UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UMA ARQUITETURA NÃO INTRUSIVA E REATIVA PARA REALIZAR O PROCESSO ETL EM TEMPO REAL EM AMBIENTES DE DATA WAREHOUSING

FLÁVIO DE ASSIS VILELA

ORIENTADOR: PROF. DR. RICARDO RODRIGUES CIFERRI

São Carlos – SP

Dezembro, 2021

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UMA ARQUITETURA NÃO INTRUSIVA E REATIVA PARA REALIZAR O PROCESSO ETL EM TEMPO REAL EM AMBIENTES DE DATA WAREHOUSING

FLÁVIO DE ASSIS VILELA

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos (UFSCar), como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação, área de concentração: Engenharia de Software, Banco de Dados e Interação Humano-Computador.

Orientador: Prof. Dr. Ricardo Rodrigues Ciferri

São Carlos – SP

Dezembro, 2021

Flávio de Assis Vilela
Uma arquitetura não intrusiva e reativa para realizar o processo ETL em tempo real em ambientes de data warehousing/ Flávio de Assis Vilela. – São Carlos – SP, Dezembro, 2021-
221 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Ricardo Rodrigues Ciferri

– Universidade Federal de São Carlos, Dezembro, 2021.

1. Data Warehousing. 2. ETL. 2. Tempo Real. I. Orientador. II. Universidade Federal de São Carlos (UFSCar). III. Centro de Ciências Exatas e de Tecnologia. IV. Uma arquitetura não intrusiva e reativa para realizar o processo ETL em tempo real em ambientes de Data Warehousing.

Flávio de Assis Vilela

Uma arquitetura não intrusiva e reativa para realizar o processo ETL em tempo real em ambientes de data warehousing

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos (UFSCar), como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação, área de concentração: Engenharia de Software, Banco de Dados e Interação Humano-Computador.

São Carlos – SP, 20 de dezembro de 2021:

Prof. Dr. Ricardo Rodrigues Ciferri (UFSCar)
Orientador

Prof. Dr. Anderson Chaves Carniel (UFSCar)

Profa. Dra. Marilde Terezinha Prado Santos (UFSCar)

Documento assinado digitalmente
PATRICIA DELLA MEA PLENTZ
Data: 29/03/2022 10:58:57-0300
CPF: 959.243.700-97
Verifique as assinaturas em https://v.ufsc.br

Profa. Dra. Patricia Della Méa Plentz (UFSC)

Documento assinado digitalmente
Renato Fileto
Data: 30/03/2022 09:09:12-0300
CPF: 106.315.998-94
Verifique as assinaturas em https://v.ufsc.br

Prof. Dr. Renato Fileto (UFSC)

São Carlos – SP
Dezembro, 2021

Este trabalho é dedicado aos meus pais Geraldo e Rosângela, à minha noiva Amanda e as minhas irmãs Larissa e Lorena.

AGRADECIMENTOS

Ao meu pai Geraldo e a minha mãe Rosângela pelo carinho, amor, apoio e estímulo contante.

A minha noiva Amanda pelo amor, carinho, paciência e companheirismo em todas as horas.

A minha irmã Lorena, meu cunhado Fabrício e minhas afilhadas Larissa e Maria Alice, pela paciência e apoio nos momentos que sempre precisei.

A minha irmã Larissa, por me acompanhar espiritualmente desde a época da especialização, por ter me mostrado, infelizmente de forma trágica, que a vida é maior e melhor do que qualquer bem material, por nunca ter deixado eu cair quando eu não quis mais ficar de pé, por me abençoar a cada dia e me mostrar os passos que devo seguir.

Aos meus avós José e Jerônima, pois, se eu sou como eu sou, eu devo a esse casal que muito se doou para criar e educar os filhos e netos.

Aos meus avós Irineu e Alice, pois, enquanto vivos, me guiou e auxiliou na construção do meu caráter.

Aos meus tios Adeneide, Enivaldo, Reginaldo, Agnaldo, Fernando, Mário, Maurílio e Júlio pela paciência e pelo apoio incondicional, desde o meu nascimento até hoje.

Às minhas tias Rosilda, Vilma, Sirlene, Iramar, Cristina, Maria Angélica, Cláudia e Vanilda pelo carinho e zelo por mim em todos os momentos.

Aos meus primos e primas pelo apoio e motivação de sempre.

Aos meus afilhados Ulysses, Angélica, Arthur, Larissa e Maria Alice pela paciência em minha ausência enquanto estive cursando o doutorado.

Ao meu sogro Sandro, a minha sogra Katia e meu cunhado Victor pela paciência e apoio nos momentos mais difíceis.

Ao meu orientador de doutorado, professor Ricardo Rodrigues Ciferri. Primeiro, por ter me aceitado como seu orientando de doutorado. Segundo, pelo apoio, paciência, educação, empenho e sabedoria em me conduzir ao longo de todo o Doutorado. Terceiro, por ter me ensinado a ser critico sem ser vulgar, a ter me ensinado a ser pesquisador, a ter me ensinado que a humildade é maior que qualquer vaidade. Esses fatos foram e serão essenciais para a minha formação pessoal, espiritual, humana e profissional.

Ao Pedro Henrique Fonseca Bertoleti e Ronitti Juner da Silva Rodrigues pelo apoio no assunto de Internet das Coisas e sensores.

Aos meus colegas do grupo de pesquisa em Banco de Dados da UFSCar, pelo apoio e incentivo

ao longo do Doutorado.
Aos meus colegas do IFG, pelo apoio e companheirismo de sempre nos momentos que estive ausente da sala de aula e até mesmo quando retornei.
Aos meus amigos Gideone Rosa, Diego Evangelista e Daniel Canedo pelo apoio moral e incentivo nos momentos mais difíceis.
A todas as pessoas que me apoiaram e me ampararam assim que cheguei em São Carlos.
À UFSCar, por me permitir cursar o Doutorado em uma instituição de qualidade e com professores reconhecidos mundialmente.
Ao IFG, por ter me concedido a dispensa das minhas atividades de docência durante 4 anos para cursar o Doutorado.

RESUMO

É cada vez maior o interesse em se obter dados que apoiem o processo de tomada de decisão estratégica nas organizações. Esses dados estão disponíveis em fontes de dados no ambiente operacional, as quais são autônomas, heterogêneas e distribuídas. Os dados são obtidos por meio do processo de Extração, Transformação e Carga (do inglês *Extract, Transform, and Loading* - ETL)) e armazenados no ambiente informacional em uma base de dados homogênea e dimensional chamada *data warehouse*. O processo ETL ocorre tradicionalmente em momentos predefinidos, tais como diariamente, semanalmente, mensalmente ou de acordo com as regras de atualização de dados da organização. Entretanto, existem aplicações que necessitam obter os dados operacionais o mais rápido possível ou imediatamente após os dados serem produzidos nas fontes de dados. Exemplos dessas aplicações são sistemas médicos, sistemas de controle de rodovias e sistemas para agropecuária digital. Portanto, o processo ETL tradicional e as técnicas disponíveis atualmente são incapazes de disponibilizar os dados para tomada de decisão em tempo real, garantindo os requisitos de disponibilidade, baixo tempo de resposta e escalabilidade. Este trabalho apresenta uma inovadora arquitetura não intrusiva e reativa, chamada Imã de Dados, a partir da qual é possível realizar o processo ETL em tempo real em ambientes de *data warehousing*. A característica não intrusiva permite que a solução não necessite buscar os dados no ambiente operacional e desta forma não é necessário realizar a conexão com as fontes de dados e nem lidar diretamente com a heterogeneidade dos dados. Já a característica reativa indica que a solução irá reagir a eventos ocorridos no ambiente operacional e executar uma ação automaticamente de forma a garantir os requisitos de tempo real. Dois testes experimentais foram realizados, o primeiro em ambiente real no domínio da pecuária leiteira e o segundo em um ambiente sintético, mostraram que o Imã de Dados é capaz de processar corretamente todo o fluxo de ETL em tempo real. Além disso, o Imã de Dados apresentou um bom desempenho com baixo tempo de resposta, garantiu disponibilidade e apresentou escalabilidade à medida que ocorreu o aumento do volume de dados. Em especial, o Imã de Dados produziu um grande ganho de desempenho considerando o tempo médio, ao ser comparado com a tradicional técnica de gatilhos, comumente usada em processos ETL de tempo real.

Palavras-chave: ETL, tempo real, data warehousing, data warehouse, extração de dados, carga de dados, carregamento de dados.

ABSTRACT

There is a great interest in obtaining data that support the decision-making process in business. These data are available in data sources in the operational environment, which are autonomous, heterogeneous, and distributed. The data are extracted through the Extract, Transform, and Load process (ETL) and stored in the informational environment in a homogeneous, integrated, and dimensional database called data warehouse. The ETL process traditionally takes place at predefined periods, such as daily, weekly, monthly, or according to the organization's data update rules. However, there are applications that need operational data as quickly as possible or immediately after the data is available from data sources. Examples of these applications are medical systems, highway control systems and digital farming systems. Therefore, the traditional ETL process and currently available techniques are unable to make the data available for decision making in real-time, ensuring availability, low elapsed time, and scalability. This work presents an innovative, non-intrusive and reactive architecture, called Data Magnet, from which it is possible to perform the ETL process in real time in data warehousing environments. The non-intrusive feature means that the solution does not need to search for data in the operating environment and, therefore, it is not necessary to make a connection with the data sources or deal directly with the heterogeneity of the data. The reactive feature indicates that the solution will react to events in the operating environment and perform an automatic action in order to guarantee real-time requirements. Two experimental tests were performed, the first one in a real environment in the field of dairy farming, and the second one in a synthetic environment, in order to assess the Data Magnet with a high volume of data. In addition, the Data Magnet produced a good performance with low elapsed time, guaranteed availability and great scalability as the data volume increased. The Data Magnet also produced a huge performance gain for the average metric with regard to the traditional trigger technique commonly used in real-time ETL process.

Keywords: *ETL, real-time, data warehousing, data warehouse, data extraction, data loading*.

LISTA DE SIGLAS

ETL	*Extract, Transform, and Load*
XML	*Extensible Markup Language*
JSON	*JavaScript Object Notation*
OLAP	*Online Analytical Processing*
OLTP	*Online Transaction Processing*
BPMN	*Business Process Model Notation*
SQL	*Structured Query Languag*
ODBC	*Open Database Connectivity*
API	*Application Programming Interface*
IoT	*Internet of Things*
WSN	*Wireless Sensor Network*
SoA	*Service-oriented Architecture*
ATD	Área de Tratamento de Dados
FDT	Fonte de Dados Temporária
MOM	*Middleware* Orientado à Mensagem
CDC	*Change Data Capture*
DW	*Data Warehouse*
WAN	*Wide Area Networks*
IdC	Internet das Coisas

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

SUMÁRIO

Capítulo 1
INTRODUÇÃO

Este capítulo descreve a motivação para se obter dados em tempo real para tomada de decisão estratégica por meio do processo de Extração, Transformação e Carga (do inglês *Extract, Transform, and Loading* - ETL)). Para isso, a seção 1.1 retrata o cenário tradicional de um ambiente de *data warehousing*. A seção 1.2 descreve como ocorre o processo ETL tradicional. A seção 1.3 retrata o surgimento de novos domínios de negócios, os quais necessitam de dados em tempo real para tomada de decisão estratégica. Além disso, essa seção mostra o problema identificado ao surgir novas demandas por dados em tempo real. A seção 1.4 apresenta o objetivo geral e os objetivos específicos desta pesquisa de Doutorado. A seção 1.5 apresenta as hipóteses da pesquisa. Por fim, a seção 1.6 descreve a organização em capítulos desta tese de Doutorado.

1.1 Contextualização

É cada vez maior o interesse em se obter dados que apoiem o processo de tomada de decisão estratégica nas organizações. Esses dados estão disponíveis no ambiente operacional, isto é, no ambiente no qual ocorrem as transações dos usuários e estão armazenados em diferentes formatos e em fontes de dados heterogêneas e distribuídas. Por exemplo, os dados podem estar armazenados em arquivos em *clusters* de computadores, em bancos de dados relacionais usando servidores distribuídos com localização geográfica distinta, em fontes de dados semi-estruturadas e não estruturadas, em arquivos de texto, em arquivos de e-mail e outros meios de armazenamento (MUKHERJEE; KAR, 2017). Os dados também podem ser mantidos nos formatos de arquivos texto CSV (*comma-separated-values*), XML, JSON e planilhas eletrônicas (YADRANJIAGHDAM et al., 2017). Além disso, os dados podem ser gerados: 1) manualmente, por meio da interação do usuário com sistemas, tais como redes sociais, sistemas de controle de gestão e sistemas *Web*; 2) automaticamente por meio de sensores, tais como sensores de presença, radares em rodovias e sensores conectados ao corpo de pessoas e animais; 3) automaticamente por meio de satélites, os quais enviam dados constantemente para estações de coleta de dados (por exemplo, dados climáticos e tráfego de veículos) e, a partir desses dados, as decisões são tomadas. Portanto, para que os dados possam ter um valor significativo para as organizações e efetivamente

auxiliar na tomada de decisão estratégica, eles devem passar pelas atividades de extração, limpeza, filtragem, transformação, integração, processamento e carga para serem posteriormente armazenados em uma base de dados no ambiente de *data warehousing* (CHANDRA, 2018; SABRY; ALI, 2014; MUKHERJEE; KAR, 2017).

Um ambiente de *data warehousing* compõe o ambiente informacional, isto é, o ambiente que apoia a gerência de dados e a tomada de decisão estratégica, cujo objetivo é promover novas ações de mercado, identificar possíveis falhas no processo organizacional, planejar novas atividades, determinar análise de tendências e outras análises (CIFERRI, 2002). Esse ambiente é composto por arquiteturas, algoritmos e ferramentas que possibilitam a obtenção de dados de fontes autônomas, heterogêneas e distribuídas e armazena-os em uma base de dados integrada, homogênea e dimensional chamada *data warehouse* (DW) (CIFERRI, 2002). Além disso, o ambiente informacional é caracterizado por lidar com dados históricos, ou seja, faz uso de fatos que já ocorreram para tomar decisões futuras. Como consequência ao acúmulo de dados, o volume de dados manipulado por esse ambiente se torna cada vez maior, podendo alcançar Petabytes de armazenamento (CIFERRI, 2002; SABRY; ALI, 2014; MUDDASIR; RAGHUVEER, 2017).

Os dados armazenados no DW são originados de itens de dados do ambiente operacional (por exemplo, nome, endereço, idade). Contudo, apenas os itens de dados de interesse para tomada de decisão estratégica são atualizados no DW (MUDDASIR; RAGHUVEER, 2017). Ao longo desta tese, os itens de dados de interesse do ambiente operacional serão chamados de dados de interesse. Por exemplo: um ambiente operacional pode gerar dados de interesse referentes as compras realizadas nos fornecedores. Cada compra pode ser representada pelos atributos Id da compra, Data da compra, Fornecedor e Valor Total da compra. Além disso, o ambiente operacional também gera itens de dados das vendas feitas aos clientes. Essas vendas podem ser representadas pelos atributos Id da venda, Data da venda, Produto, Vendedor, Local e Hora da venda. Entretanto, os usuários tomadores de decisões estratégicas utilizam apenas um subconjunto desses dados para tomar decisões, ou seja, por exemplo, Id da venda, Produto e Data da venda. Dessa forma, esse subconjunto de dados gerados e de interesse para os usuários tomadores de decisões é chamado de dados de interesse e somente eles serão armazenados no DW.

1.2 Processo ETL tradicional

Independentemente do item de dado de interesse do ambiente operacional, do seu formato e de sua forma de obtenção, o processo ETL deve ser aplicado de modo que os dados de interesse sejam enviados do ambiente operacional para o ambiente de *data warehousing*. Basicamente, o processo ETL é composto por um *workflow* de tarefas. Essas tarefas são organizadas logicamente por meio de três fases: 1) **Extração (E)**: o objetivo dessa fase é conectar às fontes de dados operacionais e obter os dados de interesse. Tipicamente, essa fase é realizada com o auxilio

de um *wrapper* ou alguma técnica já definida em banco de dados, tais como *trigger*, arquivo de *log*, arquivo delta ou replicação de dados. Nessa fase ocorre a comunicação com as fontes de dados operacionais, nas quais os dados de interesse estão armazenados. Além disso, essa fase deve lidar com a heterogeneidade das fontes de dados, visto que o ambiente operacional pode manter os dados com formatos e meios de acesso distintos. 2) **Transformação** (T): nessa fase ocorre a limpeza, a filtragem, a transformação, a integração e o processamento dos dados. Mais especificamente, os dados de interesse são limpos, ou seja, são eliminados os conflitos de valores e dados inconsistentes. Os dados limpos e integrados são mantidos temporariamente em uma base de dados chamada *Data Staging Area* (DSA), a qual tem um formato padrão para todos os dados armazenados, além de ser mantida em um local distinto do ambiente operacional. 3) **Carga**: essa fase é responsável por carregar os dados transformados da DSA para o DW (MUDDASIR; RAGHUVEER, 2017; KIMBALL et al., 2004). O resultado final dessa fase e consequentemente de todo o processo ETL é permitir um valor semântico para os dados de interesse e que tais dados sejam armazenados no DW de forma integrada e consistente. Como consequência a esse processo, favorece-se a tomada de decisão estratégica a partir de dados consistentes (KAKISH; KRAFT, 2012; MUDDASIR; RAGHUVEER, 2017).

O processo ETL é um elemento inseparável de um ambiente de *data warehousing*, além de ser o principal elemento desse ambiente. Segundo Kimball e Caserta (KIMBALL et al., 2004), cerca de 70% de todo o custo de desenvolvimento de um ambiente de *data warehousing* é gasto com o processo ETL. Além disso, os autores ainda dizem que o processo ETL pode alavancar ou afundar todo o ambiente de *data warehousing*. Esse fato sugere que o sucesso das consultas analíticas, da qualidade dos dados armazenados no DW e das tomadas de decisões estratégicas estão diretamente relacionadas com o processo ETL (PHANIKANTH; SUDARSAN, 2017; MAJEED; RÖHRIG, 2012). Isso se deve ao fato de que o processo ETL cria um valor semântico para um dado, isto é, faz com que um dado extraído do ambiente operacional possa ter um valor significativo para um usuário tomador de decisão. Uma vez que um valor semântico incorreto for gerado para um determinado dado, toda a tomada de decisão a partir desse valor semântico incorreto pode ter consequências catastróficas.

Tradicionalmente, a execução do processo ETL para atualizar os dados de interesse para o DW ocorre em momentos específicos e predefinidos, isto é, numa frequência diária no período da janela noturna, semanalmente, mensalmente ou de acordo com as regras de atualização de dados da organização. (N; K, 2017; MUKHERJEE; KAR, 2017; WIBOWO, 2015). Neste caso, o ambiente operacional e o ambiente de *data warehousing* comumente estão fora de uso para que a atualização ocorra rapidamente. Isso quer dizer que durante o processo de atualização, as consultas OLAP (*Online Analytical Processing*) não vão gerar resultados com base em dados atualizados. Esse fato cria limites para o uso do processo ETL tradicional a medida que à janela de atualização se torne maior. Isso cria desafios para organizações que possuem escritórios em várias partes do mundo, visto que o fuso horário de cada região impede de se usar uma mesma janela de atualização. Segundo Muddasir e Mohammed (MUDDASIR; RAGHUVEER, 2017),

esse fato se torna aceitável para ambientes que permitem essa característica e focam em objetivos a longo prazo e, neste caso, os dados podem ser atualizados com uma frequência menor. Além disso, essa abordagem de atualização é simples, amplamente estudada e aplicada, consegue resolver a demanda solicitada e com custo menor em comparação a outras abordagens (SABTU et al., 2017).

1.3 Novas demandas por dados atualizados em tempo real

A partir da evolução da forma de se gerar e manipular dados, surgiram aplicações que necessitam de dados atualizados em tempo real (SABTU et al., 2017; PHANIKANTH; SUDARSAN, 2017), tais como sistemas médicos (JAIN et al., 2012) e sistemas que lidam com sensores (MESITI et al., 2016). Além disso, aplicações nos domínios de *smart farming* e agropecuária digital (RYU et al., 2015; KULATUNGA et al., 2017; ZAMORA-IZQUIERDO et al., 2019; FUENTES et al., 2020) e também controle de tráfego de rodovias (FIGUEIRAS et al., 2017), apesar de geralmente não gerarem grandes volumes de dados, precisam garantir que os dados sejam armazenados no DW em tempo real de forma a tratar adequadamente o processo de tomada de decisão estratégica (SABTU et al., 2017; PHANIKANTH; SUDARSAN, 2017).

Segundo Sabtu et al. (SABTU et al., 2017), a característica de atualização do processo ETL tradicional afeta negativamente o processo de tomada de decisão para domínios que necessitam de dados em tempo real. De fato, as consultas OLAP processadas durante um período do dia podem retornar dados inconsistentes em comparação com a real situação da organização. Isso se deve ao fato de que o DW é atualizado em alguns momentos específicos (tais como diariamente, semanalmente, mensalmente) e o ambiente operacional gera dados continuamente ao longo do tempo (LI; MAO, 2015; MUDDASIR; RAGHUVEER, 2017). Desta forma, se torna um desafio adotar o processo ETL tradicional em domínios que necessitam de dados em tempo real. Além do mais, Mesiti et al. (MESITI et al., 2016) dizem que, atualmente, o grande número de sensores existentes, cada vez mais baratos e utilizados em domínios distintos para diferentes propostas, favorece o desenvolvimento de novos serviços, os quais precisam dos dados gerados pelos sensores em tempo real. Alguns exemplos desses serviços são: monitoramento de desastres naturais, controle de temperatura e pressão atmosférica. Do ponto de vista produtivo, alguns exemplos de serviços são: controle de congestionamento em rodovias, controle de umidade da terra em hortas e controle de pesagem de animais em fazendas. Entretanto, é necessário oferecer novos meios eficientes de se executar o processo ETL, de modo que tais dados oriundos de sensores possam ser disponibilizados em tempo real no DW para tomada de decisão estratégica.

A partir desse cenário, o processo ETL evoluiu para ser executado de forma distinta da tradicional e consequentemente o DW passou a ser atualizado por meio de três abordagens principais, as quais buscam diminuir o tempo de execução do processo ETL e consequentemente diminuir o tempo de atualização dos dados no DW: 1) ***on-demand:*** as etapas de extração e

carga de dados são executadas e os dados operacionais são armazenados diretamente no DW sem a garantia da realização das atividades de limpeza, filtragem, transformação, integração e processamento. A cada solicitação de consulta OLAP, as referidas atividades são executadas e o resultado da consulta é apresentado ao usuário. Isso obviamente encarece o custo de processamento de consultas OLAP, mas reduz o custo do processo ETL; 2) ***near real-time:*** os dados são armazenados no DW em um intervalo de tempo menor do que a forma tradicional. Assim, essa abordagem permite que o processo ETL seja executado não em tempo real, mas várias vezes ao dia. Neste caso, aceita-se um determinado tempo de execução acima do esperado. De acordo com Muddasir e Mohammed (MUDDASIR; RAGHUVEER, 2017), Kakish e Kraft (KAKISH; KRAFT, 2012), Langseth, J. (LANGSETH, 2004) e Sabry e Ali (SABRY; ALI, 2014) existem três diferentes técnicas para lidar com a abordagem *near real-time*. Essas técnicas são: *Direct trickle feed*, *Trickle and flip* and *External real-time data cache*. Basicamente, a principal diferença entre as três técnicas é a forma que se lida com os dados históricos e os dados de tempo real. Contudo, todas as técnicas permitem executar o processo em tempo quase real com o uso de uma estratégia intrusiva, isto é, acessam diretamente o ambiente operacional para buscar os dados de interesse.

Como pode ser notado a partir das duas abordagens *on-demand* e *near real-time*, a solução imediata para a atualização constante do DW é aumentar a frequência na qual é executado o processo ETL, isto é, mais de uma vez por dia, num período ocioso da aplicação ou sempre que houver alguma alteração nos dados no ambiente operacional (LI; MAO, 2015). Entretanto, esse processo pode causar outros três problemas: 1) **leitura de dados no ambiente operacional**: as fontes de dados do ambiente operacional servem a grupos de usuários que realizam transações continuamente, e assim, resulta em uma sobrecarga em operações de consulta para extrair os dados armazenados nessas fontes de dados (CHANDRA, 2018; N; K, 2017); 2) **escrita de dados no ambiente informacional**: o DW serve a outros grupos de usuários do ambiente informacional que executam consultas analíticas sobre os dados. Essas consultas normalmente são complexas e envolvem um enorme volume de dados e, por consequência, demandam um tempo excessivo para serem executadas. Desta forma, o DW estará indisponível para ser atualizado até que a execução das consultas seja finalizada (LI; MAO, 2015); 3) **tempo de atualização e volume de dados**: imagine que ambos os ambientes estejam ociosos, logo, uma atualização no DW poderia ser realizada. Entretanto, se houver um grande volume de dados a ser atualizado do ambiente operacional para o informacional, o processo se torna ainda mais demorado e todos os usuários terão que esperar o processo ser finalizado para ter acesso a ambos os ambientes (MUDDASIR; RAGHUVEER, 2017).

A partir das duas abordagens *on-demand* e *near real-time*, assim como os problemas que podem ser identificados ao aplicá-las no contexto de tempo real, novas estratégias foram introduzidas em ambientes de *data warehousing*, de modo a permitir a reação imediata do ambiente de *data warehousing* a eventos ocorridos no ambiente operacional. Assim, a terceira abordagem é chamada de abordagem ***real-time***. Embora o termo tempo real não seja homogêneo

na literatura de ambientes de *data warehousing*, isto é, são utilizados com mais de um significado para um mesmo contexto, pode-se estabelecer uma definição para tal conceito com base na literatura base de tempo real. Por definição, sistemas de tempo real devem reagir a eventos ocorridos no ambiente operacional, respeitando as restrições temporais impostas pela aplicação. O correto funcionamento desses sistemas depende não somente dos resultados da computação, mas também do tempo necessário para executá-los (PLENTZ, 2008). Já Liu, J. (LIU, 2000) define sistemas de tempo real como aqueles sistemas que devem completar seu trabalho e entregar o serviço para o qual foi projetado em um intervalo de tempo máximo predefinido. Kopetz, H. (KOPETZ, 2002) diz que esses sistemas devem cumprir suas tarefas baseando-se em um *deadline*, ou seja, cumprir as tarefas e respeitar um tempo máximo de execução.

Além do conceito de sistemas de tempo real, o termo tempo real pode ser classificado em *hard real-time* e *soft real-time* (LIU, 2000). *Hard real-time* significa que uma falha em cumprir uma restrição de tempo imposta pela aplicação pode causar uma consequência catastrófica, ao passo que *Soft real-time* significa que uma falha em cumprir uma restrição de tempo imposta pela aplicação não causa nenhum dano catastrófico. Independentemente do conceito de tempo real adotado para desenvolver alguma solução, deve-se respeitar o *overhead* implícito na troca de dados entre o ambiente operacional e o ambiente de *data warehousing*. Esses conceitos serão detalhados na seção 2.1.

Ao aplicar o conceito de tempo real no processo ETL em ambientes de *data warehousing*, todo o processo deve garantir três requisitos básicos, os quais compõem ambientes de tempo real: 1) **baixo tempo de execução**, isto é, os dados de interesse devem ser armazenados no DW imediatamente após serem disponibilizados no ambiente operacional; 2) **disponibilidade**, ou seja, o processo ETL deve estar disponível a qualquer momento para enviar os dados para o DW; 3) **escalabilidade**, ou seja, à medida que aumenta a frequência e o volume de dados a ser armazenado no DW, o processo ETL deve manter o desempenho estável e continuar oferecendo baixo tempo de execução. Além disso, é desejável que outros dois requisitos sejam respeitados: **recuperabilidade** e **desacoplamento** do processo ETL com o ambiente operacional (SABTU et al., 2017).

Diversas pesquisas vem sendo realizadas nos últimos anos sobre o processo ETL, cujo o objetivo é fornecer novos meios para se executar o processo ETL em tempo real em ambientes de *data warehousing*. Contudo, todas as soluções propostas são caracterizadas por serem intrusivas e não reativas. Ao serem intrusivas, as soluções devem acessar diretamente as fontes de dados do ambiente operacional em busca dos dados de interesse. Além disso, as soluções devem conhecer a forma de acesso e lidar com a heterogeneidade dos dados. A forma de acesso às fontes de dados e consequentemente aos dados de interesse é decorrente das técnicas do processo ETL tradicional, isto é, uso de gatilhos, arquivo de *log*, arquivo delta, replicação de dados ou outro meio de acesso aos dados disponibilizado por sistemas gerenciadores de bancos de dados ou por sistemas de tomada de decisão. Esse fato faz com que se crie um forte acoplamento entre o processo ETL e o

ambiente operacional, que por sua vez pode afetar negativamente o desempenho do processo ETL. Ao ser não reativo, as soluções não são preparadas para reagir a eventos ocorridos no ambiente operacional e executar alguma ação automaticamente.

1.4 Objetivos

A partir da contextualização, da motivação e do problema apresentado, o objetivo desta pesquisa de Doutorado foi investigar uma solução inovadora para realizar o processo ETL em tempo real. A solução é voltada especificamente para ser usada em ambientes de *data warehousing*, apesar de que pode ser também usada em outros domínios, tais como na descoberta de conhecimento em bancos de dados e mineração de dados. Nesse sentido, foi desenvolvida uma arquitetura não intrusiva e reativa, chamada de Imã de Dados, para lidar com o processo ETL em tempo real em ambientes de *data warehousing*, considerando o conceito de *soft real-time*, visto que o não cumprimento de um *deadline* não implica em falhas catastróficas, perdas financeiras ou descarte de animais. As fases do processo ETL que são consideradas no Imã de Dados são a extração e carregamento. Já a fase de transformação é considerada um *workflow* específico para cada domínio de aplicação e está fora do escopo deste trabalho. Para alcançar esse objetivo, os seguintes objetivos específicos foram considerados:

- Investigar arquiteturas, métodos, metodologias, técnicas, estratégias e estudos em geral aplicados no processo ETL em ambientes de *data warehousing*, os quais permitem executar o processo ETL em tempo real e de forma não intrusiva e reativa.
- Investigar como a heterogeneidade do ambiente de *data warehousing* afeta o processo ETL em tempo real.
- Investigar como os dados produzidos em um ambiente de tempo real afetam os requisitos de tempo real ao executar o processo ETL.
- Investigar a forma de gerenciamento dos metadados envolvidos na arquitetura.
- Propor uma arquitetura que seja capaz de apoiar a execução do processo ETL em tempo real em ambientes de *data warehousing*.

1.5 Hipóteses

As hipóteses de pesquisa foram as seguintes:

- A proposta de uma arquitetura não intrusiva e reativa permite a correta execução de um *workflow* de ETL em tempo real, com a garantia dos requisitos de disponibilidade, baixo tempo de execução, escalabilidade e recuperabilidade.

- A criação de uma arquitetura não intrusiva e reativa permite que o processo ETL em tempo real seja executado independentemente das características do ambiente operacional.
- A utilização do conceito de *tag* em arquitetura não intrusiva e reativa garante o desacoplamento entre o ambiente operacional e o processo ETL, favorecendo o desempenho e a escalabilidade da arquitetura.
- A utilização de um modelo *publish/subscribe* (pub/sub) auxilia para alcançar que o processo ETL seja executado em tempo real.

1.6 Organização do trabalho

Esta tese está organizada da seguinte forma: o Capítulo 2 descreve a fundamentação teórica sobre os principais conceitos envolvidos nesta pesquisa de Doutorado. O Capítulo 3 descreve o processo ETL, assim como suas fases e principais características. O Capítulo 4 mostra as características e particularidades de um ambiente de *data warehousing* de tempo real. O Capítulo 5 descreve detalhadamente a arquitetura do Imã de Dados. O Capítulo 6 descreve os dois experimentos, com dados reais e com dados sintéticos, os quais serviram para validar a viabilidade do Imã de Dados, assim como validar a garantia do atendimento aos requisitos de tempo real. Por fim, o Capítulo 7 descreve e analisa os trabalhos relacionados ao assunto desta pesquisa de Doutorado. Como leitura complementar, o Apêndice A descreve a revisão sistemática realizada para encontrar o maior número de trabalhos relacionados e consequentemente criar o embasamento necessário para o desenvolvimento desta pesquisa de Doutorado.

Capítulo 2
FUNDAMENTAÇÃO TEÓRICA

Este capítulo relaciona e descreve os conceitos que foram aplicados para desenvolver o Imã de Dados. A explicação desses conceitos é altamente importante para que o leitor possa entender os fundamentos que cercam o Imã de Dados, assim como entender o motivo pelo qual tal conceito foi aplicado. Assim, a seção 2.1 mostra os conceitos de um ambiente de tempo real. A seção 2.2 retrata os principais conceitos de Internet das Coisas (IdC). A seção 2.3 descreve as principais características de sensores. Além disso, a subseção 2.3.1 mostra a aplicação de sensores no domínio de Pecuária Digital. A seção 2.4 retrata o conceito e características de um ambiente intrusivo e reativo. A seção 2.5 mostra os conceitos e aplicações do sistema Publish/Subscribe (Pub/Sub). Por fim, a seção 2.6 finaliza o capítulo com as considerações finais.

2.1 Ambientes de tempo real

Diferentemente do ambiente convencional, um ambiente de tempo real é um tipo de ambiente no qual os dados são gerados no menor intervalo de tempo possível e por meio de diferentes formas. Além disso, esse ambiente é composto por sistemas de tempo real, ou seja, o tempo de resposta entre a ação de entrada e saída de dados é sempre em tempo real (ELLIS, 2014). Além disso, Liu, J. (LIU, 2000) define sistemas de tempo real como aqueles sistemas que devem completar seu trabalho e entregar o serviço para o qual foi projetado em um intervalo de tempo pré-definido. Segundo Plentz, P. (PLENTZ, 2008), sistemas de tempo real são aqueles que devem executar alguma ação baseado em algum evento ocorrido. Contudo, devem respeitar os atributos temporais definidos pelo sistema, além de cumprir suas tarefas baseados em *deadline*, ou seja, cumprir as tarefas em um tempo máximo de execução (KOPETZ, 2002). Alguns exemplos de sistemas de tempo real são: sistemas que monitoram decolagem e pouso de aeronaves, sistemas embarcados que monitoram pressão sanguínea e batimentos cardíacos e sistemas que monitoram algum componente de um veículo (LIU, 2000).

Esses conceitos e exemplos apresentados dizem respeito ao conceito clássico de sistemas de tempo real. No mesmo sentido, em um ambiente de *data warehousing* de tempo real esses conceitos também podem ser aplicados, pois nesse ambiente os dados disponíveis no ambiente

operacional devem ser enviados para o DW em tempo real. Contudo, deve-se respeitar um determinado *deadline* para que todo o processo seja concluído. Esse *deadline* é imposto pelo próprio ambiente em que é composto um ambiente de *data warehousing*. Além disso, esse tempo total pode aumentar em decorrência do *overhead* implícito na troca de dados entre o ambiente operacional e o ambiente de *data warehousing*. Isso ocorre devido a limitações de *hardware*, protocolos de rede, limitação na taxa de transferência dados, o uso da Internet, custos para executar a extração, limpeza, processamento, integração e carregamento dos dados para o DW ou qualquer outro tipo de restrição que possa afetar todo o processo ETL e consequentemente aumentar o tempo de resposta ao enviar os dados do ambiente operacional para o DW.

Os termos *overhead* implícito e *deadline* normalmente não são considerados ao abordar o conceito de tempo real em ambientes de *data warehousing*. Isso se deve ao fato de que grande parte da literatura assume o termo tempo real como sendo um tempo em milésimo de segundos, o que, de fato, não é. O *overhead* implícito é encontrado em qualquer aplicação em que há troca de dados entre ambientes heterogêneos. Além disso, com o advento da Internet, essa troca de dados se tornou cada vez mais comum e tornou cada vez mais prático o compartilhamento de dados. Contudo, a própria estrutura do ambiente pode afetar o tempo total gasto para realizar algum processo de em tempo real. Com relação ao *deadline*, qualquer aplicação que deseja compartilhar dados deve ter um tempo limite para que esse processo seja realizado. Dessa forma, o *deadline* é outro fator que afeta diretamente o tempo total gasto para realizar algum processo em tempo real.

