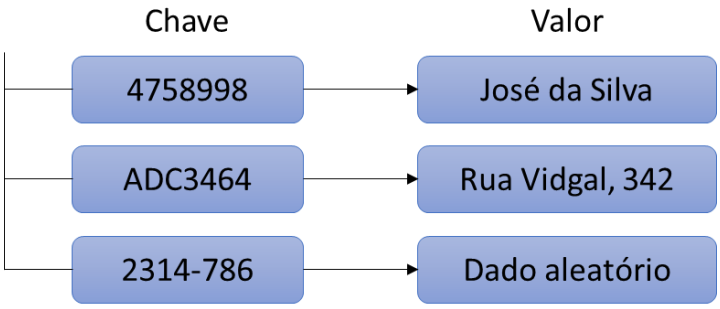
Diferentes tipos de bases *NoSQL* coexistem. Os modelos mais comuns são (HAN; HAIHONG *et al.*, 2011), (CATTEL, 2010), (GUDIVADA; RAO e RAGHAVAN, 2014):

- armazenamento chave-valor;
- orientado a coluna e;
- orientado a documentos.

Outros modelos como: grafo, XML, objeto e híbrido existem, porém não são discutidos neste trabalho. O modelo orientado a coluna é detalhado em uma seção posterior.

O modelo chave-valor guarda valores com uma chave única associada, podendo esse valor ser um documento estruturado ou não, ou até mesmo um conjunto de outras chaves. Esse é o modelo mais primitivo para armazenar dados não estruturados e não é otimizado para consultas. Não é necessário um esquema para o manter o conjunto de dados (CELKO, 2013). Utilizando um vetor associativo, um exemplo de como é realizado o armazenamento é dado na Figura 20:

Figura 20 - Exemplo de armazenamento chave-valor



(CELKO, 2013)

O *NoSQL* orientado a documentos é similar ao modelo chave-valor e utilizado para guardar documentos semiestruturados.
Em comparação com as bases relacionais, os documentos se comportam como linhas, porém os dados são semiestruturados não precisando portanto manter a mesma estrutura de uma linha para a outra. Um exemplo utilizando *JSON (avaScript Object Notation,* um formato leve para intercâmbio de dados computacionais)

Um documento pode fornecer informações parciais de um funcionário (não são conhecidas as demais informações):

```
{
  "ID_FUNCIONARIO": "FU1",
  "PRIMEIRO_NOME" : "ANA",
  "ULTIMO_NOME"  : "SILVA",
  "IDADE"       : 45,
  "SALARIO"    : 24500
}
```

Um segundo documento contém mais detalhes de outro funcionário

```
{
  "ID_FUNCIONARIO": "FU2",
  "PRIMEIRO_NOME" : "Anand",
  "IDADE"       : 34,
  "SALARIO"    : 4000,
  "ENDERECO"   : {
    "RUA"  : "RUA PEDRO DA SILVA, 234",
    "CIDADE"   : "SAO PAULO",
    "ESTADO"  : "SAO PAULO"
  },
  "PROJETO"  : [
    "MIGRACAO",
  ]
}
```

E por último, o terceiro documento que contém informações de localização de trabalho:

```
{
  "ID_LOCAL"        : "LO1",
  "NOME"    : "ESCRITORIO NOSQL"
  "ENDERECO" : {
    "RUA"    : "RUA PEDRO DA SILVA, 123",
    "CIDADE"     : "SAO PAULO",
    "ESTADO"    : "SAO PAULO"
  },
}
```

É possível notar que as estruturas dos dois primeiros documentos são similares, no entanto com quantidades de informações divergentes, e o terceiro caso ligeiramente

diferente, com propriedades de uma localidade. Os *IDs* informados nos documentos não são necessariamente sua identificação ou chave, contudo é conveniente que a identificação fique em local padrão na estrutura do documento e seja utilizado como chave.

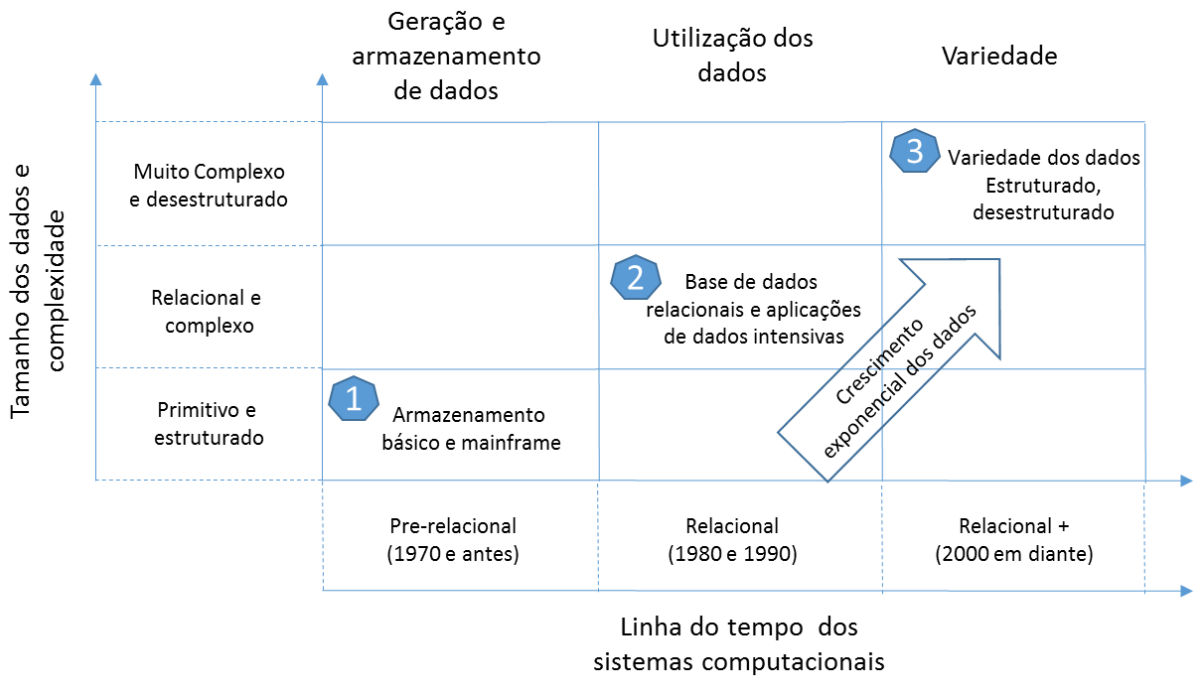
A base orientada a documentos promove uma grande flexibilidade, ou seja, os documentos podem ser armazenados sem ou com esquema, inclusive ainda serem modificados posteriormente. Por essa característica é um dos modelos mais implementados (VAISH, 2013).

Os estudos de desempenho conduzidos por Carniel et al. (2012) mostram um funcionamento excepcional para os sistemas *NoSQL* usando arquitetura de *Data Warehouse* com esquema floco de neve. As bases utilizadas foram: *FastBit, LucidDB e MongoDB.*

4.3 Dados não estruturados e serviços

A maior parte dos dados analisados até o final da década de 1990 eram informações providas de repositórios estruturados, eventualmente alimentados no *Data Warehouse*.Tecnologia essa capaz de integrar, acessar e realizar análises sobre esses dados, todavia limitada em recursos, como a falta de técnicas próprias para analisar e capacidade restrita da estrutura para guardar os dados não estruturados (ALQARNI e PARDEDE, 2012). Atualmente o volume de dados não estruturados é quantivamente maior e exponencialmente crescente em relação aos dados estruturados (ALQARNI e PARDEDE, 2012), (PORNAIN, 2014), (GRIMES, 2005). No intuito de sanar esse entrave, as técnicas de mineração existentes foram feitas para trabalhar com banco de dados contendo esquemas (LOMOTEY e DETERS, 2014). A Figura 21 apresenta o histórico desse tipo de dado:

Figura 21 - Evolução dos dados



(MOHANTY, JAGADEESH e SRIVATSA, 2013)

O segundo "V" (variedade) do *Big Data* está ligado diretamente com os dados não estruturados. Informações como mídias sociais, e-mails, blogs, páginas web, vídeos, áudios, imagens, documentos etc. constituem dados não estruturados. Lomotey e Deters (2014) define os dados como:

a) Os dados são representados em vários formatos;
b) Não contêm esquema, ou seja, não há padrões ou restrições na geração ou no armazenamento;
c) As fontes são diversas, incluindo dados da web, computadores, celulares etc.
d) Requer muitas interfaces de programação de aplicações para agregar os dados.

Big Data é uma plataforma muito comum para armazenar, consumir e tratar dados não estruturados devido à sua grande flexibilidade. Os *Data Warehouses* tipicamente tratam itens sem esquema em arquivos binários (*BLOB - Binary Large Objects*), contudo algumas propostas recentes de Gupta e Gosain (2013), Inmon e Krishnan (2011) e Alqarni e Pardede (2012) foram construídas para gerar valor sobre esses elementos.

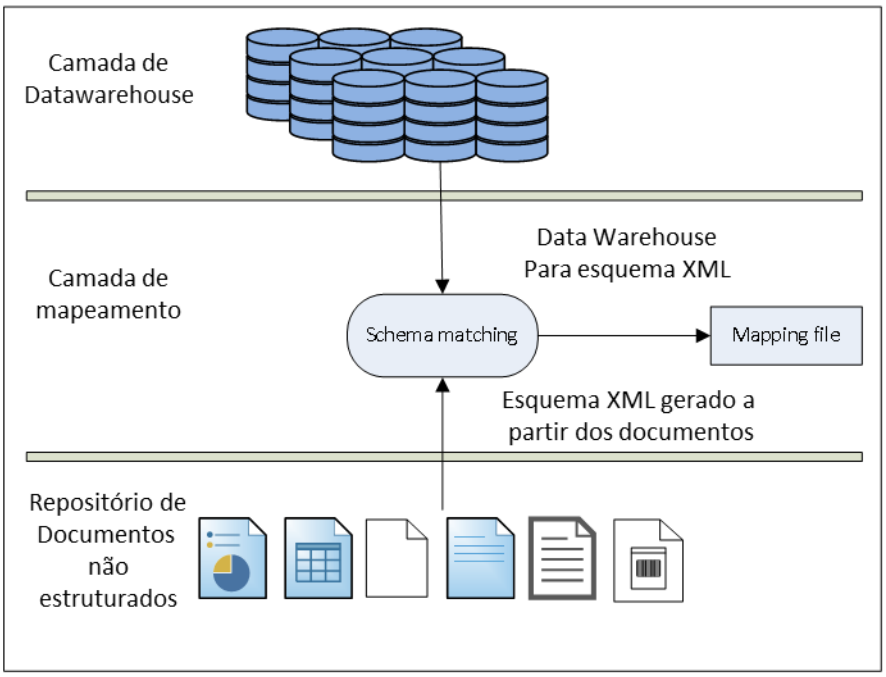
A maior parte dos dados não estruturados para *DW* está em forma textual (ou transformado em texto) (INMON e KRISHNAN, 2011) e incluem documentos, e-mails, planilhas e texto embutido.

Algumas técnicas conhecidas são utilizadas para realizar mineração sobre os documentos e incluem: extração de termos, processamento natural de linguagem, regras de associação, sumarização de documentos e dicionários.

Quando montada sobre sistemas *RDBMS*, análises semânticas são oferecidas nativamente por alguns fornecedores como Microsoft, Oracle e Vertica.

Exemplo de um *Data Warehouse* com análise de documentos é dado por Alqarni e Pardede (2012), que propõem uma abordagem para facilitar a integração de dados não estruturados com *RDBMS*. O esquema genérico de *XML* para os vários tipos de documentos foi projetado para separar as partições agrupando-os pelo tipo de documento. O esquema da solução pode ser visto na Figura 22:

Figura 22 - Integração de documentos não estruturados



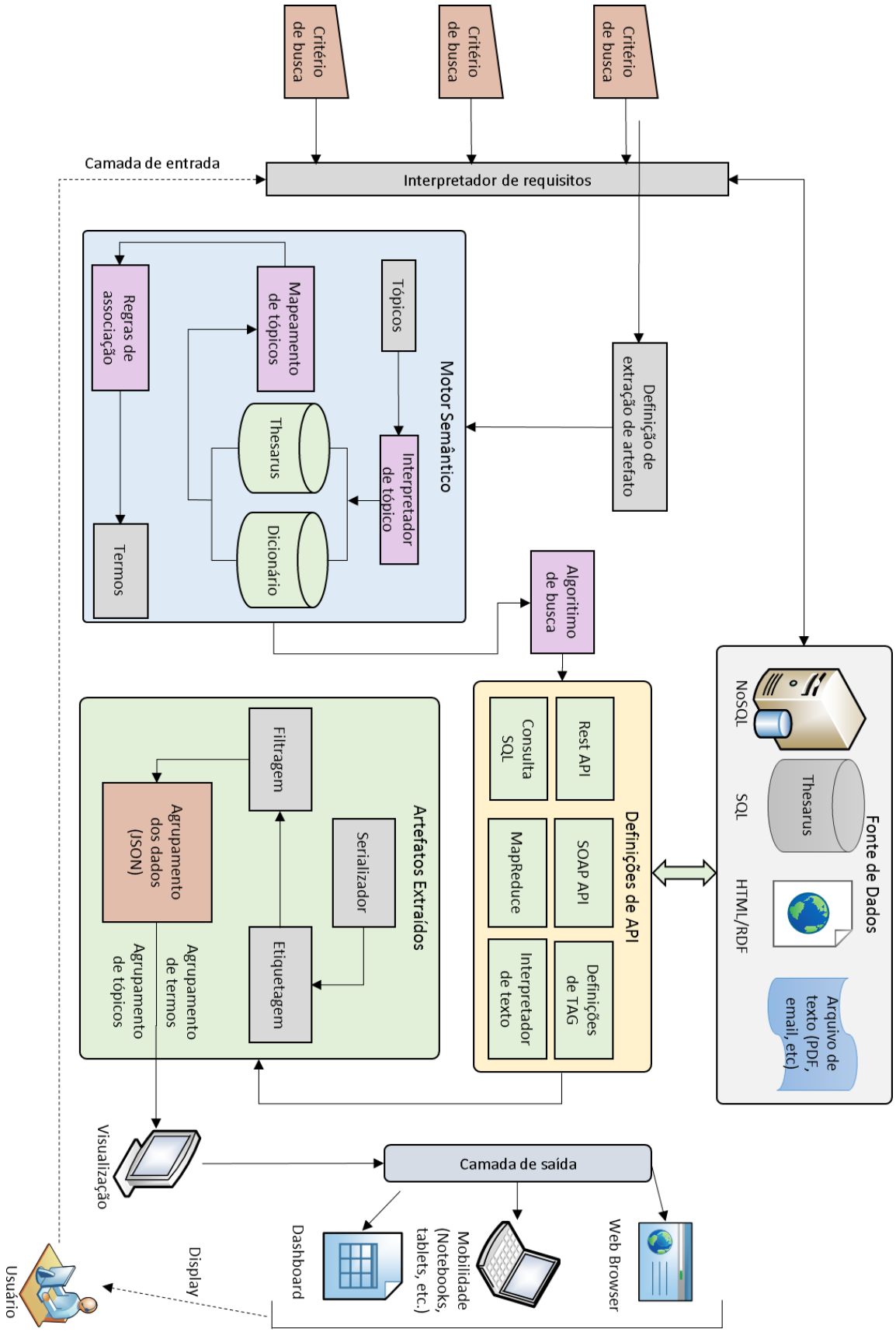
(ALQARNI e PARDEDE, 2012)

O método tenta manter a organização de um *DW* tradicional adicionando um esquema *XML* para suportar os dados. A camada central contém esse esquema, o qual é obtido sobre as duas camadas adjacentes, formando assim uma estrutura genérica para organização dos dados.

Como visto na seção anterior, bases *NoSQL* contêm armazenamentos que contribuem para o alojamento dos dados não estruturados. Porém, a análise deles é um problema completamente diferente.

Os serviços especializados (em nuvem) auxiliam no contorno desse problema. AaaS (*Analytics as a Service*) está se tornando mais comum devido a especialidade desses fornecedores na análise dos dados não estruturados. A IBM (2013) propõe um serviço especializado para transformar esses dados em valor. A análise e o ambiente são de responsabilidades do fornecedor e agregados à ele o custo sobre o serviço. É válido explorar a proposição de Lomotey e Deters (2014), cuja maioria dos elementos comumente podem ser oferecidos em um serviço em nuvem. O esquema pode ser visto na Figura 23:

Figura 23 – Análise de dados como serviço



(LOMOTEY e DETERS, 2014)

Nesse esquema, o usuário envia o comando para o interpretador de requisitos, que é a interface do serviço. Após a validação do componente “Definição de extração de artefato”, o motor semântico passa o pedido por uma série de revisões que procuram melhorar a qualidade do data mining. O algoritmo de busca determina qual metodologia se aplica ao buscar as fontes de dados (dentre as localizadas nas definições de *API*) e por fim, a camada “Artefatos Extraídos” é onde a extração de conhecimento é extraída e apresentada.

Os serviços na nuvem oferecem infraestrutura e recursos de maneira rápida, elástica e eficiente, tornando a solução extremamente atrativa para várias aplicações, especialmente para as que consomem mais recursos (ZHANG *et al.*, 2013).

Existem duas implicações para serviços de *BD* em nuvem: Os dados devem ser migrados para os servidores, sendo que o trabalho pode ser árduo e custoso considerando o volume de dados, e a privacidade afetada, pois a custódia dos dados será tutelada terceiros (GROLINGER; HAYES *et al.*, 2014).

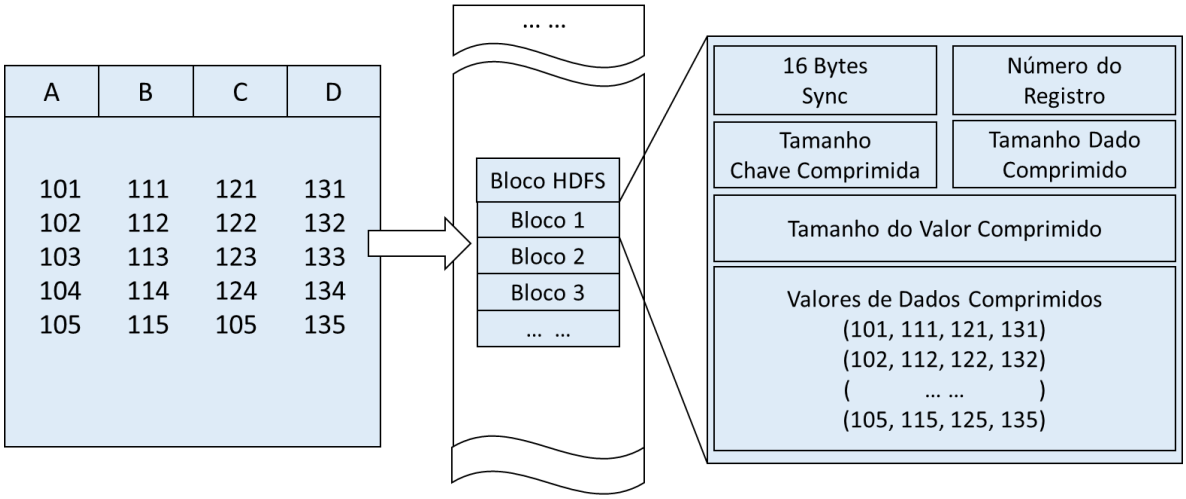
De acordo com a IBM (2013), AaaS pode agregar valor ao negócio, ajudando na transformação de dados e redução de custos.

4.4 Índices de linha, coluna e híbridos

Outra maneira de endereçar as consultas em grandes volumes de dados é a utilização de índices apropriados. Tradicionalmente, o banco de dados orientado a linhas (*row store*) sempre foi utilizado e ainda domina o mercado. Nesse modo, quando acessado, o conteúdo de todo o registro (ou tupla) deve ser lido (YAMAN, 2012).
A leitura de todas as colunas para buscar os atributos necessários de uma consulta gera trabalho desnecessário para fornecer a informação requerida (MATEI, 2010), portanto, a principal desvantagem para essa estrutura são as leituras e as consultas que costumam ser menos performáticas devido ao acesso adicional na maioria dos casos. Outro ponto importante é sua compressão, cuja leitura não costuma ser adequada, devido a variedade de tipos de dados em cada coluna (HE; LEE *et al.*,

2011). A Figura 24 mostra a representação de um armazenamento em linha utilizado no *HDFS*:

Figura 24 - Índice de linhas utilizado no HDFS

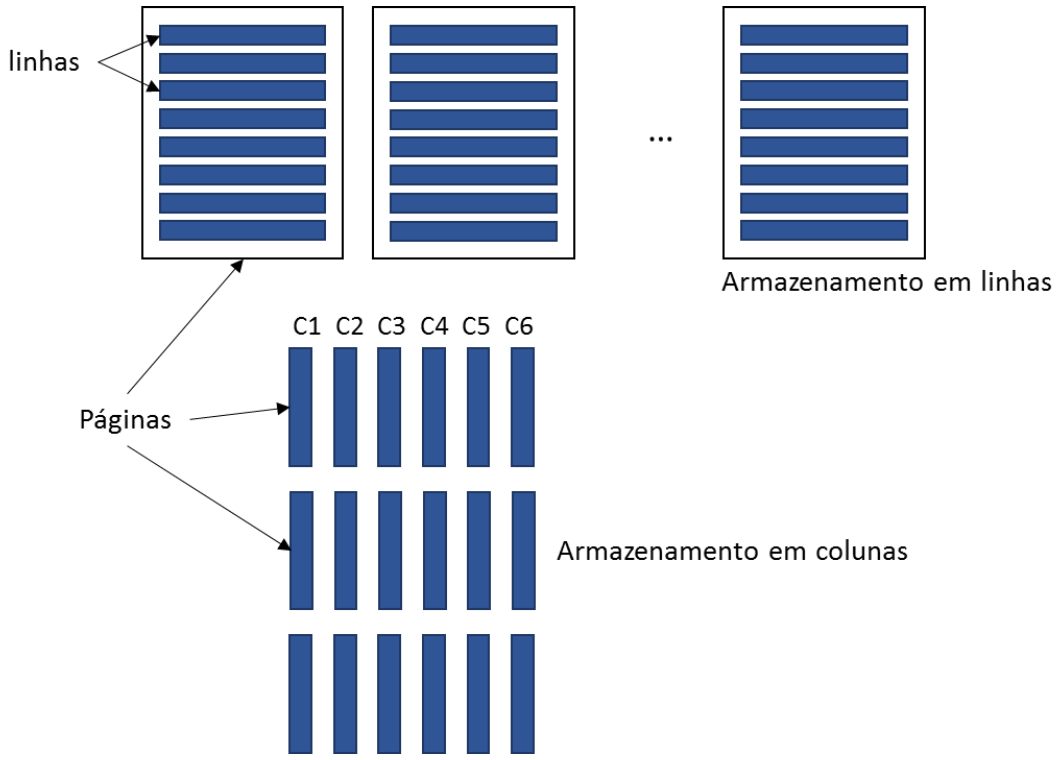


(HE, LEE,* et al.*, 2011)

A técnica de armazenar as tabelas em seções de colunas, ao invés de linhas, pode ser utilizada principalmente em aplicações com grandes quantidades de dados e alto volume de leitura, como o *Data Warehouse* (STONEBRAKER; ABADI *et al.*, 2005).