A partir do fato que se deve considerar os termos *overhead* implícito e *deadline* em conjunto ao termo tempo real para ambientes de *data warehousing*, pode-se classificar o termo tempo real em *hard real-time* e *soft real-time* (LIU, 2000). *Hard real-time* quer dizer que, uma falha em cumprir uma restrição de tempo imposta pela aplicação pode causar uma consequência catastrófica. Um exemplo da aplicação do conceito de *hard real-time* são sistemas utilizados em aeronaves. As aeronaves executam uma variedade de tarefas durante o voo de forma automática e autônoma com base em eventos ocorridos. Essas tarefas devem ser executadas respeitando os aspectos temporais impostos pelo próprio ambiente. Por exemplo: se, por algum motivo, o avião perder altitude, o piloto automático deve imediatamente alinhar o avião conforme o plano de voo, sob pena de uma falha catastrófica ocorrer, por exemplo, a colisão com outra aeronave no mesmo sentido ou até mesmo a queda da aeronave. *Soft real-time* quer dizer que, uma falha em cumprir uma restrição de tempo imposta pela aplicação não causa nenhum dano catastrófico. Além disso, sistemas desenvolvidos com base no conceito de *soft real-time* normalmente não precisam provar os requisitos de desempenho em tempo real. Dessa forma, executar todas as tarefas em um determinado tempo estabelecido não é o único aspecto a ser considerado em relação a todo o processo. Um exemplo da aplicação do conceito de *soft real-time* são sistemas *on-line*. Sistemas *on-line* são sistemas de computador os quais a entrada de dados ocorre diretamente em um terminal e a saída de dados é transmitida diretamente para outro terminal.

A definição e classificação do termo tempo real, assim como a definição e consideração dos termos *overhead* implícito e *deadline* em ambientes de *data warehousing*, favorece a identificar novas possibilidades de pesquisa, assim como identificar aplicações já existentes que fazem uso de tais conceitos. Independentemente do fim para o qual a aplicação é desenvolvida, tais conceitos devem ser sempre considerados. Assim, alguns exemplos da aplicação desse conceito são (ELLIS, 2014):

- **Monitoramento de sistema**: em um sistema no qual os dados são gerados a cada cinco segundos, pode-se haver a necessidade de monitorar os dados que trafegam pela rede. Esses dados podem ser úteis para analisar potenciais ameaças aos usuários da rede ou analisar o comportamento dos usuários. Outra situação pode ser a necessidade de monitorar a temperatura dos dispositivos físicos conectados ao sistema. Nesse contexto, os dados de interesse para ambas as situações de análise são gerados em tempo real, assim como as ações desejadas para esses dados.

- **Análise de dados da *Web***: com a crescente demanda por transações feitas pela *Web*, tais como *e-commerce* e transações bancárias, a necessidade de dados para analisar os acessos e o comportamento dos usuários se tornou cada vez maior. Além disso, devido ao volume de usuário transacionais ser também cada vez maior, esses dados passaram a ser gerados em tempo real. Dessa forma, se tornou necessária a análise de tais dados em tempo real, a fim de atender a demanda por parte dos analistas de dados e tomadores de decisões.

- **Redes sociais**: as redes sociais são outro exemplo de ambientes de tempo real. Facebook, Twitter e Instagram, por exemplo, representam uma grande parcela dos usuários de redes sociais. Devido ao grande número de usuários, o volume de dados gerados por ambas as redes são em tempo real, pois o número de usuário fazendo comentários, curtidas e compartilhamento de dados é tão grande que tais dados são gerados em tempo real. Dessa forma, empresas de publicidade, propaganda e *e-commerce* utilizam esses dados gerados pelos usuários para oferecer ofertas, produtos, descontos e demais serviços baseados nos dados gerados. Assim, os dados gerados em tempo real devem ser manipulados, de modo a serem disponibilizados para os tomadores de decisões também em tempo real.

- **IdC**: com o surgimento de dispositivos sensoriais, tais como *smartphones*, sensores em relógios, sensores em câmeras de segurança e outros tipos de sensores, fez com que aplicações pudessem realizar tarefas sem a necessidade da interação entre dispositivos e o homem. Nesse sentido, uma organização pode requerer dados de interesse produzidos por meio de sensores. Além disso, esses sensores podem se comunicar entre si e realizar diversas tarefas automáticas, inclusive, armazenar esses dados de interesse no DW.

A partir dos conceitos apresentados e de exemplos de ambiente de *data warehousing* de tempo real os quais são aplicados, vale a pena destacar que existem alguns requisitos em

comum a todos as aplicações que atuam nesses ambientes de tempo real. Esses requisitos criam restrições para as aplicações, as quais devem ser desenvolvidas a partir de tais requisitos. Assim, pode-se dizer que os requisitos desejáveis para um ambiente de *data warehousing* de tempo real são (ELLIS, 2014; SABTU et al., 2017):

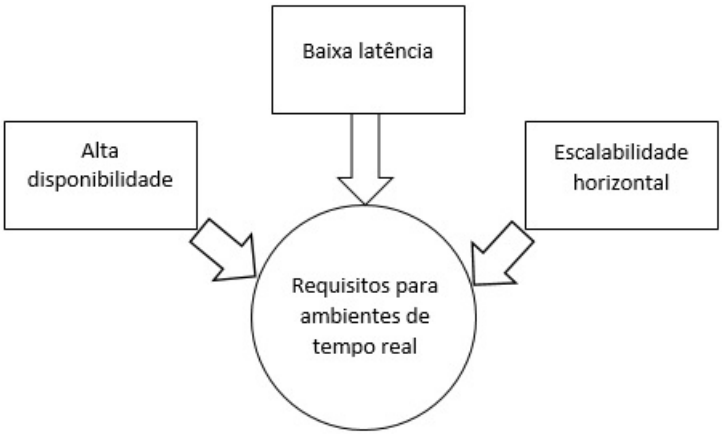
- **Disponibilidade**: em um ambiente de *data warehousing* convencional, alguma falha estrutural pode ocorrer, tais como indisponibilidade de rede, falhas de conexão ou outro motivo técnico. Essas falhas fazem com que os usuários aguardem o problema ser corrigido para continuar suas atividades de análise de dados e tomada de decisão. Por outro lado, em ambientes de tempo real, esse requisito deve ser considerado ao desenvolver alguma solução. Isso ocorre pois os usuários necessitam dos dados gerados em tempo real para a tomada de decisão. Dessa forma, a inatividade do ambiente torna o processo de análise de dados em tempo real impreciso, visto que a geração de dados e as consultas OLAP são afetadas. Assim, as soluções desenvolvidas para ambientes de tempo real devem priorizar a disponibilidade, ou seja, o ambiente deve estar disponível para geração e análise de dados mesmo em situações de falhas inesperadas.

- **Baixo tempo de resposta**: em ambientes convencionais, o tempo de resposta entre a geração de dados no ambiente operacional e a tomada de decisão a partir desses dados é outro requisito a ser considerado. Esse fato ocorre pois os usuários tomadores de decisões necessitam dos dados gerados imediatamente após serem produzidos no ambiente operacional para a tomada de decisão. Dessa forma, as soluções desenvolvidas para ambientes de tempo real devem priorizar técnicas para garantir o menor tempo de resposta possível.

  Vale a pena destacar que, geralmente, o termo latência e tempo de resposta são conceitos utilizados igualmente para um mesmo problema abordado, mas, de fato, são diferentes (LIU, 2000; KLEPPMANN, 2017). Latência é o tempo em que uma determinada tarefa espera (por exemplo, em uma lista de tarefas) para ser inicializada. Por outro lado, o tempo de resposta é o tempo total gasto para a tarefa ser realizada, incluindo a latência e algum *overhead* implícito para executar a tarefa e que atinge diretamente o tempo total gasto. Portanto, para ambientes de *data warehousing* em tempo real, o termo a ser considerado é tempo de resposta.

- **Escalabilidade**: as características de ambientes de tempo real fazem com que o volume de dados gerado e analisado se torne cada vez maior a medida que há novos usuários e novos meios de se produzir dados. O aumento do volume de dados afeta diretamente o desempenho de todo o ambiente, visto que as consultas OLAP exigidas pelos usuários são custosas e volumosas. Dessa forma, um ambiente escalável tem a característica de permitir que todo o ambiente tenha um desempenho adequado a medida que aumenta o volume de dados gerado.

Figura 1 – Representação dos requisitos para ambientes de tempo real.



Fonte: Adaptado de (SABTU et al., 2017).

A Figura 1 representa os requisitos necessários para sistemas aplicados em ambientes de tempo real. Segundo Sabtu et al. (SABTU et al., 2017) e Ellis, B. (ELLIS, 2014), a não aplicação ou aplicação parcial de um desses requisitos afeta diretamente ou limita as funcionalidades do sistema. Entretanto, pode-se notar que os autores não foram categóricos e não dizem que a falta desses requisitos descaracteriza um sistema para ambientes de tempo real. Dessa forma, pode-se interpretar como: mesmo com a falha ou a aplicação parcial de um dos requisitos de ambiente de tempo real, o sistema ainda será considerado um sistema para ambientes de tempo real.

Com a popularização do conceito de Internet das Coisas (IdC) e Sensores, o conceito de ambiente de tempo real se tornou ainda mais claro. Isso se deve ao fato que essas tecnologias lidam com dados em tempo real, isto é, produzem dados em alta frequência e manipulam e disponibilizam esses dados no menor tempo de resposta possível. Dessa forma, a integração dessas tecnologias ao conceito de ambiente de tempo real pretendido neste trabalho favorece a exemplificação do ambiente, além de favorecer o desenvolvimento de soluções tecnológicas modernas. Assim, nas próximas seções será apresentado o conceito dessas tecnologias, assim como exemplos de aplicações.

2.2 Internet das Coisas

Todos os dias, "coisas"são construídas e equipadas com sensores ou outros componentes que permitem seu controle remoto ou sua intercomunicação. Tais coisas podem ser veículos, eletrodomésticos, celulares, equipamentos médicos e outros tipos de objetos. Esses objetos são compostos por arquiteturas que permitem ser rastreados, monitorados e controlados por aplicativos de celular ou até mesmo interagir com seres humanos de forma autônoma. Além disso, uma característica importante é que os objetos podem ser controlados em tempo real, ou seja, o tempo entre a geração de um dado e uma consequente ação ocorrida é imediata. Por exemplo: um veículo pode se comunicar com outro veículo ou permitir seu rastreamento a partir

de dados coletados por sensores. Outro exemplo são celulares que podem se conectarem uns aos outros e executar tarefas, tais como troca de arquivos, envio de mensagens e etc.

Esses objetos, também chamados de dispositivos ou coisas, compartilham um aspecto em comum: são equipados com sensores ou outro meio de comunicação sem fio, os quais permitem a coleta de dados, sua intercomunicação e executam ações de forma autônoma. Dessa forma, surgiu o conceito de IdC, que é uma infraestrutura composta por dispositivos e tecnologias que permitem o sensoriamento, comunicação, compartilhamento e processamento de dados. Assim, é possível que objetos interajam entre si de forma automática, autônoma e em tempo real.

O conceito de IdC surgiu não somente com o uso de sensores, mas sim baseado em meios de comunicação sem fio em geral. Esses meios permitem enviar e receber dados, assim como podem permitir que objetos interajam entre si. Os principais meios de comunicação sem fio que deram origem ao conceito de IdC são:

- ***Radio-Frequency Identification (RFID)***: essa tecnologia de comunicação é composta por: 1) RFID *tag*: permite, a partir de um chip, registrar dados de interesse; 2) RFID *reader*: responsável por fazer a leitura do RFID *tag* e colocar os dados de interesse. Por meio do RFID, é possível obter dados de pessoas ou objetivos, além de rastreá-los e monitorá-los a partir das *tags* definidas. O RFID é amplamente aplicado em diferentes áreas, tais como praças de pedágio, empresas de logística, indústria farmacêutica e hospitais.

- ***Near Field Communication* (NFC)**: essa tecnologia de comunicação permite que dispositivos interajam entre si por meio de comunicação via rádio quando aproximados. Normalmente, essa tecnologia é integrada a *smartphones* para permitir a troca de dados entre si.

- ***Sensor Network***: sensores são dispositivos que monitoram características de objetos ou ambientes. Tais características podem ser umidade, temperatura, movimento e etc. Vários sensores podem ser utilizados em conjunto e realizar ações entre si. Dessa forma, quando isso ocorre, tem-se a *Wireless Sensor Network* (WSN). Nesse sentido, quando alguma característica é detectada pelos sensores, o atuador tem a função de executar alguma ação sobre o objeto ou ambiente. Tais ações podem ser: emitir sons, luzes, ondas de rádio ou outra ação desejada. Dessa forma, é possível que combinar os papeis dos sensores e atuadores e fazer com que ocorra de forma autônoma a comunicação de dispositivos entre si ou com humanos.

Os avanços dessas tecnologias de comunicação, assim como a descoberta de novos dispositivos que compartilham essas tecnologias, tornou a aplicação de IdC ainda mais difundida na indústria. Setores como fábricas e industrias em geral passaram a adotar tecnologias IdC em suas produções de modo a construir objetos inteligentes que permita obter informações detalhadas, atualizadas e em tempo real sobre um determinado contexto. Além da indústria, é

possível encontrar a aplicação de IdC em outros setores, tal como no setor da saúde. Por exemplo: em UTI hospitalar, sensores podem ser conectados em um paciente. Esses sensores disponibilizam dados sobre algumas características desejadas, tais como pressão arterial, batimento cardíaco e etc. A partir desses dados, outros dispositivos que recebem e leem esses dados podem executar ações automaticamente, por exemplo, emitir um alerta, prever alguma anomalia com o paciente, disponibilizar sugestões sobre ações a serem tomadas ou outras ações automáticas e/ou autônomas.

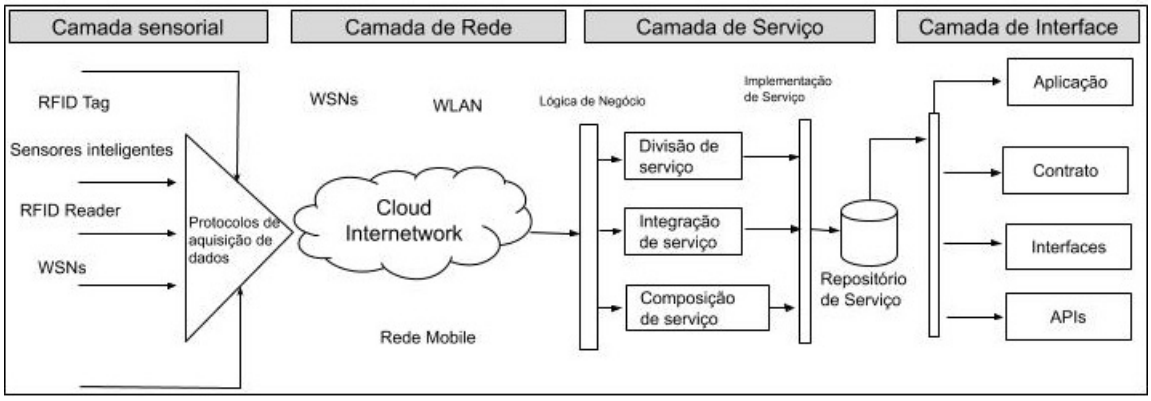
Embora a aplicação de IdC se tornou bastante difundida nos últimos anos, alguns desafios devem ser superados pelas organizações que desejam construir suas soluções IdC. Além disso, segundo Onyalo et al. (ONYALO et al., 2015), os desafios de segurança e privacidade são os dois aspectos mais críticos que devem ser superados, devido a mobilidade e a complexidade do ambiente IdC. Dessa forma, Whitmore et al. (WHITMORE et al., 2015) descrevem a segurança e privacidade como:

- **Segurança**: os dispositivos utilizados em IdC são essencialmente conectados em uma rede sem fio. Nessas redes podem conter diversos tipos de dispositivos e a criptografia dos dados trafegados pela rede se torna indispensável. Entretanto, alguns dispositivos não são equipados com tecnologias de criptografia suficientes para manter o dispositivo seguro em ambientes IdC. Dessa forma, novos algoritmos devem ser propostos de modo a garantir a segurança dos dados.
- **Privacidade**: quanto mais dispositivos são acrescentados em um ambiente IdC, mais a privacidade e garantia de propriedade dos dados devem ser consideradas. O proprietário de um dado deve ser informado e ter a garantia de que suas informações não serão utilizadas sem sua autorização, principalmente se tais informações forem compartilhadas. Assim, políticas de privacidade de dados devem garantir esse desafio.

Um aspecto importante de IdC é permitir que dispositivos executem ações entre si. Entretanto, além dos desafios citados, desenvolver tais soluções envolve outros fatores, tais como protocolos de comunicação, processamento de dados, modelos de dados e outros fatores que interferem diretamente no projeto. Além disso, os dispositivos podem ser movidos geograficamente, mas mesmo assim continuar se comunicando entre si em tempo real. Nesse sentido, a adaptabilidade, escalabilidade e interoperabilidade entre dispositivos heterogêneos devem ser considerados (ONYALO et al., 2015; XU et al., 2014).

Para garantir esses requisitos de integração entre diferentes dispositivos, *service-oriented architecture* (SoA) (arquitetura baseada em serviço) é a arquitetura base para o desenvolvimento de soluções IdC. A arquitetura SoA é amplamente aplicada em áreas como computação em nuvem, redes WSN e redes veiculares, além de permitir criar soluções multicamadas (XU et al., 2014). Dessa forma, pode-se garantir a interoperabilidade entre dispositivos heterogêneos (ONYALO et al., 2015).

Figura 2 – Arquitetura base de SoA para IdC.



Fonte: Adaptado de (ONYALO et al., 2015).

A Figura 2 representa a arquitetura base à aplicação da arquitetura SoA para IdC. A arquitetura é composta por quatro camadas: Camada Sensorial, Camada de Rede, Camada de Serviço e Camada de *Interface*.

- **Camada Sensorial**: essa camada representa os dispositivos físicos equipados com sensores ou RFID que tem a capacidade trocar ou compartilhar informações entre si. De acordo com Onyalo et al. (ONYALO et al., 2015), na construção dessa camada deve ser considerado os seguintes aspectos: 1) custo, tamanho, recursos e consumo de energia: esses fatores devem ser observados devido a grande quantidade de sensores que podem ser requeridos por um dispositivo ou aplicação; 2) desenvolvimento: esse fator deve ser considerado pois dados podem ser obtidos de sensores em tempo real, sob demanda ou de forma aleatória; 3) heterogeneidade: a variedade de dispositivos faz com que existam diversas formas de se manipular os sensores; 4) comunicação: devido a heterogeneidade, a comunicação também se torna outro aspecto importante, pois podem existir diferentes protocolos de comunicação. Dessa forma, implica diretamente no acesso e obtenção de dados.

- **Camada de Rede**: essa camada representa a infraestrutura de conexão responsável por estabelecer a comunicação entre os dispositivos. Além disso, essa camada agrega informações de diferentes infraestruturas já existentes. Esse fato resulta em uma heterogeneidade de conexões, fazendo com que se torne essencial a mudança automática de rede para que os dispositivos permaneçam conectados.

  Nessa camada, alguns aspectos devem ser considerados, tais como eficiência de energia, requisitos de qualidade dos serviços oferecidos, processamento de dados e de sinais, segurança e privacidade.

- **Camada de Serviço**: essa camada é composta por um *middleware* cuja responsabilidade é fornecer meios de se integrar os serviços e aplicações. Esse componente permite que *hardwares* e *softwares* sejam reutilizados. A especificação do *middleware* bem definida,

implica em disponibilizar uma API e protocolos para as aplicações, serviços e necessidades dos usuários. Além disso, essa camada é responsável pelo processamento dos serviços, como exemplo troca de informações, gerenciamento dos dados e comunicação.

- **Camada de *Interface***: a compatibilidade entre dispositivos se torna importante, pois eles são desenvolvidos por diferentes fornecedores e não possuem um mesmo padrão de comunicação. Dessa forma, essa camada é responsável por permitir que dispositivos sejam compatíveis entre si e economize o máximo de recurso possível. Além disso, essa camada pode ser considerada uma subcamada da camada de serviços, que permite especificar as funcionalidades necessárias da aplicação para serem encaminhadas para os respectivos serviços.

2.3 Sensores

De modo geral, os sensores são dispositivos utilizados para coletar características especificas de um ambiente ou objeto. Tais características podem ser: presença de um objeto em um ambiente, presença de gases, dados atmosféricos, temperatura de objetos e outros elementos que podem ser coletados por meio de sensores. Os sensores podem auxiliar usuários de sistemas e controladores de ambientes a entender melhor as características de tais ambientes ou objetos, além de monitorar ou rastrear objetos por meio dessas características. Essa característica dos sensores faz com que os mesmos possam ser aplicados em diferentes contextos e permite realizar o monitoramento ou rastreamento de objetos, dispositivos ou coisas em tempo real.

Em um ambiente podem haver diversos tipos de sensores, por exemplo: um sensor disponibiliza a temperatura do ambiente, outro sensor verifica a presença de um objeto e outro sensor verifica um som específico. Nesse cenário, tem-se uma rede do tipo WSN, que é uma rede composta por diversos tipos de sensores. Os sensores em uma rede WSN são dispostos em locais estratégicos, os quais são adequados para captar os dados de interesse. Os dados coletados a partir dos sensores são transmitidos para um servidor central, os quais são processados e disponibilizados para os usuários (YICK et al., 2008; SRIVASTAVA et al., 2019).

Os sensores em uma rede WSN tem a característica de autoprocessamento, ou seja, os sensores tem a capacidade de executar um processamento sobre os dados coletados antes de enviá-los aos servidores. Isso se deve ao fato de sensores inteligentes serem compostos por processador, memória, alimentação de energia, dispositivo de receber e enviar sinais de radio e um atuador. Dessa forma, os sensores podem transmitir apenas os dados processados, ou apenas os dados de interesse ou dados parciais coletados de um ambiente ou objeto (SRIVASTAVA et al., 2019). Esse aspecto resulta em um volume menor de dados transmitido, fazendo com que ocorra uma otimização no tempo de transmissão dos dados. Por sua vez, o atuador pode ser incorporado ao sensores de modo a permitir que ações possam ser realizadas automaticamente (YICK et al., 2008).

As WSN contém diversos tipos de sensores, cada qual com suas características e recursos. Entretanto, esse ambiente heterogêneo possui restrições e desafios que devem ser analisados ao serem implantados. As restrições e desafios do ponto de vista técnico são: limitações de processamento e armazenamento interno, capacidade limitada de energia e baixo alcance de comunicação. As restrições e desafios do ponto de vista de projeto são ditadas pelo ambiente no qual os sensores serão aplicados. Contudo, os aspectos a serem considerados são: o esquema utilizado, a topologia de rede utilizada e o tamanho da rede (YICK et al., 2008).

Os sensores tem grande potencial de ser aplicados em diferentes contextos. As características do domínio e a necessidade do usuário são os aspectos analisados ao construir uma WSN. Contudo, alguns exemplos de domínios em que uma WSN pode ser construída são: monitorar a presença de pessoas em um ambiente, verificar a presença de gases nocivos a saúde humana, prever ou monitorar fenômenos naturais e monitorar pacientes em hospitais. Os sensores podem ser aplicados de modo a obter as características desejadas e fornecer os dados de interesse aos usuários. Entretanto, quanto maior o ambiente ou quanto mais objetos a serem analisados, maior será a quantidade de sensores a ser utilizado. Dessa forma, o desenvolvimento de WSN *ad hoc* deve ser considerado para suprir as especificidades de cada tipo de ambiente (YICK et al., 2008).

2.3.1 Uso de sensores no domínio de Pecuária Digital

Como já foi comentado ao longo desta seção, a disseminação de sensores em diversas áreas de negócios favoreceu a obtenção de dados em tempo real. Esses dados podem ser utilizados para análises de dados em tempo real ou servem como fonte de dados para que dispositivos comuniquem entre si. Essa tecnologia passou a ser aplicada no domínio rural, a qual favoreceu a criação do conceito de Pecuária Digital. Esse conceito surgiu pelo fato de que os equipamentos desse domínio passaram a ser equipados com sensores. Dessa forma, pode-se propor soluções que permitem o controle e execução de tarefas automaticamente e em tempo real.

Na agricultura, os sensores podem ser encontrados nos tratores, colheitadeiras, implementos ou no próprio campo. Segundo Zamora-Izquierdo et al. (ZAMORA-IZQUIERDO et al., 2019), nos últimos anos, a introdução da tecnologia no campo favoreceu o gerenciamento automático de tarefas por meio de sistemas de monitoramento agronômico. Os sensores podem fornecer valores altamente acurados do estado de uma determinada cultura plantada, e com a aplicação de técnicas de IdC, é possível, por exemplo, controlar a irrigação de determinadas áreas e até mesmo aplicar nutrientes no solo. Além disso, as tecnologias impulsionam a agricultura de precisão e diminuem a necessidade da interação humana em atividades cotidianas.

Na pecuária, pode-se encontrar sensores nos reservatórios de leite, no ambiente no qual é produzido o leite e até mesmo nas próprias vacas. Segundo Kulatunga et al. (KULATUNGA et al., 2017), a aplicação de técnicas de IdC, Big Data e ferramentas de análises de dados para tomadas de decisões podem resolver alguns problemas enfrentados nesse domínio, tais como falta de mão-de-obra, problemas climáticos e questões de gerenciamento ambiental. Isso se deve

ao fato de que diversos sensores e outros dispositivos podem ser conectados a uma rede em uma fazenda. Dessa forma, pode-se extrair esses dados e disponibilizá-los aos usuários tomadores de decisões de forma integrada e em tempo real.

Chetan et al. (Chetan Dwarkani et al., 2015) afirmam que as tecnologias de informação e comunicação aplicadas na agropecuária digital permitem que os usuários tomem decisões com base em dados extraídos a partir de seus equipamentos. Dessa forma, pode-se automatizar diversas atividades que antes eram realizadas manualmente. Além disso, os autores descrevem alguns contextos nos quais foram aplicados técnicas de IdC, sensores e Big Data. Os contextos são: 1) controle de irrigação com base em dados obtidos de sensores na cultura de vegetais; 2) controle de pragas em plantação de tomate; 3) avaliação de um sistema de alerta utilizando uma rede de sensores e tecnologias de monitoramento de gado.

2.4 Ambiente intrusivo e reativo

Nos trabalhos encontrados na literatura até o momento sobre *data warehousing* de tempo real, as abordagens adotadas são consideradas intrusivas. Isso quer dizer que o processo ETL deve acessar diretamente os provedores de dados do ambiente operacional e buscar os dados de interesse. Para isso, o processo ETL deve lidar com a forma de acesso e a heterogeneidade dos dados do ambiente operacional. Só após lidar com todas essas tarefas, as fases do processo ETL são capazes de ser realizadas.

Essa abordagem, embora seja aplicada com sucesso durante anos, passou a enfrentar problemas assim que surgiu a necessidade de integrar as características de tempo real em ambientes de *data warehousing*. Alguns problemas enfrentados são: 1) as fontes de dados operacionais podem estar inativas, fazendo com que o processo ETL seja incapaz de buscar os dados periodicamente; 2) o ambiente operacional pode gerar um volume grande de dados e em uma frequência cada vez maior. Isso faz com que o processo ETL se torne custoso e demorado para realizar suas tarefas sobre um grande volume de dados. Esse fato, juntamente com o item 1, pode fazer com que as propriedades de tempo real, em específico as de baixo tempo de resposta e escalabilidade, não sejam respeitadas; 3) por ser intrusivo, é necessário conhecer toda a estrutura (esquema e forma de conexão) dos provedores de dados do ambiente operacional. Isso faz com que haja uma sobrecarga de informações por parte dos usuários administradores do ambiente de *data warehousing*; 4) ao ser intrusivo, o processo de extração de dados deve lidar com a heterogeneidade das fontes de dados operacionais.

A partir desse contexto, nota-se que a característica intrusiva se tornou inadequada para o contexto de *data warehousing* de tempo real. Isso se deve ao fato de que o ambiente de *data warehousing* de tempo real deve ser preparado para receber (e não ir buscar) os dados de interesse do ambiente operacional imediatamente. Dessa forma, para retirar essa característica intrusiva do processo ETL, neste trabalho serão adotadas duas abordagens fundamentais: não intrusiva

e reativa. Uma abordagem não intrusiva indica que uma solução não irá buscar os dados no ambiente operacional, ou seja, não realiza a conexão com provedores de dados operacionais e não lida com a heterogeneidade dos dados. Por consequência, não é necessário conhecer o esquema e a forma de acesso aos provedores de dados operacionais. Contudo, é necessário apenas conhecer o local e quais os dados de interesse que serão armazenados futuramente no DW. Já a abordagem reativa indica que uma solução irá reagir a eventos ocorridos no ambiente operacional e executar uma ação automaticamente. Por exemplo: 1) a partir de um valor gerado pelo usuário no ambiente operacional, uma aplicação pode automaticamente armazenar esse dado gerado no DW; 2) a partir de um dado recebido por meio de um arquivo, pode-se gerar relatórios automaticamente; 3) a partir de um dado recebido por meio de um sensor, pode-se gerar algum tipo de alerta para o usuário; 4) a partir de dados recebido por meio de sensores, pode-se processar esse dado, integrá-lo com outros dados de outros sensores e armazenar o resultado final no DW.

Percebe-se que o objetivo principal de um ambiente não intrusivo e reativo é reduzir a sobrecarga de informações necessárias para se extrair dados do ambiente operacional. Por consequência, o processo ETL deve lidar com menos tarefas a serem realizadas e, consequentemente, se torna menos custoso de ser executado. Além disso, se torna possível reduzir a sobrecarga de trabalho e a informação retida no usuário administrador, pois são eles que mantém informações sobre as fontes de dados e dados interesse do ambiente operacional (COOPERSTOCK et al., 1995). Dessa forma, ao ser não intrusivo, o sistema não busca os dados, mas sim recebe um dado gerado no ambiente operacional. Após ocorrer esse evento, a característica reativa faz com que seja executada uma ação automaticamente.

A partir dessas características de um ambiente não intrusivo e reativo, percebe-se que um ambiente com essas características difere de um ambiente intrusivo pela sua forma de aplicação. Em um ambiente intrusivo, o usuário é responsável por realizar a principal tarefa ou ação, isto é, emitir um relatório, emitir um alerta, ligar ou desligar dispositivos ou buscar dados em uma fonte de dados. Além disso, é necessário o prévio conhecimento do esquema e forma de acesso das fontes de dados operacionais e lidar com a heterogeneidade das mesmas. Isso faz com que, inevitavelmente, ocorra uma sobrecarga de informações e tarefas a serem realizadas pelo processo ETL e pelos usuários. Por outro lado, em um ambiente não intrusivo e reativo, essas tarefas são realizadas sem o prévio conhecimento das fontes de dados do ambiente operacional e de forma automática por meio de um sistema ou outros meios, os quais são dotados dessas características.

Especificamente em um ambiente de *data warehousing* de tempo real, as características não intrusiva e reativa se tornam essenciais. Isso se deve ao fato de que o processo ETL é executado sem a sobrecarga de tarefas citadas anteriormente. Por consequência, o processo ETL é desacoplado do ambiente operacional, fazendo com que o desempenho e a escalabilidade seja afetada positivamente. Nesse sentido, o ambiente de *data warehousing* de tempo real deve ser

preparado para receber, validar e armazenar os dados de interesse no DW imediatamente após serem recebidos do ambiente operacional. Assim, especificamente a fase de extração de dados se torna menos custosa de ser realizada, devido ao fato de não ter que lidar com a forma de acesso às fontes de dados operacionais e lidar com a heterogeneidade dos dados. Vale destacar que o ambiente operacional pode gerar um dado de forma manual por meio da interação do usuário com um sistema ou automática por meio de sensores. Em ambos os casos, um ambiente de *data warehousing* de tempo real não intrusivo e reativo deve ser preparado para receber esse dado gerado e executar as demais tarefas do processo ETL de forma automática.

Segundo Cooperstock et al. (COOPERSTOCK et al., 1995), uma forma de tornar o ambiente reativo é utilizar sensores. Isso se deve ao fato de que os sensores são capazes de detectar algum dado ou um comportamento esperado. O resultado dos dados obtidos dos sensores são encaminhados para sistemas computadorizados ou microcontroladores que, por sua vez, executa alguma ação determinada para um certo ambiente. No contexto de *data warehousing* de tempo real, um sensor pode ser utilizado para coletar dados de interesse dos usuários no ambiente operacional. Uma vez que o dado de interesse foi gerado, o mesmo é enviado para uma arquitetura que, por sua vez, valida e armazena o dado automaticamente no DW. Percebe-se que, nesse exemplo fictício, a arquitetura em questão não acessa o ambiente operacional e tampouco sabe suas características. Ao contrário, a arquitetura recebe (e não vai buscar) o dado de interesse originado pelo sensor e, por fim, executa as demais atividades automaticamente.

2.5 Sistemas publish/subscribe

Como já comentado ao longo deste trabalho, os dados de negócios para tomada de decisão estratégica passaram a ser gerados a partir de ambientes heterogêneos, os quais são compostos por diferentes tipos de provedores de dados, que por sua vez possuem diferentes tipos de acessos. Especificamente, em ambientes de *data warehousing*, esses dados se tornaram essenciais para os tomadores de decisões. Segundo Toshev, M. (TOSHEV, 2015), a principal questão ao lidar com ambientes com essas características é como implementar um meio de comunicação entre ambientes heterogêneos. Além disso, o autor diz que alguns fatores devem ser considerados ao projetar um ambiente com tais características heterogêneas. Os fatores que devem ser observados são:

- **Desacoplamento**: as aplicações são capazes de ser operadas independentemente?
- **Processamento de dados em tempo real**: o quão rápido é a comunicação entre as aplicações?
- **Escalabilidade**: como o ambiente se comporta a medida que aumenta o volume de dados manipulado?
- **Manutenibilidade**: o quão difícil é manter os sistemas integrados?

- **Extensibilidade**: o quão fácil é integrar novos sistemas ao ambiente?

De acordo com Toshev, M. (TOSHEV, 2015), duas abordagens podem ser adotadas para lidar com ambientes com essas características. A primeira é utilizar compartilhamento de arquivos. Isso quer dizer que cada aplicação deve exportar os dados de interesse em um formato de arquivo específico para permitir a troca de informações entre ambientes. Contudo, essa alternativa exige que as aplicações sejam capazes de importar e exportar os arquivos compartilhados de um ambiente para outro. Além disso, esse mecanismo pode levar a problemas de desempenho, escalabilidade e comunicação em tempo real entre os ambientes. A segunda abordagem é utilizar um banco de dados compartilhado entre as aplicações. Neste caso, todos as aplicações dependem do mesmo esquema do banco de dados compartilhado. Entretanto, todos os sistemas deverão ser projetados para lidar com um esquema de dados global. Além disso, os requisitos de escalabilidade e manutenibilidade são afetados por essa abordagem, principalmente se o banco de dados utilizado for baseado em um banco de dados relacional. No mesmo sentido, Dossot, D. (DOSSOT, 2014) diz que uma maneira simples de realizar a comunicação entre ambientes diferentes é de forma síncrona. Em outras palavras, um componente de origem (cliente) interage com um outro componente de destino (servidor) de forma direta por meio de chamadas de procedimentos. Embora essa alternativa seja considerada simples e não exige muita habilidade para ser construída, ela proporciona um alto acoplamento dos componentes envolvidos. Esse fator afeta diretamente na escalabilidade e desempenho de todo o ambiente.