Um *index column-store* (índice de coluna) grava cada coluna em agrupamento de páginas separadas, ao invés de gravar múltiplas linhas por página conforme o modelo tradicional de banco de dados (HANSON, 2010). A diferença está ilustrada na Figura 25:

Figura 25 - Índice de colunas



(HANSON, 2010)

A desvantagem no índice de colunas ocorre na escrita ou inserção de dados que costumar ser mais custosa devido à disposição e localidade física de cada coluna no armazenamento (YAMAN, 2012).

Um *Data Warehouse* fictício foi utilizado para efeitos de comparação. No teste a seguir, uma consulta é realizada na tabela fato *"*DetalhesDeCompra*"*. Para obter a soma dos produtos vendidos e valor médio dos mesmos, a seguinte *query* é executada duas vezes. Com índice de coluna e com índice convencional em sua chave primária:

```
SELECT ProductKey, SUM(UnitPrice) SumUnitPrice, AVG(UnitPrice)AvgUnitPrice,
SUM(OrderQty) SumOrderQty, AVG(OrderQty) AvgOrderQty
FROM [dbo].[DetalhesDeCompra]
GROUP BY ProductID
ORDER BY ProductID
```

O número de leituras lógicas diminui consideravelmente quando o índice de coluna é utilizado (Figura 26 e Tabela 4):

Figura 26 - Efeito de um index Column Store

```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

(158 row(s) affected)
Table 'PerformanceTable'. Scan count 4, logical reads 54, physical reads 7, read-ahead reads 25, lob logical reads 0, lob physical
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0,

 SQL Server Execution Times:
   CPU time = 109 ms,  elapsed time = 110 ms.

SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.
SQL Server parse and compile time:
   CPU time = 15 ms, elapsed time = 21 ms.

(158 row(s) affected)
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0,
Table 'PerformanceTable'. Scan count 5, logical reads 55333, physical reads 1, read-ahead reads 55008, lob logical reads 0, lob ph

 SQL Server Execution Times:
   CPU time = 1078 ms,  elapsed time = 11674 ms.
```

Fonte: Autor

Tabela 4 - Comparação entre os índices

	Tempo de CPU (ms)	Tempo total (ms)	Leituras lógicas
Índice de coluna	109	110	54
Índice de linha	1078	11674	55333
Ganho	9.9x	106.1x	1024.7x

Fonte: Autor

A performance extra pode mitigar o problema de desempenho que um *Data Warehouse* de enorme volume de dados pode ter (MATEI, 2010). De acordo com Hanson (2010), é possível reduzir até o tempo de *ETL*, pois apesar do índice precisar mais tempo para ser construído, é possível descartar os agregados de dados que eram necessários com uma configuração tradicional de gravação em linhas (*row store*).

Como benefícios destacam-se a alta compressão, alto desempenho para consultas analíticas, facilidade em construir os índices, menos acesso ao disco, junções e agregações mais rápidas em consultas. Em contrapartida, a escrita fica mais difícil devido à natureza de inserção em linhas, e a construção de tuplas mais complicada (STONEBRAKER; ABADI *et al.*, 2005), (MATEI, 2010), (ABADI; BONCZ e HARIZOPOULOS, 2009).

Existem ainda sistemas híbridos entre os índices de coluna e linha, como o *PAX* (*Partition Attributes Across*), que é descrito por Ailamaki; Dewitt *et al.* (2001). Mantendo o armazenamento de tuplas (linhas), ele garante que as colunas de um grupo se mantenham na mesma página, facilitando a reconstrução. Duas principais propriedades negativas estão associadas com o PAX que dificulta o uso em sistemas de *BI*: A solução não é associada com compressão de dados e não melhora o desempenho de entrada e saída (HE; LEE *et al.*, 2011).

4.5 Stream Data

Aplicações como publicidade *online*, monitoramento de mídia social, otimização de processo industrial ou algoritmo de aplicações financeiras exigem o trabalho mais ágil dos dados, ligados diretamente com o terceiro V, ou velocidade do *Big Data.* Essa análise tem a premissa de que o processamento deve ser contínuo (*streaming*) e depende da atualidade dos dados. Ou seja, nesse paradigma, os dados devem ser acessados o mais rápido possível para que derivem resultados (HU; WEN *et al.*, 2014). As propriedades desejadas são (GOLAB e JOHNSON, 2014):

- Processo rápido de ETL;
- Estrutura dos dados organizada;
- Manutenção de Visões materializadas constantes;
- Controle de concorrência.

Dentre as otimizações a serem realizadas, Golab e Johnson (2014) citam a importância do rápido acesso e estrutura para minimizar a latência da transposição dos dados. A Tabela 5 aponta as principais diferenças com bases tradicionais:

Tabela 5 - Diferença entre Stream Data e Dados tradicionais

	Processamento contínuo *(streaming)*	**Processamento em quantidade** *(batch)*
Entrada	Alimentação contínua ou atualizações	Atualização em partes e programada
Tamanho dos dados	Sem conhecimento prévio	Conhecido e finito
Armazenamento	Memória e partições não triviais	Disco rígido comum
Hardware	Tipicamente uma única unidade de armazenamento (memória)	Particionado
Processamento	Única leitura e processamento dos dados	Processada em fases
Tempo	Segundos ou milissegundos	Diário, semanal, mensal
Aplicações	Sensores de rede, monitoramento de tráfego	Em quase todos os outros domínios

(HU, WEN,* et al.*, 2014)

As complexidades e os problemas desse paradigma são (GOLAB e JOHNSON, 2014):

- Administração de grandes bases distribuídas;
- Arquiteturas de sistemas híbridos compostos de dados rápidos e *batch* são difíceis de se gerenciar;
- As memórias de alta velocidade não são grandes os suficientes.

As plataformas que trabalham com dados relacionais, que mais se aproximam das análises convencionais e tradicionais, devem ser modificadas extensivamente (como por exemplo, “Truviso”, “DataDepot” e “Deja Vu”, que são plataformas de análise contínuas, e utilizam base de dados relacionais) para comportar esse tipo de dado. De maneira similar, o *Hadoop* contém extensões como “Muppet”, “HOP” e “Nova”, para trabalhar com uma menor latência de análise e gerenciamento contínuo.

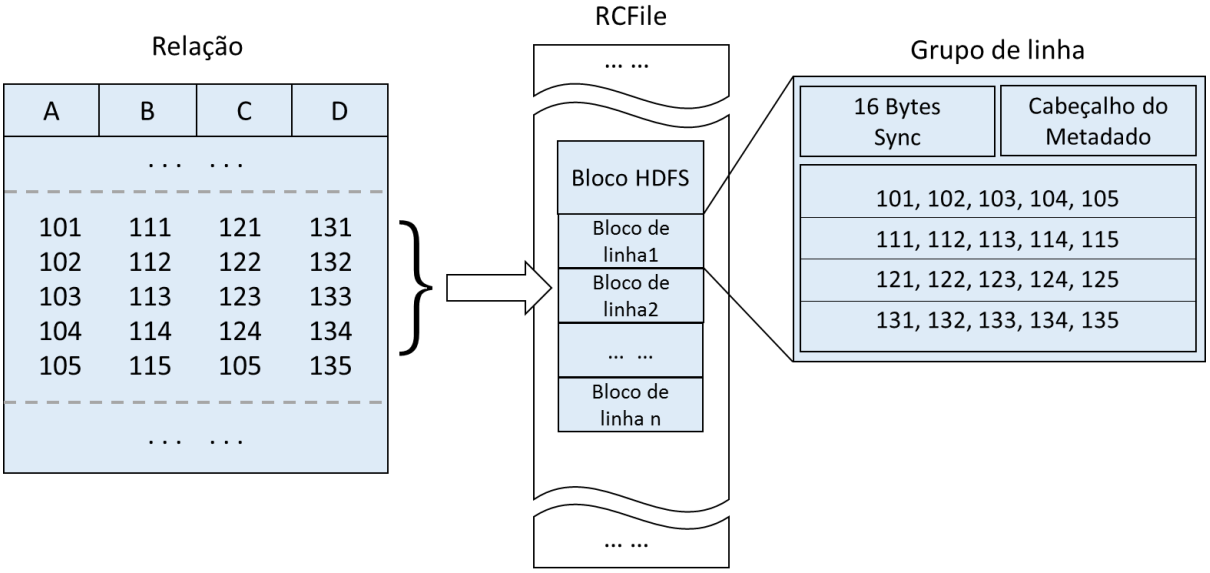
4.6 Soluções alternativas

As soluções nas seções seguintes exibem alternativas que agregam ou utilizam as tecnologias já apresentadas. E podem ser utilizadas em conjunto, substituindo ou complementando um *Data Warehouse* ou *Big Data*.

4.6.1 *RCFile* – Sistema de *DataWarehouse*

O *RCFile* (*Record Columnar File*) projetado por HE et al. (2011) é implementado com o *Hadoop*, e sua estrutura é representada na Figura 27. Ele é uma estrutura de repositório de dados para sistemas de Warehouse que utilizam o *Hadoop*.

Figura 27 - Estrutura dos dados utilizada pelo RCFile



(HE, LEE,* et al.*, 2011)

Para gravar uma tabela, a solução utiliza as seguintes premissas:

1. De acordo com a estrutura *HDFS*, uma tabela pode estar contida em múltiplos blocos;
2. Dentro de cada bloco, o *RCFile* organiza a estrutura mais básica com um agrupamento de linhas (tuplas);
3. Um grupo de linhas contém 3 sessões. A primeira sessão é um marcador síncrono que é colocado no início do grupo. Ele é utilizado para separar os grupos. O segundo é um cabeçalho de metadados para o grupo, que guarda informações de quantas linhas e o tamanho de cada coluna é utilizado. A terceira sessão é na verdade um *column store*. Nessa sessão as colunas são gravadas juntas, continuamente.

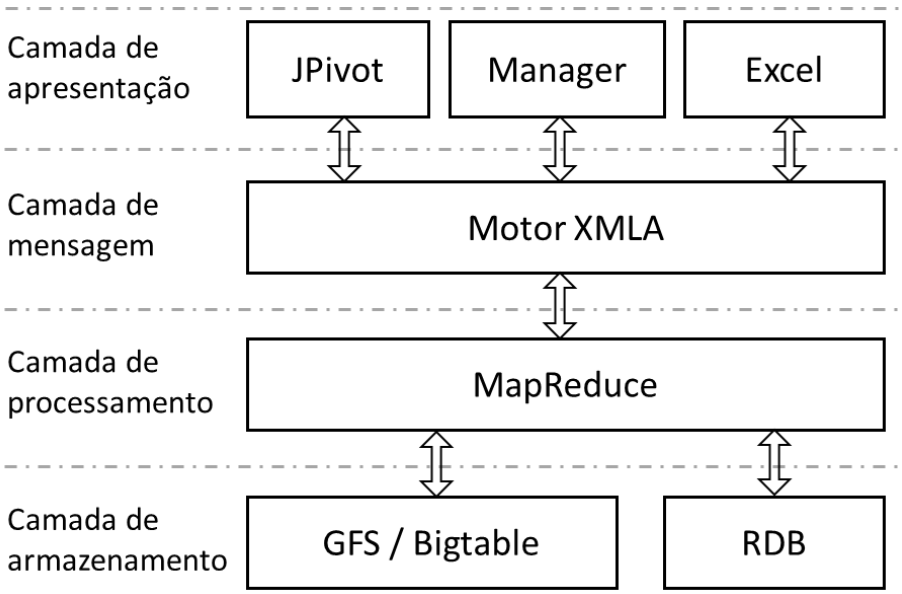
Sob o *MapReduce*, um mapeador (*mapper*) é iniciado para o bloco *HDFS.* Esse mapeador irá sequencialmente processar cada grupo de linhas no *HDFS.* A solução

aplica um conceito similar ao *PAX,* que primeiro particiona as páginas por tuplas e em seguida por colunas. Essa alternativa ficou conhecida pela utilização no facebook.

4.6.2 *Data Warehouse* de Alto desempenho

Proposto por YOU et al. (2008), a arquitetura consiste em 4 camadas que estão ilustradas na Figura 28.

Figura 28 - Arquitetura Data Warehouse de alto desempenho



(YOU, XI,* et al.*, 2008)

A camada de repositório grava o metadado e o sumário em *Bigtable* ou *GFS* (*Bigtable* é uma forma de armazenamento comprimida de alta performance construída sob o *GFS*, de propriedade do Google). Ela também é responsável por extrair os dados de bases relacionais.

A camada de cálculo utiliza *MapReduce* para paralelizar tarefas computacionais como cubos de dados, ela acessa o repositório via interface.

A camada de mensagem integra a interface com o repositório e os cálculos, e incluí mecanismo que interpreta *XMLA*.

A camada de apresentação abrange uma interface compatível com *XMLA* permitindo visualização dos dados e um painel para controle.

O sistema de alto desempenho, ou como YOU et al. (2008) cunha, *HDW*, é baseado no *Hadoop* e utiliza o *MapReduce* para distribuir as consultas pelos nós, realizando construções de cubos de dados assim como consultas *OLAP*.

De acordo com a conclusão de You; Xi *et al.* (2008), o *HDW* é ótimo para acomodar TeraBytes de dados em *Data Warehouse* com centenas de nós.

4.6.3 Solução híbrida

Proposto por ABOUZEID et al. (2009), a ideia básica por trás do *HadoopDB* é utilizar o *MapReduce* como camada de comunicação sob múltiplos nós executando instancias de banco de dados relacionais. As consultas são realizadas em *SQL,* traduzidas para *MapReduce* utilizando ferramentas existentes e o máximo de trabalho é jogado nos nós mais potentes.
Situacionalmente, a solução executa muito do processamento dentro do mecanismo das bases de dados paralelas que podem obter performances excelentes quando as consultas têm alta complexidade e utiliza o *MapReduce* para garantir a tolerância a erros e trabalhar em ambientes heterogêneos.

Quatro componentes fazem parte do *HadoopDB*:

A. Database Connector

Age como a interface entre banco de dados independentes. Cada *job* do *MapReduce* fornece para o conector uma *query SQL* e os parâmetros de conexão.

B. Catalog

Mantém o metadado sobre os bancos de dados, incluindo parâmetros de conexão, data sets contidos num cluster, localização das réplicas e propriedades do particionamento de dados.

C. Data Loader

Responsável pela repartição dos dados na carga, quebrar nós únicos em partições menores (ou *chunks*) e finalmente realizar a leitura dos dados

D. SMS Planner

O "*SQL to MapReduce to SQL*", ou *SMS Planner* estende a funcionalidade do Hive, transformando o HiveQL em *jobs* SQL que serão posteriormente traduzidos para o *MapReduce.* Em alguns casos, o motor de um banco de dados relacional em paralelo pode se comportar melhor do que a distribuição do *MapReduce*, e o *SMS Planner* entrega toda a lógica *SQL* para ser realizada pelo banco relacional. Para a realização do planejamento da forma mais eficaz, é necessário atualizar e analisar os metadados previamente à execução.

A solução se mostrou favorável para sistemas que exigem alta tolerância a erros, além de atingir performance similar ao sistema de banco de dados em paralelo e ter a versatilidade que o *Hadoop* permite.

4.6.1 Outras soluções

Ainda outras extensões do *Hadoop* são apresentadas por KARUN e CHITHARANJAN (2013). São elas:

- Hadoop ++
- Co-Hadoop
- HAIL
- Elastic Replica Manegement System For HDFS (ERMS)
- Trojan HDFS
- Cheeta
- DARE
- Clydesdale

Nas tabelas 6 e 7 é feito um comparativo das técnicas acima listadas.

Tabela 6 - Comparação das extensões do Hadoop

	HadoopDB	Hadoop++	HAIL	Co-Hadoop	ERMS
Tolerância a falhas	Praticamente o mesmo do Hadoop	Menor que do Hadoop	Quase a mesma do Hadoop	Mesma do Hadoop	Mesma do Hadoop
Escalabilidade	Melhor que base de dados paralelas	Altamente escalável	Altamente escalável	Altamente escalável	Altamente escalável
Localidade dos dados	-	-	Considerado	Considerado	-
Tempo de carregamento	Maior que o do Hadoop	Maior que do Hadoop DB	Menor que do Hadoop++	Menor que do Hadoop++	Maior que do Hadoop
Velocidade e performance	Aproximado ao de bases paralelas	Maior que do Hadoop	Maior que do Hadoop++	Maior que o Hadoop++	Aproximado ao de paralelas
Mudanças no MapReduce e interface	Mudança na interface SQL	Sem mudança	Sem mudanças	Sem mudanças	Sem mudanças
Mudanças no Hadoop	Grandes mudanças	Sem mudança	Mínima	Mínima	Mínima
Indexação e estrutura	Sem indexação. Tabular como SGBDR	Indexação Trojan	Indexação PAX	Sem indexação, estrutura padrão	Sem indexação, estrutura padrão
Compressão dos dados	Mesmo do PostgreeSQL	-	-	-	-

(KARUN e CHITHARANJAN, 2013) - I

Tabela 7 - Comparação das extensões do Hadoop

	Trojan HDFS	RCFile	DARE	Cheetah	Clydesdale
Tolerância a falhas	Mesma do Hadoop	Mesma do Hadoop	Mesma do Hadoop	Mesma do Hadoop	Mesma do Hadoop
Escalabilidade	Altamente escalável	Altamente escalável	Altamente escalável	Altamente escalável	Mesma do Hadoop
Localidade dos dados	Considerado	-	Considerado	-	Considerado
Tempo de carregamento	Maior que do Hadoop	Menor que estrutura de colunas	Maior que do Hadoop	Maior que do Hadoop	Mesma do Hadoop
Velocidade e performance	Maior que o Hadoop++	Maior que do Hadoop	Maior que do Hadoop	Alta	Alta
Mudanças no MapReduce e interface	Sem mudanças	Sem mudanças	Sem mudanças	JDBC e interface SQL	Interface dos registro por "Avro"
Mudanças no Hadoop	Mínima	Mínima	Muitas mudanças	-	Sem mudanças
Indexação e estrutura	Indexação Trojan	Sem indexação, estrutura híbrida	Sem indexação, estrutura padrão	Sem indexação, estrutura padrão	Sem indexação
Compressão dos dados	-	Mesmo do PostgreeSQL	-	Compressão de coluna	Compressão utilizando cache distribuído

(KARUN e CHITHARANJAN, 2013) - II

4.7 Conclusão do capítulo

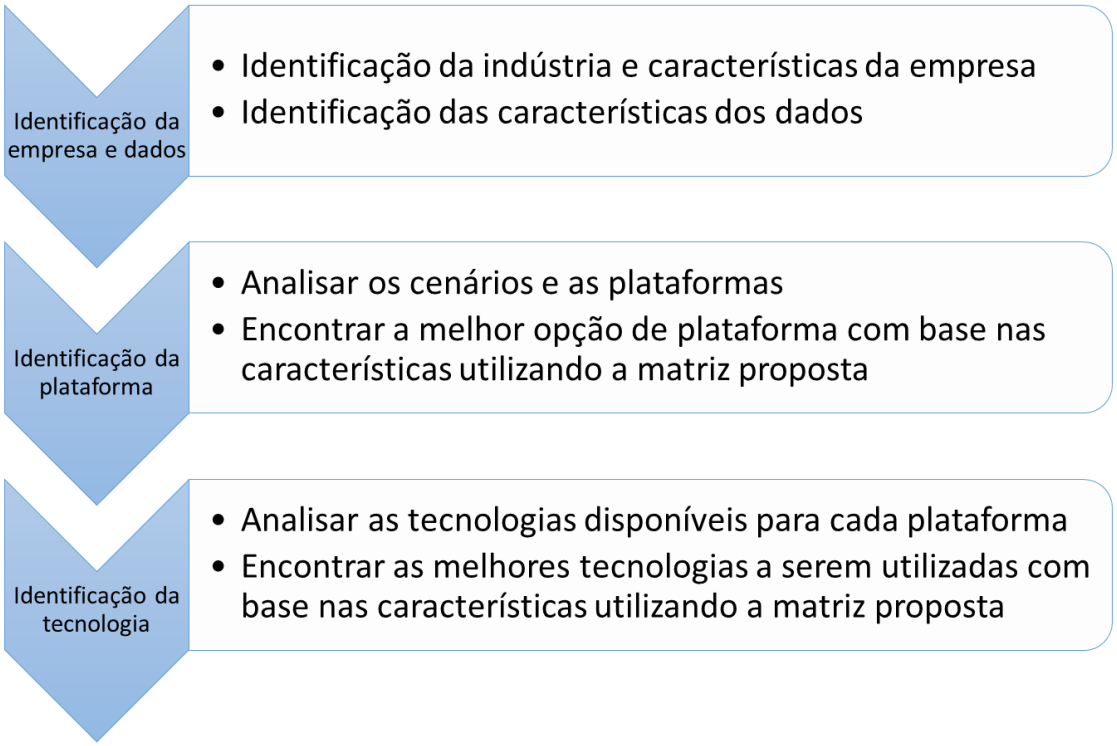
Históricamente, o *Data Warehouse* atendia as soluções de Inteligência de negócio e análise de dados, todavia com o aumento do volume, a velocidade e a complexidade, as empresas e pesquisadores foram obrigados a inovar, criando o fenômeno conhecido como *Big Data*. E tão logo se fez necessário uma revolução no *Data Warehouse* para que este permanecesse competitivo, culminando em uma série de tecnologias utilizadas por ambas as frentes. Atualmente, as plataformas se combinam em ambientes e técnicas no intuito de dar a melhor solução ao que é pretendido por cada empresa ou segmento de área.

5. PLATAFORMAS E ARQUITETURAS PARA CENÁRIOS CORPORATIVOS

As estratégias representadas neste capítulo são baseadas nas tecnologias e plataformas descritas nas sessões anteriores. Elas podem ser classificadas desde tradicional, que dependem de bases de dados internas, estruturadas e esquematizadas, até uma solução mais complexa e inovadora que trabalha com dados externos, mídias e variados tipos de fontes.

A estratégia para escolha de uma plataforma e tecnologia que vai definir a arquitetura geral do sistema foi dividida em três partes, e podem ser visualizadas na Figura 29.