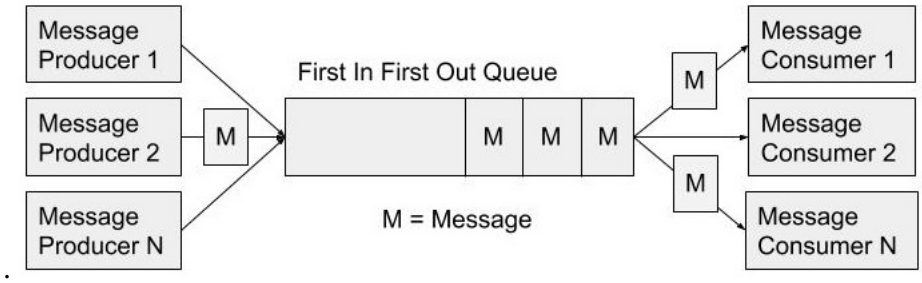
A partir desse cenário apresentado, se torna essencial algum meio de realizar a comunicação entre diferentes tipos de sistemas, respeitando os requisitos de tempo real citados na seção 2.1 e os pontos citados por Toshev, M. (TOSHEV, 2015). Portanto, deve-se permitir a comunicação entre diferentes tipos de componentes em um ambiente composto por várias fontes geradoras de dados e outras fontes que armazenam dados para tomada de decisão estratégica. Por exemplo: em um ambiente de *data warehousing* de tempo real, o DW é atualizado com dados gerados por aplicações implantadas em computadores e em *smartphones* no ambiente operacional, os quais são alimentados por meio da interação com o usuário. Além disso, esse ambiente é composto por sensores que detectam a quantidade de pessoas no ambiente. Logo, os dados coletados pelos sensores, assim como os dados produzidos pelos computadores e *smartphones*, devem ser integrados e armazenados no DW. Embora os provedores de dados sejam heterogêneas e disponibilizam seus dados em formatos diferentes, os dados de ambos os provedores devem ser enviados para um ou vários componentes de destino (nesse exemplo, para uma ou vários DW).

Para que essa comunicação ocorra respeitando os requisitos citados, destaca-se o padrão de comunicação chamado *Publish/Subscribe* (ao longo deste trabalho será utilizada a expressão modelo pub/sub para se referir a esse paradigma de comunicação). Basicamente, o modelo pub/sub permite a comunicação e o compartilhamento de dados entre ambientes heterogêneos. Além disso, permite detectar, isolar e entregar notificações de eventos ocorridos dinamicamente para destinos interessados (TARKOMA, 2012). Onica et al. (ONICA et al., 2016) definem

um modelo pub/sub como um modelo que permite disseminar informações em ambientes distribuídos, cujo os dados são produzidos por meio de diferentes provedores de dados e são enviados para diferentes destinos interessados nesses dados produzidos. Já Walkenbach, J. (WALKENBACH, 2010) diz que um modelo pub/sub é um padrão de comunicação caracterizado por um componente *sender*, uma mensagem e um componente *receiver*. Basicamente, esse padrão de comunicação permite a troca de mensagens entre um *sender* e um *receiver* com o auxilio de um servidor (*broker*).

É importante destacar que o modelo pub/sub é baseado em *Middleware* Orientado a Mensagem (MOM). Basicamente, um MOM é um método de comunicação que envia mensagens de um componente de origem (*sender*) para um componente de destino (*receiver*) (TARKOMA, 2012). O envio de mensagens ocorre com o auxilio de um componente chamado fila de mensagem (*queue*). Uma fila de mensagem é mantida no *sender* e mantém temporariamente uma mensagem antes de enviá-la ao *receiver*. Após o envio, a mensagem pode ser removida da fila ou, se por algum motivo não ocorrer a entrega da mensagem, a mesma pode ser reenviada (TARKOMA, 2012). Segundo Curry, E. (CURRY, 2005), o modelo MOM opera de forma assíncrona, isto é, permite que os componentes de origem e destino continuem trocando mensagens sem a necessidade de esperar o processamento de uma mensagem trocada anteriormente. Dessa forma, é possível que um *sender* continue processando mensagens logo após a última mensagem ter sido enviada ao *receiver*. Assim, o modelo MOM fornece as características de desacoplamento dos componentes envolvidos, além de confiabilidade, escalabilidade e disponibilidade (CURRY, 2005). Ainda segundo Curry, E. (CURRY, 2005), existem duas implementações principais do modelo MOM, que são o modelo *point-to-point* e o modelo pub/sub, sendo o último o objeto de estudo dessa seção.

Figura 3 – Arquitetura básica do modelo MOM



Fonte: (CURRY, 2005)

A Figura 3 mostra uma visão básica do modelo de comunicação MOM. O componente *Message Producer* representa os sistemas do ambiente operacional, os quais produzem eventos que são enviados em forma de mensagem e consumidos pelo componente *Message Consumer*. Por sua vez, o componente *Message Consumer* representa as entidades que recebem uma mensagem de um evento produzido por um *Message Producer*. Entre esses dois componentes, tem-se uma estrutura de dados fila, a qual é responsável por manter temporariamente as mensagens que

foram enviadas do *Message Producer* para um *Message Consumer*. Vale destacar que a Figura 3 foi redesenhada e a escrita foi mantida em inglês. Isso se deve ao fato de que a tradução para português resulta na perda de significado dos componentes.

Visto que o modelo pub/sub foi desenvolvido com base no modelo de comunicação MOM, pode-se dizer que o mesmo é utilizado para disseminar informações entre ambientes heterogêneos. Isso se deve ao fato de que o modelo pub/sub é basicamente uma implementação do modelo MOM. Contudo, o modelo pub/sub é composto por componentes que suprem determinadas lacunas do modelo MOM. Dessa forma, o modelo pub/sub fornece um mecanismo de distribuição de dados dos tipos *one-to-many* ou *many-to-many*. Isso quer dizer que um ou vários componentes de origem podem enviar mensagens para um ou vários componentes de destino, mesmo que tais componentes sejam heterogêneos e anônimos. Segundo Dobbelaere e Esmaili (DOBBELAERE; ESMAILI, 2017), os três principais princípios sobre os quais o modelo pub/sub foi desenvolvido são: 1) desacoplar os componentes de origem e destino; 2) os componentes envolvidos em interações não precisam estar ativos ao mesmo tempo; 3) a comunicação entre os componentes deve ser de forma assíncrona, ou seja, os componentes envolvidos não podem ser bloqueados enquanto outros componentes realizam tarefas. Outra característica definida como fundamental pelos autores é sua forma de roteamento de mensagens por meio de um tópico (correspondente a uma fila no método MOM). Basicamente, um tópico é responsável por manter uma mensagem enviada por um componente de origem até que um componente de destino a consuma. Além disso, um tópico representa ou refere-se a um assunto em específico, no qual é possível agrupar informações sobre um mesmo assunto (WADHWA, 2015).

Dada as definições sobre o modelo pub/sub, Tarkoma, S. (TARKOMA, 2012) diz que os principais componentes do modelo pub/sub são:

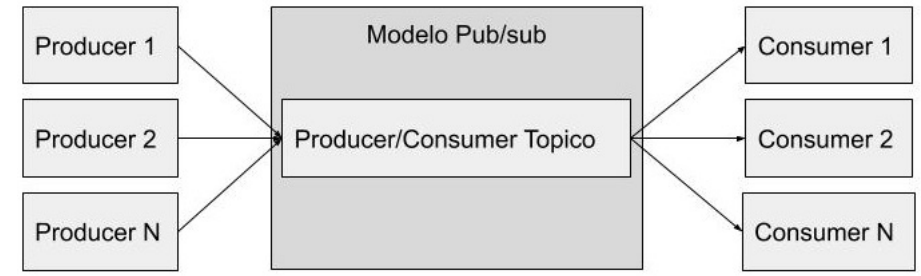
- **Evento**: um evento é considerado qualquer mudança de estado ocorrida em um objeto em um ambiente. Alguns exemplos de eventos são: um login bem sucedido de um usuário a um sistema, o registro de um novo dado em um banco de dados, a detecção de um vírus malicioso em uma rede ou a detecção de um dado gerado por meio de um sensor. Sempre que um evento ocorrer, o conteúdo desse evento é enviado em forma de mensagem para outros componentes interessados nesse evento, sem que outros componentes de origem tenham conhecimento do componente que gerou o evento.

- ***Producer*** ou ***Publisher*** (ao longo deste trabalho esse componente será chamado de *producer*): esse componente (componente de origem) é responsável por detectar e enviar uma mensagem, a qual foi gerada por meio de um evento ocorrido. Basicamente, ele detecta um evento ocorrido e envia o conteúdo do evento em forma de mensagem.

- ***Consumer*** ou ***Subscriber*** (ao longo deste trabalho esse componente será chamado de *consumer*): este componente (componente de destino) é responsável por receber uma mensagem enviada pelo *producer*. Para que essa entrega ocorra, esse componente deve

expressar interesse em algum evento produzido por um *producer* em específico. Em outras palavras, um *consumer* deve realizar uma inscrição e deixar explicito de qual *producer* deseja receber mensagens.

Figura 4 – Arquitetura básica do modelo pub/sub



Fonte: Adaptado de (CURRY, 2005)

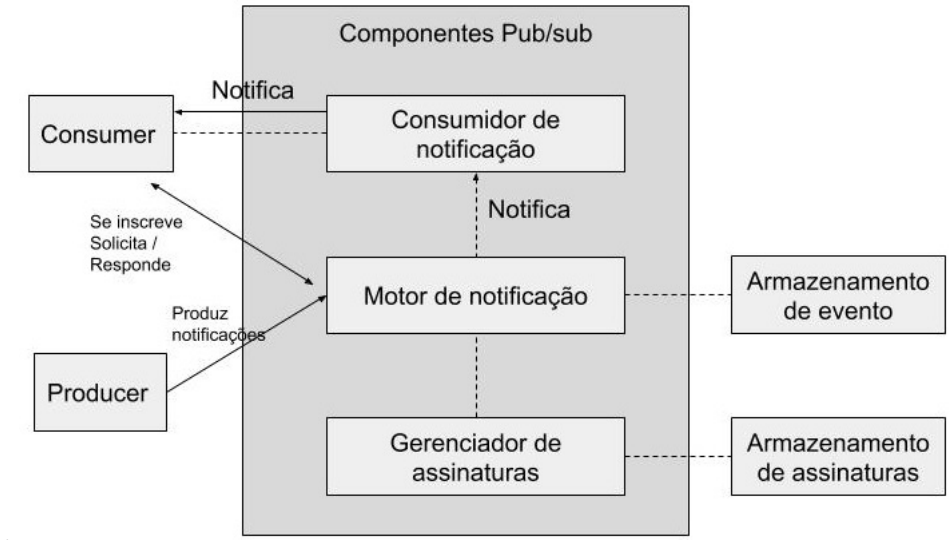
A Figura 4 representa de forma geral o funcionamento do modelo pub/sub. Basicamente, o modelo pub/sub é composto por diversos componentes *producers* e diversos componentes *consumers*. Além disso, o modelo é composto por um servidor (*broker*), o qual é responsável por gerenciar o envio e recebimento de mensagens. Além dessas características do modelo pub/sub, um *producer* deve disponibilizar uma notificação de evento aos *consumers* inscritos e que tem interesse em suas notificações. Vale destacar que, para que ocorra o recebimento de uma notificação por parte do *consumer*, o mesmo deve expressar interesse em visualizar as notificações de eventos de um *producer* específico. Para que isso ocorra, o modelo pub/sub deve ser projetado com base em uma das seguintes estratégias:

- A notificação é enviada para uma rede e os dispositivos conectados a essa rede podem visualizar a notificação. Um sistema baseado no modelo pub/sub é instalado nos dispositivos, os quais devem processar as mensagens por meio do *producer* e as enviá-las para os devidos *consumers*.
- A notificação é disponibilizada em uma rede que tem recursos de *multicast*. Assim, uma rede específica é utilizada para entregar uma notificação de evento de um *producer* para vários *consumers*.
- A notificação é enviada diretamente de um *producer* para um *consumer* que tenha expressado interesse em notificações de um dado *producer*. Neste caso, utiliza-se o protocolo de entrega de mensagens *one-to-one*.

Vale destacar que essas estratégias são consideradas centralizada e são válidas de acordo com o contexto no qual o modelo pub/sub é aplicado. As duas primeiras estratégias são dependentes da rede na qual o sistema é implementado. Além disso, essas estratégias não são adequadas

para aplicações baseadas na *Internet*. Isso se deve ao fato de as notificações podem não ser enviadas para todos os *consumers* envolvidos no ambiente. A terceira estratégia é adequada para os casos em que a quantidade de *consumers* é considerada pequena. Isso porque a entrega de notificações é diretamente de um *producer* para um *consumer*. Logo, a medida que a quantidade de *consumer* aumenta, a escalabilidade do sistema pode ser afetada.

Figura 5 – Componentes de um modelo pub/sub centralizado



Fonte: Adaptado de (TARKOMA, 2012)

A Figura 5 mostra um esquema básico de um modelo pub/sub centralizado. O componente *Producer* é responsável por observar o ambiente no qual um modelo pub/sub foi implantado. Sempre que um determinado evento ocorrer, esse componente cria e envia uma notificação de evento para o componente Motor de Notificação. Após o recebimento da notificação, o Motor de Notificação é responsável por identificar os *consumers* inscritos e que são interessados na notificação. Para isso, o Motor de Notificação mantém uma estrutura de índices de assinaturas e usa essa estrutura para identificar os *consumers* correspondentes à notificação recebida. Além disso, esse componente oferece uma *interface* adequada para o *producer* e para o *consumer* realizar suas tarefas. As tarefas realizadas pelo Motor de Notificação são auxiliadas pelo componente Gerenciador de Assinaturas. Sua responsabilidade é manter as assinaturas de *consumers* interessados em notificações de um *producer*. Dessa forma, esse componente mantém a relação entre um *consumer* e um *producer*. O componente Consumidor de Notificação é responsável por receber uma notificação de evento ocorrido do componente Motor de Notificação. Esse recebimento é feito após o Motor de Notificação identificar o *consumer* corresponde à notificação recebida. Após esse recebimento, o Consumidor de Notificação encaminha a notificação para o *consumer* apropriado. Por fim, o componente *Consumer* é responsável por receber a notificação de um evento ocorrido no *producer* para o qual foi inscrito.

Embora a representação dos componentes de um modelo pub/sub mostradas na Figura 5

sejam válidas para determinados contextos, para um ambiente de *data warehousing* de tempo real, o qual tem características distribuídas, essa representação pode não ser adequada. Isso se deve ao fato de que esse ambiente é composto por fontes de dados heterogêneas e distribuídas. Além disso, há um grande volume de dados sendo gerado e consumido pelos usuários. Esse fato faz com que a escalabilidade do ambiente seja afetada a medida que aumenta a quantidade de *producers*, a quantidade de *consumers* e consequentemente aumenta o volume de dados manipulado. Por exemplo: se houver um grande volume de notificações sendo geradas e uma grande quantidade de *consumers* para receber essas notificações, pode ocorrer problemas no desempenho de todo o ambiente, pois um *consumer* deverá aguardar a entrega das notificações para outros *consumers* até chegar sua vez.

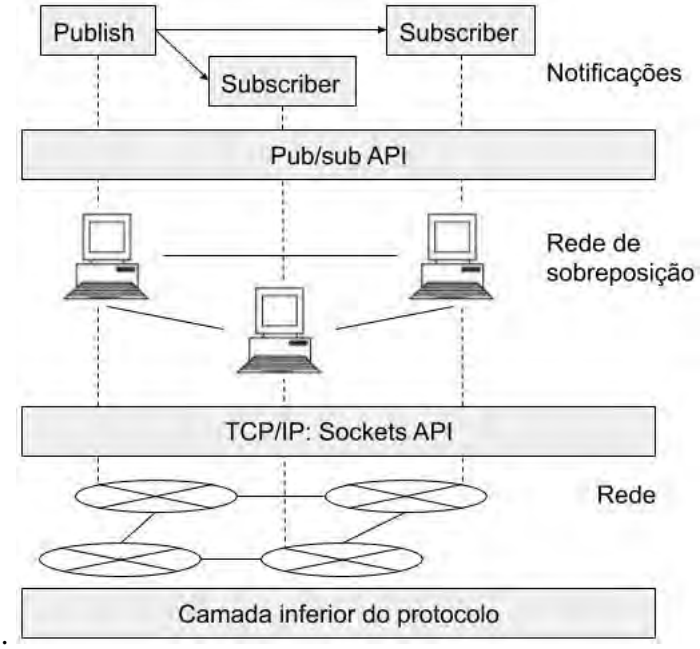
Como forma de propor soluções para resolver os problemas de escalabilidade e desempenho em ambientes com características distribuídas, foram propostas outras duas estratégias para o projeto de um modelo pub/sub. As estratégias são:

- A notificação é inicialmente enviada do *producer* para um servidor (*broker*). Então, o servidor disponibiliza a notificação de evento para os *consumers* que expressaram interesse em receber as notificações de eventos do servidor.
- A notificação é primeiramente enviada do *producer* para uma rede de servidores. Então, o *consumer* recebe a notificação de evento a partir do servidor de interesse.

Essas estratégias são consideradas distribuídas e são compostas por um servidor ou rede de servidores, os quais mantém as notificações de eventos enviadas pelo *producer*. Isso quer dizer que há o roteamento das notificações, fazendo com que possa haver vários *consumers* interessados em notificações de eventos de vários *producers*. Como consequência, ocorre um ganho desempenho e consequentemente ganha-se em escalabilidade de todo o ambiente. Vale destacar que para este trabalho de Doutorado serão consideradas apenas essas duas estratégias. Ao aplicar essas estratégias, deve-se construir uma nova camada de rede de aplicação sobre a camada de rede já existente, a qual é composta por servidores que mantém as notificações de eventos, assim como uma indicação do *producer* que enviou a notificação. Além disso, essa nova camada de rede de aplicação fornece novos recursos para serem aplicados, tais como armazenamento distribuído, recursos de tolerância a falhas e recursos de recuperação em casos de falha na rede.

A Figura 6 mostra um exemplo da nova camada de rede implantada sobre a camada de rede já existente. Nota-se que entre a camada de rede TCP/IP e o ambiente no qual gera as notificações de eventos, existe uma nova camada de rede composta por servidores, também conhecido na literatura do modelo pub/sub como *brokers*. Esses servidores são responsáveis por manter as notificações de eventos enviadas por um *producer*. Além disso, essa camada é responsável por enviar tais notificações de eventos para os respectivos *consumers* interessados.

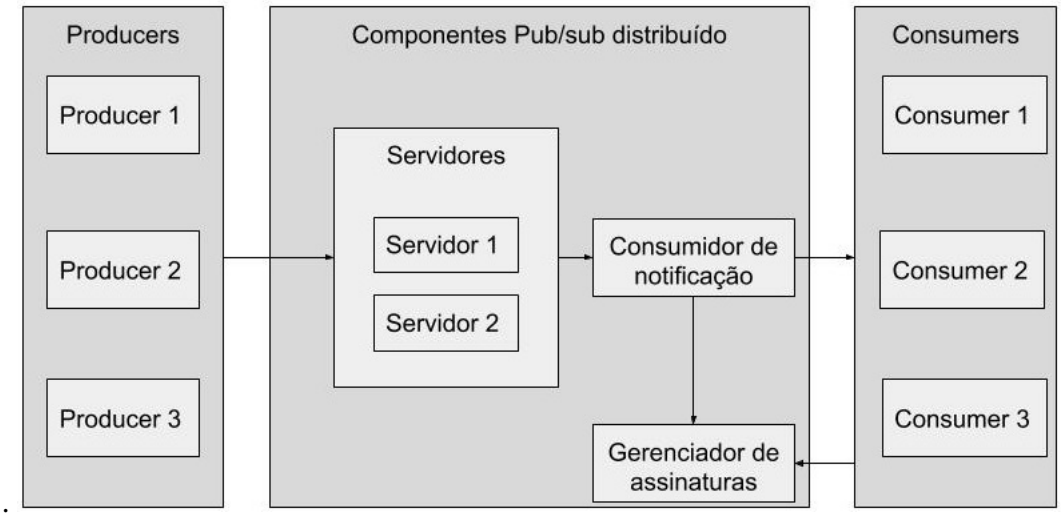
Figura 6 – Representação da nova camada de aplicação



Fonte: Adaptado de (TARKOMA, 2012)

Para isso, essa camada é composta por novos recursos, os quais permitem a comunicação entre as camadas do modelo pub/sub.

Figura 7 – Componentes de um modelo pub/sub distribuído



Fonte: Inspirado em (ONICA et al., 2016)

Já a Figura 7 mostra os componentes básicos de um modelo pub/sub aplicado nessa nova camada de aplicação distribuída. Neste caso, nota-se que podem existir vários servidores que recebem as notificações de eventos, os quais estão implantados na nova camada de rede de aplicação. A partir dessa abordagem, pode-se ter várias entidades que representam o *producer*, assim como podem haver várias entidades que representam o *consumer*. Dessa forma, a nova

camada de rede é responsável por gerenciar as notificações de eventos enviadas por um *producer* e quais são os *consumers* interessados em receber tais notificações. Um *consumer* expressa o interesse em receber notificações de eventos de um determinado *producer*. Para isso, o componente Gerenciador de Assinaturas recebe de um *consumer* uma assinatura de interesse. Sempre que uma nova notificação de evento for enviada por um *producer*, o componente Consumidor de Notificação verifica qual *consumer* tem interesse na notificação recebida. Essa verificação é feita com o auxilio do Gerenciador de Assinaturas. Dessa forma, o Consumidor de Notificação é capaz de enviar a notificação de evento recebida de um *producer* para o *consumer* interessado.

Como pode-se notar ainda na Figura 7, sempre que um *producer* envia uma notificação de evento, todos os servidores recebem essa notificação. Isso faz com que o modelo pub/sub favoreça a replicabilidade dos dados, o desempenho e escalabilidade de todo o ambiente. Contudo, podem não haver *consumers* interessados nessa tal notificação enviada. Para evitar o envio de notificações desnecessárias, a nova camada de rede fornece o recurso de filtro. Basicamente, esse recurso permite que os servidores recebam apenas notificações pelas quais tem pelo menos um *consumer* interessado. Com a utilização de filtros, é possível reduzir o volume de notificações de eventos enviadas para os servidores. Com consequência, pode ocorrer uma melhora de desempenho devido o volume menor de notificações de eventos a serem processadas.

É importante destacar que ao utilizar o modelo pub/sub em um ambiente com características distribuídas, pode haver perda de mensagens trocadas, assim como pode haver duplicação de mensagens entregues a um *consumer*. Isso ocorre devido a falhas que podem ocorrer durante a entrega de uma mensagem, por exemplo: no momento em que uma mensagem for entregue, pode ocorrer falha de rede ou o servidor não estar disponível. Logo, o modelo pub/sub permite que essa mensagem possa ser reenviada ou cancelada a operação. Em casos em que ocorrer o reenvio da mensagem, pode ocorrer a duplicidade da mensagem no *consumer*. Já em casos em que a operação for cancelada, pode ocorrer a perda de mensagens. Dessa forma, o modelo pub/sub foi projetado para fornecer recursos baseados em *Quality-of-service Guarantees* (QoS). De modo geral, essa garantia de qualidade de serviço tem a ver com a confiabilidade do modelo pub/sub e consiste em definir como ocorrerá a entrega das mensagens enviadas de um *producer* para um *consumer*. Assim, pode-se destacar três principais modos de garantia entrega de mensagens (DOBBELAERE; ESMAILI, 2017; TARKOMA, 2012):

- ***At most once***: neste modo, uma tentativa de entrega de uma mensagem ocorrerá no máximo uma vez. Se alguma falha ocorrer a mensagem pode ser perdida. Portanto, esse modo não garante que a mensagem será entregue ao *consumer*. Embora esse modo de entrega possa ocorrer perda de dados, é o modo que permite o melhor desempenho. Além disso, este modo garante a não duplicação de dados.

- ***At least once***: neste modo, uma tentativa de entrega de uma mensagem ocorrerá pelo menos uma vez. Se alguma falha ocorrer, outras tentativas de entrega da mensagem serão

feitas até que a mensagem seja entregue. Esse modo garante a não perda de mensagens enviadas. Contudo, pode-se haver duplicação de mensagens entregue ao *consumer*.

- ***Exactly once***: neste modo, uma tentativa de entrega de uma mensagem ocorrerá exatamente uma vez. A mensagem será entregue mesmo se um servidor estiver indisponível ou alguma falha de rede ocorrer. Esse modo garante que nenhuma mensagem será perdida e que não ocorrerá duplicação de mensagens no *consumer*.

A partir dessa garantia de entrega de mensagens, percebe-se que dependendo do contexto no qual o modelo pub/sub for aplicado, é possível fazer com que ocorra o melhor caso na entrega de mensagens, isto é, sem perda e sem duplicação de mensagens. Nesse mesmo sentido, como já discutido na seção 2.4, um ambiente reativo reage a algum evento ocorrido no ambiente operacional e executa alguma ação automaticamente. Segundo Tarkoma, S. (TARKOMA, 2012), a construção de um ambiente reativo por meio do modelo pub/sub é uma alternativa para implementar um ambiente no qual deve reagir a vários tipos de eventos ocorridos. Além disso, para o contexto discutido na seção 2.4, o modelo pub/sub pode ser implementado em conjunto à abordagem chamada *event loop*. Essa abordagem basicamente permite que o ambiente reaja imediatamente após um *producer* produzir alguma notificação de evento, ao mesmo tempo que são identificados os *consumers* interessados nessa notificação. Dessa forma, pode-se ter um ambiente reativo em que ocorra o envio e recebimento de mensagens por meio do modelo pub/sub e com a garantia da entrega de tais mensagens.

Em resumo, pode-se observar que o modelo pub/sub é um modelo de comunicação e troca de mensagens em ambientes distribuídos e heterogêneos, por meio do qual torna-se possível a troca de mensagens entre vários componentes de origem e vários componentes de destino. Além disso, essa troca de mensagens ocorre por meio de notificações, as quais podem ser enviadas e recebidas de componentes heterogêneos, distribuídos, autônomos e anônimos.

2.6 Considerações finais

Este capítulo descreveu os conceitos essenciais para o entendimento de um ambiente de tempo real. Esse ambiente se contrapõe a ambientes convencionais, pois os dados são gerados em tempo real que, por sua vez, são dados gerados baseados em um evento ocorrido, respeitando os requisitos temporais. Além disso, com a evolução na forma de geração de dados, diversos tipos de sensores foram incorporados nesses ambientes. Dessa forma, o requisito de baixo tempo de resposta passou a ser cada vez mais restrito, isto é, os dados passaram a ser produzidos cada vez mais rápido, assim como as tomadas de decisões estratégicas passaram a ser realizadas em tempo real.

Esses conceitos foram importantes para melhor entender o conteúdo dos dois próximos capítulos, nos quais retratarão o processo ETL e ambientes de *data warehousing* em tempo real.

Basicamente, o processo ETL em ambiente de *data warehousing* em tempo real é um ambiente que executa o processo ETL em tempo real e, simultaneamente e na medida do possível, com características do ambiente de *data warehousing* convencional e do ambiente de tempo real. Assim, o Capítulo 3 tratará dos conceitos, características e particularidades do processo ETL e o Capítulo 4 tratará dos conceitos, características e particularidades de ambientes de *data warehousing* em tempo real.

Capítulo 3
PROCESSO ETL

O processo ETL surgiu como forma de integrar dados distribuídos em fontes de dados heterogêneas e armazená-los em uma fonte de dados homogênea, de modo a permitir consultas analíticas para tomada de decisão estratégica. Cada etapa do processo ETL é responsável por realizar atividades específicas e customizadas, cujo o objetivo é disponibilizar os dados integrados e consistentes.

Sendo assim, este capítulo descreve os principais conceitos do processo ETL, desde o contexto para o qual ele foi criado até exemplos de aplicação. Na seção 3.1 são mostrados os principais conceitos e as características do processo ETL e na seção 3.2 é descrita cada uma das fases do processo ETL, assim como as suas especificidades.

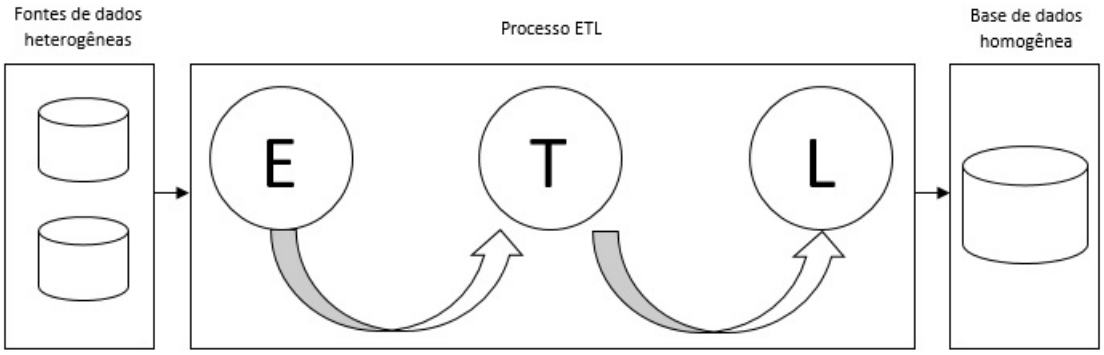
3.1 Processo ETL

Basicamente, o processo ETL consiste em capturar os dados armazenados em fontes de dados do ambiente operacional, tratar estes dados e posteriormente permitir o armazenamento deles em um DW. Sua principal tarefa é extrair dados de interesse a partir de fontes de dados do ambiente operacional, transformar esses dados de modo a disponibiliza-los de forma integrada e consistente e carrega-los para uma fonte de dados de destino para servir como fonte de informação para tomada de decisão estratégica. Esse processo é visto como um elemento inseparável em um ambiente de *data warehousing*, pois é por meio dele que é possível que os dados sejam carregados em um DW. Dessa forma, o processo ETL pode favorecer ou impedir a utilização de um DW. Estima-se que cerca de 70% a 80% do custo para construir um ambiente de *data warehousing* seja destinada a implementação e manutenção do processo ETL. Esse fato ocorre, pois o processo ETL deve superar diversos desafios até realizar todas as suas tarefas.

Do ponto de vista prático, o processo ETL não é simplesmente integrar dados de fontes de dados heterogêneas. O resultado final desse processo é adicionar um valor semântico a um dado, de modo que represente uma informação válida ao usuário final para tomada de decisão estratégica. Esse aspecto implica diretamente na qualidade dos dados manipulados. Para isso, o processo ETL deve garantir a consistência dos dados armazenados no DW. Além disso, o

processo ETL deve permitir ao usuário auditar o fluxo dos dados, desde a sua extração do ambiente operacional até o seu armazenamento no DW.

Figura 8 – Esquema básico do processo ETL



Fonte: próprio autor

A Figura 8 mostra um exemplo de um fluxo básico que representa o processo ETL. Conceitualmente, nota-se que o processo ETL é, no fundo, um grande *workflow* orientado a dados. Isso quer dizer que qualquer dado que trafegue por esse fluxo de dados passará por todas as fases estabelecidas. O início do processo ocorre na extração de dados de fontes de dados heterogêneas. Após a extração, o dado passa pela fase de transformação e finaliza com a fase de carregamento para uma base de dados homogênea.

Tradicionalmente, o processo ETL deve enfrentar cinco principais desafios: 1) **grande volume de dados**: o ambiente operacional gera dados constantemente e em grande volume, o que implica diretamente nas técnicas, nos métodos e nas ferramentas adotadas no processo ETL; 2) **qualidade dos dados**: os dados do ambiente operacional podem estar distribuídos em diferentes fontes de dados e em diferentes formatos, mas representando um mesmo objeto ou entidade do mundo real. Esse fato implica na necessidade de se realizar uma limpeza desses dados de modo a representa-los de forma integrada e consistente; 3) **evolução das fontes de dados**: tanto o DW quanto as fontes do ambiente operacional sofrem constantes mudanças devido ao grande volume de dados manipulado, o que implica em constantes manutenções por parte da equipe técnica; 4) **atualização**: os dados de interesse para tomada de decisão estratégica são continuamente inseridos no ambiente operacional. Logo, é necessário que essas mudanças ocorridas nesses itens de dados reflitam diretamente no DW; 5) **desempenho**: o grande volume de dados manipulado por um ambiente de *data warehousing* implica diretamente no desempenho do processo ETL, desde a extração de dados até as consultas realizadas pelos usuários (VASSILIADIS; SIMITSIS, 2009).

Com base nesse contexto, percebe-se que não é possível construir um ambiente de *data warehousing* sem aplicar o processo ETL. Isso se deve ao fato de que as características do ambiente serem equivalentes aos recursos fornecidos pelo processo. Se um conjunto de dados precisa ser analisado, mas este não for manipulado com base nos conceitos de ETL, certamente

não poderá ter consultas analíticas e tomadas de decisões corretas sobre esses dados. Esse aspecto implica em dizer que podem haver diversas ferramentas aplicadas para o carregamento de dados no DW, mas, independente das ferramentas, o conceito de ETL deve ser aplicado.

O processo ETL envolve, além de desafios, de restrições, de conceitos, de técnicas e de ferramentas, grupos de pessoas com diferentes interesses. Esses grupos podem ser projetistas de bancos de dados, gerentes de negócios, usuários envolvidos em tomadas de decisões e usuários responsáveis pela manutenção do ambiente. Dessa forma, a complexidade no desenvolvimento do processo se torna cada vez maior a medida que vários desses elementos são integrados ao projeto. Visto que o processo ETL envolve diversos fatores técnicos e humanos, além de toda a complexidade descrita, vários trabalhos foram propostos na literatura em busca de oferecer não somente arquiteturas, mas também métodos e técnicas focadas no processo ETL. Essas pesquisas focam também em realizar estudos que favorecem uma melhor organização do fluxo de atividades que envolve a construção do processo ETL, desde as fases mais abstratas até a fase de implementação.

3.2 Fases do processo ETL

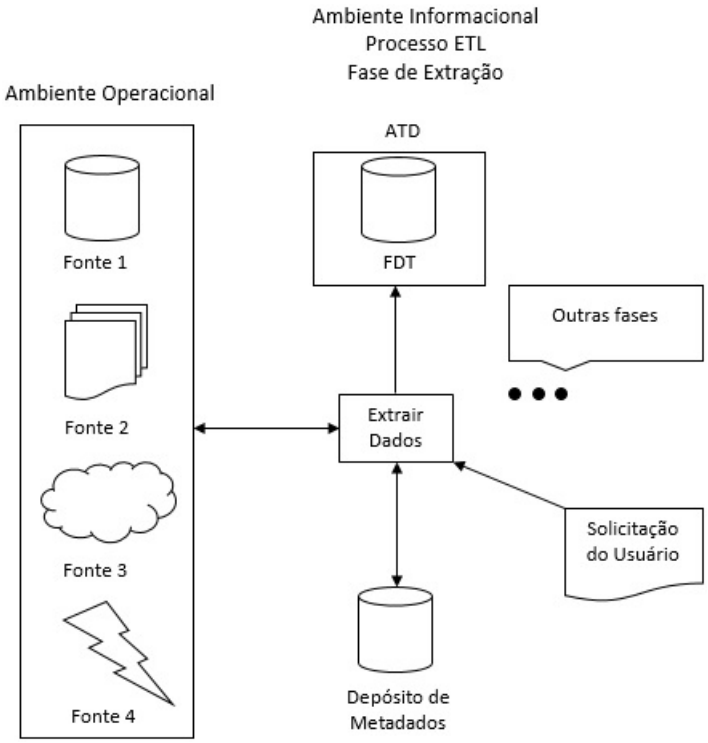
O processo ETL é composto por três fases genéricas, que são extração, transformação e carregamento. Entretanto, há estudos que apontam que, em cada fase, podem haver outras etapas que auxiliam no processo. A seguir, são descritas detalhadamente cada fase do processo ETL, além de outras fases específicas identificadas na literatura.

3.2.1 Extração

A fase de extração de dados é a primeira fase do processo ETL. Ela consiste em conectar-se às fontes de dados do ambiente operacional e extrair os dados de interesse do usuário. Esses dados são extraídos sem nenhum tratamento, isto é, mesmo se um dado estiver em branco, inválido ou apresentar qualquer erro, o mesmo será extraído, desde que seja de interesse. Somente os dados de interesse são extraídos e, portanto, tais dados não compõem uma cópia integral de todos os dados do ambiente operacional. Embora as fontes de dados de origem possam ser mantidas localmente ou externamente, a fase de extração deve garantir o acesso a essas fontes de dados. Isso quer dizer que o processo ETL deve conhecer previamente quais as fontes de dados que contém os dados de interesse. Além disso, as fontes de dados podem disponibilizar os dados em diferentes formatos. Mesmo assim, a fase de extração deve assegurar que os dados, ao serem extraídos, serão mantidos em um formato homogêneo para servir a outras fases do processo ETL (SABRY; ALI, 2014).

A Figura 9 mostra como ocorre a fase de extração de dados. O ambiente operacional é composto por diversas fontes de dados, cada qual com seu formato, localização, formas de acesso e restrições. No ambiente informacional, a fase de extração é composta por uma área

Figura 9 – Exemplo da fase de extração de dados



Fonte: próprio autor

de dados denominada **área de tratamento de dados (ATD)**, a qual tem a responsabilidade de manter as transformações ocorridas nos dados nas etapas de transformação e limpeza, assim como o fluxo de dados desde o ambiente operacional até o DW. Além disso, a ATD é composta por uma fonte de dados temporária e homogênea chamada **Fonte de Dados Temporária (FDT)**, cujo o objetivo é manter temporariamente os dados extraídos do ambiente operacional para servir como dados de entrada para as fases posteriores do processo ETL. Ainda em relação ao ambiente informacional, o **depósito de metadados** é uma fonte de dados que mantém todas as configurações de metadados estruturais e de conexão necessárias para a executar o processo. Tais metadados de conexão podem ser: nome de usuário e senha para conexão, número de porta, nome do servidor e outras informações relevantes para estabelecer conexão com as fontes de dados. Os metadados estruturais podem ser: esquema das fontes de dados de origem, regras de extração, regras de mapeamento entre os diversos esquemas existentes, regras de transformação de dados, regras de controle de acesso, perfis de usuários e demais regras estruturais referentes ao processo.

Inicialmente, o usuário solicita dados de negócios, os quais estão distribuídos em fontes de dados do ambiente operacional. Após a solicitação, o extrator de dados identifica quais as fontes de dados envolvidas na consulta solicitada. Após essa identificação, o extrator faz uma consulta no depósito de metadados para buscar os metadados de conexão das fontes de dados identificadas. Além disso, são identificados os metadados estruturais necessários para acessar as

fontes de dados, assim como as regras de mapeamento de esquemas entre as fontes de dados e a FDT. Após esse processo, os dados de interesse são extraídos de suas fontes de dados, mapeados para um esquema homogêneo e armazenados na FDT.

A fase de extração é considerada, segundo alguns pesquisadores, a fase mais custosa de ser realizada do processo ETL. Esse fato decorre pelos seguintes motivos: 1) a fase de extração de dados depende de agentes externos, tais como sistemas operacionais diferentes, sistemas de gerenciamento de banco de dados diferentes, protocolos, *hardwares* e outros agentes; 2) os dados utilizados na fase de extração estão armazenados em fontes de dados externas. Essas fontes de dados, no pior caso, podem conter esquemas diferentes, formas de acesso diferentes, restrições diferentes e disponibilizar seus dados em diferentes formatos. Isso quer dizer que a fase de extração de dados deve lidar também com a heterogeneidade dos dados; 3) conceitualmente, a fase de extração deveria ser tão simples quanto uma consulta SQL. Entretanto, as fontes de dados do ambiente operacional podem sofrer constantes sobrecargas. Essas fontes de dados servem a um grupo de usuários interessados em realizar operações de escrita e a fase de extração, interessada em realizar consultas para recuperar dados modificados; 4) o resultado produzido pela fase de extração servirá como dados de entrada para as outras fases do processo ETL. Esse resultado implicará positivamente ou negativamente nas demais fases do processo; 5) de acordo com a escala do projeto, essa tarefa se torna ainda mais complexa. Quanto maior o projeto, mais fontes de dados envolvidas, mais pessoas envolvidas, mais ferramentas e técnicas aplicadas e um maior volume de dados manipulado.

Tabela 1 – Exemplo de dados que foram extraídos na fase de extração de dados.

Dados fictícios				
Nome	**Endereço**	**Idade**	**CPF**	**Fonte**
João da Silva	Rua XZX	60	123.456	Fonte A
João da silva	Rua XXX	60	123.456	Fonte B
João Silva	Rua XYX	350	123456	Fonte C
José Souza e Silva	Rua YYY	50	444.555	Fonte A
José Souza	Rua YYY	800	444.555	Fonte C

A Tabela 1 mostra um exemplo de dados que foram extraídos de diferentes fontes de dados. Ao analisar a tabela, alguns aspectos podem ser destacados: os dados foram extraídos de três fontes de dados diferentes, sendo Fonte A, Fonte B e Fonte C. Além disso, esses dados são correspondentes a duas pessoas, João da Silva e José Souza e Silva. Entretanto, alguns erros podem ser notados: a pessoa João da Silva tem o nome diferente em cada fonte de dados. Além disso, existem três endereços diferentes, sua idade, na Fonte C, está informada 350 e seu CPF está sem formatação. Com relação ao José Souza e Silva, seu nome também está diferente em cada fonte de dados e a idade, na Fonte C, está informada 800.

Essas divergências nos valores dos dados ocorrem devido a erros dos usuários do ambiente operacional, por não haver um padrão de inserção de dados nos diferentes sistemas que

alimentam as fontes de dados. Além disso, cada um desses sistemas tem sua própria forma de inserir dados e ainda podem disponibilizar os dados em diferentes formatos e meios. Com isso, ao ser executada a fase de extração, os dados podem ser extraídos sem nenhum tratamento para validar sua consistência.

Vale a pena destacar que o fato dos dados serem mantidos em diferentes fontes de dados e, por sua vez, disponibilizar os dados em diferentes formatos, deve haver uma forma para se comunicar com as fontes de dados por meio de um mecanismo padrão de acesso a dados. Além disso, a fase de extração é a primeira fase do processo ETL e, com isso, essa atenção deve ser tomada nessa fase. Dessa forma, Kimball et al. (KIMBALL et al., 2004) sugerem que uma forma estabelecer um padrão de comunicação é por meio de *Open Database Connectivity* (ODBC). As conexões ODBC surgiram para estabelecer uma forma padrão de acessar uma fonte de dados. Dessa forma, permite que uma aplicação se torne portátil, isto é, independe da fonte de dados na qual os dados estão armazenados. O uso de ODBC, no entanto, pode ser substituído por métodos de acesso mais modernos, tais como *Application Programming Interface* (API) customizadas para diferentes plataformas e linguagens de programação.

Além das características citadas, a fase de extração de dados deve garantir os dados em um formato homogêneo para que todo o restante do processo possa acessá-lo. Além do mais, deve garantir que todos os dados de interesse, selecionados no ambiente operacional, trafegue por todo o fluxo de dados do processo ETL até ser armazenado no DW. É fácil imaginar que realizar uma carga completa dos dados do ambiente operacional para o DW resolva o problema de atualização de dados. Entretanto, isso implica em outro problema ainda maior: o desempenho da aplicação. Realizar uma consulta e/ou carga de dados em uma fonte de dados cujo o volume de dados é grande, implica em perda de desempenho e consequentemente na disponibilidade do DW. Portanto, uma solução é executar o processo de atualização incremental, isto é, atualizar apenas os dados que sofreram alterações do ambiente operacional para o DW.

A atualização incremental auxilia o processo de manutenção do DW, pois permite identificar e encaminhar um volume menor de dados. Esse fato interfere diretamente e positivamente no desempenho e na disponibilidade do DW. Basicamente, a atualização incremental realiza uma filtragem dos dados do ambiente operacional e disponibiliza apenas os dados que sofreram alterações e que serão utilizados no restante do processo de integração de dados. Essa tarefa requer a habilidade dos desenvolvedores em criar mecanismos que facilitam a extração dos dados no ambiente operacional. Esses mecanismos devem favorecer a atualização incremental dos dados, uma vez que os dados de interesse para atualização são apenas os dados alterados. Nesse cenário, o mecanismo *Change Data Capture* (CDC) surgiu como um elemento chave para capturar as atualizações dos dados em tempo real. Esse mecanismo é utilizado com o objetivo de identificar, capturar e disponibilizar as mudanças ocorridas nos dados no ambiente operacional (BOKADE et al., 2013). Essa característica faz do mecanismo CDC um método de extração *on-line*, isto é, as alterações ocorridas no ambiente operacional são disponibilizadas no mesmo

momento em que sofrem alterações. Esses dados são disponibilizados por meio de uma estrutura predefinida, como por exemplo tabelas e *logs*. Além disso, podem ser disponibilizados por recursos da própria fonte de dados, como por exemplo gatilhos.

A captura de dados alterados no ambiente operacional é crucial para o bom funcionamento do DW (KIMBALL et al., 2004). Além disso, a aplicação do mecanismo CDC permite melhorar a eficiência e o desempenho do processo ETL, sobretudo na fase de extração de dados. Isso se deve ao fato de que permite fornecer um menor volume de dados a ser extraído e consequentemente utilizado no restante do processo ETL. Tecnicamente, o mecanismo é um processo utilizado para realizar a sincronização entre o ambiente operacional e o *data warehousing* em tempo real (BOKADE et al., 2013).

A sincronização fornecida pela aplicação do mecanismo CDC pode ser em dois modos (JAIN et al., 2012): 1) **síncrono**: as mudanças são capturadas imediatamente. Esse processo é realizado no ambiente operacional por meio de *triggers* nas tabelas envolvidas; 2) **assíncrono**: as mudanças são encaminhadas para um arquivo externo após as tabelas do ambiente operacional efetivar uma transação. Esses arquivos podem ser *log*, XML ou outro tipo de arquivo acessível aos processos interessados na importação e/ou exportação dos dados.

A partir dessa necessidade de aplicar o mecanismo CDC, Kimball et al. (KIMBALL et al., 2004) sugerem técnicas de captura de dados que podem ser aplicadas a fim de permitir a atualização incremental dos dados no DW. Essas técnicas são:

- **Colunas auditáveis**

  Essa técnica consiste em criar uma coluna que armazena a data e a hora que um determinado registro foi inserido ou alterado. Essa coluna é atualizada automaticamente por meio do uso de *trigger* assim que os comandos de inserção ou alteração são executados no banco de dados. Entretanto, por questões de desempenho, essa coluna pode ser atualizada automaticamente pela aplicação. Caso essa coluna seja atualizada pela aplicação, é importante não deixar isso sob a responsabilidade direta do usuário, pois podem haver problemas com campos nulos. A sugestão dada pelos autores é inserir a data e a hora diretamente no código da aplicação. Assim, evita-se possíveis erros de inserção de dados e pode-se definir critérios para inserir o valor esperado, por exemplo: pode-se inserir a data e hora atual do servidor do banco de dados. Segundo os autores, pode-se criar uma tabela separa, relacionada com a tabela de fatos e dimensões, para inserir a data e a hora.

- **Eliminação de dados**

  Essa técnica consiste em manter uma cópia dos dados previamente extraídos na FDT (como já mostrado na Figura 9). Com base nesses dados, é feita uma comparação entre os dados que já estão na FDT e os dados que foram copiados. Dessa forma, apenas os dados diferentes são armazenados no DW. Percebe-se que essa técnica percorre todos os registros

armazenados na FDT e, dessa forma, a probabilidade de erros é menor. Entretanto, pode se tornar uma técnica mais custosa de ser implementada, visto que deve ser feita uma cópia completa dos dados do ambiente operacional em uma dada frequência para que a comparação possa ser executada.

- **Carregamento inicial e incremental**

  Essa técnica permite manter duas tabelas: uma para os dados carregamentos anteriormente (tabela anterior) e outra para dados atuais (tabela atual, também chamada de arquivo Delta). Inicialmente, os dados são carregamentos por completo para a tabela atual. Se nenhuma mudança for detectada, os dados são transformados e carregados para o DW. Quando o processo termina, a tabela atual é renomeada para tabela anterior e a tabela atual é recriada para receber novos dados. A próxima vez que o processo for executado, a tabela atual é alimentada e é feita uma comparação entre a tabela atual e a tabela anterior. Os registros diferentes entende-se que são as mudanças ocorridas. Dessa forma, esses registros diferentes são encaminhados para as próximas fases do processo ETL.

Entretanto, com a evolução das pesquisas sobre CDC, outras técnicas foram propostas com o objetivo de fornecer novos meios de captura de dados para melhorar as técnicas já existentes. As outras técnicas propostas na literatura são:

- **Arquivos de *log* das transações**

  Grande parte dos bancos de dados utilizam arquivos de *log* para registrar as alterações ocorridas no banco de dados durante uma transação. Uma alternativa para capturar as mudanças ocorridas é analisar o arquivo de *log* e obter os dados alterados do *log*. O aspecto positivo dessa técnica é que não afeta as fontes de dados do ambiente operacional. Entretanto, um ponto negativo é que, se houver um grande volume de dados a ser analisado, o processo pode se tornar custoso e demorado.

- ***Triggers***

  Uma das técnicas mais aplicados é a criação de *triggers* nas tabelas do ambiente operacional que devem ser gerenciadas para controle de alterações. As *triggers* podem rastrear as mudanças ocorridas nos dados em conjunto com um atributo do tipo *timestamp*, a fim de identificar o exato momento em que uma alteração ocorreu. A cada comando de inserção, alteração ou remoção executado na tabela, a *trigger* executa alguma função para disponibilizar os dados alterados e atualizar o atributo com a data e a hora exata da alteração.

- **Replicação**

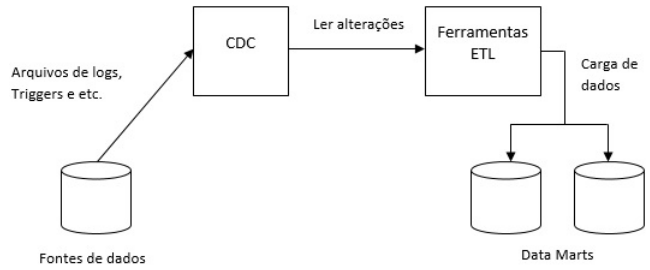
  Essa técnica permite distribuir uma cópia dos dados para diversos servidores, mantendo os dados sincronizados entre as fontes de dados e o DW. De acordo com Bokade et al.

(BOKADE et al., 2013), esse método não é essencialmente uma técnica CDC, pois exige um esforço adicional para satisfazer os requisitos da técnica CDC. Além disso, não é uma boa solução quando as fontes de dados são heterogêneas, pois essa técnica é fornecida nativamente pelas fontes de dados. Dessa forma, se torna um problema replicar dados cujo o esquema é diferente.

- **Snapshot**

  Essa técnica consiste em capturar os dados de uma fonte de dados em um exato momento. Esses dados são mantidos temporariamente em uma memória *cache* e podem ser utilizados como dados de entrada para outros processos, por exemplo: durante o processo de extração de dados, pode-se executar um *snapshot* de uma tabela específica. Os dados capturados podem ser armazenados diretamente no DW ou podem ser servir como fonte de dados para consultas.

Figura 10 – Arquitetura da técnica CDC



Fonte: (JAIN et al., 2012)

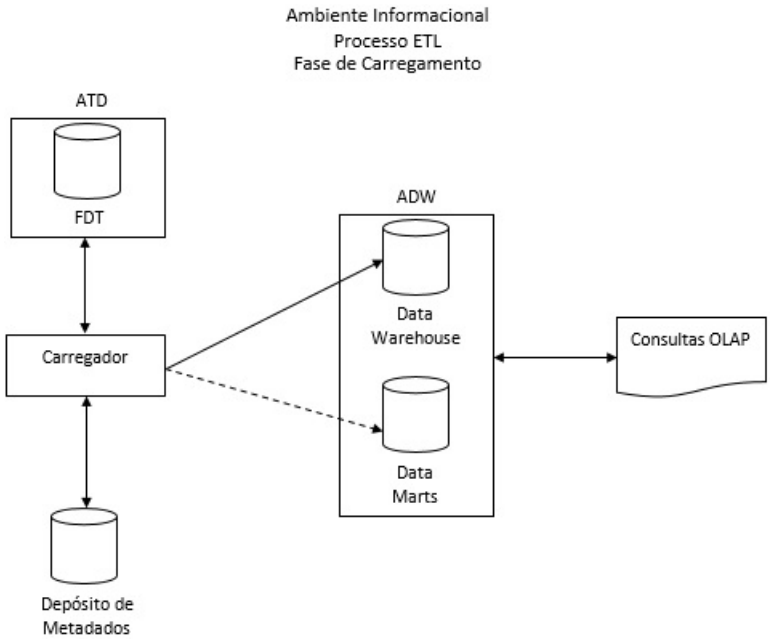
A Figura 10 mostra um exemplo de como ocorre a aplicação da técnica CDC. A figura mostra uma representação das fontes de dados do ambiente operacional, da técnica CDC, das ferramentas ETL e do *data mart* (nesse caso, considera-se também o DW). A técnica CDC captura as alterações ocorridas nas fontes de dados por meio de arquivos de *logs*, *triggers* ou outra técnica e disponibiliza os dados para ser utilizado pelo processo ETL. Dessa forma, o processo ETL tem acesso apenas aos dados alterados no ambiente operacional e que devem ser atualizados no DW. Contudo, vale a pena destacar que, tanto essas técnicas quanto as técnicas apresentadas por Kimball et al. (KIMBALL et al., 2004), podem ser aplicadas em conjunto. Dessa forma, é necessário conhecer o ambiente o ambiente de *data warehousing*, sobretudo o processo ETL, para definir quais as melhores técnicas a serem aplicadas.

3.2.2 Carregamento

Após as fases de extração e transformação, os dados devem ser armazenados definitivamente em uma fonte de dados de destino para servir de insumo para análises de negócios. Normalmente, essa fonte de dados de destino é o DW, mas pode ser qualquer outra fonte de

dados de interesse da organização. Independentemente da fonte de dados utilizada, essa deve permitir a criação de um esquema homogêneo e oferecer mecanismos de processamento de consulta sobre os dados armazenados. Além disso, essa fonte de dados deve permitir a criação de índices, restrições de integridade, agregações e demais elementos de banco de dados. Dessa forma, torna-se possível a criação do esquema estrela para o projeto lógico do DW. Normalmente, com base nessas características, utiliza-se bancos de dados cuja estrutura é relacional, em que se trata de uma abordagem amplamente fundamentada, aplicada e estudada. Além disso, os sistemas gerenciadores de bancos de dados relacionais oferecem mecanismos avançados de consulta, evitando assim a construção de outras ferramentas para processamento de consulta.

Figura 11 – Esquema em alto nível da fase de carregamento de dados para o DW.



Fonte: Próprio autor

A Figura 11 mostra como ocorre a fase de carregamento de dados. Inicialmente, o objeto **carregador** acessa a **FDT** para buscar os dados que foram transformados e limpos na fase anterior. Além disso, o carregador acessa o **depósito de metadados** para buscar os metadados de conexão do DW. Após esse processo, o carregador se comunica com a área de DW **(ADW)**, na qual são mantidos o DW e, dependendo do caso, os *data marts* da organização.

É importante destacar que essa fase é a última fase do processo ETL e é por meio dela que os dados são disponibilizados para as consultas analíticas. Entretanto, essa fase tem suas limitações e complexidades, tais como: 1) elementos como índices, relacionamentos e chaves devem ser desabilitados para que o processo de carregamento de dados possa ser executado. Além disso, após o término do processo, esses elementos devem ser recriados. Esse aspecto torna o processo custoso quando existe um grande volume de dados sendo manipulado; 2) os dados podem ser disponibilizados em *data marts*, o que implica em vários processos de carregamento, um para cada *data mart* existente; 3) além de *data marts*, os dados podem ser estar particionados

em várias tabelas fato, como visto na seção 4.4. Isso implica em novas técnicas de direcionar cada dado para suas respectivas partições. Sendo assim, cada obstáculo colocado no processo de carregamento, torna o tempo mais demorado para executar todo o processo.

Alguns mecanismos para carregamento de dados no DW são (SABRY; ALI, 2014; KAKISH; KRAFT, 2012):

- Tabelas externas: esse mecanismo permite que os dados externos sejam utilizados como uma tabela virtual, nas quais podem ser aplicadas consultas e junções antes de carregar para o DW.
- *Oracle Call Interface* (OCI) e *Application Programming Interface* (API): são métodos aplicados quando as transformações são feitas fora do ambiente do DW.
- Importar/Exportar: esse método é usado se não há nenhuma transformação complexa sobre os dados. É considerado o método mais simples de se carregar os dados para o DW.

3.3 Considerações finais

Este capítulo apresentou os principais conceitos do processo ETL, as características específicas de cada fase, sua modelagem conceitual, lógica e física e suas particularidades. Dado o contexto apresentado, ficou claro que o processo ETL é uma peça fundamental na construção de um ambiente de *data warehousing*. Isso se deve ao fato de que é por meio dele que os dados operacionais são carregados no DW. Além disso, como foi mostrado, o custo maior na construção de um ambiente de *data warehousing* é o processo ETL. Isso quer dizer tal processo é essencial para o sucesso ou fracasso de tal ambiente.

Além dos conceitos, foi possível notar que existem diversas formas para abordar o processo ETL. Grande parte dos trabalhos focam em uma abordagem teórica, cujo o foco é voltado para estudos. Contudo, Kimball et al. (KIMBALL et al., 2004) propõem uma abordagem prática, na qual é possível ter uma outra visão do processo ETL quando aplicado em um ambiente real. Além disso, foi possível identificar variações para o processo ETL, as quais tentam fornecer novas formas de lidar com os dados extraídos das fontes de dados operacionais, mas mantendo o conceito de extração, transformação e carga de dados.

No Capítulo 4 serão tratados os principais conceitos e características de um ambiente de *data warehousing* em tempo real.

Capítulo 4

AMBIENTE DE DATA WAREHOUSING EM TEMPO REAL

Como já refletido ao longo deste trabalho, tradicionalmente, o DW foi projetado para ser atualizado pelo processo ETL em momentos específicos. Contudo, após o surgimento de novos domínios de negócios, passaram a surgir novas demandas por dados atualizados em tempo real. Dessa forma, tanto o DW quanto o processo ETL passaram por ajustes para ser possível atender essas demandas de dados em tempo real.

Nesse sentido, este capítulo retrata as características e ajustes feitos no DW e no processo ETL. A seção 4.1 retrata as características do ambiente de *data warehousing* de tempo real. A seção 4.2 mostra os requisitos a serem respeitados ao se pensar em um ambiente de *data warehousing* de tempo real. A seção 4.3 descreve como ocorre o processo ETL e suas respectivas fases em um ambiente de *data warehousing* de tempo real. A seção 4.4 mostra as mudanças ocorridas para permitir que a tabela de fatos do DW seja preparada para receber dados em tempo real. Por fim, a seção 4.5 finaliza o capítulo com as considerações finais.

4.1 DW em um ambiente de data warehousing de tempo real

A evolução e o aumento da competitividade entre diversos setores de negócios impulsionaram a forma como as organizações lidam com o negócio. Com isso, um dos aspectos que sofreu transformação foi a demanda por dados analíticos que possam favorecer uma tomada de decisão mais rápida e eficiente. Com o surgimento do DW, essas tomadas de decisões passaram a ser tomadas com mais qualidade e rapidez devido as características já citadas neste trabalho. Além disso, o DW passou a ser atualizado por meio do processo ETL numa frequência específica, isto é, os dados de interesse passaram a ser armazenados para o DW diariamente, semanalmente ou durante um período ocioso da organização. Dessa forma, os usuários sempre tem dados atualizados para as tomadas de decisões.

Contudo, essa característica de atualização do DW se tornou inadequada para determinados domínios de negócios que necessitam de dados atualizados em tempo real (SABTU et al.,

2017; PHANIKANTH; SUDARSAN, 2017), tais como sistemas médicos (JAIN et al., 2012) e sistemas que lidam com sensores (MESITI et al., 2016). Além disso, aplicações no domínio de *smart farming* (RYU et al., 2015; KULATUNGA et al., 2017; ZAMORA-IZQUIERDO et al., 2019; FUENTES et al., 2020) e controle de trafego de rodovias (FIGUEIRAS et al., 2017) geralmente não geram dados em grande volume, mas precisam garantir que os dados serão armazenados no DW em tempo real para garantir o processo de tomada de decisão estratégica (SABTU et al., 2017; PHANIKANTH; SUDARSAN, 2017).

A partir do surgimento de novos meio de se produzir dados e a demanda cada vez maior por dados analíticos atualizados, novas soluções foram desenvolvidas para tornar possível executar o processo ETL em uma frequência cada vez maior e consequentemente atualizar o DW. Dessa forma, surgiu o conceito de ambientes de *data warehousing* em tempo real, o qual segue o mesmo paradigma de um ambiente de *data warehousing* tradicional, isto é, os conceitos e a finalidade para o qual é aplicado são os mesmos, em que as tomadas de decisões devem acontecer por meio das consultas OLAP a partir de dados analíticos armazenados no DW. Além disso, o processo ETL deve ser executado para manter o DW sempre atualizado. Entretanto, dois fatores principais devem ser alterados: 1) os dados devem ser enviados dos provedores de dados para o DW no menor tempo de resposta possível; 2) o DW deve ser remodelado de modo a permitir que inserções sejam feitas concomitantemente a submissão e processamento de consultas OLAP.

A Tabela 2 apresenta uma comparação com relação aos objetivos de um ambiente de *data warehousing* convencional e em tempo real. O ambiente de *data warehousing* em tempo real foi proposto para resolver o principal problema de um ambiente de *data warehousing* tradicional: o tempo de atualização dos dados, o que resultou em novos recursos agregados à adaptação de sua estrutura original.

De acordo com Bouaziz et al.(BOUAZIZ et al., 2017), existem dois argumentos que justificam a utilização de um ambiente de *data warehousing* em tempo real: 1) a atualização de dados do DW convencional ocorre numa frequência diária, semanal ou até mensal. Entretanto, alguns domínios de negócios necessitam de dados atualizados com maior frequência; 2) com os dados permanecendo estáticos por um longo período de tempo, os relatórios podem ser emitidos contendo informações desatualizadas e consequentemente decisões erradas podem ser tomadas (JAIN et al., 2012).

A maior dificuldade ao propor um ambiente de *data warehousing*, tanto convencional quanto em tempo real, é construir o processo ETL (LANGSETH, 2004). Segundo Kimball e Caserta (KIMBALL et al., 2004), cerca de 70% de todo o custo de desenvolvimento de um ambiente de *data warehousing* é gasto com o processo ETL. Para realizar a atualização do DW, é necessário que o ambiente de *data warehousing* esteja fora de uso. Esse fato ocorre pois se torna custoso inserir novos dados, reconstruir chaves, índices e demais elementos de banco de dados com uma fonte de dados em execução. Além disso, o ambiente operacional pode sofrer uma sobrecarga nas fontes de dados, pois já servem a outro grupo de usuários que realizam operações

Tabela 2 – Comparação entre um *data warehousing* convencional e um *data warehousing* em tempo real

Ambiente de *data warehousing* convencional	**Ambiente de *data warehousing* em tempo real**
Somente para decisões estratégicas	Decisões estratégicas e táticas
Dados atualizados periodicamente	Dados atualizados em tempo real
Dados disponíveis diariamente, semanalmente e mensalmente são aceitáveis	Somente dados disponíveis em minutos são aceitáveis
Relatórios restritivos usados para verificar processos e padrões existentes	Relatórios *ad hoc* flexíveis para descobrir novos relacionamentos e hipóteses
Resultados difíceis de medir	Resultados medidos com operações
Poucos usuários acessando	Vários usuários acessando

Fonte: Adaptado de (BOUAZIZ et al., 2017)

de escrita sobre os dados. Essa inatividade e sobrecarga pode se transformar em restrições ao implantar esse processo em determinadas organizações, pois os usuários podem não aceitar tais limitações em seus sistemas (SANTOS et al., 2011). Com isso, o ambiente de *data warehousing* em tempo real deve lidar com dois principais desafios: 1) realizar atualização continua dos dados sem a inatividade do DW, principalmente no que diz respeito a inserção de dados; 2) a atualização do DW deve ocorrer concomitantemente com consultas OLAP. Além disso, o ambiente de *data warehousing* em tempo real deve se preocupar com o impacto e o desempenho causado pela consultas OLAP sendo executadas conjuntamente com a atualização dos dados (SANTOS et al., 2011).

No ambiente de *data warehousing* em tempo real, o termo tempo real pode ser interpretado como sendo uma atualização que ocorre em milésimos de segundos, o que pode até acontecer, mas, de forma geral, não acontece. Como discutido nas seções 4.3.1 e 4.4, de acordo com a técnica adotada, pode-se alcançar uma maior frequência de atualização com a técnica mais simples para atualização de dados. Essa técnica consiste em diminuir a frequência de atualização do DW, por exemplo: de uma vez por dia para uma vez a cada hora ou em alguns minutos. Em casos mais críticos, pode-se utilizar a técnica mais clássica e mais adequada, distribuindo o processamento de dados de tempo real em uma fonte de dados externa e os dados históricos no DW (VAISMAN; ZIMÁNYI, 2014). Em ambos os casos, o processo não é literalmente em tempo real, mas sim "imediatamente após". Esse termo é interpretado como: assim que um novo dado for produzido no ambiente operacional, o mesmo deve ser enviado imediatamente para o DW. Dessa forma, para a grande maioria dos ambientes de *data warehousing* em tempo real, o conceito de *soft real-time* é considerado. Contudo, um determinado *deadline* deve ser respeitado,

e o não cumprimento desse tempo não causa nenhum dano catastrófico. Além disso, deve-se respeitar as limitações de *hardware*, protocolos de rede, limitações de taxa de transferência de dados ou qualquer outro tipo de restrição que possa afetar o tempo gasto para armazenar os dados no DW.

4.2 Requisitos para um ambiente de data warehousing em tempo real

Como pode-se notar ao longo desse capítulo, o ambiente de *data warehousing* requer novas estratégias para lidar com os dados gerados em ambientes de tempo real. Além disso, como já discutido na seção 2.1, o ambiente de tempo real tem características específicas que devem ser respeitadas para que a aplicação possa ser executada. Contudo, um ambiente de *data warehousing* em tempo real deve respeitar tais características e, como consequência, proporcionar seus próprios requisitos. Isso ocorre pois, além de ser aplicado em ambiente de tempo real, o DW tem suas próprias especificidades, restrições e objetivos. Dessa forma, para ser executado em um ambiente de tempo real, o DW deve obedecer os seguintes requisitos:

- **Integração de dados contínua**

  O processo ETL deve ser executado continuamente. Isso quer dizer que não se espera o tempo de atualização da abordagem convencional, em que a atualização ocorre diariamente ou num momento oportuno. As atualizações do DW devem ocorrer imediatamente após alterações ocorrerem no ambiente operacional.

- **Realização conjunta da análise de dados e da atualização de dados**

  Mesmo durante o processo de atualização de dados, deve ser permitido executar consultas OLAP para tomadas de decisões. Isso quer dizer que um grupo de usuários deve realizar operações de geração de dados, ao mesmo tempo que outro grupo de usuários realizam operações de consulta sobre dados atualizados.

- **Disponibilidade**

  Na abordagem convencional, a disponibilidade do DW é afetada sempre que o processo ETL é executado para atualizar os dados. Em alguns momentos, não é possível ter disponibilidade, nem do ambiente operacional nem do DW. Entretanto, alguns pesquisadores definem a disponibilidade do DW em tempo real como 24/7, ou seja, vinte e quatro horas por dia, sete dias por semana. Isso quer dizer que o DW em tempo real deve sempre estar disponível, independente da operação realizada.

- **Escalabilidade horizontal**

Na escalabilidade horizontal, o processamento dos dados são distribuídos em diferentes servidores. Dessa forma, evita-se a perda de disponibilidade e ganha-se em desempenho de consultas sobre os dados. Em um ambiente de *data warehousing* em tempo real, a escalabilidade horizontal deve ser fortemente considerada, visto que a disponibilidade também é um requisito fundamental. Se a escalabilidade vertical for adotada, os dados seriam armazenados em apenas um servidor centralizado. Dessa forma, se esse servidor se tornar inativo por algum motivo, o seu acesso será imediatamente afetado. Como consequência, a disponibilidade e desempenho também será diretamente afetada.

4.3 Processo ETL em um ambiente de data warehousing em tempo real

Como já foi explicado no Capítulo 3, o processo ETL foi projetado para ser aplicado em um ambiente no qual a exigência de atualização é diferente dos requisitos de tempo real. Normalmente, essa atualização ocorre em um momento em que não há transações em andamento no ambiente operacional e nem consultas OLAP sendo executadas. Contudo, ao ser aplicado em um ambiente de tempo real, o processo ETL deve sofrer ajustes em ambas as fases para conseguir superar os desafios encontrados em um ambiente de tempo real.

4.3.1 Extração de dados em tempo real

Um dos principais desafios do processo ETL é identificar os dados de interesse que devem ser extraídos do ambiente operacional. Segundo Muddasir e Raghuveer (MUDDASIR; RAGHUVEER, 2017), Sabry e Ali (SABRY; ALI, 2014), Valencio et al. (VALENCIO et al., 2014) e outros pesquisadores, a fase de extração de dados é a fase mais importante e mais desafiadora de todo o processo ETL. Isso se deve a vários motivos: 1) a fase de extração é a primeira fase do processo. Logo, todas as outras fases depende da fase de extração de dados; 2) o resultado da fase de extração, tanto custo operacional quanto os dados extraídos, serão aplicados em todas as outras fases. Se um dado for identificado e extraído erroneamente, todo o restante do processo também será afetado; 3) a fase de extração deve lidar com a heterogeneidade dos dados, isto é, as fontes de dados operacionais podem ser heterogêneas e com diferentes formas de acesso.

Embora a fase de extração do processo ETL seja objeto de estudo de diversos pesquisadores durante anos, nenhuma técnica CDC diferente das apresentadas no Capítulo 3 foi criada especificamente para lidar com a extração de dados em tempo real. Ao contrário, diversas estratégias e técnicas foram propostas em busca de utilizar as técnicas CDC já existentes conjuntamente com ajustes feitos para permitir a extração de dados em tempo real (mostrado no Capítulo 7). A partir desses ajustes, diversos problemas surgiram ao lidar com a extração de dados em tempo real, tais como integração de fontes de dados heterogêneas, sobrecarga nas fontes de dados, perda

de desempenho da aplicação (WIBOWO, 2015). Tais problemas foram solucionados por meio de novos ajustes feitos e, dessa forma, a extração de dados passou a ser realizada em tempo quase real.

Independentemente das técnicas utilizadas para extração de dados, pode-se perceber que o processo ETL é acionado de forma intrusiva. Em outras palavras, a técnica, a estratégia ou a arquitetura proposta deve ser capaz de acessar as fontes de dados operacionais e extrair os dados de interesse. Além disso, mesmo utilizando as técnicas CDC de *log* ou gatilho, não é uma garantia de que o processo ETL não acessará o ambiente operacional e nem qual base de dados do ambiente de *data warehousing* receberá o dado extraído por meio da técnica adotada.

4.3.2 Transformação de dados em tempo real

Após a fase de extração de dados, os dados devem ser transformados em uma única representação consistente, antes de serem carregados para o DW. Contudo, para realizar a fase de transformação, é necessária a utilização da *Data Staging Area*. Segundo Wibowo, A. (WIBOWO, 2015), mesmo em tempo real, é necessário uma base de dados para que os dados transformados sejam mantidos temporariamente até que todo o processo finalize. Esse aspecto faz com que a fase de transformação seja custosa ao ser executada em tempo real.

Uma alternativa para solucionar esse problema é utilizar a abordagem ELT proposta por Freudenreich et al. (FREUDENREICH et al., 2013) (mostrada no Capítulo 7). Contudo, com base no conhecimento adquirido até o momento, nenhuma técnica para transformação de dados foi proposta especificamente para o processo ETL em tempo real.

4.3.3 Carregamento de dados em tempo real

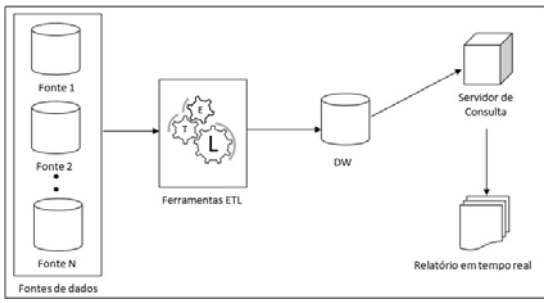
Após a fase de transformação, os dados devem ser carregados para o DW. Diferentemente da fase de transformação, a fase de carregamento de dados sofreu alterações de modo a permitir um melhor desempenho em ambientes de *data warehousing* em tempo real. Isso se deve ao fato de que a fase de carregamento de dados deve se preocupar em disponibilizar os dados atualizados e permitir que consultas sejam executadas ao mesmo tempo. Dessa forma, Langseth (LANGSETH, 2004) propôs algumas técnicas que permitem que tais características possam ser respeitadas. Vale a pena destacar que, embora essas técnicas foram propostas em 2004, elas são referenciadas e aplicadas atualmente. As técnicas propostas são:

- **Near Real-Time ETL**

  Essa técnica não é considerada verdadeiramente em tempo real. Entretanto, é apontada como a mais fácil e mais barata de ser implementada, e permite aproximar o DW dos requisitos de tempo real. De acordo com Langseth (LANGSETH, 2004), nem todas as aplicações se justificam utilizar uma solução verdadeiramente em tempo real, pois isso

implica custos que, as vezes, não se justificam. Uma solução é diminuir a frequência de atualização do DW, por exemplo: se o DW é atualizado semanalmente, pode-se reduzir a frequência para diariamente ou duas vezes por dia.