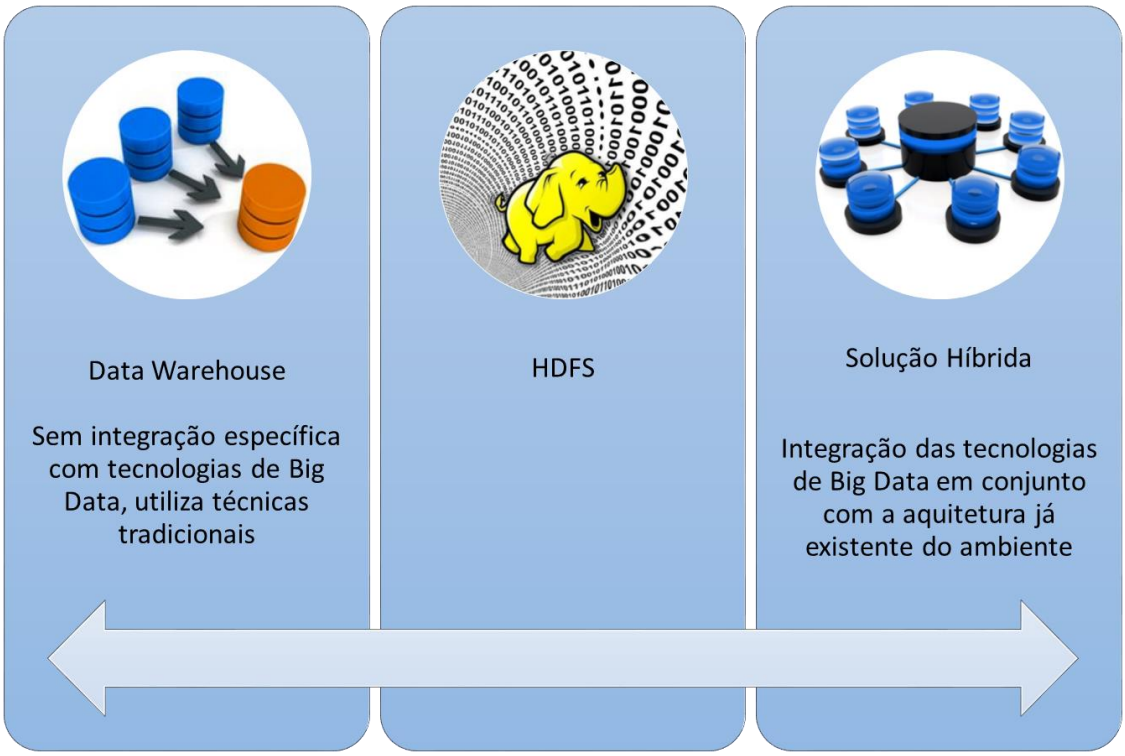
Figura 29 - Passos para identificação dos cenários de BI



(EBNER, BUHNEN e URBACH, 2014) (MADDEN, 2012) (PAVLO, PAULSON,* et al.*, 2009)

Para o segundo passo da Figura 29, as três principais plataformas extraídas do estudo foram identificadas e estão separadas conforme a evolução da integração dos dados representada na Figura 30.

Figura 30 - Identificação das plataformas



(EBNER, BUHNEN e URBACH, 2014)

Na esquerda, a figura mostra um ambiente sem integração externa com a maior parte dos dados provenientes de bases internas e dados estruturados. Movendo-se para o lado direito da figura há um ambiente totalmente integrado com estruturas e dados externos, arquitetura mais complexa e completa.

A sessões seguintes detalham cada uma das plataformas, apresentando particularidades e benefícios que a escolha pode proporcionar.

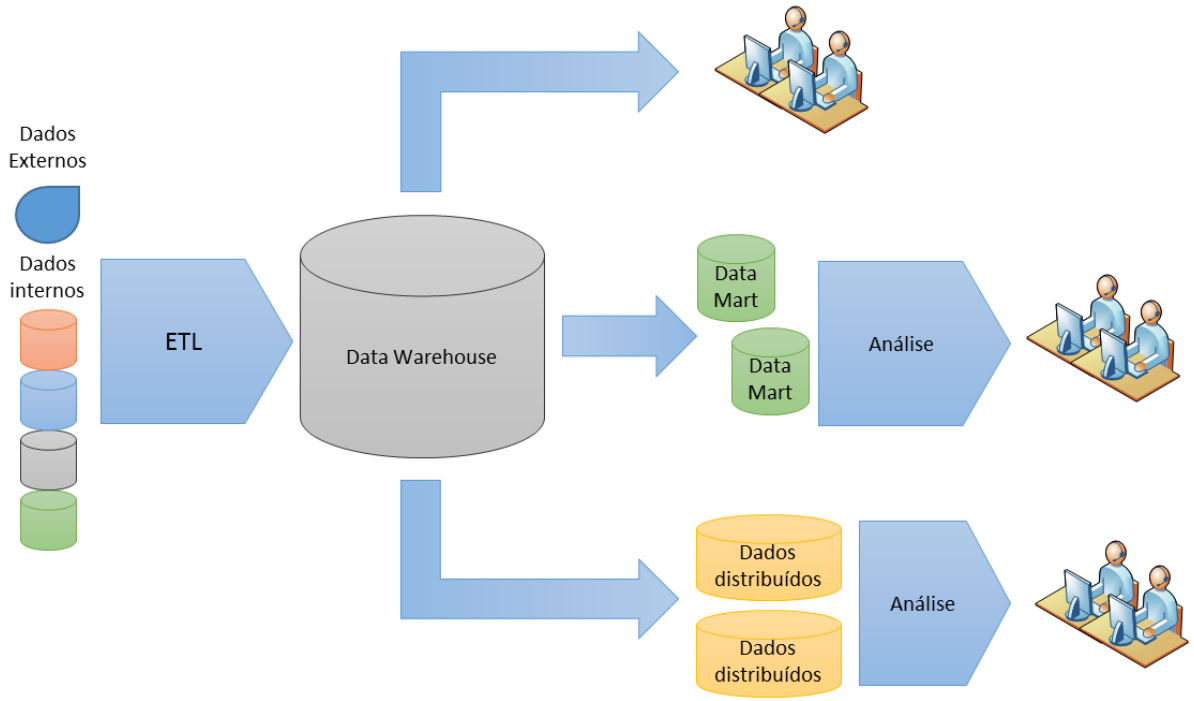
5.1 Data Warehouse

A primeira estratégia e provavelmente a mais simples é depender de SGBDR, que estão presentes na maioria das empresas hoje e integrá-los para o gerenciamento e análise dos dados dos dados em um DW se torna uma tarefa simples. O modelo estrela é a forma mais custosa para se processar uma consulta em Data Warehouse e deve ser evitada em grandes volumes. Algumas técnicas, como fragmentação dos dados, visões materializadas, pré-agregação e compressão dos dados podem ser

utilizadas para melhorar o desempenho e o armazenamento, porém podem deixar os sistemas inflexíveis (CARNIEL, DE AGUIAR SA, *et al.*, 2012).

Duas possíveis alternativas derivam desse cenário: de Inmon que produz um DW centralizado, visto na primeira opção superior da Figura 31, e de Kimbal, que distribuí os dados coletados em Data Marts. Ainda uma terceira opção seria distribuir os dados de ambas as alternativas anteriores em bases paralelas, sem modificar a estrutura ou o esquema dos dados (alternativa inferior na Figura 31).

Figura 31 - Arquiteturas do Data Warehouse



(KIMBALL e ROSS, 2013) (INMON, 2005) (HAJMOOSAEI, KASHFI e KAILASAM, 2011)

Os SGBDR paralelos oferecem um desempenho no mínimo equivalente e muitas vezes melhores que o MapReduce com poucos nós, principalmente em tarefas simples como junções de tabelas (PAVLO, PAULSON, *et al.*, 2009), e ainda mantém a facilidade que uma linguagem de alto nível oferece (SQL). A escala nesse caso é proporcionada pela arquitetura *Shared Nothing*.

O tratamento da variedade e velocidade dos dados são pontos fracos nessa plataforma, principalmente no processo ETL.

O objetivo primário de um EDW é criar um ambiente integrado e padronizado que facilita o estudo analítico e o relatório de dados, culminando numa maior capacidade de tomada de decisões e solução de problemas (MOHAMMAD RIFAIEA, 2008). Os benefícios de sua utilização são: fácil uso e entendimento, tecnologia madura e consolidada, pode ser utilizado por muitos usuários, promove uma visão única dos dados e simplifica a tomada de decisões (FURLOW, 2002) (BALLARD; HERREMAN *et al.*, 1998) (GOSS e VEERAMUTHU, 2013).

Entre as desvantagens, estão: O processo ETL é demorado e complicado, a edição dos dados para utilização é mais sensível, sua modelagem deve ser estudada e construída, o processo inicial envolve custo, tempo e deve ter apoio da alta gerência, os dados devem ser estruturados, não suporta volumes gigantescos, tem tolerância a erros somente no nível de transação, o hardware deve ser homogêneo e os nós devem ser construídos igualmente (FURLOW, 2002) (GOSS e VEERAMUTHU, 2013).

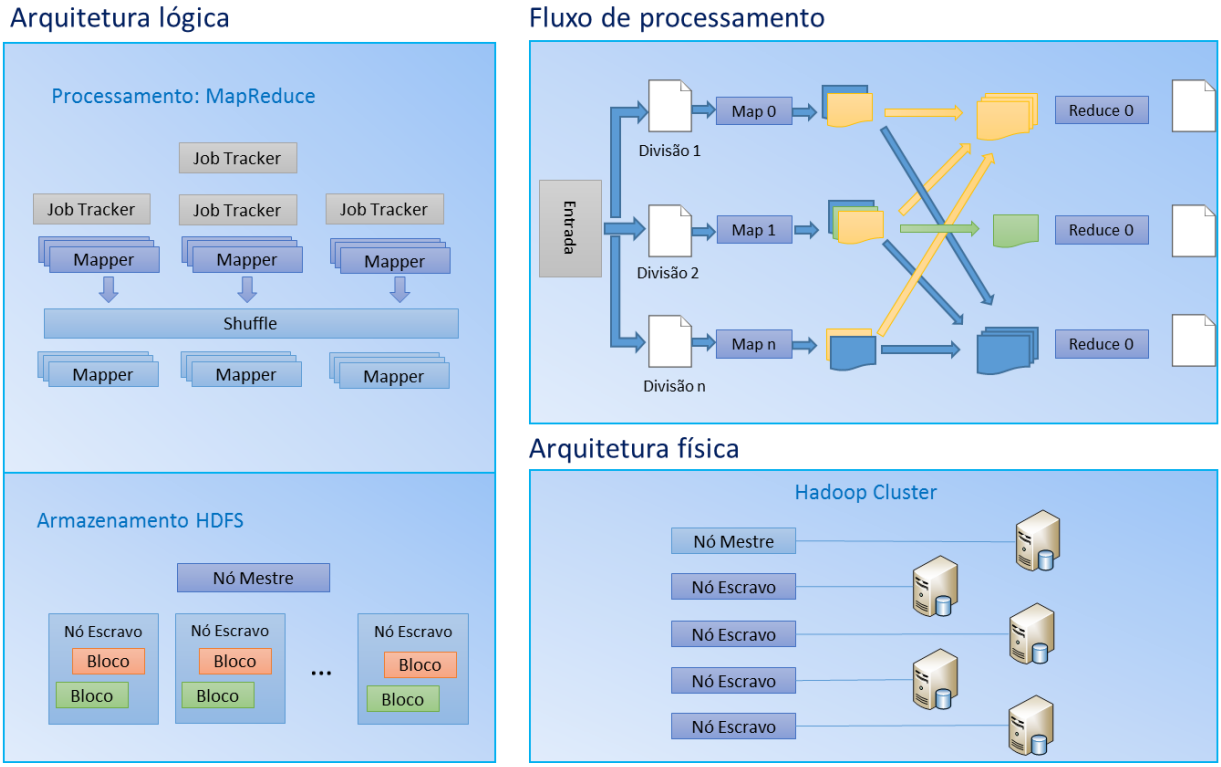
5.2 HDFS e MapReduce

A segunda plataforma para análise de Big Data consiste no uso de sistemas de arquivos distribuídos (DFS), mais especificamente, o sistema distribuído do Hadoop (HDFS). Composto de arquitetura *open source* e com excelente escalabilidade, os arquivos são gravados nos armazenamentos distribuídos e processados no MapReduce. Ao contrário do Data Warehouse tradicional, o MapReduce exige programação e tempo para escrever um programa de consulta, que pode ser significantemente mais demorado, tornando consultas *ad hoc* mais complicadas de se fazer (EBNER; BUHNEN e URBACH, 2014). A técnica de agregação ou pouca granularidade, comum em Data Warehouse, parte do pressuposto que alguns dados são mais importantes que outros, sendo que o Hadoop tende a tratar todos com igual valor (PROFFITT, 2012).