Figura 12 – Exemplo da técnica *Near Real-Time* ETL



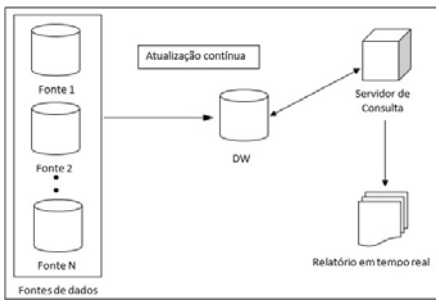
Fonte: Próprio autor

A Figura 12 mostra a arquitetura da técnica *Near Real-Time* ETL. Essa figura representa as fontes de dados do ambiente operacional, as ferramentas ETL, o DW, o servidor de consultas OLAP e uma representação do relatório final do usuário. Basicamente, a técnica consiste em manter o fluxo do processo ETL convencional, mas com uma frequência de atualização maior. De acordo com alguns pesquisadores, alguns domínios de negócios ficam ociosos durante o dia devido a diversos fatores, por exemplo: pausa para reuniões, pausa para descanso ou outros fatos que implicam na inatividade temporária da organização. Com isso, durante esse período ocioso, o processo ETL pode ser executado.

- **Direct Trickle Feed**

  Essa técnica é considerada verdadeiramente em tempo real, e consiste em carregar o DW diretamente com os dados do ambiente operacional. Neste caso, pode-se inserir os dados diretamente nas tabelas do DW ou em tabelas separadas (seção 4.4). Um aspecto positivo dessa técnica é que permite, de fato, atualizações do DW em tempo real. Entretanto, alguns pontos negativos devem ser considerados: 1) escalabilidade e 2) atualização de dados executada no mesmo momento de uma consulta OLAP pode afetar o desempenho.

Figura 13 – Exemplo da técnica *Direct Trickle Feed*



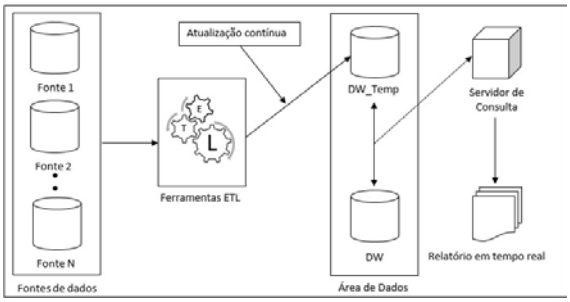
Fonte: Próprio autor

A Figura 13 mostra a arquitetura da técnica *Direct Trickle Feed*. Essa figura tem a mesma representação da figura anterior, exceto a representação das ferramentas ETL que foi eliminada. Neste caso, qualquer alteração ocorrida no ambiente operacional será encaminhada para o DW. A diferença dessa técnica para a anterior é que aqui a atualização é contínua e na *Near Real-Time* ETL a atualização ocorre em períodos definidos. Além disso, a atualização contínua é executada pelo processo ETL, mas o mesmo foi retirado da figura para fins ilustrativos.

- **Trickle And Flip**

  De acordo com Langseth (LANGSETH, 2004), essa técnica evita problemas de executar consultas OLAP ao mesmo tempo que o DW sofre atualizações. Ao invés de carregar os dados de tempo real para o DW, eles são carregados continuamente para uma fonte de dados temporária, cujo esquema é igual ao esquema do DW. Dependendo do caso (seção 4.4), a fonte de dados temporária pode conter uma cópia completa dos dados do DW ou apenas os dados do dia corrente.

Figura 14 – Exemplo da técnica *Trickle And Flip*



Fonte: Próprio autor

A Figura 14 mostra um exemplo da técnica *Trickle And Flip*. Essa figura representa as fontes de dados do ambiente operacional, as ferramentas ETL, o DW, uma representação da fonte de dados temporária, cujo esquema é igual ao esquema do DW, o servidor de consultas OLAP e uma representação do relatório final do usuário. Percebe-se pela figura que tanto a fonte de dados temporária quanto o DW são mantidos em um mesmo servidor, mas em fontes de dados separadas. Além disso, o processo ETL carrega os dados de tempo real diretamente para a fonte de dados temporária. Neste caso, apenas o ambiente operacional executa operações de escrita nessa fonte de dados, ao passo que operações de leitura podem ser executadas, tanto nessa fonte de dados quanto no DW. Quando solicitada uma consulta, pode-se obter os dados da fonte de dados temporária (dados de tempo real), dos dados armazenados no DW (dados históricos) ou dados de ambas as fontes de dados.

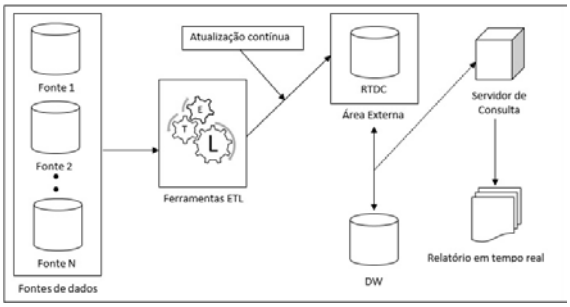
Após um período pré-estabelecido, os dados da fonte de dados temporária são integrados com os dados históricos do DW. Após esse processo de integração, os dados da fonte de

dados temporária são excluídos e mantidos apenas os dados no DW. Dessa forma, o DW é atualizado com os dados do ambiente operacional.

- **External Real-time Data Cache**

  Em alguns casos, há a necessidade de se manipular um grande volume de dados numa latência menor ou obter um melhor desempenho no processamento de consultas. Esse fato faz com que o servidor responsável por executar as tarefas sobre os dados históricos e os dados de tempo real se torne incapaz de oferecer um desempenho adequado para o processo ETL e o processamento de consultas OLAP. Neste caso, uma alternativa é armazenar os dados de tempo real em uma fonte de dados externa ao DW, como por exemplo em memória. Essa fonte de dados deve ser dedicada apenas a carregar, armazenar e processar os dados de tempo real. Dessa forma, retira a sobrecarga do servidor do DW em processar os dados de tempo real e executar as consultas OLAP. Além disso, esse repositório de dados deve conter o mesmo esquema do DW (seção 4.4), pois ambas as fontes de dados serão sincronizadas no fim.

Figura 15 – Exemplo da técnica *External Real-time Data Cache*



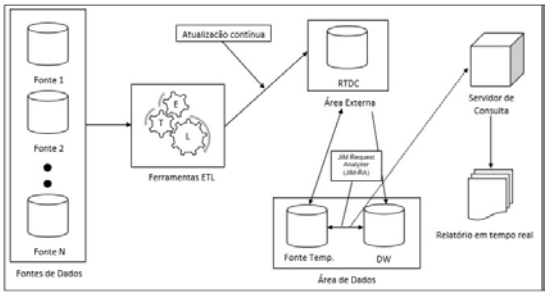
Fonte: Próprio autor

A Figura 15 mostra um exemplo da técnica *External Real-time Data Cache* (RTDC). Essa figura representa as fontes de dados do ambiente operacional, as ferramentas ETL, uma representação de uma fonte de dados externa cujo o esquema é igual do DW, o DW, o servidor de consultas OLAP e uma representação do relatório final do usuário. Percebe-se pela figura que as fontes de dados são mantidas em servidores separados, mas mantendo o mesmo esquema. Além disso, o processo ETL carrega os dados de tempo real diretamente para a fonte de dados externa para que os mesmos possam ser processados em um servidor exclusivo para tal. Assim como na técnica *Trickle And Flip*, apenas o ambiente operacional executa operações de escrita nessa fonte de dados externa, ao passo que operações de leitura podem ser executadas, tanto nessa fonte de dados externa quanto no DW. Quando solicitada uma consulta, pode-se obter os dados da fonte de dados externa (dados de tempo real), dos dados armazenados no DW (dados históricos) ou dados de ambas as fontes de dados. Após um determinado período, os dados da fonte de dados externa e do DW são integrados e consequentemente o DW é atualizado.

- **Just-in-time Information com External Real-time Data Cache**

  Em casos mais extremos, um ambiente pode ter algumas das seguintes características: 1) exigir dados em tempo real (e não em tempo quase real); 2) envolver um grande volume de transações por segundo; 3) envolver um grande volume de usuários; 4) analisar dados históricos e dados de tempo real simultaneamente. Nesse cenário, Langseth (LANGSETH, 2004) diz que é recomendado que o ambiente permita pelo menos escalabilidade e disponibilidade alta. Para isso, o autor sugere a utilização de uma abordagem híbrida. Essa abordagem envolve a técnica *External Real-time Data Cache* com a técnica chamada *Just-in-time Information*.

Figura 16 – Exemplo da técnica *Just-in-time Information* com *External Real-time Data Cache*



Fonte: Próprio autor

A Figura 16 mostra um exemplo da técnica *Just-in-time Information* com *External Real-time Data Cache*. Essa figura representa as fontes de dados do ambiente operacional, as ferramentas ETL, uma representação de uma fonte de dados externa cujo esquema é igual ao esquema do DW, o DW, o servidor de consultas OLAP e uma representação do relatório final do usuário. Essa abordagem segue a seguinte lógica: os dados de tempo real são mantidos em uma fonte de dados externa. Sempre que necessário, os dados são obtidos da fonte de dados externa por meio de um *snapshot*. Após isso, esses dados são armazenados temporariamente em tabelas no DW. Uma vez armazenados junto ao DW, esses dados podem ser utilizados como fonte de consulta. Dessa forma, elimina-se a necessidade de executar consultas diretamente na fonte de dados externa, favorecendo assim o desempenho e disponibilidade da aplicação.

4.4 Modelagem das tabelas do data warehouse em tempo real

A modelagem das tabelas do DW exerce um papel fundamental para a qualidade das tomadas de decisões. Isso se deve ao fato de que os dados devem responder exatamente à perguntas que os usuários desejam resposta imediata. Dessa forma, a modelagem das tabelas do DW em ambientes de tempo real também deve ser considerada durante sua construção para potencializar o desempenho e as características esperadas. A ideia base é fornecer técnicas de modelagem para serem aplicadas em determinados contextos de negócios. Algumas técnicas

permitem armazenar dados de tempo real juntamente com os dados históricos, armazena-los em locais diferentes ou até mesmo não criar uma distinção entre ambos. Contudo, não existe uma técnica certa ou errada, tudo depende do contexto e do ambiente no qual o DW será implementado. Langseth (LANGSETH, 2004) sugere algumas técnicas a serem adotadas durante a modelagem do DW:

- **Carregamento direto à tabela de fatos**

  Quando as técnicas *Direct Trickle Feed* ou *Trickle and Flip* são aplicadas e os dados de tempo real são armazenados juntamente com os dados históricos, nenhuma modelagem especial deve ser considerada. Isso se deve ao fato de que as ferramentas de consulta são incapazes de diferenciar os dados em tempo real dos dados históricos estando mantidos em um mesmo local.

- **Particionar a tabela de fatos**

  Embora o DW seja uma fonte de dados única, a tabela de fatos pode ser particionada a fim de melhorar o desempenho no processamento de consultas OLAP, diminuir o volume de dados sobre uma partição ou outro recurso desejado. As ferramentas de consulta devem capazes de acessar qualquer partição criada e recuperar os dados desejados. Entretanto, para facilitar o trabalho, um conjunto de tabelas em que estão os dados históricos pode apontar para outro conjunto de tabelas em que estão os dados de tempo real, e dessa forma, podem ser aplicadas operações complexas de consulta, por exemplo, *drill-across*, para retornar o resultado da consulta desejada. Do ponto de vista das ferramentas de consulta e do administrador dos dados, essa abordagem é mais complexa do que a anterior. Além disso, essa abordagem requer o entendimento de onde estão os dados armazenados e como acessa-los.

- **Dados de tempo real integrados por meio de visões**

  Outra alternativa para modelar as tabelas do DW é armazenar os dados de tempo real separados dos dados históricos, mas no mesmo servidor do DW. Por meio das visões que são fornecidas pelos bancos de dados, os dados de tempo real e os dados históricos são integrados e apresentados na consulta. Essa abordagem é similar à primeira abordagem apresentada, entretanto essa abordagem permite armazenar os dados de tempo real em tabelas separadas, o que proporciona inserções e alterações mais fáceis de administrar. Do ponto de vista das ferramentas de consulta, essa abordagem é simples e é tratada como sendo uma única tabela.

  Vale a pena destacar que, neste caso, haverá uma perda natural de desempenho, pois a visão será processada no momento em que for acionada. Dessa forma, é possível que, mesmo em um ambiente de tempo real, ocorra uma latência maior no processamento de consultas OLAP do que utilizando outras técnicas.

- **Dados de tempo real armazenados externamente**

  Quando essa abordagem é utilizada, nenhuma modelagem especial é requerida, pois os dados de tempo real armazenados em fontes de dados externa são modelados iguais ao DW. Essa abordagem tem a vantagem de eliminar possíveis problemas de desempenho, pois permite utilizar um armazenamento externo ao DW para carregar, armazenar e processar os dados de tempo real. As ferramentas de consulta devem acessar os dados em ambas as fontes de dados para apresentar o resultado da consulta desejado.

Vale a pena destacar que essas técnicas de modelagem não sobrepõem as técnicas para carregamento de dados apresentadas na seção 4.3.3. Essas técnicas são para mostrar que existem diferentes formas de se projetar as tabelas do DW. Além disso, se utilizada com a solução de carregamento de dados adequada, pode otimizar ainda mais a solução desenvolvida.

4.5 Considerações finais

O objetivo principal deste capítulo foi descrever técnicas, ferramentas e abordagens para que, em conjunto, possam otimizar o desempenho, integração contínua e consistência de dados em um ambiente de *data warehousing* em tempo real. A partir das técnicas apresentadas, é importante destacar que elas não se sobrepõem. Cada técnica representa um aspecto diferente e desejável no ambiente de *data warehousing*. Dessa forma, pode-se agregar as técnicas apresentadas neste capítulo juntamente com as técnicas CDC apresentadas no Capítulo 3. Assim, pode-se obter uma frequência de atualização e desempenho melhor do que às técnicas convencionais.

Ao final deste capítulo, percebe-se que as técnicas para executar o processo ETL em ambientes de *data warehousing* em tempo real são as mesmas aplicadas em um ambiente de *data warehousing* convencional. Contudo, em cada fase, foram propostas novas soluções para se obter um melhor desempenho e escalabilidade de acordo com o ambiente. Entretanto, ao serem aplicados em um ambiente de *data warehousing* em tempo real, essas soluções não se comportam como o esperado, e novas soluções devem ser propostas para permitir a execução do processo ETL e ao mesmo tempo preservar os requisitos de ambientes de tempo real.

Dessa forma, no Capítulo 5, será apresentada a arquitetura chamada de Imã de Dados, a qual foi desenvolvida para permitir executar o processo ETL em ambientes de *data warehousing* em tempo real de forma não intrusiva e reativa. A partir do Imã de Dados, é possível executar o processo ETL nesses ambientes e manter os requisitos de baixo de tempo de resposta, escalabilidade e disponibilidade.

Capítulo 5
IMÃ DE DADOS

A partir da contextualização e do problema apresentado no Capítulo 1, dos conceitos que cercam todo o ambiente de *data warehousing* apresentados no Capítulo 2, do conceito e fases do processo ETL descritos no Capítulo 3 e dos conceitos e técnicas para desenvolver um ambiente de *data warehousing* em tempo real apresentados no Capítulo 4, neste capítulo é descrita a arquitetura do Imã de Dados, por meio da qual é possível executar o processo ETL em tempo real em ambientes de *data warehousing*.

Na seção 5.1 é mostrado o contexto no qual a proposta da arquitetura é aplicada. Na seção 5.2 é mostrado o objetivo definido para o desenvolvimento do Imã de Dados. Na seção 5.3 é detalhada a arquitetura do Imã de Dados. Nesse momento, serão detalhados todos os componentes envolvidos na arquitetura do Imã de Dados.

5.1 Contextualização

A execução do processo ETL para atualizar o DW comumente ocorre em uma frequência predefinida. Durante essa execução, tanto o ambiente operacional quanto o DW devem estar inativos para não haver conflitos de atualização e consulta de dados, além de não sobrecarregar as fontes de dados operacionais. Contudo, com o surgimento de novas possibilidades de geração de dados, sobretudo a geração de dados em tempo real por meio de sensores ou outros meios de geração de um fluxo contínuo de dados, surgiu a necessidade do DW ser atualizado com dados em tempo real. Entretanto, as soluções tradicionais para executar o processo ETL não são adequadas para ambientes de tempo real, pois não são capazes de respeitar os requisitos de tempo real.

Por meio da revisão sistemática apresentada no Apêndice A e dos trabalhos relacionados apresentados no Capítulo 7 foi possível identificar os trabalhos que tratam do processo ETL em tempo real em ambientes de *data warehousing*. Entretanto, após analisar todos os trabalhos fortemente relacionados ao tema de processo ETL em tempo real em ambientes de *data warehousing*, foi possível constatar elementos que serviram como motivação para o desenvolvimento do Imã de Dados: 1) todos os trabalhos que lidam com a extração de dados adotam o mecanismo CDC para

obter os dados de interesse. Isso quer dizer que, dependendo do volume de dados e da frequência na qual os dados são produzidos, o processo ETL como um todo continuará custoso; 2) nenhum trabalho relacionado considera o fato de que um dado armazenado no DW tem relação direta com um dado de interesse; 3) poucos trabalhos lidam com a extração de dados de sensores para armazená-los no DW; 4) todos os trabalhos realizam o processo ETL convencional de forma intrusiva; 5) no mesmo sentido do item 4, nenhum trabalho considera o fato de que o processo ETL pode ser executado de forma reativa, seguindo um protocolo pré-estabelecido.

Vale destacar que a utilização de sensores é um requisito fortemente considerado para a execução e validação do Imã de Dados. Isso se deve ao fato de que sensores podem gerar um fluxo contínuo de dados, o que faz com que o volume de dados e frequência na geração de dados se torne um desafio para a execução e validação do Imã de Dados. Além disso, sensores de diversos tipos podem ser encontrados em diferentes locais e em diferentes domínios. Dessa forma, se torna possível solucionar problemas atuais envolvendo dados gerados em tempo real por sensores e seu armazenamento no DW. Consequentemente, as análises de dados e tomadas de decisões podem ser realizadas também em tempo real.

Portanto, percebe-se que o estudo relacionado ao processo ETL aplicado em ambientes de tempo real com dados gerados por meio de sensores é um tema que carece de novas contribuições originais e relevantes na fronteira do estado da arte da área de banco de dados. Por consequência, diversas áreas de negócios ainda podem ser estudadas, assim como problemas podem ser resolvidos com a utilização do processo ETL em um grande fluxo de dados com o DW como repositório.

5.2 Objetivos

Após identificar o problema e as soluções propostas até então, foi identificado os pontos negativos (ou limitações) de tais soluções, assim como as motivações já citadas. Após esse processo, percebeu-se que a fase de extração de dados se torna complexa quando há uma heterogeneidade no ambiente operacional. Além disso, essa heterogeneidade torna o processo ainda mais complexo quando há um grande volume de dados gerado continuamente e simultaneamente por vários usuários. Isso se deve ao fato das soluções propostas até então não terem realizado experimentos que validem suas propostas em ambientes com requisitos de tempo real. Dessa forma, o Imã de Dados foi desenvolvido para investigar o problema de executar o processo ETL contendo um fluxo contínuo de dados gerados em tempo real e o DW como repositório desses dados. Assim, foi possível definir a arquitetura do Imã de Dados, a qual será descrita neste capítulo.

Antes de detalhar a arquitetura, é importante destacar os cenários em que o Imã de Dados pode ser aplicado: o primeiro cenário é uma organização que, ainda na fase inicial de implantação de todo o ambiente computacional, deseja realizar o projeto do ambiente de *data*

warehousing. Entretanto, ainda não há dados sendo gerados no ambiente operacional e tampouco dados já armazenados no DW para tomada de decisão estratégica. Contudo, deseja-se já projetar o ambiente como um todo, de modo a permitir que os dados operacionais sejam, em algum momento futuro, enviados para o DW. Dessa forma, tal ambiente pode ser construído seguindo os padrões do Imã de Dados, de modo que, no futuro, possa fazer uso dos seus recursos. O segundo cenário é uma organização que já tem dados sendo produzidos continuamente pelo ambiente operacional e deseja fazer uso dos recursos fornecidos pelo Imã de Dados. Neste caso, visto que o ambiente operacional já tem dados sendo gerados e o DW está vazio, deve-se realizar dois passos: 1) o primeiro passo é realizar uma carga completa dos dados de interesse do ambiente operacional para o DW. Para isso, deve-se usar técnicas tradicionais disponíveis para realizar o processo ETL, por exemplo o uso de CDC. Isso se deve ao fato de que os tomadores de decisões podem querer tomar decisões baseando-se nos dados já produzidos na organização. Nesse caso, a carga completa dos dados ocorrerá apenas uma vez, ou seja, logo e somente na implantação do Imã de Dados. Contudo, o processo de carga completa pode levar tempo para ser finalizado, dependendo do volume de dados a ser enviado do ambiente operacional para o DW; 2) o segundo passo compreende passar a atualizar o DW em tempo real a partir do momento da implantação do Imã de Dados.

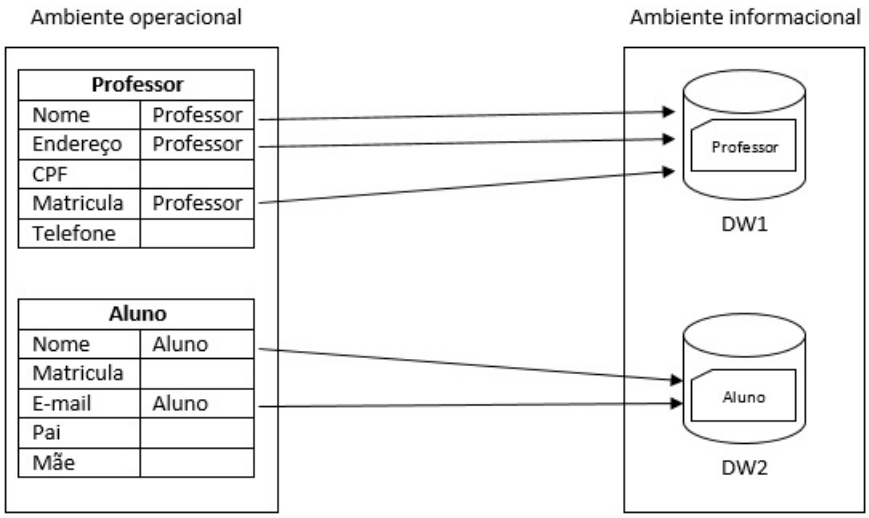
5.3 Imã de Dados

A ideia base do Imã de Dados é associar um item de dado do ambiente operacional a uma representação do mesmo no ambiente informacional. Dessa forma, assim que um dado for gerado e armazenado em uma fonte de dados do ambiente operacional, o dado deve ser imediatamente enviado para o ambiente de *data warehousing* a partir de um forte relacionamento entre o dado gerado e seu local de destino, que pode ser, por exemplo, o seu armazenamento em um determinado DW. Para que isso ocorra, deve-se definir uma *tag* para cada item de dado de interesse que é mantido nas fontes de dados. Basicamente, uma *tag* é um indicativo de que um dado de interesse será utilizado em algum momento no ambiente de *data warehousing*. Assim, o projetista do banco de dados deve definir quais itens de dados operacionais poderão ser armazenados em algum DW e que servirão para as tomadas de decisões. Nesse momento, cria-se o conceito de definir dados de interesse, ou seja, definir quais itens de dados operacionais serão utilizados no ambiente de *data warehousing* para futuras consultas OLAP.

Dado esse cenário, é possível imaginar o seguinte exemplo: pode-se definir uma *tag* chamada *professor* para os atributos *nome, endereço e matrícula* de cada professor de uma tabela de um banco de dados relacional armazenada no Departamento de Computação da UFSCar. Da mesma forma, pode-se definir uma *tag* chamada *aluno* para os atributos *nome* e *e-mail* de cada aluno. Já no ambiente informacional, pode-se ter diferentes bases de dados que serão utilizadas como DW. Por exemplo: o DW1 armazena dados de professor e o DW2 armazena dados de aluno. Então, deve-se definir também quais *tags* são de interesse e terão dados armazenados em

cada DW. No exemplo corrente, imagina-se que a *tag professor* será utilizada pelo DW1 e a *tag aluno* será utilizada pelo DW2. Dessa forma, é possível estabelecer uma relação entre um item de dados do ambiente operacional e sua representação no ambiente informacional. Assim, sempre que um dado de interesse for gerado no ambiente operacional, o Imã de Dados recebe esse dado gerado e, por meio das *tags* definidas, saberá qual base de dados no ambiente informacional irá armazenar o item de dado.

Figura 17 – Exemplo básico do Imã de Dados



Fonte: Próprio autor

A Figura 17 mostra um exemplo das *tags* configuradas no ambiente operacional. A partir das *tags* configuradas, o Imã de Dados é capaz de direcionar automaticamente os dados recebidos do ambiente operacional para os DWs interessados em armazenar o item de dado. Em particular, o exemplo mostra uma única fonte de dados operacional fornecendo dados para o DW. Além disso, apenas para ilustração, o exemplo foi feito utilizando o formato de tabela. Entretanto, o Imã de Dados é capaz de lidar com diferentes fontes de dados heterogêneas do ambiente operacional, assim como diversas bases de dados do ambiente informacional. Contudo, isso somente é possível se ambos as fontes de dados e os DWs permitirem a intervenção do projetista do banco de dados para configurar as *tags*. Essa intervenção tem o objetivo apenas de definir os dados de interesse, não sendo necessário realizar qualquer tipo de modificação no esquema ou conteúdo da tabela.

A partir da definição da arquitetura, percebe-se que o Imã de Dados difere das outras abordagens do processo ETL disponíveis na literatura. Primeiramente, o Imã de Dados introduz o conceito de *tag* ao executar o processo ETL em tempo real em ambientes de *data warehousing*. Esse fato faz com que um dado de interesse tenha uma relação direta com DWs que necessitam armazenar o dado de interesse. Além do conceito de *tag*, o Imã de Dados apresenta uma solução para execução do processo ETL em tempo real composta pelo modelo *publisher/subscriber* (pub/sub). A partir desse modelo, é possível o compartilhamento de dados entre ambientes

heterogêneos e manter o desempenho e escalabilidade de todo o ambiente.

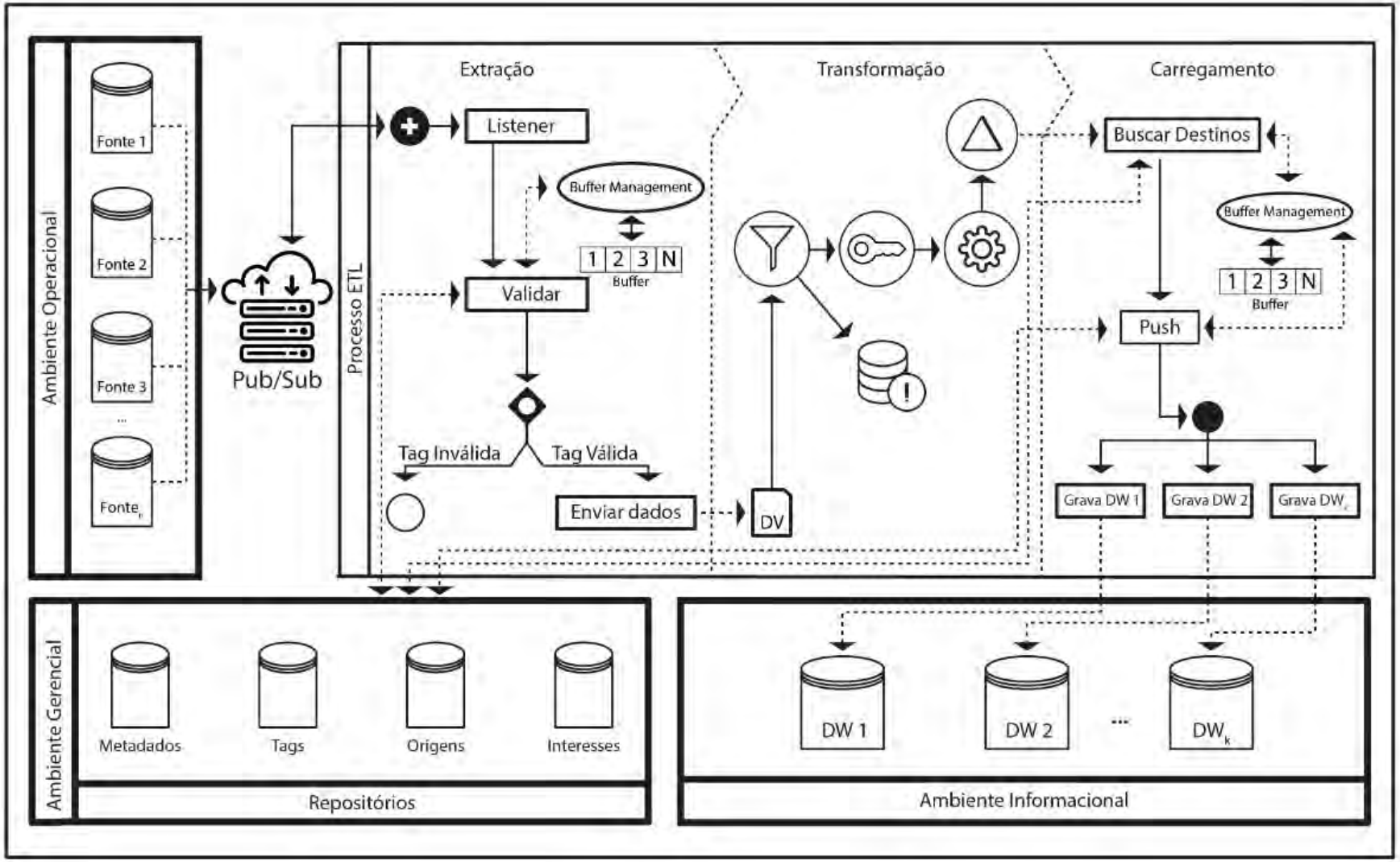
Além dessas duas características, o Imã de Dados foi projetado para ser executado de forma não intrusiva e reativa. Por não intrusiva, entende-se que o Imã de Dados não precisa conhecer o esquema das fontes de dados operacionais e tampouco acessá-las para obter os dados. Ao contrário, o Imã de Dados irá receber (e não ir buscar) as solicitações do ambiente operacional. Dessa forma, o ambiente operacional, sempre que necessário (independentemente se for numa frequência predefinida ou em tempo real), envia dados de interesse para o Imã de Dados sem ocorrer o *overhead* nas fontes de dados operacionais. A partir do conceito de *tag*, o Imã de Dados é capaz de estabelecer a relação direta entre o item de dado recebido e os DWs nos quais o item de dado será armazenado. Assim, não é necessário que o processo ETL dependa de algum *wrapper*, gatilho, percorra um arquivo de *log* ou uma tabela temporária nas fontes de dados do ambiente operacional para obter os dados de interesse. Já por reativo, entende-se que o Imã de Dados irá executar alguma ação automaticamente assim que um dado de interesse for recebido seguindo um padrão estabelecido. Algumas das ações incluem: validar o item de dado recebido, verificar quais são os DWs interessados em armazenar o item de dado. Todas as ações serão explicadas detalhadamente ao longo deste capítulo.

5.3.1 Arquitetura do Imã de Dados

A Figura 18 apresenta a arquitetura do Imã de Dados. Essa figura foi criada a partir das notações do modelo de processo *Business Process Model Notation* (BPMN). Essa abordagem foi escolhida por se tratar de uma abordagem aplicada por alguns trabalhos na representação conceitual do processo ETL. Além disso, essa ferramenta oferece mecanismos de representar todo o fluxo de atividades do processo ETL, assim como suas inter-relações. Segundo El Akkaoui e Zimanyi (AKKAOUI; ZIMANYI, 2009), a abordagem BPMN fornece uma gama de ferramentas para representação gráfica de um processo de negócio. Além disso, é possível mostrar o relacionamento de todos os processos envolvidos. Assim, as próximas seções serão dedicadas ao detalhamento de todos os componentes da arquitetura.

É importante destacar que a arquitetura é dividida em quatro partes (ou módulos): Ambiente Operacional, Processo ETL, Ambiente Informacional e Ambiente Gerencial. Além disso, o processo ETL é composto por três componentes: Extração, Transformação e Carga (ou Carregamento). O ambiente informacional é composto por bases de dados que representam o DW. Por fim, o ambiente gerencial é composto por bases de dados que representam os repositórios que armazenam as configurações do Imã de Dados. Em particular, a figura mostra a fase de transformação de dados. Contudo, ao longo de todo o capítulo, essa fase não é contemplada. A fase de transformação foi coloca na figura apenas para ilustrar que a transformação também faz parte do processo ETL. Porém, essa fase do processo ETL está fora do escopo deste trabalho e certa forma é independente da solução proposta e dependente do domínio de negócio no qual o processo ETL executa para carregar dados das fontes de dados do ambiente operacional para os

Figura 18 – Arquitetura do Imã de Dados.



Fonte: Próprio autor

repositórios de *data warehouse* no ambiente informacional.

5.3.1.1 Ambiente operacional

Em qualquer domínio de negócio, o ambiente operacional é composto por um grupo de usuários responsáveis por produzir os dados de interesse que serão utilizados por outro grupo de usuários tomadores de decisões. De acordo com cada domínio, os dados são originados de diferentes formas. Por exemplo: um domínio pode necessitar de dados gerados por meio de aplicativos instalados em *smartphones*. Por outro lado, outro domínio pode necessitar de dados gerados por meio de inserções manuais dos usuários. Além disso, podem haver domínios que necessitam de dados gerados pela interação entre diferentes dispositivos em uma rede de sensores. Em resumo, pode-se ter diferentes formas de produzir os dados que serão utilizados pelos usuários tomadores de decisões.

Portanto, na Figura 18, o componente Ambiente Operacional representa as fontes de dados do ambiente operacional, as quais vão enviar os dados de interesse para o ambiente de *data warehousing* por meio do Imã de Dados. É importante destacar que esse ambiente pode ser composto por fontes de dados heterogêneas, as quais mantêm os dados em diferentes formatos e permitem obtê-los por meio de diferentes meios. Contudo, para o Imã de Dados, esse fato não é essencial, pois o Imã de Dados vai receber (e não buscar) os dados de interesse das fontes de dados. Assim, não é necessário manter os metadados de conexão dessas fontes de dados do ambiente operacional.

5.3.1.2 Modelo Publisher/Subscriber

Para que o Imã de Dados possa ser executado a partir de ambientes heterogêneos, a arquitetura foi projetada com base no modelo publish/subscriber (pub/sub) como meio de compartilhamento de dados entre ambientes. Dessa forma, o ambiente operacional age como produtor dew dados, isto é, produzem e publicam os dados de interesse, os quais serão armazenados em DWs. Por outro lado, o Imã de Dados atua como consumidor de dados, ou seja, se inscreve no *broker* do modelo pub/sub e demonstra interesse em receber notificações de dados do ambiente operacional. Sempre que uma nova notificação de dado for recebida, o Imã de Dados é capaz de receber o dado produzido (independentemente do ambiente operacional e sem ter conhecimento sobre o ambiente operacional) e realizar suas tarefas automaticamente.

A partir do modelo pub/sub introduzido no Imã de Dados é possível desacoplar o ambiente operacional e o ambiente informacional. Dessa forma, é possível que as fontes de dados que mantém os dados de interesse não tomem conhecimento dos DWs que armazenarão os dados de interesse. No mesmo sentido, não é necessário que o DW conheça as fontes de dados que produzem os dados de interesse. O Imã de Dados é capaz de atuar dinamicamente, isto é, identifica novos dados de interesse e os armazena nos DWs interessados nos dados de forma automática.

5.3.1.3 Processo ETL

A partir dos dados produzidos no ambiente operacional, o Imã de Dados é acionado e os dados produzidos são recebidos pelo Imã de Dados com o auxílio do componente Pub/Sub. Após o recebimento desses dados, o Imã de Dados deve realizar tarefas referentes ao processo ETL. Essas tarefas dizem respeito a verificação e validação do padrão dos dados, identificação dos DWs interessados em armazenar o item de dado, identificação dos metadados de conexão aos DWs e o armazenamento do item de dado nos DWs. Essas validações e identificações são realizadas por subcomponentes os quais compõem o processo ETL. A seguir serão apresentados os subcomponentes, assim como suas respectivas responsabilidades.

5.3.1.4 Extração

A fase de extração é a primeira fase do processo ETL, por meio da qual o ambiente operacional se comunica com o Imã de Dados. Essa fase é responsável por receber um item de dado enviado pelas fontes de dados, validar esse item de dado e disponibilizá-lo para a próxima fase do processo ETL. Essa fase é composta por quatro componentes:

- **Listener**: esse componente é responsável por, efetivamente, receber o item de dado das fontes de dados. O *Listener* se conecta ao *broker* do pub/sub e demonstra interesse em receber solicitações de envio de dados de uma fonte de dados. Sempre que uma nova notificação de dado é enviada ao *broker*, o *Listener* recebe essa solicitação e mantém temporariamente o dado recebido.

  É importante destacar que os dados enviados pelas fontes de dados devem estar em um formato pré-estabelecido e reconhecido pelo Imã de Dados. O formato padrão proposto é o seguinte: Item(nome do item de dado, valor do dado, tipo do dado, lista de *tags*, nome da fonte de dados, *timestamp*). Por exemplo: um projetista decidiu que na fonte de dados *DepartamentoComputacao*, o campo *Nome* é um item de dado de interesse para ser armazenado no ambiente de *data warehousing*. Para isso, definiu a *tag Professor* para o campo *Nome*. Dessa forma, sempre que a fonte de dados *DepartamentoComputacao* desejar enviar o campo *Nome* para o ambiente de *data warehousing*, a fonte de dados deve acionar o Imã de Dados por meio do *Listener* e enviar o dado no formato estabelecido para o modelo pub/sub. Nesse exemplo, o formato do dado completo é: Item(Nome, "José", *String*, [Professor], DepartamentoComputacao, 01-01-2019 08:00:00).

  Após receber o item de dado, o *Listener* deve guardar temporariamente esse item de dado para que o próximo componente da arquitetura realize sua tarefa. Para isso, o item de dado é mantido temporariamente em um *buffer-pool* (ao longo do texto, o *buffer-pool* será nomeado apenas de *buffer*). Basicamente, um *buffer-pool* é um espaço na memória utilizado especificamente para armazenar temporariamente objetos localizados

em memória secundária. Esse *buffer* contém todos os dados recebidos pelo *Listener* e os mantém em ordem cronológica de recebimento.

A proposta atual da arquitetura do Imã de dados trata apenas dados convencionais que possuem relação de ordem total, tais como números, datas e cadeias de caracteres. Para dados convencionais, o formato de dados usado pelo Imã de Dados é suficiente para atender as necessidades de processamento do fluxo do processo ETL. Dados não convencionais estão fora do escopo deste trabalho, tais como dados espaciais e imagens.

- **Validar**: com os dados de interesse mantidos temporariamente na lista, é necessário que os dados sejam validados. Esse processo de validação consiste em verificar se existe alguma *tag* associada ao item de dado recebido que seja válida. Para realizar a validação, o processo Validar se comunica com o Repositório de Origens (explicado adiante) e verifica se existe alguma *tag* definida para o item de dado em questão. Se nenhuma *tag* válida for identificada nesse processo, será emitida uma resposta de erro ao ambiente operacional. Além disso, o usuário administrador do banco de dados deverá ser acionado para verificar os motivos pelos quais a *tag* não foi associada ao item de dado. Caso contrário, também será emitida uma resposta ao ambiente operacional apenas para indicar que a *tag* é válida e os dados foram validadas com sucesso.

  É importante destacar que nesse processo Validar ocorre a identificação de uma *tag* e um item de dado do ambiente operacional. Nesse momento, é verificado quais *tags* estão associadas a um item de dado que foi enviado pelo ambiente operacional. Esse processo é importante para que o processo Buscar Destinos (explicado adiante) possa associar uma base de dados de destino (ou seja, um ou mais DWs) a cada *tag* identificada nesse processo.

  Após validar o item de dado recebido, o mesmo é armazenado em um *buffer*. Esse processo é essencial para o desempenho do Imã de Dados, pois sempre que um novo item de dado for recebido pelo *Listener*, é verificado se o dado já foi validado. Se o item de dado ainda não foi validado, é feita uma conexão com o Repositório de Origens e feita a validação. Esse processo é muito custoso. Caso contrário, o item de dado já estará validado e armazenado no *buffer*. Logo, o custo para realizar o processo de validação é bem menor, visto que não é necessário acessar uma base de dados em disco e efetivamente validar o item de dado novamente. Assim, obtém-se um grande ganho de desempenho no processo de extração de dados como um todo. Apenas para ilustração, as primeiras versões do Imã de dados não possuíam o *buffer* e o desempenho da arquitetura foi bem inferior, em uma ordem de magnitude.

- **Enviar Dados**: após o item de dado ser enviado das fontes de dados do ambiente operacional, recebido pelo *Listener* e o item de dado ser validado, é necessário enviar esse item de dado válido para a fase de carga (ou carregamento de dados). Para isso, o processo Enviar Dados encaminha o item de dado validado para a fase de carregamento. Já na fase de carregamento, esses dados válidos são recebidos pelo processo chamado Buscar

Destinos. É importante destacar que a fase de transformação foi ilustrada na Figura 18 apenas para mostrar que essa fase faz parte de todo o processo ETL. Essa fase é composta por um *workflow* de tarefas específicas para cada domínio de negócio. Por esse motivo, a transformação foi apenas ilustrada na figura, mas não foi considerada para ser investigada na arquitetura do Imã de Dados.

5.3.1.5 Carga

A fase de carga de dados (ou carregamento) é a fase na qual os itens de dados válidos são efetivamente armazenados no DW. Para tanto, essa fase é composta por dois processos:

- **Buscar Destinos**: O processo Buscar Destinos é responsável por se comunicar aos repositórios de interesse (explicado adiante) e identificar quais DW tem interesse em armazenar o item de dado, o qual já está associado a *tags*. A partir de então, é possível criar a relação entre o item de dado válido e seus DWs de destino. Por exemplo: o usuário administrador do banco de dados definiu que a base de dados DW1 tem interesse na *tag Professor* e a base de dados DW2 tem interesse na *tag Aluno*. Após ambas as *tags* serem validadas, ou seja, tem um item de dado a elas associadas, foi estabelecida a relação que dados da *tag Professor* serão armazenados no DW1 e dados da *tag Aluno* serão armazenados no DW2.

- **Push**: O processo *Push* é responsável por efetivamente armazenar os dados em cada DW de destino. Para isso, para cada DW interessado em armazenar o item de dado, é feita a comunicação com o Repositório de Metadados para buscar os metadados de conexão com o DW. Dessa forma, a partir do item de dado validado, do DW interessado em armazenar esse item de dado e com base nos metadados de conexão do DW, o item de dado é efetivamente armazenado no DW de destino.

  Assim como na fase de extração, a fase de carregamento é composta pelo componente *buffer*. Esse componente armazena a relação entre o item de dado recebido pelo *Listener*, o DW de interesse e os metadados de conexão com o DW. Sempre que uma nova solicitação de armazenamento de dados no DW é recebido pelo processo Buscar Destinos, é verificado se já existe a relação [item de dado, DW de interesse, metadados de conexão] mantidos no *buffer*. Se não existir, é feita a comunicação com os repositórios de Interesse e Metadados para buscar, respectivamente, o DW interessado em armazenar o item de dado e seus metadados de conexão. Caso contrário, busca-se no *buffer* esses dados. Esse recurso permite otimizar o desempenho do Imã de Dados, visto que nos casos em que os dados já estiverem armazenados no *buffer*, não é necessário realizar uma conexão com as bases de dados gerenciais.

  É importante destacar que o Imã de dados foi projetado baseando-se também no conceito de paralelismo. Assim, a cada novo dado recebido pelo *Listener*, o mesmo é tratado de forma individual pelo Imã de Dados. Isso quer dizer que o Imã de Dados é capaz de realizar

todas as suas tarefas com vários dados recebidos das fontes de dados de forma paralela. Na Figura 18, esse aspecto é representado por meio de dois componentes: o primeiro é um componente do modelo BPMN localizado entre o *Listener* e o Pub/Sub (sinal de +), o qual representa processos paralelos; o segundo são três processos (Grava DW1, Grava DW2 e Grava DWk) representados no final da fase de carregamento.

De modo geral, pode-se observar a partir das definições apresentadas que ao usar conjuntamente os conceitos de **não intrusivo** e **reativo**, o modelo pub/sub, o conceito de *tags*, o armazenamento de dados temporariamente em um *buffer* e o paralelismo, percebe-se que o desempenho do Imã de Dados é superior ao desempenho das técnicas tradicionais para realizar o processo ETL, visto que o desacoplamento entre o ambiente operacional e o ambiente informacional, além das técnicas aplicadas, favorece ao ganho de desempenho e escalabilidade do ambiente desenvolvido. Esse ganho de desempenho e escalabilidade é mostrado no Capítulo 6, no qual são apresentados os testes experimentais realizados para validar o Imã de Dados com dados reais e sintéticos.

5.3.2 Ambiente informacional

O ambiente informacional é composto por bases de dados que representam o DW e processos, cuja a função é manter os dados no DW e permitir às consultas OLAP. Na Figura 18, esse ambiente é representado pelos componentes DW1, DW2 e DWk.

5.3.3 Ambiente gerencial

Como já foi discutido, o usuário administrador exerce um papel fundamental na configuração e na consequente execução do Imã de Dados. Isso se deve ao fato de que o Imã de Dados deve ser configurado manualmente para que suas tarefas sejam posteriormente executadas automaticamente. Dessa forma, é necessário um ambiente no qual permita-se manter as configurações realizadas pelo usuário administrador. Esse ambiente é composto por repositórios gerenciais e é chamado de Ambiente Gerencial. Os repositórios gerenciais são representadas na Figura 18 pelos componentes: Metadados, Tags, Origens, Interesses.

- **Repositório de Tags**: esse repositório tem a responsabilidade de armazenar as *tags* configuradas.

  A Tabela 3 mostra uma configuração de exemplo do Repositório de Tags. É possível configurar a *tag* unicamente ou uma hierarquia de *tags*. O atributo Pai indica o principal nível da hierarquia. O atributo Tag indica um subconjunto pertencente a hierarquia. Além disso, é possível definir uma descrição para a *tag*, a qual auxiliará o usuário administrador em entender o significado da *tag* nos outros processos que dependem das *tags* configuradas. Vale destacar que as *tags* podem ser definidas no ambiente operacional baseando-se em

Tabela 3 – Exemplo do Repositório de Tags.

Pai	**Tag**	**Descrição**
Null	Professor	Tag que representa todos os professores.
Professor	Professor-Ensino Médio	Tag que representa os professores do ensino médio.
Professor	Professor-Graduação	Tag que representa os professores da graduação.
Null	Aluno	Tag que representa os alunos

Fonte: Próprio autor

qualquer nível hierárquico. Isso quer dizer que o usuário pode obter dados operacionais a partir de uma *tag* que representa a raiz da hierarquia ou qualquer outra raiz filha.

- **Repositório de Origens**: esse repositório tem a responsabilidade de armazenar a relação entre um dado de interesse e uma *tag*. A origem definida será consultada pelo processo Validar (explicado na seção 5.3.1.4), de modo a verificar se existe alguma *tag* associada ao dado de interesse recebido do ambiente operacional.

Tabela 4 – Exemplo do Repositório de Origens

Fonte de Dados	**Entidade**	**Item de Dado**	**Tag**
Notas	Professores	Nome	Professor
Notas	Professores	CPFProfessor	Professor
Notas	Professores	EnderecoProfessor	Professor
Notas	Alunos	NomeAluno	Aluno
Notas	Alunos	NotaAluno	Aluno

Fonte: Próprio autor

A Tabela 4 mostra um exemplo do Repositório de Origens configurado. O campo Fonte de Dados representa o nome da fonte de dados que mantém os dados no ambiente operacional. O campo Entidade representa o local no qual está mantido o item de dado na Fonte de Dados. Essa Entidade pode ser uma tabela de um banco de dados relacional, um conjunto de arquivos XML, um arquivo JSON ou outra entidade que mantém um conjunto de itens de dados. O campo Item de Dado representa o dado de interesse de uma entidade da fonte de dados do ambiente operacional. O campo Tag representa o nome da *tag* associada ao dado de interesse. Percebe-se que os itens de dados Nome, CPFProfessor e EnderecoProfessor estão associados a *tag* Professor e à fonte de dados Notas. Já os itens de dados NomeAluno e NotaAluno estão associados a *tag* Aluno e também à fonte de dados Notas. Desse modo, sempre que o processo Validar receber os campos Nome, CPFProfessor e EnderecoProfessor da fonte de dados Notas pelo *Listener*, o mesmo saberá que esses itens de dados estão associados a *tag* Professor. Da mesma forma, sempre que receber os campos NomeAluno e NotaAluno da fonte de dados Notas pelo *Listener*, saberá que esses campos estão associados a *tag* Aluno. Vale destacar que a *tag* é associada a um

item de dado juntamente com sua fonte de dados. Ou seja, pode-se ter um item de dado com um mesmo nome para fontes de dados diferentes.

Por exemplo: o *Listener* recebeu o seguinte dado: {Fonte de Dados: "Notas"; Entidade: "Professores"; Item de Dado: "Nome"; Valor: "José"}. Ao receber esses dados pelo *Listener*, o processo Validar identifica que tais campos estão associados à *tag* Professor. Isso se deve ao fato de que o usuário administrador configurou o Repositório de Origens com essas definições.

- **Repositório de Metadados**: esse repositório tem a responsabilidade de armazenar os metadados de conexão dos DWs de destino. Esses metadados são essenciais para que o processo *Push*, da fase de carregamento, possa identificar os metadados de conexão e se conectar a cada DW definido.

Tabela 5 – Exemplo do Repositório de Metadados.

Nome	**Usuario**	**Senha**
DW1	D1	W1
DW2	D2	W2

Fonte: Próprio autor

A Tabela 5 mostra um exemplo de configuração do Repositório de Metadados. Nesse exemplo, é possível configurar o nome da base de dados, o nome de usuário e senha de conexão à base de dados. Entretanto, é permitido que o usuário administrador adicione novos valores de configurações, visto que podem haver outros metadados importantes que são utilizados para se conectar às bases de dados.

- **Repositório de Interesses**: esse repositório tem a responsabilidade de armazenar a relação entre a *tag* e o DW interessado em armazenar o dado de interesse associado à *tag*.

Tabela 6 – Exemplo do Repositório de Interesses

Tag	**Destino**
Professor	DW1
Professor	DW2
Aluno	DW3

Fonte: Próprio autor

A Tabela 6 mostra um exemplo do Repositório de Interesses. Nesse exemplo é mostrado o nome da *tag* e o DW de destino. Esse repositório é acessado pelo processo Buscar Destinos da fase de carregamento Imã de Dados, de modo a identificar para qual DW deve ser enviado o dado de interesse validado pelo processo Validar da fase de extração.

Vale destacar que o usuário pode configurar uma *tag* para mais de um DW de destino. Portanto, os mesmos dados de interesse recebidos do ambiente operacional podem ser

armazenados em mais de um DW diferente. Logo, o processo Buscar Destinos deve lidar com o gerenciamento do armazenamento dos dados nos DW de interesse.

Após destacar todos os componentes da arquitetura ilustrada na Figura 18, os seguintes passos mostram como ocorre o fluxo dos dados no Imã de Dados: 1) um dado de interesse é recebido pelo *Listener*; 2) o processo Validar obtém esse dado de interesse e verifica as *tags* associadas ao dado de interesse recebido junto ao Repositório de Origens. Nesse momento, tem-se um dado de interesse associado a *tags* válidas; 3) o processo Buscar Destinos busca os DWs interessados nesse dado de interesse junto ao Repositório de Interesses. Isso ocorre por meio de cada *tag* associada ao dado de interesse e da mesma *tag* associada aos DWs de destino; 4) o processo *Push* obtém o dado de interesse validado (item de dado, *tags*, bases de destino) e busca os metadados de conexão de cada DW de destino junto ao Repositório de Metadados; 5) o processo GravaDW pega os dados de interesse e os persiste em cada DW de destino.

Outro aspecto importante a ser destacado é que o usuário administrador interage diretamente com a configuração e operacionalização do Imã de Dados. Assim, a qualquer momento, o usuário administrador pode ser acionado para verificar, por exemplo, alguma *tag* não associada a um dado de interesse, alguma falha de conexão ao DW pode ocorrer ou qualquer outra anomalia que possa impedir o correto funcionamento do Imã de Dados. Dessa forma, o usuário administrador deve ser capaz de realizar operações de inserção, alteração, remoção e pesquisa sobre todos os repositórios do Ambiente Gerencial.

5.4 Considerações finais

Neste capítulo foi apresentada e descrita detalhadamente a arquitetura do Imã de Dados. Após as configurações de *tags*, da fontes de dados do ambiente operacional e da definição dos DWs de interesse, o Imã de Dados pode ser executado. Assim, sempre que um dado de interesse for recebido do ambiente operacional pelo *Listener*, o Imã de Dados deve, automaticamente e em tempo real, identificar quais DWs irão receber o dado de interesse e armazenar o dado de interesse em cada DW de destino.

Dessa forma, espera-se retirar a sobrecarga da fase de extração de dados do processo ETL em lidar com a heterogeneidade dos dados do ambiente operacional, assim como retirar a responsabilidade de gerenciar as formas de acesso à tais fontes de dados. Assim, o Imã de Dados difere de todas as outras abordagens existentes na literatura devido aos novos conceitos introduzidos ao se pensar em processo ETL em tempo real. Os conceitos de **não intrusivo** e **reativo**, a utilização do modelo pub/sub, do conceito de *tags*, do armazenamento de dados temporariamente em *buffers* e do conceito de paralelismo, faz do Imã de Dados uma mudança de paradigma ao pensar no processamento de fluxos em processos ETL em tempo real em ambientes de *data warehousing*.

Para mostrar o ganho de desempenho do Imã de Dados em relação ao estado da arte do processo ETL em ambientes de *data warehousing* em tempo real, o Imã de Dados foi testado e validado por meio de um experimento em um ambiente real contendo dados reais e em um experimentado com dados sintéticos. Portanto, no Capítulo 6 será mostrado e detalhado o experimento realizado com o Imã de Dados.

Capítulo 6
EXPERIMENTOS

Após ser apresentada a contextualização, motivação, problema e objetivos no Capítulo 1, a fundamentação teórica dos principais conceitos acerca do tema de pesquisa no Capítulo 2, uma descrição do processo ETL no Capítulo 3, as características de um ambiente de *data warehousing* em tempo real no Capítulo 4 e principalmente após a apresentação e detalhamento da arquitetura do Imã de Dados no Capítulo 5, é necessário mostrar como o Imã de Dados foi testado e validado, de modo que as hipóteses definidas na seção 1.5 sejam confirmadas ou refutadas, além de demonstrar a correta execução do processo ETL em tempo real pela arquitetura do Imã de Dados.

Portanto, este capítulo tem o objetivo de apresentar e discutir os resultados dos dois experimentos realizados com o Imã de Dados. O primeiro experimento teve por objetivo validar a viabilidade de aplicar o Imã de Dados em um ambiente IdC, o qual é composto por Internet, rede de sensores e microcontroladores. O segundo experimento teve dois objetivos: o primeiro objetivo foi validar a disponibilidade, o tempo de resposta e a escalabilidade do Imã de Dados a medida que aumenta o número de sensores, o volume de dados produzidos e a frequência na geração de dados. O segundo objetivo foi comparar o tempo de resposta do Imã de Dados com a técnica considerada estado da arte em processo ETL em ambientes de *data warehousing* em tempo real, isto é, o uso de CDC por meio da técnica de gatilhos.

Dessa forma, na seção 6.1 será apresentada uma contextualização de um ambiente de pecuária leiteira, assim como as características e como ocorre a produção de leite em um ambiente de pecuária leiteira real. Na seção 6.2 será mostrado o ambiente IdC criado para viabilizar o experimento em um ambiente real de pecuária leiteira. A seção 6.3 foi separada em três grandes subseções. Na seção 6.3.1 será apresentada a configuração realizada no Imã de Dados de modo a permitir a execução de suas tarefas. Na seção 6.3.2 será apresentada a execução completa do Imã de Dados, desde a geração de um dado no ambiente IdC até o armazenamento desse dado no DW. Já na seção 6.3.3 será mostrada algumas considerações e particularidades a respeito da execução dos experimentos. Na seção 6.3.4 serão mostrados os resultados obtidos após a execução dos experimentos.

6.1 Ambiente de pecuária leiteira

Em uma fazenda produtora de leite, sua renda principal é o próprio leite. Quanto mais leite a fazenda produz, maior será o ganho financeiro. Contudo, existem diversos fatores que podem afetar diretamente a produção de leite das vacas, tais como: nutrição, intoxicação alimentar, ataques por cobras ou outros tipos de animais, manejo diário do animal (local onde passa o dia, exposição a um ambiente muito quente, muito frio ou muito úmido) e demais fatores que afetam o bem-estar animal. Portanto, quanto melhor for o manejo do animal, melhor será o ambiente no qual esse animal estará e mais leite ele produzirá. Nesse sentido, a agropecuária digital, sobretudo a pecuária leiteira, foi adotada como domínio para aplicar o Imã de Dados, pois esse domínio é composto por características que são compatíveis ao tema deste trabalho. Os dados de produção de leite que são produzidos e consumidos por esse ambiente são gerados diariamente. Além disso, tanto os dados gerados em tempo real quanto o histórico desses dados são fatores essenciais na tomada de decisão estratégica.

Além desses fatores que afetam diretamente a produção de leite, os produtores e os veterinários responsáveis pelo manejo das vacas utilizam diversos dados históricos para realizar previsões de datas ou ações preditivas nos animais, tais como próximas vacinas, novas quantidades de alimentos, novos locais de descanso do animal e outras ações. Além dessas análises, os profissionais que atuam nesse ambiente utilizam os dados históricos para analisar a produção de leite. A análise do histórico de produção de leite é essencial, pois é por meio dessa análise que é possível verificar se a vaca está apta para a lactação, se é o momento para seu descarte no rebanho, se a mesma tem alguma doença ou anomalia e analisar a rentabilidade financeira da fazenda. Além disso, é possível prever algumas datas para cada vaca individualmente, tais como: data da próxima lactação, data de saída da lactação, data do próximo cio e data do próximo enxerto. Todas esses dados são gerados exclusivamente com base no histórico de dados de cada animal.

Todas as anomalias que afetam as vacas e consequente sua produção de leite podem ser verificadas de duas formas: 1) olhando pessoalmente o animal e analisando tais fatores; 2) analisar o histórico do animal e verificar o volume de leite produzido em cada período. Na segunda forma, se ocorrer uma queda inesperada na produção de leite, algum fator negativo afetou a vaca. Além disso, se a média diária de produção de leite do animal tem sofrido uma queda constante dia após dia, algo negativo afetou o animal. Caso alguma anomalia seja identificada com base no histórico de produção de leite, alguma ação veterinária deve ser tomada imediatamente. Além do mais e mais agravante, grande parte das fazendas produtoras de leite realizam a produção de leite no mínimo em um período e no máximo em três períodos ao dia (manhã, tarde e noite). Quanto maior a frequência de produção de leite durante o dia, maior deverá ser o controle. Isso se deve ao fato que, de um período para outro, pode haver uma queda na produção de leite de cada vaca individualmente. Isso ocorre pois a vaca pode ter sofrido algum ataque de animal peçonhento, alguma intoxicação alimentar ou ter sofrido outra anomalia poucos minutos antes de iniciar

a ordenha. Caso isso ocorra, sua produção de leite irá cair inevitavelmente naquele período e consequentemente naquele dia. Assim, por consequência de tais fatores negativos que afetaram a vaca, alguma ação deve ser tomada imediatamente. É importante destacar que todos esses fatores citados contribuem para afirmar que o valor da produção de leite não é o único dado disponível em uma fazenda produtora de leite. Contudo, pode ser considerado como o mais importante, pois todos os demais dados gerados e demais ações realizadas na fazenda são focadas em aumentar ao máximo a produção de leite.

Entretanto, grande parte das fazendas produtoras de leite não mantém um histórico dos dados do animal e não utilizam as características do DW para permitir a análise do histórico de dados de forma homogênea, centralizada e não-volátil. Ao contrario, fazem anotações manuais de cada vaca em cadernos ou papeis avulso. Esse processo manual de coleta de dados é suscetível a erros, falhas de obtenção do valor de produção do leite, não histórico e altamente volátil. É comum encontrar fazendas nas quais tem de 200 a 300 vacas em lactação. Contudo, o valor de leite produzido de cada vaca em cada período, quando coletado, é anotado em cadernos e certamente esquecido com o tempo. Isso equivale a dizer que, caso ocorra alguma anomalia com algum animal e consequentemente ocorra uma perda na produção de leite, o produtor só toma conhecimento de tais fatos dias depois do fato ocorrido. Esse fato negativo afeta diretamente a produção de leite da fazenda, a rentabilidade financeira da fazenda e, no pior caso, no descarte do animal.

Dado esse cenário de fazendas produtoras de leite, foi possível vislumbrar a aplicação do Imã de Dados no domínio de pecuária leiteira. Contudo, é importante destacar que não foi possível encontrar uma estrutura adequada nesse domínio para que os experimentos pudessem ser realizados. Nenhuma propriedade rural, que seja de conhecimento dos envolvidos neste trabalho, possui uma estrutura que permita obter os dados de volume de leite automaticamente e em tempo real. Isso se deve ao fato de que as propriedades rurais, sejam elas grandes ou pequenas, não fazem a coleta do volume de leite produzido de forma automática e/ou por meio de sensores. Além do mais, até existem equipamentos que disponibilizam os dados de volume de leite automaticamente. Contudo, três fatores puderam ser observados: 1) esse processo exige que o produtor tome nota desse valor de volume de forma manual. Isso faz com que ocorram erros de digitação, falhas no processo de obtenção desse dado e, no pior caso, não tome nota do valor; 2) outros tipos de dispositivos que disponibilizam os dados automaticamente não são acessíveis por outros dispositivos externos; 3) algumas ordenhadeiras até possuem uma estrutura pronta para coletar os dados automaticamente. Contudo, o alto custo dos equipamentos para se agregar a ordenhadeira para permitir essa coleta automática de dados inviabiliza o processo.

Todos esses fatores negativos fizeram com que fosse criada uma estrutura IdC composta por sensores e microcontroladores semelhante a já existente em algumas fazendas, para realizar os experimentos de validação do Imã de Dados. Dessa forma, os seguintes passos foram considerados para iniciar a construção do ambiente: em um ambiente de pecuária leiteira, o processo

de coleta do leite ocorre da seguinte forma: no curral da fazenda existe o local chamado de sala de ordenha. Esse local é onde a vaca entra para que seja realizada a coleta do leite. Esse local é composto pela ordenhadeira (equipamento responsável por coletar o leite), produtos de limpeza dos animais e quaisquer outros equipamentos necessários para o processo de ordenha (processo que obtém o leite da vaca). Assim que uma vaca entra na sala de ordenha, o produtor conecta um par de teteiras ao teto do animal. Pode-se realizar várias coletas ao mesmo tempo, dependendo do modelo da ordenhadeira. Em seguida, liga-se a ordenhadeira e inicia-se o processo de ordenha. O leite é obtido com o auxilio de uma bomba a vácuo, a qual suga o leite do teto do animal. Após esse processo de obtenção do leite, o mesmo é imediatamente armazenado em um tanque de resfriamento. Após o envio do leite para o tanque de resfriamento, o valor do volume do leite pode ser obtido de forma manual por meio de um dispositivo (visor) acoplado à ordenhadeira, indiretamente por meio de uma balança (medidor) ou automaticamente por meio de um medidor digital. Após ordenhar todas as vacas, todos os equipamentos envolvidos no processo de ordenha são lavados e higienizados com produtos específicos para, além de realizar a limpeza dos equipamentos, retirar qualquer impureza contida nos equipamentos.

De acordo com os próprios pecuaristas e veterinários, vários fatores são importantes no manejo de uma propriedade rural leiteira, tais como valores referentes a nutrição animal, valores que indicam a situação fisiológica do animal e valores que indicam a situação psicológica do animal. Todos esses valores são diretamente relacionados a produção de leite. Se algum desses valores sofrer grandes e repentinas alterações, toda a produção leiteira é afetada. Contudo, para obter todos esses dados automaticamente, se torna oneroso e inviável para grande parte dos produtores. Isso ocorre pois até existem sistemas de ordenhadeiras semiautomáticas (ordenha, sensores, software de gerenciamento e demais equipamentos). Porém, esses equipamentos podem custar desde R$ 40.000,00 até R$ 500.000,00, e grande parte dos produtores de leite não tem condições financeiras de arcar com esses equipamentos devido ao alto custo. Por outro lado, esses mesmos produtores necessitam de um mínimo e aceitável controle automatizado em sua propriedade, sobretudo a obtenção do valor de produção do leite. Com isso, do ponto de vista do custo-benefício, obter o valor da produção do leite automaticamente e em tempo real se torna o dado mais relevante de uma fazenda produtora de leite. Isso se deve ao fato de que, caso um animal apresente uma produção de leite inesperada para tal momento, o produtor pode analisar esse dado imediatamente e tomar as devidas decisões.

A partir do contexto de pecuária leiteira apresentado, após reuniões com o orientador e um especialista da área de pecuária leiteira, foi possível viabilizar a aplicação do Imã de Dados nesse domínio. Do ponto de vista do orientador deste trabalho, é altamente relevante aplicar o Imã de Dados nesse domínio, pois suas características são diretamente relacionadas ao tema deste trabalho. Esse domínio necessita de dados extraídos e carregados para um DW em tempo real para servir aos pecuaristas e veterinários tomarem as decisões em tempo real. Ademais, a não volatilidade e histórico dos dados de cada vaca se torna crucial na obtenção de dados para a tomada de decisão estratégica na fazenda. Assim, essas características são adequadas para

aplicar o processo ETL em tempo real para extrair os dados, assim como são adequadas para o DW manter o histórico dos dados. Do ponto de vista do especialista da área, a aplicação do Imã de Dados nesse domínio é viável pois a obtenção dos dados em tempo real auxilia o produtor a tomar decisões em tempo real. Além disso, segundo o próprio especialista, é um tema no qual outros pesquisadores também estão em busca de novas investigações e soluções.

Embora a aplicação desse experimento seja apenas para verificar a viabilidade de se aplicar o Imã de Dados em um ambiente com dados reais, já se torna altamente relevante, tanto para o produtor obter o valor do volume de leite em tempo real, quanto para a fase de experimentação e validação deste trabalho. Isso se deve ao fato de que, de acordo com o orientador e o especialista, do ponto de vista comercial, uma alternativa mais acessível pode fazer com que o produtor que não tem condições de arcar com o alto custo dos equipamentos atuais possa ter o controle da produção de leite em tempo real e com baixo custo.

6.2 Ambiente IdC para a pecuária leiteira

Após definir o domínio e o cenário para aplicar o Imã de Dados, foi feita uma análise em algumas propriedades rurais para verificar a viabilidade da implantação do Imã de Dados em tal ambiente. Dentre as propriedades analisadas, a fazenda Bom Jardim no município de Jataí-GO foi adotada para realizar o experimento com dados reais. A fazenda possui uma estrutura moderna, na qual a ordenhadeira está preparada para disponibilizar os dados de produção de leite individualmente. Contudo, dois aspectos negativos foram analisados: 1) devido o alto custo dos sensores exigidos pela ordenhadeira, os proprietários optaram por não adquirir; 2) mesmo se os proprietários adquirissem os sensores, tais equipamentos não são de código aberto, além de todos os equipamentos e *software* ser proprietários. Isso faz com que se torne difícil ou impossível o acesso aos equipamentos para obter os dados por meio do Imã de Dados.

Dado esse contexto, foi necessário criar um ambiente IdC semelhante ao identificado na tal propriedade. Esse ambiente construído considerou as seguintes características:

- Cada vaca tem um brinco (*Tag* RFID) contendo um número de identificação único.
- A sala de ordenha comporta várias vacas (um conjunto de vacas é chamado de lote) para serem ordenhadas de uma vez só.
- Cada vaca tem um local específico na sala de ordenha.
- Esse local específico é composto por um conjunto de teteira e um medidor digital.
- Ao término da ordenha de um lote, o mesmo é retirado da sala de ordenha por inteiro e outro lote é encaminhado.
- Ao saírem da sala de ordenha, as vacas saem de forma individual, uma atrás da outra, por uma saída única.

6.2.1 Equipamentos utilizados

A partir desse cenário apresentado, foi possível analisar todos os equipamentos necessários para serem implantados nesse ambiente. Assim, os equipamentos utilizados, bem como suas quantidades, foram os seguintes:

- 3 - Microcontroladores ESP32 LoRa Healtec.
- 1 - Sensor de presença de curta distância (até 60 cm) Viaonda.
- 1 - Sensor de presença de media distância (até 2.5m) Viaonda.
- 3 - Brincos RFID EPC-GEN2.
- 2 - Sensores de fluxo YF-S403.

O ESP32 LoRa é um microcontrolador responsável por gerenciar todos os dados que são gerados pelos sensores. Isso quer dizer que cada sensor, tanto de presença quanto o de fluxo, é acoplado a um ESP32, de modo que o sensor seja responsável por gerar o dado e o microcontrolador seja responsável por gerenciar o dado gerado. Esse gerenciamento inclui as seguintes tarefas: 1) capturar os dados gerados pelos sensores; 2) manter temporariamente os dados obtidos pelos sensores; 3) enviar os dados obtidos para o *broker* do sistema pub/sub. Um aspecto positivo do microcontrolador ESP32 é que ele consegue se conectar à Internet e enviar e receber dados. Uma vez que o microcontrolador ESP32 recebeu um dado de um sensor, o mesmo é capaz de enviar tal dado para um repositório de dados, por exemplo, para um banco de dados na nuvem. Para este experimento, foi utilizado um ESP32 para gerenciar os dados do sensor de curta distância, outro ESP32 para gerenciar os dados do sensor de média distância e outro ESP32 para enviar os dados para o *broker*.

O sensor de presença de curta distância foi responsável por detectar quando uma vaca entra na sala de ordenha. Para simular com exatidão o ambiente da fazenda analisada, seria necessário a aquisição de 16 sensores de curta distância e 16 ESP32. Isso porque a ordenhadeira da fazenda é capaz de ordenhar até 16 vacas ao mesmo tempo. Contudo, devido ao alto custo, os testes foram realizados apenas com 2 sensores e 3 ESP32. O sensor de curta distância foi fixado em um local no qual a vaca fique parada no momento da ordenha. Já o sensor de presença de média distância foi responsável por detectar quando uma vaca sai da sala de ordenha. Esse sensor foi fixado em um ponto estratégico no qual detecta o momento exato em que a vaca sai da sala de ordenha e não haja sua possibilidade de retorno. Isso ocorre porque a vaca pode sofrer algum tipo de estresse durante a ordenha e, em alguns casos, querer sair da sala de ordenha por outro local que não seja o correto. Nesses casos, o próprio produtor deve intervir e fazer com que a vaca saia pelo local correto.

Os brincos RFID foram colocados na orelha de cada vaca. O rebanho em lactação total (vacas aptas para tirar leite) é composto por aproximadamente 300 vacas. Contudo, foram

implantados brincos RFID em apenas 3 vacas. Esses brincos RFID, como o nome sugere, são dispositivos desenvolvidos no formato de brincos e equipados com a tecnologia RFID. Além de manter um número de identificação único do dispositivo, eles permitem ser detectados por algum sensor de presença. Assim, visto que cada vaca tem seu próprio brinco RFID, implica em dizer que cada vaca tem seu número de identificação único. Esse aspecto faz com que uma vaca seja ordenhada em um local no qual está fixado o conjunto C1 composto pelo ESP32 e o sensor de presença de curta distância. É importante destacar que a ordenhadeira da fazenda analisada é capaz de ordenhar até 16 animais por vez. Contudo, foram utilizados apenas um conjunto C1 e também apenas 3 brincos. Logo, para simular o ambiente, o produtor deve conduzir uma vaca que esteja com o brinco RFID para o local no qual está fixado o conjunto C1.