O MapReduce ganha muito em análises de dados não estruturados como arquivos de texto e mídia social quando comparado com SGBDR, adicionalmente é mais veloz no carregamento dos dados (EBNER; BUHNEN e URBACH, 2014), porém ainda assim, por si só, não é apropriado para tarefas de análise em tempo real (GROLINGER; HAYES *et al.*, 2014). Acoplando extensões apropriadas como Muppet, HOP e Nova, a plataforma se torna adequada para análises em tempo real.

A arquitetura geral do Hadoop, lógica e física, sem identificar suas extensões é apresentada na Figura 32.

Figura 32 - Arquitetura geral do Hadoop



(INTEL, 2013)

Como a plataforma é relativamente nova, os custos com consultorias, migração e treinamento podem se tornar altos, além do tempo necessário para se construir a plataforma.

O Big Data é normalmente utilizado como plataforma para inovação e o objetivo de sua construção é analisar grandes quantidades de dados e facilitar a tomada de decisões, trazendo como benefícios a escalabilidade, robustez, tolerância a falhas, baixa latência de escrita e leitura de dados, a possibilidade de se construir o sistema com softwares open source e como consequência, o baixo custo de licenças (LOSHIN, 2013), (SAGIROGLU e SINANC, 2013).

Entre suas desvantagens, estão a dificuldade em manter a qualidade dos dados devido a quantidade, a tecnologia ainda está em desenvolvimento e não é madura, exige alta qualificação, treino e expertise da equipe de TI e não tem antecipação dos

dados (CHANDARANA e VIJAYALAKSHMI, 2014) (WIKIBON, 2012) (KATAL; WAZID e GOUDAR, 2013).

5.3 Solução híbrida

Considerando o Hadoop e os SGBDR não como concorrentes, mas como sistemas com relações simbióticas, pode-se endereçar os pontos fracos de ambos paradigmas. A solução híbrida procura utilizar os SGBDR para tarefas simples que podem tirar um melhor proveito dessa plataforma e o Hadoop com sua escalabilidade e tolerância a falhas, ou seja, deve suportar trabalho analítico de ambas maneiras, seja um novo estilo exploratório e inovador de visualizar os dados ou consultas consultas analíticas em grandes quantidades de volumes.

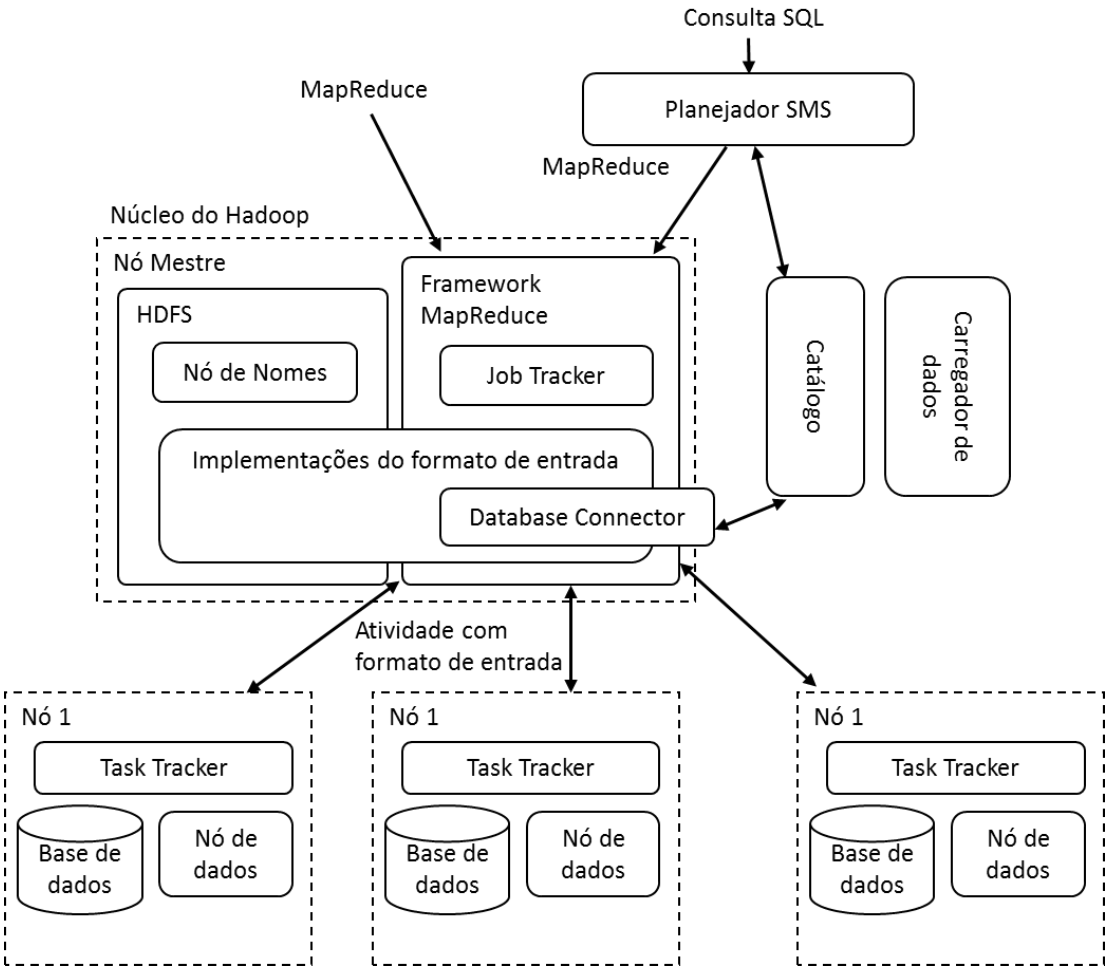
De acordo com Gruska e Martin (2010), existem três tipos de alternativas para híbridos. A que utiliza SGBDR com MapReduce acoplado, sistemas que utilizam MapReduce com base de dados relacionais acopladas ou o funcionamento independente de cada uma das partes.

Nenhuma das alternativas têm um desempenho melhor que suas estratégias não combinadas, porém, devem capturar, consultar e analisar dados num desempenho aceitável.

A solução de Abouzeid et al. (2009), que é predominantemente ligada ao MapReduce foi escolhida por mostrar um bom desempenho e utilizar apenas sistemas *open source*, porém, do ponto de vista financeiro, o custo da plataforma pode ainda assim ser mais alto que as soluções não combinadas (PROFFITT, 2012).

Como o núcleo ainda é composto pelo Hadoop, é escalável, ótimo para consultas em dados não estruturados, os nós não precisam ser homogêneos e contém uma estrutura interna que verifica se a consulta pode se beneficiar das análises realizadas pelos SGBDR. A arquitetura do HadoopDB é mostrada na Figura 33:

Figura 33 - Arquitetura do HadoopDB



(ABOUZEID, BAJDA-PAWLIKOWSKI,* et al.*, 2009)

A plataforma exige alto conhecimento técnico da equipe de TI e maior tempo de preparação. Por conta disso, o apoio da gerencia é essencial.

5.4 Escolha das plataformas

Para escolher uma plataforma adequada deve-se levar em consideração alguns fatores, como estratégia de negócio, estrutura, complexidade, e recursos disponíveis. Os fatores aqui escolhidos foram baseados nas sessões 2 e 3 deste trabalho e nos trabalhos dos autores Ebner, Buhnen e Urbach (2014) e Doshi et al. (2013).

Quanto ao volume e variedade dos dados, o crescimento pode ocorrer nas 3 principais formas:

- Cronológica: Ocorre pelo acúmulo dos dados durante os dias, meses e anos. Esse crescimento não contém muitas variações e é relativamente simples.
- Horizontal: Surge de crescimento e novos serviços ou processos que atendem as novas necessidades nas organizações.
- Vertical: Aumento nos dados devido a expansão do negócio e novas complexidades externas.

Esse crescimento está geralmente associado com os 3 Vs de Big Data, sendo que o crescimento cronológico gera volume e o horizontal e vertical geram volume e variedade ao mesmo tempo.

Duas linhas de pensamento divergem quanto à seleção dos sistemas distribuídos. Proponentes das bases de dados paralelas argumentam que elas proporcionam uma forte ênfase na performance e eficiência, o que torna a tecnologia bem adequada para análise, porém, por outro lado, outros argumentam que os sistemas baseados no MapReduce são mais adequados devido a sua escalabilidade, tolerância a falhas e flexibilidade para manusear dados não estruturados (ABOUZEID; BAJDA-PAWLIKOWSKI, *et al.*, 2009). É preferível utilizar MapReduce ao SGBDR quando o volume de dados é excessivo ou a complexidade do sistema é muita alta (centenas de nós ou mais), já que a falha de um dos nós pode ser amenizada com a estrutura do Hadoop. Consultas mais complexas ou transformações também são mais facilmente formuladas com o MapReduce, que mais se aproxima de uma linguagem de programação (GRUSKA e MARTIN, 2010). Para dados estruturados, com poucos nós com necessidade de performance, a escolha fica clara para um SGBDR paralelo.

A Tabela 8 apresenta uma comparação qualitativa dos dados que auxilia na escolha da plataforma, para os diferentes tipos de indústria:

Tabela 8 - Potencial das indústrias em Big Data

	Volume	Velocidade	Variedade	Dados sub utilizados	Potencial do Big Data
Mercando financeiro	Alto	Alto	Baixo	Médio	Alto
Comunicação e Mídia	Alto	Alto	Alto	Médio	Alto
Educação	Muito Baixo	Muito Baixo	Muito Baixo	Alto	Médio
Governo	Alto	Médio	Alto	Alto	Alto
Saúde	Médio	Alto	Médio	Médio	Alto
Seguro	Médio	Médio	Médio	Médio	Médio
Manufatura	Alto	Alto	Alto	Alto	Alto
Química e recursos naturais	Alto	Alto	Alto	Alto	Médio
Varejo	Alto	Alto	Alto	Baixo	Alto
Transporte	Médio	Médio	Médio	Alto	Médio
Utilidade	Médio	Médio	Médio	Médio	Médio

(GOSS e VEERAMUTHU, 2013)

Enquanto grandes empresas conseguem proporcionar mais recursos (no contexto, entende-se recursos como o conjunto de valores monetários, pessoas e tempo) e investimento para implementar soluções mais custosas como sistemas complexos com muitos nós ou híbridos, as empresas menores estão limitadas as soluções individuais, mais simples e consolidadas (EBNER; BUHNEN e URBACH, 2014).

A Tabela 9 apresenta um sumário das propriedades dos dados (as 3 primeiras linhas) e as propriedades das empresas (as 3 últimas linhas) que devem ser consideradas na escolha da plataforma. Os valores “Alto”, “Médio” ou “Baixo” representam a afinidade da plataforma com certa propriedade.

Tabela 9 - Relevância dos fatores na escolha da plataforma

		SGBDR	HDFS	Híbrido
Propriedade do dado	Volume	Médio	Alto	Alto
	Variedade	Baixo	Alto	Alto
	Velocidade	Baixo	Médio	Médio
Propriedade da empresa	Capacidade do time de TI	Médio	Alto	Alto
	Recurso disponível	Neutro	Neutro	Alto
	Apoio da gerência	Médio	Alto	Alto

(EBNER; BUHNEN e URBACH, 2014) (HU; WEN* et al.*, 2014) (PAVLO; PAULSON* et al.*, 2009)

Um volume de dados médio ou baixo com pouca variedade é facilmente comportado por bases relacionais e exige pequeno conhecimento técnico ou apoio da gerência, enquanto soluções mais inovadoras devem ter uma equipe mais preparada e ciência de que o investimento não trará resultados de imediato.