Por fim, o sensor de fluxo foi utilizado para coletar os dados do leite, o qual foi conectado ao teto da vaca. Esse sensor foi fixado no mesmo local no qual foram fixados o conjunto C1. Isso se deve ao fato de que esse ESP32 deve ser capaz de gerenciar o sensor de presença e o sensor de fluxo. Para mostrar como os equipamentos foram montados no curral, as próximas ilustrações mostram a distribuição dos equipamentos no ambiente que envolve o curral e o ponto de Internet.

Figura 19 – Ponto de WiFi montado para enviar sinal de Internet para o curral

(a) Perspectiva para o curral.

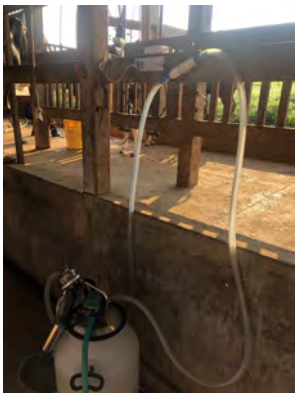
(b) Perspectiva para o ponto central de Internet.

A Figura 19 mostra como foi montado o roteador para estender o sinal de Internet para o curral. Na propriedade onde foi aplicado o experimento, o sinal de Internet é centralizado na sede da fazenda. Contudo, o curral fica aproximadamente 60 metros de distância do ponto central, e essa distância faz com que inviabilize a conexão à Internet por meio do ESP32. Para resolver esse problema, o roteador foi colocado há aproximadamente 20 metros do ponto central. Esse local é composto por um ponto de energia para alimentar o roteador. Além disso, é livre de qualquer obstáculo que possa atrapalhar o sinal de Internet até o curral.

A Figura 20 mostra como ficou montado e fixado o ESP32 (a) (neste caso, chamado de ESP32 *Slave*, pois é utilizado apenas para gerenciar os dados da vaca quando entra na sala de ordenha), o sensor de curta distância (a), o sensor de fluxo (a) e a teteira (c) na sala de ordenha (d). É importante destacar que a sala de ordenha é composta por um galão de leite de 50 litros (em outras propriedades, tem-se um resfriador de leite de até 10 mil litros), o qual é responsável por manter temporariamente o leite que é retirado das vacas. Tanto o sensor de curta distância

Figura 20 – Conjunto ESP32, sensor de presença de curta distância, sensor de fluxo, teteira e sala de ordenha

(a) Conjunto ESP32 + Sensores.



(b) Conjunto ESP32 + Sensores ampliada.

(c) Teteira que é conectada no teto da vaca.

(d) Sala de ordenha.

quanto o sensor de fluxo foram conectados ao ESP32 por meio das portas lógicas do próprio ESP32. Especificamente o sensor de fluxo, além de ser conectado ao ESP32, é conectado ao galão de leite. Isso se deve ao fato de que o leite sai da teteira, passa pelo sensor de fluxo e cai diretamente no galão de leite. Dessa forma, assim que o leite passa pelo sensor de fluxo, o valor do volume de leite é obtido e gerenciado pelo ESP32.

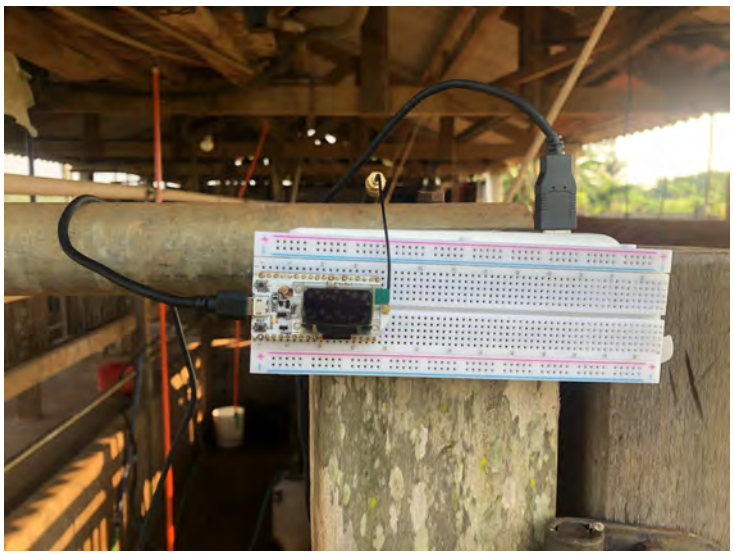
É importante dizer que sempre que uma vaca entra na sala de ordenha, a vaca é detectada por meio do sensor de curta distância. Esse sensor foi posicionado de forma estratégica para permitir detectar o brinco RFID conectado na vaca sem a intervenção humana. Assim que o sensor de curta distância detectar o brinco RFID, o ESP32 está pronto para receber e gerenciar os dados gerados pelo sensor de fluxo.

A Figura 21 mostra como foi montado o conjunto ESP32 e sensor de média distância (neste caso, chamado de ESP32 *Master*, pois identifica a saída da vaca da sala de ordenha e encerra o processo de gerenciamento de dados do leite). Esse sensor é responsável por detectar o momento em que uma vaca sai da sala de ordenha. Essa detecção é importante para identificar que uma determinada vaca finalizou sua ordenha e o valor da produção do leite possa ser enviado para o *broker*. O sensor foi fixado em um local na saída da sala de ordenha. Além disso, após a

Figura 21 – Conjunto ESP32 e sensor de média distância

vaca passar por esse local, não há mais a possibilidade de retorno para a sala de ordenha.

Figura 22 – ESP32 responsável por enviar os dados para o *broker*



A Figura 22 mostra como ficou fixado o ESP32 responsável por enviar os dados para o *broker* (neste caso, chamado de ESP32 *Sender*, pois é responsável por enviar os dados para o *broker*). Ele foi fixado no inicio na sala de ordenha, pois é o ponto ideal para se comunicar com os demais ESP32 do curral e se conectar a Internet que, por sua vez, vem do roteador mostrado na Figura 19. Assim que uma vaca entra na sala de ordenha, a vaca é detectada pelo sensor de presença de curta distância. A medida que seu leite é retirado, o sensor de fluxo em conjunto com o ESP32 coleta esse valor. Assim que a vaca finaliza sua ordenha e sai da sala de ordenha, a vaca é detectada pelo sensor de média distância. Após identificar a saída, o ESP32 *Master*

envia o número do brinco para o *Slave* para que ele possa registrar e limpar os dados da vaca que acabou de deixar a sala de ordenha. Após esse processo, o número do brinco da vaca e o volume de leite produzido é enviado do *Slave* para o *Sender*. Por sua vez, o *Sender* integra o número do brinco, o volume de leite produzido, a data, a hora e o período em um arquivo JSON e envia-o para o *broker*.

É importante destacar que o ESP32 *Sender* se mantém conectado a Internet durante todo o período da ordenha. Isso quer dizer que o ambiente deve ser equipado com um sinal de Internet estável e sem interferência externa. Além disso, caso ocorra alguma queda de energia ou no sinal da Internet, o ESP32 *Sender* é capaz de se reconectar automaticamente à Internet e aguardar o recebimento de novos dados por parte do ESP32 *Slave*.

Figura 23 – Computador conectado à Internet e com o Imã de Dados em operação

Por fim, a Figura 23 mostra um computador conectado à Internet e com o Imã de Dados em execução. Ao conectar-se à Internet e ao *broker*, o ESP32 *Sender* assume o papel de *publisher*, isto é, o ESP32 *Sender* publica novos dados no *broker* para que possam ser consumidos pelos *subscribers*. Por sua vez, o Imã de Dados, ao conectar-se à Internet e também ao *broker*, assume o papel de *subscriber*, ou seja, se inscreve ao *broker* para receber os dados publicados por um *publisher* específico. Assim, uma vez que o ESP32 *Sender* publica dados de interesse, o Imã de Dados pega esses dados publicados e imediatamente armazena-os no DW.

É importante destacar que o Imã de Dados, assim como o ESP32 *Sender*, fica conectado à Internet durante todo o período da ordenha. Além disso, assim como o ESP32 *Sender* também, caso seja perdida a conexão com a Internet, o mesmo se conecta automaticamente assim que o sinal for restabelecido.

6.2.2 Tecnologias computacionais utilizadas

Além dos equipamentos físicos necessários para a construção do ambiente, algumas tecnologias computacionais (*software*, bases de dados e outras tecnologias computacionais) foram utilizadas a fim de construir o ambiente que permita a execução do experimento. As tecnologias computacionais são:

6.2.2.1 Internet

O acesso a Internet em fazendas se tornou cada vez mais comum nos últimos anos. Além disso, principalmente com a difusão do conceito de agropecuária digital, a Internet se tornou uma tecnologia indispensável para tal domínio. Para realizar os experimentos, a conexão com a Internet foi um requisito não funcional essencial. Isso se deve ao fato de que é por meio da Internet que os dados gerados pelos sensores e mantidos nos microcontroladores são enviados para uma base de dados na nuvem e posteriormente disponíveis para consultas analíticas.

6.2.2.2 Modelo pub/sub em um ambiente de pecuária leiteira

Como já discutido no Capítulo 2, um modelo pub/sub é responsável por permitir que diversos dispositivos heterogêneos de origem enviem dados para diversos dispositivos heterogêneos de destino. Além disso, um modelo pub/sub permite integrar sistemas heterogêneos, no qual é composto por dispositivos de origem, os quais são chamados de *publishers*, isto é, geram/publicam dados para um tópico armazenado em um servidor de dados. Por outro lado, os dispositivos de destino são chamados de *subscribers*, os quais se inscrevem (demonstram interesse) em tópicos armazenados nesse servidor de dados para que possam consumir tais dados.

No domínio de pecuária leiteira, essa tecnologia se torna uma maneira altamente relevante de integrar o ambiente operacional (ambiente da sala de ordenha composta pelos ESP32, sensores e microcontroladores) e o ambiente informacional (DW e o próprio produtor com um aplicativo ou uma página *Web* que consome os dados de volume de leite). Para o restante dos experimentos, será considerado apenas o uso de um aplicativo. Nesse domínio, o ambiente na sala de ordenha é considerado *publisher*, ou seja, o microcontrolador ESP32, o sensor de presença de curta distância e o conjunto de teteira geram os dados de interesse para o produtor. Já o produtor, responsável por tomar as decisões, ao fazer uso de um aplicativo que consome esses dados gerados é considerado *subscriber*.

Ao utilizar um modelo pub/sub, além do *publisher* e *subscriber*, é necessário utilizar um *broker* para receber os dados do *publisher* e avisar aos *subscribers* que existem novos dados a serem consumidos (como foi descrito no Capítulo 2). Para realizar os experimentos, foi utilizada a implementação Mosquitto, a qual é um modelo pub/sub disponibilizado pela Eclipse (disponível em mqtt.eclipse.org). Esse serviço fornece um *broker*, por meio do qual é possível que *publishers* gerem dados para que os *subscribers* possam consumi-los. Além disso, esse serviço não teve custo devido o projeto ser apenas experimental. Além do mais, algumas

API's são fornecidas para ser utilizadas pelos desenvolvedores. Assim, as tarefas de publicar e consumir dados são feitas com o auxilio de uma API fornecida pelo próprio Eclipse.

É importante destacar que esse serviço não permite o acesso direto ao *broker* para que possa ser visualizado os dados enviados pelo *publisher*. Isso quer dizer que é necessário algum meio (*software*, uma API ou outros meios de visualização de dados) para visualizar os dados enviados pelo *publisher*. Além disso, mesmo após a leitura da documentação do serviço, não foi possível identificar qual *framework* pub/sub que compõe esse serviço, o qual gerencia o envio e recebimento dos dados. Tal *framework* poderia ser RabbitMQ ou Kafka, que são *frameworks* de processamento de dados em um ambiente pub/sub. Independentemente do *framework* utilizado, esse modelo pub/sub foi considerado adequado para a condução dos experimento, visto que ele permite executar as tarefas baseando-se no modelo pub/sub.

6.2.2.3 Sistema gerenciador de banco de dados MySQL

Como já explicado ao longo de todo o trabalho, o DW será utilizado para armazenar os dados produzidos pelo ambiente operacional. Logo, para simular a base de dados que representa o DW no ambiente de pecuária leiteira, foi utilizado o sistema gerenciador de banco de dados MySQL na nuvem. Esse serviço é pago e fornecido pela empresa WebLink. É fácil imaginar que pudesse ser utilizado o MySQL localmente. Contudo, a intenção de utilizar esse serviço na nuvem foi para simular o ambiente no qual existe um DW externo ao ambiente operacional. Dessa forma, com o banco de dados na nuvem, o ambiente informacional se torna mais real e funcional. Isso se deve ao fato de ser possível realizar testes de desempenho, prever certas anomalias ao gravar um dado na nuvem que foi gerado no ambiente operacional e permitir o acesso externo por meio de um aplicativo ou página *Web*.

Além do DW, o Imã de Dados também necessita de um banco de dados para que suas configurações possam ser armazenadas. As configurações necessárias foram explicadas e definidas na seção 5.3.3. Essas configurações do Imã de Dados devem ser persistidas para que, sempre que necessário, seja possível recuperá-las. Assim, para as configurações do Imã de Dados, também foi adotado o MySQL, porém de forma local. Isso se deve ao fato de simular a utilização de outro banco de dados diferente a do DW (de acordo com as boas práticas de DW, deve-se ter um banco de dados exclusivo para o DW). Além disso, não é essencial a manutenção desses dados de configuração na nuvem, pois eles não são acessados externamente por terceiros. Além do mais, essas configurações são restritas a apenas usuários administradores, logo, é suficiente e adequada a manutenção desses dados localmente.

6.2.2.4 Imã de Dados

Como o próprio trabalho sugere, o Imã de Dados deve ser utilizado para extrair os dados em tempo real do ambiente operacional e armazená-los no DW para posteriormente servir como fonte de dados para as consultas analíticas. Dessa forma, o Imã de Dados deve ser configurado

para que possa realizar as tarefas para as quais ele foi projetado. Para realizar essa configuração, foi desenvolvido um *software* configurador, o qual permite que o usuário administrador interaja com o Imã de Dados do ponto de vista de configuração. Uma vez configurado, o Imã de Dados é capaz de ser executado de forma autônoma para realizar as tarefas de extração e armazenamento de dados em tempo real no DW.

6.2.2.5 Aplicativo para monitoramento da produção de leite em tempo real

Ainda em relação ao tema deste trabalho, além de extrair os dados em tempo real do ambiente operacional, esses mesmos dados devem ser disponibilizados ao usuário em tempo real. Para isso, foi desenvolvido um aplicativo chamado Monitoramento de Volume de Leite em Tempo Real (MVLTR), o qual foi baseado na plataforma iOS. Sua responsabilidade é monitorar o DW no ambiente informacional e receber em tempo real os novos dados nele armazenados. Sempre que ocorrer uma nova geração de dado por parte dos sensores na sala de ordenha, é suficiente o aplicativo aberto para que o usuário tenha acesso aos novos dados em tempo real.

6.2.2.6 Linguagens de programação

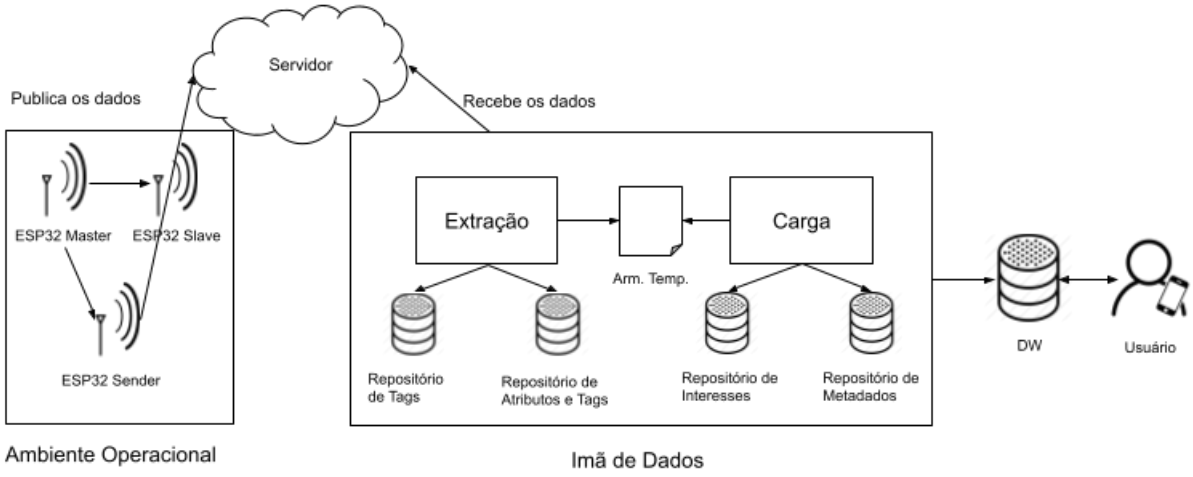
Ao longo deste trabalho foi necessário desenvolver alguns produtos de *software* para que todo o Imã de Dados pudesse ser executado. Os produtos desenvolvidos foram: 1) Imã de Dados; 2) Configurador do Imã de Dados; 3) Ambiente IdC composto por sensores e microcontroladores; 4) Aplicativo MVLTR; 5) Gerador de Dados Sintéticos (GDS) (explicado na seção 6.3.5); 6) Gerador de Dados Sintéticos para Gatilhos (GDSG) (explicado na seção 6.3.5). Cada produto de *software* criado tem uma finalidade específica e foi desenvolvido a partir de uma linguagem de programação específica.

Portanto, esses produtos foram desenvolvidos a partir das seguintes linguagens de programação: 1) Para o ambiente IdC, a linguagem de programação utilizada foi C++. Isso se deve ao fato de que os sensores e os microcontroladores são baseados em C++. Essa linguagem é considerada de baixo nível, atuando diretamente no microcontrolador sem a necessidade de um *middleware* ou outro meio de acesso entre a linguagem de programação e o microcontrolador. Além do mais, foi utilizada a IDE (*Integrated Development Environment*) de desenvolvimento VS Code (Visual Studio Code); 2) Já para o Imã de Dados, para o Configurador do Imã de Dados, para o GDS e para o GDSG foi utilizada a linguagem de programação Java. Essa linguagem de programação foi adotada devido a alta aceitação no desenvolvimento de *software*, ser adequada para este trabalho e pelo conhecimento dos envolvidos. A IDE de desenvolvimento utilizada foi a IntelliJ; 3) Por fim, o aplicativo MVLTR foi desenvolvido sobre a linguagem de programação *Swift*. Essa linguagem de programação é exclusivamente focada em desenvolvimento de aplicativos para iOS. Além do mais, a IDE utilizada para esse desenvolvimento foi a XCode.

6.3 Aplicação do Imã de Dados no ambiente de pecuária leiteira

Após apontar todos os equipamentos e tecnologias utilizadas para a construção do ambiente IdC para a pecuária leiteira, é o momento de detalhar a aplicação do Imã de Dados em tal ambiente. Vale destacar que esse detalhamento inclui sua configuração, o esquema das tabelas responsáveis por manter essas configurações, o fluxo de dados desde a geração do dado por meio dos sensores até a visualização desse dado pelo usuário por meio do aplicativo MVLTR. Além disso, será detalhado o esquema das tabelas que representam o DW, assim como sua configuração e formas de acesso por meio do Imã de Dados.

Figura 24 – Ciclo de vida da aplicação do Imã de Dados



Fonte: Próprio autor

Para viabilizar e facilitar a explicação da aplicação do Imã de Dados em um ambiente real, é necessário entender o fluxo de dados do ambiente real. A Figura 24 mostra de forma abstrata o cenário no qual foi aplicado o Imã de Dados. De um lado tem-se o ambiente operacional composto por sensores, e estes acoplados em um ESP32. Esses sensores produzem dados, os quais são enviados para o *broker* por meio do ESP32 *Sender*. Do outro lado, tem-se o Imã de Dados, o qual recebe os dados publicados, realiza suas tarefas e armazena os dados no DW.

6.3.1 Configuração do Imã de Dados

Inicialmente, o usuário administrador deve identificar nas entidades (tabela, arquivo JSON, arquivo XML ou qualquer que seja o local) que armazenam os dados de produção de leite, quais dados são de interesse para as tomada de decisão (número do brinco da vaca, volume do leite produzido, data, hora e período). Vale destacar que, no caso deste experimento, essa análise foi feita a partir do arquivo JSON gerado pelos microcontroladores. Esse arquivo JSON contém

os dados de produção de leite e é gerado pelo ESP32 *Sender* logo após o término da ordenha de cada vaca.

É importante destacar que os dados gerados pelos microcontroladores poderiam ter sido gerados em outro tipo de arquivo diferente de JSON, tais como XML, arquivo texto ou outro tipo de arquivo. A escolha pelo JSON foi devido a alta aceitação desse formato de arquivo na integração de sistemas e pelo fato desse formato ser semiestruturado. Isso faz com que seja possível criar estruturas de códigos que seja possível facilmente encontrar atributos e valores para os atributos. Um exemplo para a utilização desse formato de arquivo é o Kafka, em que faz o envio e recebimento de dados por meio de arquivos JSON.

O arquivo JSON no qual é disponibilizado os dados de produção de leite tem a seguinte estrutura:

{"milk_id_animal":"226052181542400623817420 7",
"milk_milk_volume":"5",
"milk_time_started":"08:30",
"milk_date":"2020-09-09",
"milk_time":"11:57:53",
"milk_period":"2",
"time_arrive":"17:00",
"EntityName":"MilkYield"}

Os atributos disponibilizados no arquivo JSON foram os seguintes: *milk_id_animal*, o qual representa o número do brinco do animal. Esse valor é obtido por meio do sensor de presença de curta distância assim que a vaca entra na sala de ordenha. O atributo *milk_milk_volume* é obtido pelo sensor de fluxo assim que finaliza o processo de ordenha. Os atributos *milk_time_started* e *milk_date* representam, respectivamente, a hora na qual iniciou a ordenha e a data da ordenha. Esses dados são obtidos por meio do microcontrolador assim que inicia o processo de ordenha. O atributo *milk_time* representa a hora que finalizou a ordenha. Esse valor é obtido por meio do microcontrolador assim que a vaca deixa a sala de ordenha. O atributo *milk_period* indica qual o período do dia (manhã, tarde ou noite, ou período 1, período 2 ou período 3) ocorreu a lactação. Esse valor é obtido por meio do microcontrolador assim que a vaca deixa a sala de ordenha e segue a seguinte lógica: se a hora da saída é entre 4h e 7h59, então é indicado o período 1. Se a hora da saída é entre 8h e 15h59, então é indicado o período 2. Se a hora da saída é entre 16h e 23h59, então é indicado o período 3. Diferentemente dos outros atributos que indicam tempo, o atributo *time_arrive* é gerado pelo processo de Carga do Imã de Dados e é agregado ao arquivo JSON posteriormente. Esse atributo indica a hora na qual os dados foram armazenados no DW, ou seja, esse atributo indica a hora na qual os dados chegaram ao destino. Por fim, o atributo *Entityname* indica o nome da entidade na qual ocorreu a geração dos dados.

Como pode ser notado, o Imã de Dados utiliza o nome da entidade e dos atributos contidos no arquivo JSON para identificar os dados de interesse e associá-los a uma *tag*. Portanto,

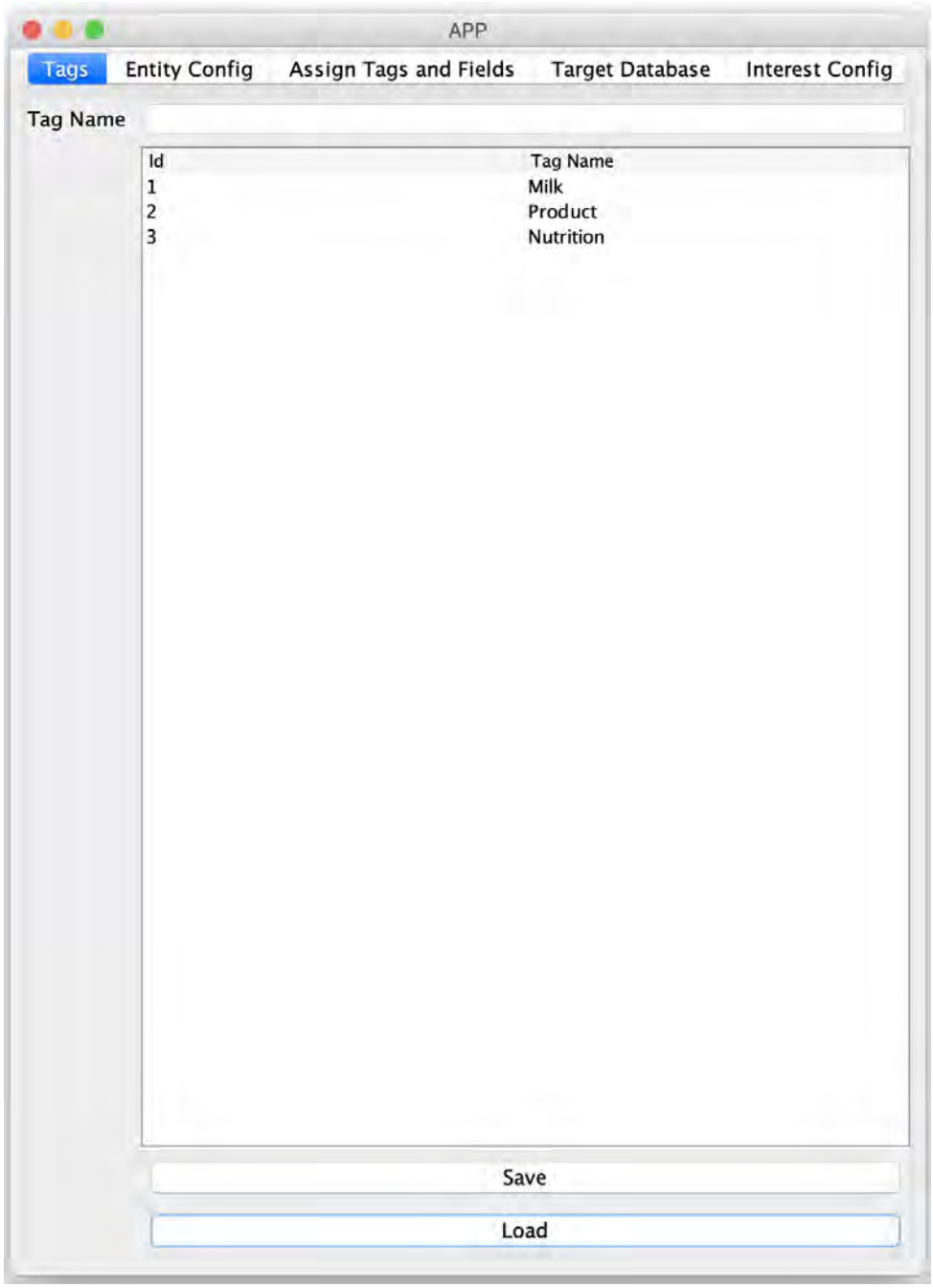
é por meio do atributo *EntityName* que é indicado o nome da entidade que mantém os dados de produção de leite. Vale destacar que o valor desse atributo foi fixado durante o processo de desenvolvimento do ambiente IdC. Em outras palavras, os atributos do arquivo JSON são preenchidos dinamicamente a medida que os dados são gerados pelos sensores na sala de ordenha. Contudo, o atributo *EntityName* é fixado com um nome padrão para qualquer arquivo JSON gerado na sala de ordenha referente a produção de leite. Assim, para esse cenário de obter dados de produção de leite automaticamente, a entidade geradora desses dados sempre será a fixada no atributo *EntityName*. Logo, é o nome dessa entidade que será utilizada para configurar o Imã de Dados juntamente com seus dados de interesse. Ademais, caso outro conjunto de sensores seja criado na fazenda para, por exemplo, disponibilizar dados de temperatura e quantidade de passos das vacas, certamente esses dados serão disponibilizados em outro arquivo JSON. Além do mais, esse outro arquivo JSON será projetado contendo apenas os dados de temperatura e quantidade de passos das vacas. Logo, o valor do atributo *EntityName* desse outro arquivo JSON deverá ser diferente do arquivo JSON gerado pela estrutura de geração de dados de produção de leite. Dessa forma, o usuário administrador poderá configurar os dados de interesse de ambas as entidades (ambos os arquivos JSON).

Após identificar os atributos, o usuário administrador deve realizar a primeira configuração do Imã de Dados. Essa configuração consiste em definir o nome para a *tag*, a qual será associada a esses dados de interesse. Essa configuração deve ser feita por meio do Repositório de Tags. A Figura 25 é composta por 6 sub-figuras, as quais representam respectivamente a configuração das *tags*, a configuração das entidades, a configuração de associação de campos às entidades, a configuração de uma *tag* a um dado de interesse, a configuração dos metadados de conexão aos DW de interesse e a configuração dos DW de interesse.

Após o processo de definição das *tags*, o usuário deve configurar as entidades e o nome dos atributos que mantém os dados de produção de leite e associá-los a uma *tag* anteriormente criada. Para isso, por meio do Repositório de Atributos e Tags da Figura 24, deve-se informar o nome da entidade, o nome do atributo e o nome da *tag* associada a esse atributo. Esse processo deve ser feito para todos os dados de interesse do arquivo JSON que mantém os dados de produção de leite.

Após o processo de identificar os atributos de interesse e associá-los a *tag*, o usuário administrador deve identificar no ambiente informacional, qual base de dados que representa o DW tem interesse nos dados de produção de leite gerados, cuja *tag* foi associada. Para isso, o usuário administrador deve configurar o DW com sua respectiva *tag* de interesse no Repositório de Interesse da Figura 24. Percebe-se que, nesse momento, tem-se uma *tag* associada aos atributos que armazenam dados de produção de leite, e essa mesma *tag* associada a um DW. Esse aspecto faz com que o Imã de Dados seja capaz de identificar, por meio da *tag*, um dado de interesse e ao mesmo tempo identificar qual DW tem interesse nesse dado. Além disso, para que o Imã de dados possa realizar a tarefa de carregamento de dados no DW de forma automática, o usuário

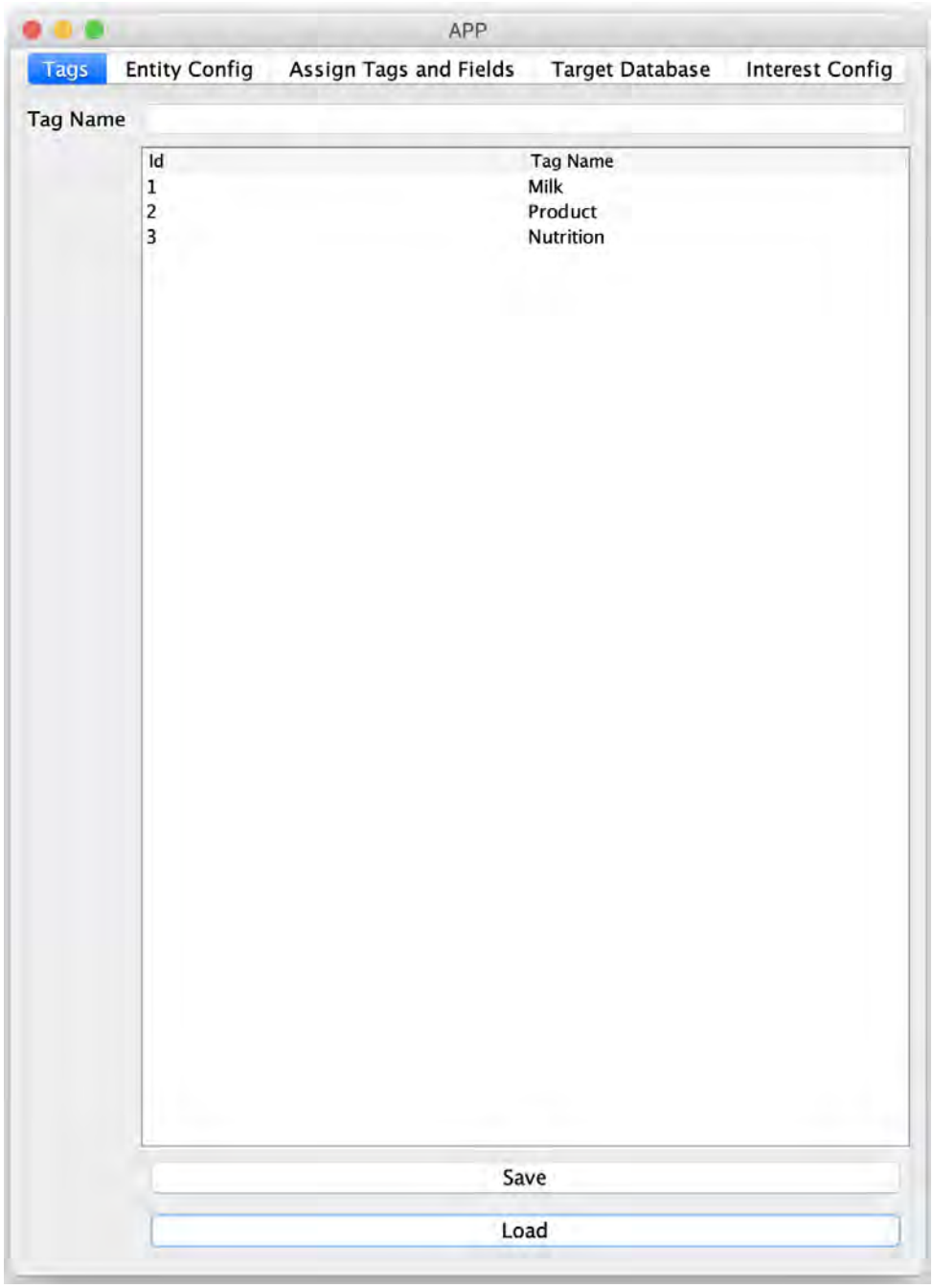
Figura 25 – A - Configuração de Tags



administrador deve configurar os metadados de conexão ao DW no Repositório de Metadados da Figura 24.