5.5 Escolhas das tecnologias

Baseando-se nas tecnologias apresentadas na sessão 5 do trabalho, as plataformas podem conter ambientes, estruturas e configurações distintas. Essa etapa final concluí os ajustes necessários na escolha de uma solução para a análise dos dados. A Tabela 10 mostra o resumo:

Tabela 10 - Matriz de seleção de tecnologias

	Expectativa de crescimento	Estruturação dos dados	Tolerância a falhas	Frequência de escrita
Sistema distribuído	Baixa	-	Baixa para SGBDR, alta para HDFS	-
Sistema único	Alta	-	Alta	-
Base SQL	Baixa	Estruturado	Baixa	Média
Base NoSQL	Alta	Não estruturado	Média	Alta
Hadoop Distributed File System	Alta	Não estruturado		Média
Índice - linha	-	-	-	Alta
Índice - coluna	-	-	-	Baixa
Serviço em nuvem	Alta	-	-	Média
Serviço onsite	Média	-	-	Alta

(EBNER; BUHNEN e URBACH, 2014) (HAJMOOSAEI; KASHFI e KAILASAM, 2011) (PAVLO; PAULSON* et al.*, 2009)

A escolha de sistemas únicos ou distribuídos se aplicam apenas para os SGBDR, já que o HDFS sempre é configurado na arquitetura Mestre escravo. O mesmo ocorre no caso de índices, onde podem ser configurados para bases de dados, porém a utilização no Hadoop ainda é limitada (PROKOPP, 2013) e extensões como o HBase deve ser utilizada para aproveitar a estrutura de colunas. As bases SQL e NoSQL podem ser utilizadas em qualquer plataforma (para o Hadoop, as bases precisam apenas suportar conexão JDBC, *Java Database Connectivity,* um conjunto de classes e interfaces escritas em Java que fazem o envio de instruções SQL).

Empresas de pequeno porte podem se beneficiar das vantagens de estudos analíticos na nuvem. Não só é muito mais simples escalar (basta adquirir mais serviços), como a contratação de uma equipe técnica normalmente ajuda o time muitas vezes pequeno pelo porte da empresa.

5.6 Estratégia de seleção e exemplo

Dois cenários são dados a seguir para colocar em prática a proposta do trabalho e exemplificar como a construção das arquiteturas pode ser beneficiada de maneira eficaz com os passos e matrizes auxiliando na escolha.

Suponha a empresa ABC, no ramo de transporte, com a pretensão de melhorar seus serviços e a distribuição localidades. A empresa contém muitos dados internos, e pretende acoplar meios de pesquisa e IBGE.

O volume de dados nesse caso ainda é pequeno, a empresa não exatamente planeja realizar nenhuma inovação, apenas tomar decisões estratégicas. Independentemente dos recursos, um Data Warehouse já é suficiente para atender a empresa. Os dados podem ser no mínimo transformados para estruturados e como não se espera nenhum tipo de crescimento significativo, um Data Warehouse interno é suficiente, mas pode ser migrado para nuvem se a redução de custo for significativa.

Em outro exemplo, um banco de investimento XYZ, decide ampliar seus serviços, descobrir novas oportunidades e aumentar a base de clientes. A empresa já tem muitas informações em bases internas e planeja integrar dados da bolsa de valores,

dados históricos e mídia sociais, buscando contato com novos clientes e um canal de resolução ativa de reclamações

A quantidade de dados nesse caso não só é muito grande como também é diversificada, porém a análise não é em tempo real. Como as informações provêm de diferentes fontes, a informação é mais facilmente extraída com linguagem de programação e base própria para guardar os dados não estruturados. Adicionalmente, esse tipo de solicitação é geralmente dado pela gerência, portanto o apoio deve ser obtido espontaneamente. A escolha da plataforma HDFS atende todas as exigências do projeto e uma plataforma híbrida também pode ser considerada se existir a necessidade de muitas consultas sobre os dados estruturados já existentes.
Se a privacidade dos dados for uma preocupação e o time de TI for capacitado, a melhor opção seria manter e ampliar o ambiente interno, utilizando o próprio HDFS ou substituindo o armazenamento para uma base NoSQL. Caso contrário, um serviço na nuvem pode ser utilizado.

5.7 Conclusão do capítulo

As plataformas e tecnologias aqui escolhidas tiveram base nos estudos das sessões anteriores. Os exemplos mostram como podemos utilizar as matrizes de apoio e realizar os ajustes necessários para diferentes situações.

6. CONCLUSÕES

6.1 Considerações finais

Este trabalho é embasado principalmente em literatura teórica pela habilidade em expor potenciais de pesquisa e novas tecnologias, além do vasto material fornecido e recente na área.

Dito isso, escolher uma tecnologia ou plataforma não apenas envolve comparação técnica. Vários outros fatores relacionados a documentação, manutenção, estabilidade, maturidade, fornecedor, suporte, comunidade, licença, preço e o futuro do produto ou da organização por trás também contém papéis importantes. Ainda assim, a comparação técnica deveria continuar tendo um papel essencial.

É claro que nem todos os possíveis cenários serão solucionados apenas seguindo os passos sugeridos, mas com estudo e conhecimento, é possível traçar as melhores opções dado o ambiente de trabalho. Se existirem lacunas não solucionadas, as mesmas podem ser preenchidas com as novas tendências do mercado ou até mesmo soluções customizadas, assim como ocorreu com as maiores empresas de tecnologia atuais (como o Google com o MapReduce, Bigfile ou Facebook, com o RCFile).

Não existe uma linha clara ou distinção entre Big Data e Data Warehouse, que são melhores vistos como complementos do que concorrentes. A importância do crescimento do Big Data e a mudança recente nas análises de Data Warehouse são uma mera parte nos desafios dos novos projetos de infraestrutura de armazenamento e análise dos dados.

6.2 Contribuições e trabalhos futuros.

A pesquisa procurou dar uma contribuição as novas tendências do mercado e expor as diferentes possibilidades na análise de dados, analisando novas bases de dados, sistemas distribuídos, índices e maneira de estruturar os dados.

Temas não abordados podem ser desenvolvidos em trabalhos futuros, como segurança e privacidade dos dados nas plataformas e tecnologias. Um estudo recente mostra grande preocupação nessas áreas (MOHANTY; JAGADEESH e SRIVATSA, 2013).

Outra evolução importante que ganha espaço no mercado é o aprendizado de máquina (*machine learning*), que utiliza de forma automática as predições analíticas de máquina para determinar cenários e tendências futuras, sendo acoplado cada vez mais aos sistemas de BI.

REFERÊNCIAS

ABADI, D. J.; BONCZ, P. A.; HARIZOPOULOS, S. Column-oriented Database Systems. Proceedings of the VLDB Endowment**, Lyon, France, v. 2, n. 2, p. 1664-1665, Agosto 2009. ISSN 2150-8097.**

ABOUZEID, A. et al. HadoopDB: an architectural hybrid of MapReduce and DBMS technologies for analytical workloads. Proceedings of the VLDB Endowment**, v. 2, n. 1, p. 922-933, Agosto 2009. ISSN 10.14778/1687627.1687731.**

AILAMAKI, A. et al. Weaving Relations for Cache Performance**. Proceedings of the 27th VLDB Conference. Roma, Italy: ACM. 2001. p. 169-180.**

ALQARNI, A. A.; PARDEDE, E. Integration of Data Warehouse and Unstructured Business Documents**. 2012 15th International Conference on Network-Based Information Systems (NBiS). Melbourne, VIC: IEEE. 2012. p. 32 - 37.**

ANTUNES, R.; FURTADO, P. Hardware Capacity Evaluation in Shared-Nothing Data Warehouses**. Parallel and Distributed Processing Symposium. Long Beach, CA: IEEE. 2007. p. 1 - 6.**

APACHE. Hadoop. Apache Hadoop**, 2014. Disponivel em: <http://hadoop.apache.org/>. Acesso em: 01 dez. 2014.**

BALLARD, C. et al. Data Modeling Techniques for Data Warehousing**. 1. ed. San Jose: IBM Redbooks, 1998.**

BEDI, P.; JINDAL, V.; GAUTAM, A. Beginning with big data simplified**. 2014 International Conference on Data Mining and Intelligent Computing (ICDMIC). New Delhi: IEEE. 2014. p. 1-7.**

CARNIEL, A. C. et al. Query processing over data warehouse using relational databases and NoSQL**. Conferencia Latinoamericana En Informatica (CLEI), 2012 XXXVIII. Medellin: IEEE. 2012. p. 1 - 9.**

CATTEL, R. Scalable SQL and NoSQL data stores. ACM SIGMOD Record**, New York, v. 39, n. 4, p. 12-27, Dezembro 2010.**

CELKO, J. Joe Celko's Complete Guide to NoSQL**. Waltham: Morgan Kaufmann, 2013.**

CHANDARANA, P.; VIJAYALAKSHMI, M. Big Data analytics frameworks**. 2014 International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA). Mumbai: IEEE. 2014. p. 430 - 434.**

CHAUDHARY, S.; MURALA, D. P.; SRIVASTAV, V. K. A Critical Review of Data Warehouse. Global Journal of Business Management and Information Technology**, Panipat, v. 1, n. 2, p. 95-103, 2011.**

COX, M.; ELLSWORTH, D. Application-controlled demand paging for out-of-core visualization**. Proceedings Visualization '97. Phoenix, AZ, USA: IEEE. 1997. p. 235 - 244.**

DEAN, J.; GHEMAWAT, S. MapReduce Simplified Data Processing on Large Clusters**. Sixth Symposium on Operating System Design and Implementation. San Francisco, CA: Google Research Publications. 2004. p. 1-13.**

DEMCHENKO, Y.; DE LAAT, C.; MEMBREY, P. Defining architecture components of the Big Data Ecosystem**. 2014 International Conference on Collaboration Technologies and Systems (CTS). Minneapolis, MN: IEEE. 2014. p. 104 - 112.**

DOSHI, K. A. et al. Blending SQL and NewSQL Approaches: **Reference Architectures for Enterprise Big Data Challenges. 2013 International Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC). Beijing: IEEE. 2013. p. 163 - 170.**

DWHWORLD. DWH Schemas. dwhworld**, 2010. Disponivel em: <http://www.dwhworld.com/dwh-schemas/>. Acesso em: 01 dez. 2014.**

EBNER, K.; BUHNEN, T.; URBACH, N. Think Big with Big Data: **Identifying Suitable Big Data Strategies in Corporate Environments. 2014 47th Hawaii International Conference on System Sciences (HICSS). Waikoloa, HI: IEEE. 2014. p. 3748 - 3757.**

ECOMCANADA. Storing and querying data Big Data in HDFS. Ecomcanada**, 2013. Disponivel em: <http://ecomcanada.wordpress.com/2012/11/14/storing-and-querying-bigdata-in-hadoop-hdfs/>. Acesso em: 01 dez. 2014.**

FISHER, D. et al. Interactions with Big Data Analytics. interactions**, New York, Maio 2012. 50-59.**

FOWLER, M.; SADALAGE, P. J. NoSQL Distilled: **A Brief Guide to the Emerging World of Polyglot Persistence. Crawfordsville, Indiana: Addison-Wesley Professional, 2012.**

FURLOW, G. The Case for Building a Data Warehouse. IT Professional**, v. 3, n. 4, p. 31 - 34, Agosto 2002. ISSN 1520-9202.**

GOLAB, L.; JOHNSON, T. Data stream warehousing**. 2014 IEEE 30th International Conference on Data Engineering (ICDE). Chicago, IL: IEEE. 2014. p. 1290 - 1293.**

GOSS, R. G.; VEERAMUTHU, K. Heading towards big data building a better data warehouse for more data, more speed, and more users**. 2013 24th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC). Saratoga Springs, NY: IEEE. 2013. p. 220 - 225.**

GRIMES, S. Structure, Models and Meanin. InformationWeek**, 2005. Disponivel em: <http://www.informationweek.com/software/information-management/structure-models-and-meaning/d/d-id/1030187?>. Acesso em: 01 2014 Dezembro.**

GROLINGER, K. et al. Challenges for MapReduce in Big Data**. 2014 IEEE World Congress on Services (SERVICES). Anchorage, AK: IEEE. 2014. p. 182 - 189.**

GRUSKA, N.; MARTIN, P. Integrating MapReduce and RDBMSs**. Proceedings of the 2010 Conference of the Center for Advanced Studies on Collaborative Research. Riverton: IBM Corp. 2010. p. 212-223.**

GUDIVADA, V. N.; RAO, D.; RAGHAVAN, V. V. NoSQL Systems for Big Data Management**. 2014 IEEE World Congress on. Anchorage, AK: IEEE. 2014. p. 190 - 197.**

GUPTA, V.; GOSAIN, A. A Comprehensive Review of Unstructured Data Management Approaches in Data Warehouse**. 2013 International Symposium on Computational and Business Intelligence (ISCBI). New Delhi: IEEE. 2013. p. 64 - 67.**

HAJMOOSAEI, A.; KASHFI, M.; KAILASAM, P. Comparison plan for data warehouse system architectures**. 2011 3rd International Conference on Data Mining and Intelligent Information Technology Applications (ICMiA). Macao: IEEE. 2011. p. 290 - 293.**

HAN, J. et al. Survey on NoSQL database**. 6th International Conference on Pervasive Computing and Applications (ICPCA). Port Elizabeth: IEEE. 2011. p. 363-366.**

HANSON, E. N. Columnstore Indexes for Fast Data Warehouse Query Processing in SQL Server 11.0**. Microsoft. [S.l.], p. 8. 2010.**

HE, Y. et al. RCFile: **A fast and space-efficient data placement structure in MapReduce-based warehouse systems. 2011 IEEE 27th International Conference on Data Engineering (ICDE). Hannover: IEEE. 2011. p. 1199 - 1208.**

HPCC. HPCC Systems. HPCC Systems**, 2014. Disponivel em: <http://hpccsystems.com/>. Acesso em: 01 dez. 2014.**

HU, H. et al. Toward Scalable Systems for Big Data Analytics: A Technology Tutorial. IEEE Access**, v. 2, p. 652 - 687, Junho 2014. ISSN 2169-3536.**

IBM. Analytics-as-a-Service Platform. Watson IBM**, 2013. Disponivel em: <http://researcher.watson.ibm.com/researcher/view_group.php?id=3992>. Acesso em: 01 dez. 2014.**

INMON, W. H. Building the Data Warehouse**. 4. ed. Indianapolis: John Wiley & Sons, 2005.**

INMON, W. H.; KRISHNAN, K. Building the Unstructured Data Warehouse: **Architecture, Analysis, and Design. 1. ed. Bradley Beach: Technics Publications, LLC, 2011.**

INTEL. White Paper: Extract, Transform, and Load Big Data with Apache Hadoop**. Intel. [S.l.], p. 1-9. 2013.**

KALA KARUN, A.; CHITHARANJAN, K. A review on hadoop — HDFS infrastructure extensions**. 2013 IEEE Conference on Information & Communication Technologies (ICT). JeJu Island: IEEE. 2013. p. 132 - 137.**

KATAL, A.; WAZID, M.; GOUDAR, R. H. Big data: **Issues, challenges, tools and Good practices. 2013 Sixth International Conference on Contemporary Computing (IC3). Noida: IEEE. 2013. p. 404 - 409.**

KIMBALL, R.; ROSS, M. The Data Warehouse Toolkit**. 3. ed. New York: John Wiley & Sons, 2013.**

LANEY, D. 3D Data Management: Controlling Data Volume, Velocity and Variaety. Application Delivery Strategies - Meta Group**, Stamford, n. File No. 949, p. 1-4, Fevereiro 2001.**

LENZERINI, M. Data Integration: **A Theoretical Perspective. Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. New York: ACM. 2002. p. 233 - 246.**

LOMOTEY, R. K.; DETERS, R. Towards Knowledge Discovery in Big Data**. 2014 IEEE 8th International Symposium on Service Oriented System Engineering (SOSE). Oxford: IEEE. 2014. p. 7-11.**

LOSHIN, D. Big Data Analytics**. Waltham: Morgan Kaufmann, 2013.**

MADDEN, S. From Databases To Big Data. IEEE Internet Computing**, Los Alamitos, CA, Junho 2012. 4-6.**

MANIKANDAN, S. G.; RAVI, S. Big Data Analysis Using Apache Hadoop**. 2014 International Conference on IT Convergence and Security (ICITCS). Beijing: IEEE. 2014. p. 1-4.**

MATEI, G. Column-Oriented Databases, an Alternative for Analytical Environment. Database Systems Journal**, Bucharest, ROMANIA, v. I, n. 2, p. 3-16, 2010.**

MATHUR, A. et al. A New Perspective to Data Processing: **Big Data. 2014 International Conference on Computing for Sustainable Global Development (INDIACom). New Delhi: IEEE. 2014. p. 110-114.**

MOHAMMAD RIFAIEA, K. K. R. A. M. J. R. Data warehouse architecture and design**. IEEE International Conference on Information Reuse and Integration, 2008. IRI 2008. Las Vegas, NV, USA: IEEE. 2008. p. 58-63.**

MOHANTY, S.; JAGADEESH, M.; SRIVATSA, H. Big Data Imperatives: **Enterprise 'Big Data' Warehouse, 'BI' Implementations and Analytics. New York: Apress, 2013.**

MORASCHI, D. Business Intelligence with MicroStrategy Cookbook**. Birmingham: Packt Publishing, 2013.**

PAVLO, A. et al. A Comparison of Approaches to Large-Scale Data Analysis**. Proceedings of the 2009 ACM SIGMOD International Conference on Management of data. New York: SIGMOD. 2009. p. 165-178.**

PORNAIN, X. Rewiring to Tackle Unstructured Data. Wired**, 2014. Disponivel em: <http://www.wired.com/2014/07/rewiring-tackle-unstructured-data/>. Acesso em: 01 dez. 2014.**

PROFFITT, B. Hadoop vs. an RDBMS: How much (less) would you pay. ITWorld**, 2012. Disponivel em: <http://www.itworld.com/article/2827051/big-data/hadoop-vs--an-rdbms--how-much--less--would-you-pay-.html>. Acesso em: 01 Dezembro 2014.**

PROKOPP, C. 32x Faster Hadoop and MapReduce With Indexing. Big Data Science and Cloud Computing**, 2013. Disponivel em: <http://www.semantikoz.com/blog/hadoop-index-optimise-map-reduce-data-access-result-in-32x-speedup/>. Acesso em: 01 dez. 2014.**

R. KELLY RAINER, C. G. C. Introduction to Information Systems**. 4. ed. [S.l.]: Wiley, 2012.**

RABL, T. et al. Solving Big Data Challenges for Enterprise Application Performance Management**. Proceedings of the 38th Conference on Very Large Databases (VLDB). Istanbul: VLDB. 2012. p. 1724-1735.**

RAHMAN, N. SQL optimization in a parallel processing database system**. 2013 26th Annual IEEE Canadian Conference on Electrical and Computer Engineering (CCECE). Regina, SK: IEEE. 2013. p. 1-5.**

RAY, C. Distributed Database Systems**. Kolkata: Pearson India, 2009.**

RUPANAGUNTA, K.; ZAKKAM, D.; RAO, H. How to Mine Unstructured Data. Information Management**, 2012. Disponivel em: <http://www.information-management.com/newsletters/data-mining-unstructured-big-data-youtube--10022781-1.html>. Acesso em: 01 dez. 2014.**

S.G.YAMAN. Introduction to Column-Oriented Database Systems**. Columnar Databases Seminar, Helsinki, Finland: [s.n.]. 2012. p. 1-6.**

SAGIROGLU, S.; SINANC, D. Big data: **A review. 2013 International Conference on Collaboration Technologies and Systems (CTS). San Diego, CA: IEEE. 2013. p. 42 - 47.**

SIMONSON, W. E.; ALSBROOKS, W. T. A DBMS for the U. S. Bureau of the Census**. Proceedings of the 1st International Conference on Very Large Data Bases. New York: ACM. 1975. p. 496-498.**

STONEBRAKER, M. et al. C-Store: **A Column-oriented DBMS. Proceedings of the 31st VLDB Conference on Very large data bases. Trondheim, Norway: VLDB Endowment. 2005. p. 553 - 564.**

TEAM TREFIS. Strong Database And Application Software Sales Could Lift Oracle Higher. forbes.com**, 2014. Disponivel em: <http://www.forbes.com/sites/greatspeculations/2014/11/18/salesforce-q3fy15-preview-growing-customer-base-should-drive-revenues-higher/>. Acesso em: 01 dez. 2014.**

TEKINER, F.; KEANE, J. A. Big Data Framework**. 2013 IEEE International Conference on Systems, Man, and Cybernetics (SMC). Manchester: IEEE. 2013. p. 1494 - 1499.**

USP. Diretrizes para Apresentação deDissertações e Teses. 4ª. Edição**. 4. ed. São Paulo: Divisão de Biblioteca - EPUSP, 2013.**

VAISH, G. Getting started with NoSQL**. Birmingham: Packt Publishing, 2013.**

VASSILIADIS, P.; SIMITSIS, A. EXTRACTION, TRANSFORMATION, AND LOADING. In: LIU, L.; ÖZSU; TAMER, M. Encyclopedia of Database Systems**. [S.l.]: Springer, 2009. p. 1095-1101. ISBN 978-0-387-35544-3.**

VIQARUNNISA, P.; LAKSMIWATI, H.; AZIZAH, F. N. Generic data model pattern for data warehouse**. 2011 International Conference on Electrical Engineering and Informatics (ICEEI). Bandung: IEEE. 2011. p. 1-8.**

WIKIBON. Wikibon Blog. Wikibon**, 2012. Disponivel em: <wikibon.org/blog/big-data-statistics/>. Acesso em: 01 dez. 2014.**

YOU, J. et al. HDW: **A High Performance Large Scale Data Warehouse. International Multisymposiums on Computer and Computational Sciences. Shanghai: IEEE. 2008. p. 200-202.**

ZAÏANE, D. O. R. Principles of Knowledge Discovery in Databases**. University of Alberta. Alberta, p. 1-8. 1999.**

ZHANG, J.; HUANG, M. L. 5Ws model for BigData Analysis and Visualization**. 2013 IEEE 16th International Conference on Computational Science and Engineering. Sydney, Australia: IEEE. 2013. p. 1021-1028.**

ZHANG, L. et al. Moving Big Data to The Cloud: An Online Cost-Minimizing Approach. IEEE Journal on Selected Areas in Communications**, v. 31, n. 12, p. 2710 - 2721, dezembro 2013. ISSN 0733-8716.**

ZHIJUAN, W.; HONGCHANG, W.; XUEFANG, W. A Data Warehouse Design Method**. 2012 International Conference on Computer Science & Service System (CSSS). Nanjing: IEEE. 2012. p. 2063 - 2066.**

APÊNDICE A – EXEMPLO DE CÓDIGO PARA CONTAGEM DE PALAVRAS NUMA COLEÇÃO DE DOCUMENTOS NO MAPREDUCE

```
map(String key, String value):
// key: document name
// value: document contents
for each word w in value:
EmitIntermediate(w, "1");
reduce(String key, Iterator values):
// key: a word
// values: a list of counts
int result = 0;
for each v in values:
result += ParseInt(v);
Emit(AsString(result));
```