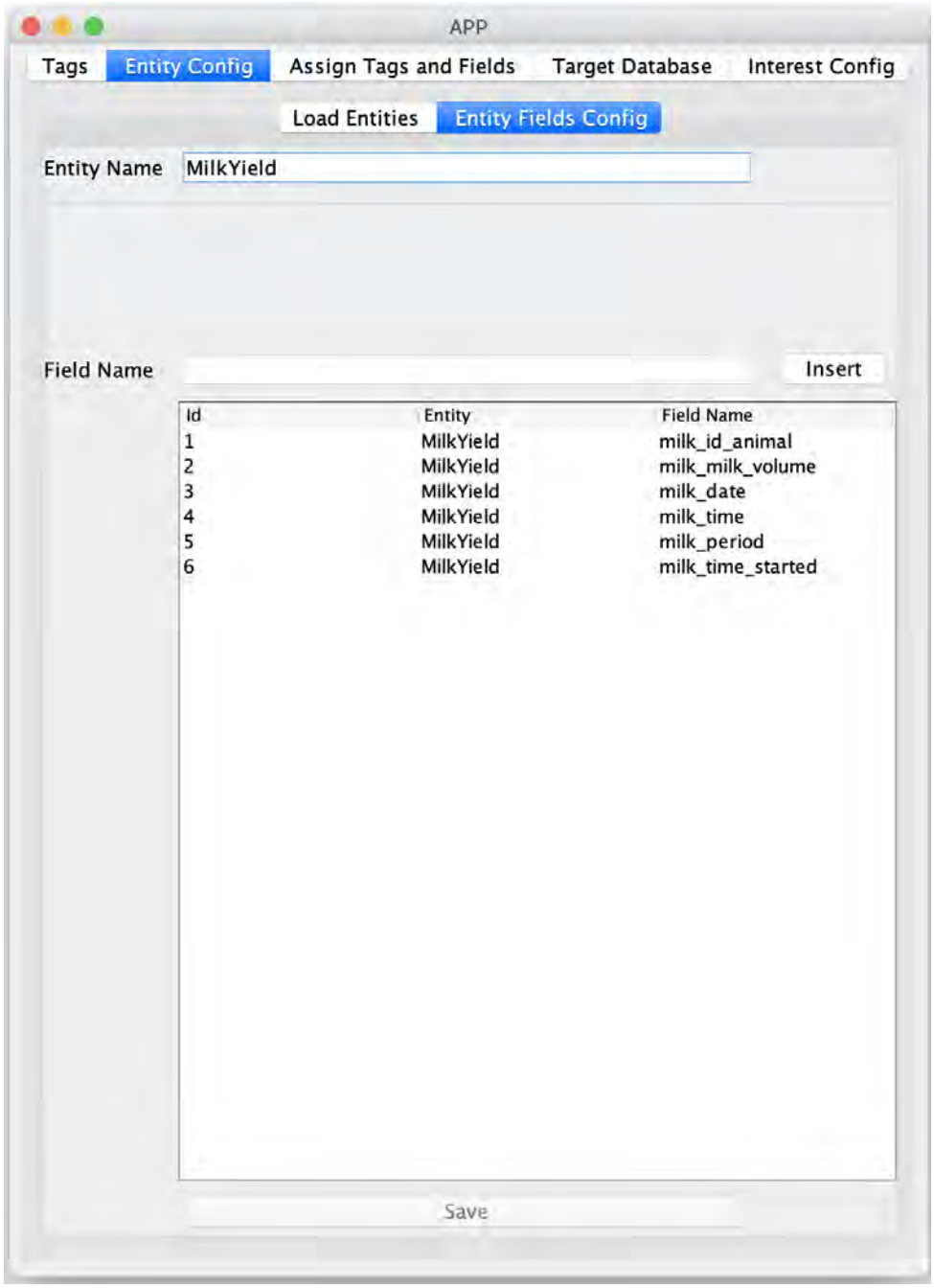
Ainda no ambiente informacional, para simular um DW real, foi necessário criar uma tabela dentro do DW, cujo o nome dos atributos são o mesmo nome dos dados de interesse do arquivo JSON. Isso se deve ao fato de que no processo de carga (explicado adiante), será criada uma *string* de inserção para que os dados de interesse possam ser inseridos no DW. Logo, o nome dos atributos utilizados nessa *string* de inserção serão extraídos dinamicamente do arquivo JSON.

Figura 25 – B - Configuração de Entidades



Em resumo, antes de executar o Imã de Dados no ambiente de pecuária leiteira, o usuário administrador deve realizar todas as configurações citadas para que o Imã de Dados possa ser executado de forma automática e em tempo real. Dessa forma, para mostrar a configuração real do Imã de Dados no ambiente em questão, a Figura 26 mostra o esquema das tabelas responsáveis por armazenar os dados de configuração do Imã de Dados. Vale destacar que essas configurações são armazenadas em tabelas do banco de dados MySQL localmente, de modo que seja acessível ao Imã de Dados sempre que for necessário realizar uma consulta para buscar valores de configurações. Visto que essas configurações não serão disponíveis para acesso externo e serão utilizadas apenas pelo Imã de Dados, torna-se adequado e suficiente o armazenamento

Figura 25 – C - Associação de campos às entidades



dessas configurações localmente.

- ***Tags***: essa tabela é composta pelo atributo Id e pelo atributo *TagName*, o qual é responsável por armazenar o nome da *tag* definida pelo usuário. Na Figura 24, essa tabela é mantida no Repositório de Tags.
- ***Entities***: essa tabela é composta pelo atributo Id e pelo atributo *EntityName*, o qual é responsável por armazenar o nome da entidade mantida no arquivo JSON. Além disso, essa tabela é associada a outras duas tabelas por meio de agregação. Essas duas tabelas são:

Figura 25 – D - Configuração de uma *tag* e um dado de interesse

- ***EntitiesFields***: essa tabela é composta pelo atributo Id, pelo atributo *IdEntity*, o qual faz referência à tabela *Entities* e pelo atributo *FieldName*, o qual é responsável por armazenar o nome dos atributos que compõem a tabela *Entities*. Além disso, essa tabela se relaciona com a tabela *Entities* com a cardinalidade 1..*, isto é, pode-se ter uma entidade com vários atributos.

- ***EntitiesFieldsTags***: essa tabela é composta pelo atributo Id, pelo atributo *IdTag* e pelo atributo *IdEntityField*. Essa tabela é responsável por manter a relação de um atributo do arquivo JSON e uma *tag*. Essa tabela se relaciona com a tabela *EntitiesFields* por meio do

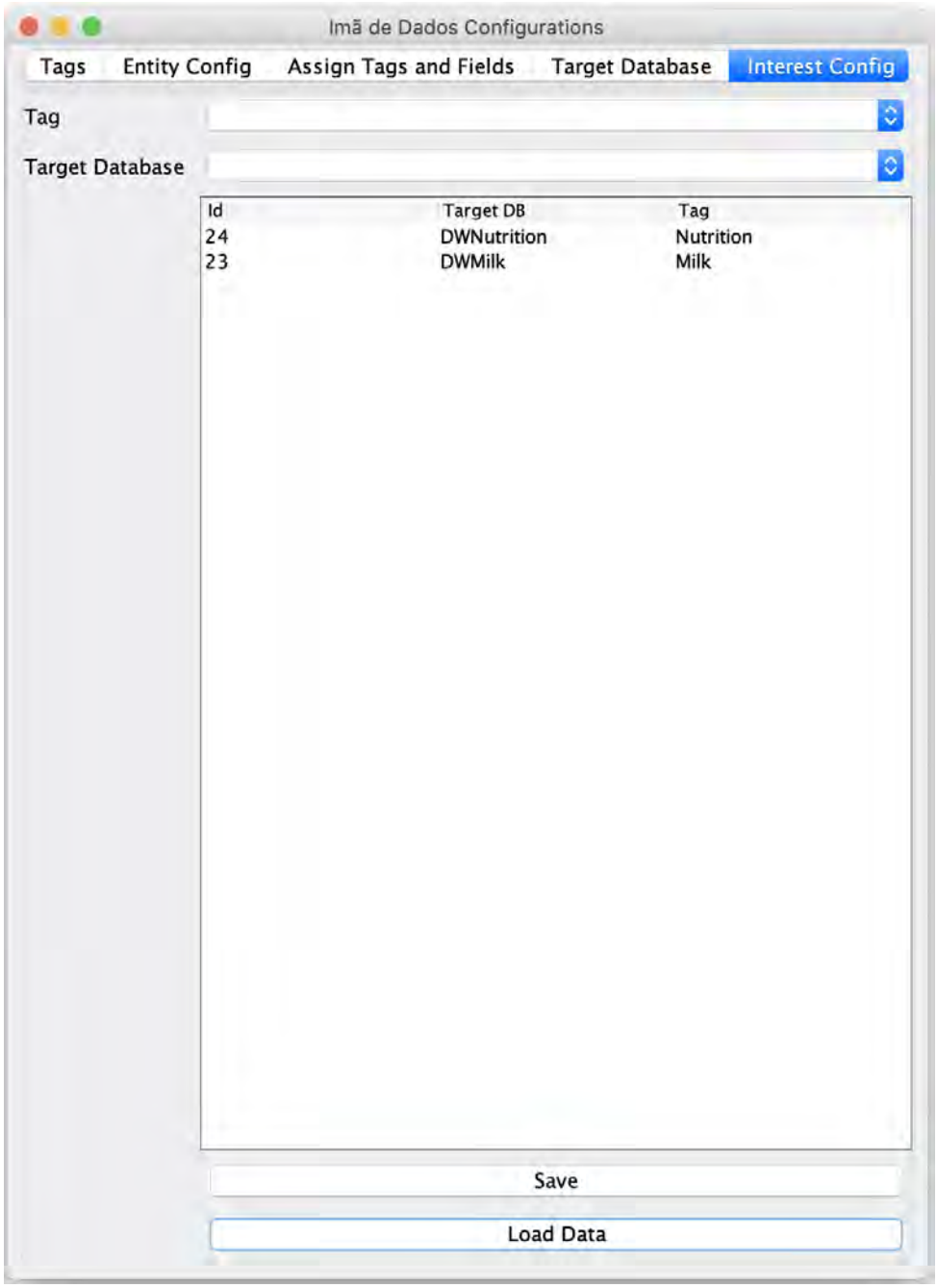
Figura 25 – E - Configuração de metadados de conexão aos DW de interesse

atributo *IdEntityField*. Além disso, sua relação de cardinalidade com a tabela *EntitiesFields* é 1..1, ou seja, uma *tag* pode ser associada a apenas um atributo de uma entidade. Contudo, pode-se ter uma *tag* associada a um mesmo atributo, mas de entidades diferentes.

Vale destacar que na Figura 24, as tabelas *Entities*, *EntitiesFields* e *EntitiesFieldsTags* são representadas pelo Repositório de Atributos e Tags.

- ***TargetDatabaseConfig***: essa tabela é responsável por manter os metadados de conexão às fontes de dados do ambiente informacional, as quais representam o DW. Essa tabela é composta pelo atributo Id e pelos seguintes atributos: *DBNameAlias*, *DBName*, *DBPath*,

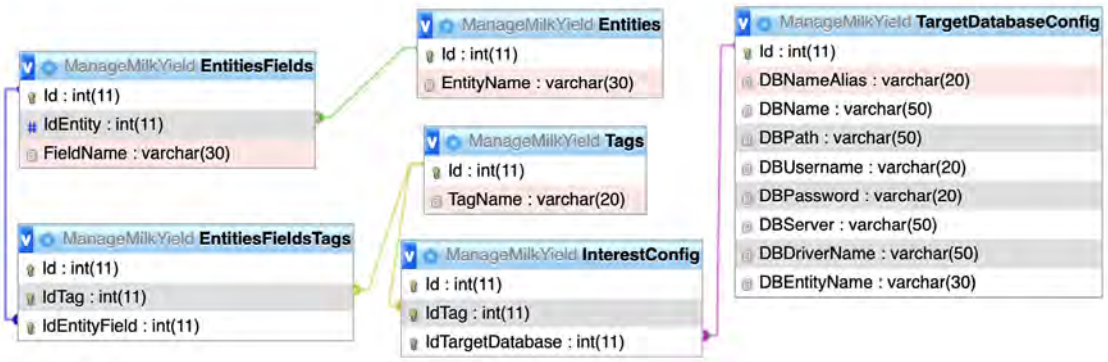
Figura 25 – F - Configuração dos DW de interesse



DBUsername, *DBPassword*, *DBServer*, *DBDriverName*, *DBEntityName*. Esses atributos descrevem os metadados de conexão responsáveis por estabelecer a conexão com tais fontes de dados informacionais. Na Figura 24, essa tabela é representada no Repositório de Metadados.

Vale destacar que o atributo *DBEntityName* é responsável por definir o nome da tabela do DW, a qual armazenará os dados extraídos do arquivo JSON. Isso equivale a dizer que a fonte de dados que representa o DW é composta por apenas uma tabela, sendo essa tabela a responsável por armazenar os dados. Isso se deve ao fato de que, seguindo os conceitos de DW, a fonte de dados que o representa deve ser mantida em um servidor separado de

Figura 26 – Diagrama relacional do conjunto de tabelas de configuração do Imã de Dados



todas as outras fontes de dados. Além disso, para este experimento foi considerado apenas um esquema estrela e desnormalizado (explicado posteriormente). Logo, os dados no DW serão armazenados em apenas uma tabela (apenas em um esquema estrela).

- ***InterestConfig***: essa tabela é composta pelo atributo Id, pelo atributo *IdTag* e pelo atributo *IdTargetDatabase*. Essa tabela é responsável por manter a relação entre uma *tag* e uma fonte de dados que representa o DW, isto é, essa tabela armazena o relacionamento entre uma *tag* e um DW de interesse. Na Figura 24, essa tabela é representada no Repositório de Interesses.

Após o entendimento de como são gerados os dados do arquivo JSON, com base no esquema das tabelas de configuração do Imã de Dados e na análise do arquivo JSON que disponibiliza os dados de produção de leite, os valores de configuração para a execução do Imã de Dados foram os seguintes:

Tabela 7 – Configuração da tabela *Tags*

Tags	
Id	**TagName**
1	Milk
2	Product
3	Nutrition

A Tabela 7 mostra a configuração da tabela *Tags*. O atributo *TagName* mantém o nome das *tags* definidas pelo usuário, os quais são *Milk*, *Product*, *Nutrition*. Vale destacar que as *tags* *Product* e *Nutrition* foram definidas apenas para ilustrar a configuração de *tags*. Dessa forma, ao executar o Imã de Dados, o mesmo irá considerar apenas a *tag Milk* (o porque desse fato será mostrado adiante).

A Tabela 8 mostra a configuração da tabela *TargetDatabaseConfig*. Essa tabela é responsável por armazenar as configurações de metadados de conexão do DW. Alguns metadados listados foram suprimidos devido ao espaço disponível, porém, os atributos que são mantidos

Tabela 8 – Configuração da tabela *TargetDatabaseConfig*

TargetDatabaseConfig					
Id	**DBNameAlias**	**DBName**	**DBPath**	**DBUsername**	. . .
1	DWNutri	sql10327005	jdbc:mysql:..	sql10327005	...
2	DWMilk	sql10327005	jdbc:mysql:..	sql10327005	...

nessa tabela são: DBNameAlias, o qual mantém um nome amigável para o banco de dados, DBName, o qual mantém o nome real do banco de dados, DBPath, mantém o caminho do banco de dados, DBUsername e DBPassword, os quais mantém respectivamente o nome e senha do usuário do banco de dados, DBServer, o qual mantém o nome do servidor do banco de dados e DBDriverName o nome do *driver* de conexão ao banco de dados. Vale destacar que é por meio dessa tabela que os metadados de conexão são identificados para se conectar ao DW e realizar a inserção dos dados. Em especial, o atributo *DBEntityName* é responsável por armazenar o nome da tabela do DW na qual será inserido os dados extraídos pelo Imã de Dados. Além disso, os demais dados disponíveis para configuração foram obtidos por meio de um arquivo de configuração do próprio MySQL. Esse arquivo mantém todos os metadados necessários para se estabelecer conexão remota com a base de dados.

Tabela 9 – Configuração da tabela *Entities*

Entities	
Id	**EntityName**
1	MilkYield
2	Sale
3	Nutri
10	TestMilk

A Tabela 9 mostra a configuração da tabela *Entities*, a qual mantém o nome das entidades das fontes de dados do ambiente operacional as quais mantém os dados de interesse. No caso deste experimento, mantém dados de produção de leite. Vale destacar que as entidades *Sale* e *Nutri* definidas na configuração não serão utilizadas para esse experimento. A configuração dessas duas entidades foram feitas exclusivamente para ilustrar a inserção de dados nessa tabela. Além disso, durante a execução do Imã de Dados, será mostrado que o Imã de Dados irá considerar apenas a entidade que esteja definida no arquivo JSON.

A Tabela 10 mostra a configuração da tabela *EntitiesFields*. Essa tabela é responsável por manter a relação de cardinalidade 1..* com a tabela *Entities*. O atributo *IdEntity* mantém o relacionamento com o Id da entidade configurado na tabela *Entities*. O atributo *FieldName* mantém o nome dos atributos que compõem a tabela *Entities*. Dessa forma, pode-se observar que a entidade *MilkYield* tem 6 atributos a ela associados. Já a entidade *Sale* tem 3 atributos definidos. A entidade *Nutri* tem 4 atributos definidos a ela. Por fim, a entidade *TestMilk* tem 3 atributos associados a ela. Vale destacar que os atributos das entidades *Sale*, *Nutri* e *TestMilk* foram

Tabela 10 – Configuração da tabela *EntitiesFields*

EntitiesFields		
Id	**IdEntity**	**FieldName**
1	1	milk_id_animal
2	1	milk_milk_volume
3	1	milk_date
4	1	milk_time
5	1	milk_period
6	1	milk_time_started
7	2	product_name
8	2	product_price
9	2	product_date_sale
10	3	nutrition_id_animal
11	3	nutrition_id_animal
12	3	nutrition_weight_animal
13	3	nutrition_food_animal
42	10	milk_id_animal
43	10	milk_volume
44	10	weight

configurados apenas para exemplificar a configuração dessa tabela. Dessa forma, na execução do Imã de Dados, serão considerados apenas os atributos da entidade *MilkYield*, cujo atributo *IdEntity* é igual a 1.

Tabela 11 – Configuração da tabela *EntitiesFieldsTags*

EntitiesFieldsTags		
Id	**IdTag**	**IdEntityField**
1	1	1
2	1	2
3	1	3
4	1	4
5	1	5
6	1	6
17	3	42
18	3	44

A Tabela 11 mostra a configuração da entidade *EntitiesFieldsTags*. Essa tabela é responsável por manter a relação entre os atributos de uma entidade e uma *tag*. É importante destacar que os atributos mostrados na Tabela 10 estão todos os atributos das entidades definidas na tabela *Entities*, ou seja, todos os atributos disponíveis no arquivo JSON de produção de leite estão definidos na tabela *EntitiesFields*. Entretanto, a tabela *EntitiesFieldsTags* é responsável por relacionar um atributo de uma entidade e uma *tag*. Neste momento, cria-se o conceito de **definir os dados de interesse do usuário para tomada de decisão**. Isso equivale dizer que, no momento da execução, o Imã de Dados recebe o arquivo JSON no módulo extração e, por

meio dessa tabela, é feita a verificação se algum dos atributos do arquivo JSON tem *tag* a eles relacionados. Para ilustrar o ambiente de produção de leite com todos os dados relevantes para o produtor tomador de decisão, todos os atributos disponíveis no arquivo JSON serão considerados como dados de interesse.

Tabela 12 – Configuração da tabela *InterestConfig*

InterestConfig		
Id	**IdTag**	**IdTargetDatabase**
1	1	2
2	3	1

A Tabela 12 mostra a configuração da tabela *InterestConfig*. Essa tabela mantém a relação de um DW, representado pelo atributo *IdTargetDatabase* e uma *tag*, representada pelo atributo *IdTag*. É nessa tabela que é mantido o relacionado de uma *tag* com um DW de interesse. Percebe-se que o DW *DBMilk* tem interesse em dados de atributos cuja *tag Milk* esteja relacionada. Por outro lado, o DW *DBNutri* tem interesse em dados de atributos cuja *tag Nutri* esteja relacionada. Isso equivale a dizer que, no processo de carregamento do Imã de Dados, assim que os atributos do arquivo JSON e suas respectivas *tags* forem detectados, é feita uma consulta nessa tabela e verificado qual o DW tem interesse nos dados que estão associados a tal *tag*.

Figura 27 – Esquema estrela do DW preparado para receber os dados

Para encerrar as configurações do experimento, além das configurações reais do Imã de Dados, é importante mostrar o esquema estrela adotado para representar o DW. A Figura 27 ilustra a tabela de fatos criada para representar o DW, a qual receberá os dados de produção de leite. Em fazendas produtoras de leite, a coleta de leite pode ocorrer em até 3 períodos do dia e uma mesma vaca só pode ter seu leite coletado uma vez por período. Em função disso, o número do brinco da vaca, a data e o período da coleta do leite foram escolhidos para formar a chave primária.

É importante destacar que essa tabela de fatos foi criada com base no arquivo JSON que disponibiliza os dados do ambiente de pecuária leiteira. Mais especificamente, após identificar como o ESP32 disponibiliza os dados que trafegam por ele e quais são esses dados, foi possível criar essa tabela de fatos com os devidos atributos. Assim, os atributos são: *milk_id_animal*, o qual representa o número do brinco do animal. O atributo *milk_milk_volume* representa o

volume de leite produzido pela vaca. O atributo *milk_time_started* indica a hora na qual iniciou a ordenha. O atributo *milk_date* indica a data da ordenha. Já o atributo *milk_time* representa a hora em que ocorreu o termino da ordenha. Já o atributo *milk_period* indica o período do dia que corresponde a hora da ordenha. Por fim, o atributo *time_arrive* representa a hora em que o Imã de Dados armazenou os dados no DW.

Após mostrar as tabelas que armazenam dados de configurações do Imã de Dados, mostrar os dados reais dessas configurações que foram utilizadas para realizar o experimento e a tabela de fatos utilizada para representar o DW, é importante mostrar a execução de todos os componentes que compreendem o Imã de Dados. Essa execução compreende todo o Imã de Dados e tudo o que foi feito neste trabalho. Além disso, essa execução inclui desde gerar um dado por meio dos sensores na sala de ordenha até esse mesmo dado ser armazenado no DW. Dessa forma, de forma genérica, os passos a serem executados são: 1) gerar os dado pelos sensores; 2) gerenciar os dados por meio do microcontrolador; 3) enviar os dados para o *broker*; 4) receber os dados do *broker*; 5) construir o arquivo JSON; 6) detectar as *tags*; 7) detectar o DW interessado nos dados; 8) armazenar os dados no DW.

Sendo assim, na próxima seção será detalhado como ocorre esses passos e a execução do Imã de Dados. Inicialmente, será mostrado como cada passo é executado. Além disso, serão utilizados pseudocódigos para ilustrar as tarefas executadas. Após apresentar a execução do Imã de Dados, serão apresentadas ilustrações reais do Imã de Dados em execução, os quais mostrarão os dados sendo gerenciados pelo Imã de Dados e esses mesmos dados sendo visualizados em tempo real no aplicativo MVLTR.

6.3.2 Execução do experimento

Após ter mostrado os equipamentos utilizados no experimento, como eles foram montados no ambiente de pecuária leiteira e explicado o processo de configuração do Imã de Dados, é o momento do Imã de Dados ser executado. Essa execução será mostrada por meio de algoritmos que descrevem passo a passo as tarefas realizadas. Além disso, serão mostradas fotos do ESP32 enviando os dados, assim como do Imã de Dados recebendo os dados do ESP32 *Sender* e armazenando-os no DW. Contudo, antes de executar o Imã de Dados, é necessário que o ambiente IdC gere os dados de produção de leite. Portanto, a seguir serão descritos os passos para gerar os dados de produção de leite e posteriormente serão descritos os passos para executar o Imã de Dados.

6.3.2.1 Execução do ambiente IdC

A execução do ambiente IdC envolve os sensores de presença, o sensor de fluxo e os microcontroladores serem executados e os dados produzidos serem enviados para o *broker*. Essa execução ocorre em diversas etapas, as quais são dependentes uma das outras para que todo o

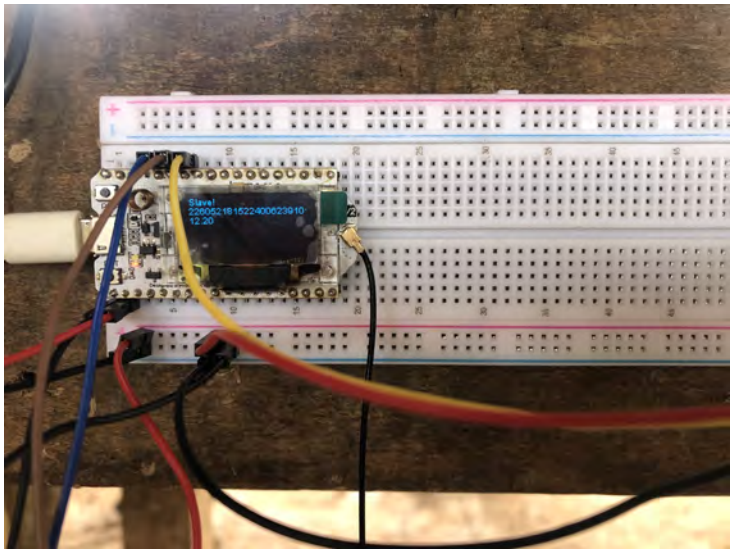
ciclo de coleta de dados em tempo real possa ser completada. Portanto, a execução do ambiente IdC será descrita com algoritmos e fotos reais do experimento.

Algoritmo 1 Gerar dado de produção de leite

Entrada: Número do brinco da vaca
Saída: Valor de produção do leite
início
    Identifica a vaca por meio do sensor de curta distância.
    Mantém o número do brinco da vaca temporariamente no ESP32 *Slave*.
    Conecta a teteira no teto da vaca.
    Inicializa sensor de fluxo.
    repita
        Ler o valor do leite por meio do sensor de fluxo.
        Manter o valor do leite temporariamente no ESP32 *Slave*.
    até *até a vaca passar pelo sensor de presença de media distância E a teteira estiver conectada*;
fim

O Algoritmo 1 mostra os passos executados para que o sensor de presença de curta distância possa identificar uma vaca quando a mesma entrar na sala de ordenha. Além disso, esse algoritmo mostra a coleta da produção do leite por meio do sensor de fluxo. Assim que uma vaca entra na sala de ordenha, a mesma é identificada por meio do sensor de curta distância. Isso se deve ao fato da vaca estar equipada pelo brinco RFID contendo um número de identificação único. Após isso, com o auxilio humano, a teteira é conectada ao teto da vaca. Além do mais, o ESP32 *Slave* mantém temporariamente o número do brinco da vaca e o valor do leite a medida que o mesmo é gerado pelo sensor de fluxo. Enquanto a vaca não sair da sala de ordenha, passar pelo sensor de presença de media distância e não for retirada a teteira da vaca, o sensor de fluxo deve coletar o valor do leite produzido.

Figura 28 – ESP32 *Slave* em execução após o sensor de curta distância detectar uma vaca



A Figura 28 mostra o ESP32 *Slave* em execução após o sensor de curta distância detectar uma vaca na sala de ordenha. O ESP32 é composto por um visor digital, por meio do qual é possível visualizar as tarefas sendo executadas pelo ESP32. A primeira linha tem o nome *Slave*, a qual indica que esse ESP32 é o responsável por gerenciar os dados do sensor de curta distância. A segunda linha é o número do brinco da vaca lido do brinco RFID pelo sensor de curta distância. Já a terceira linha indica o valor da produção do leite extraído do sensor de fluxo. A medida que o leite é retirado da vaca, esse valor aumenta em tempo real e é mostrado no visor.

Algoritmo 2 Identificar a saída da vaca da sala de ordenha

Entrada: Número do brinco da vaca lido pelo sensor de media distância
Saída: Número do brinco enviado para o ESP32 *Slave*.
início
    Identifica a vaca por meio do sensor de media distância.
    Envia o número do brinco detectado para o ESP32 *Slave*.
fim

Após retirar a teteira da vaca, o sensor de fluxo deve parar de ler o valor do leite produzido. Além disso, a vaca deve passar pelo sensor de media distância para registrar sua saída da ordenha. O Algoritmo 2 mostra a identificação da vaca por meio desse sensor e o que ocorre após identificar a saída da vaca. Assim que a vaca passa pelo sensor de media distância, ocorre a leitura e identificação da vaca por meio do brinco RFID. Então, o ESP32 *Master* mantém temporariamente o número do brinco da vaca identificado pelo sensor, e envia o número do brinco de volta para o ESP32 *Slave* para indicar que a tal vaca saiu da sala de ordenha. Por fim, o ESP32 *Master* é reinicializado para que possa realizar a leitura e gerenciamento da próxima vaca que passar pelo sensor.

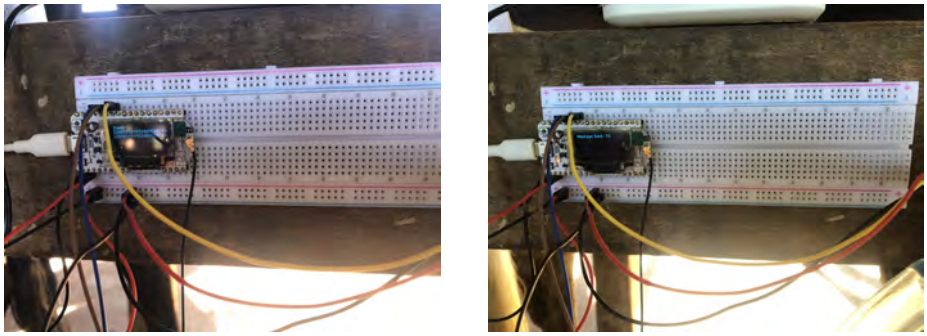
Algoritmo 3 Enviar dados para o ESP32 *Sender*

Entrada: Número do brinco lido pelo ESP32 *Master*.
Saída: Dados enviados ao ESP32 *Sender*.
início
    Recebe o número do brinco da vaca do ESP32 *Master*.
    se *número do brinco lido é igual ao número do brinco da última vaca* **então**
        Pega o valor do leite produzido por meio do sensor de fluxo.
        Pega o número do brinco da vaca.
        Encapsula esses dados em um arquivo JSON.
        Envia os dados para o ESP32 *Sender*.
    fim
fim

Após identificar a vaca que saiu da ordenha, o ESP32 *Master* deve enviar o número do brinco da vaca para o ESP32 *Slave*, no qual a vaca estava sendo ordenhada, para finalizar o ciclo da ordenha da vaca. O Algoritmo 3 mostra como ocorre esse processo. O ESP32 *Slave* verifica se o número do brinco recebido é o mesmo número do brinco da última vaca que foi

ordenhada e gerenciada por ele. Isso se deve ao fato de que, no ambiente em questão, pode ter vários sensores de presença de curta distância conectados a um ESP32, isto é, que identifica a entrada de várias vacas ao mesmo tempo na sala de ordenha, e apenas um sensor de media distância, ou seja, que identifica a saída da vaca por um único local na sala de ordenha. Dessa forma, é possível identificar e garantir qual o ESP32 *Slave* gerenciou os dados de produção de leite da vaca que acabou de ser identificada pelo ESP32 *Master*. Após esse processo, o ESP32 *Slave* envia para o ESP32 *Sender* os seguintes dados no formato JSON: número do brinco e valor do leite produzido.

Figura 29 – ESP32 *Slave* em execução após receber o número do brinco do ESP32 *Master*



(a) ESP32 *Slave* recebendo o número do brinco do ESP32 *Master*.

(b) ESP32 *Slave* enviando dados para o ESP32 *Sender*.

A Figura 29 (a) mostra o ESP32 *Slave* no momento em que recebe o número do brinco do ESP32 *Master*. A primeira linha indica que esse ESP32 é o *Slave* (na imagem é mostrada a palavra *Slave*). A segunda linha indica o número do brinco da vaca que estava sendo ordenhada nesse ESP32 *Slave*. A terceira linha indica o número do brinco da vaca que passou pelo sensor de media distância e foi enviado pelo ESP32 *Master*. É importante destacar que quando o número do brinco da vaca é enviado do ESP32 *Master* para o ESP32 *Slave*, esse dado é enviado para todos os ESP32 disponíveis no ambiente. Contudo, cada ESP32 é capaz de detectar se o número do brinco da vaca enviado pelo ESP32 *Master* é o número da vaca que estava sendo ordenhada nesse ESP32 *Slave*. Se o número do brinco igual, então indica que a vaca que estava sendo ordenhada nesse ESP32 *Slave* acabou de sair da sala de ordenha. Com isso, o ESP32 *Slave* em questão envia os dados para o ESP32 *Sender*. Esse comportamento é indicado na Figura 29 (b), na qual mostra a expressão "Mensagem Enviada"no visor do ESP32 *Slave* para indicar ao usuário que os dados foram enviados para o ESP32 *Sender*. Caso o número do brinco for diferente, o processo de enviar dados para o ESP32 *Sender* é ignorado e o ESP32 *Slave* continua preparado para ler o valor do leite produzido.

Algoritmo 4 Enviar dados do ambiente IdC para o *broker*.

Entrada: Dados recebidos pelo ESP32 *Slave*.
Saída: Dados enviados ao *broker*.
início
    Recebe o número do brinco da vaca do ESP32 *Slave*.
    Recebe o valor do leite produzido do ESP32 *Slave*.
    Pega a data e a hora atual.
    Gera o período do dia no qual corresponde a hora atual.
    Encapsula esses dados em um arquivo JSON.
    Conecta-se ao *broker*.
    Publica no *broker* o JSON gerado.
fim

Por fim, o ESP32 *Sender* recebe os dados enviados pelo ESP32 *Slave* e envia-os para o *broker*. O Algoritmo 4 mostra o envio dos dados ao *broker*. Ao receber os dados do ESP32 *Slave*, o ESP32 *Sender* identifica a data e a hora atual, e identifica o período do dia no qual corresponde a hora atual (já explicado na seção 6.3.1). Após esse processo, o ESP32 *Sender* integra esses dados em um arquivo JSON juntamente com os dados recebidos pelo ESP32 *Slave*, se conecta ao *broker* e publica os dados gerados.

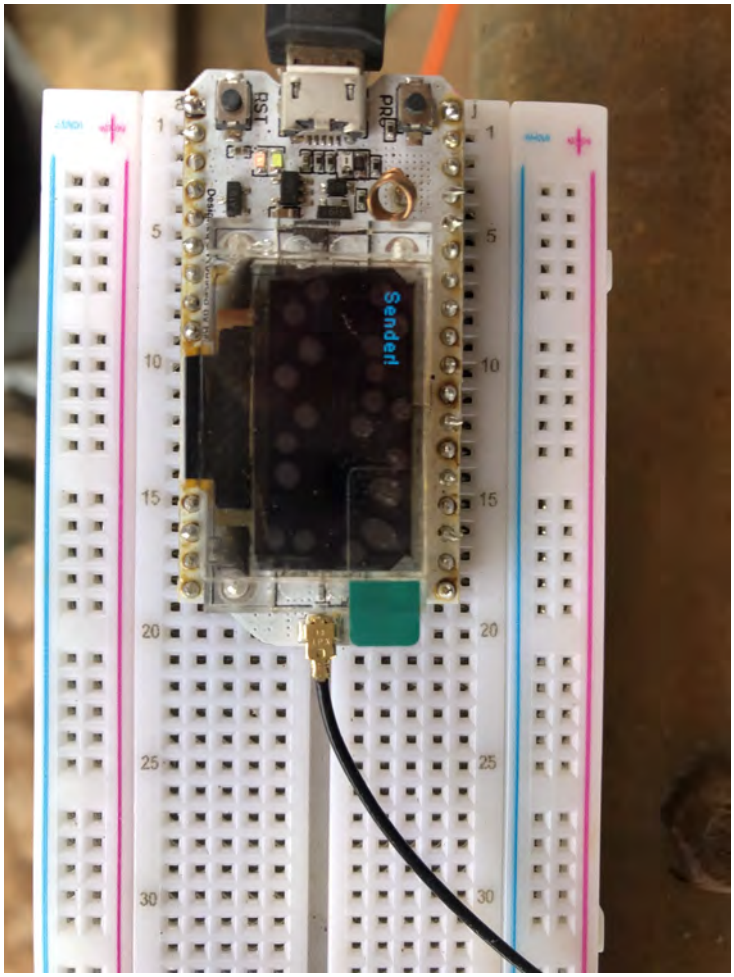
É importante destacar que o ESP32 *Sender* é o único ESP32 conectado à Internet. Isso se deve ao fato de que ele foi fixado em um local estratégico, o qual é possível obter acesso à Internet e ao mesmo tempo se comunicar internamente com os demais ESP32. Além disso, tanto o local no qual foi fixado o ESP32 *Slave* quanto o local no qual foi fixado o ESP32 *Master* não possui acesso à Internet. Isso ocorre devido a posição geográfica do curral na fazenda não ser favorável para cobrir o sinal de Internet em todos os pontos do curral. Com isso, o único local possível de se ter acesso a Internet por meio de um ESP32 é onde foi fixado o ESP32 *Sender*.

A Figura 30 mostra o ESP32 *Sender* em execução. Sempre que um novo dado é enviado do ESP32 *Slave* para o ESP32 *Sender*, a mensagem *Sender* é exibida no visor do ESP32 apenas para mostrar ao usuário que os dados foram enviados ao *broker*. Dessa forma, é finalizado o ciclo de geração de dados no ambiente de pecuária leiteira e o envio desses dados para o *broker*. Na seção 6.3.2.2 será mostrada a execução do Imã de Dados, isto é, como ocorre o recebimento desses dados do ambiente IdC e seu armazenamento no DW.

6.3.2.2 Execução do Imã de Dados

Paralelamente a execução do ambiente IdC para gerar os dados de produção de leite, o Imã de Dados também é executado. O módulo Extração do Imã de Dados atua como *subscriber*, isto é, se conecta ao *broker* e se inscreve no tópico no qual foi publicado os dados pelo *publisher* (ESP32 *Sender*) e recebe os dados a ele enviados. Após esse processo de recebimento dos dados (extração), os dados recebidos são mantidos temporariamente no Imã de Dados. Isso é necessário

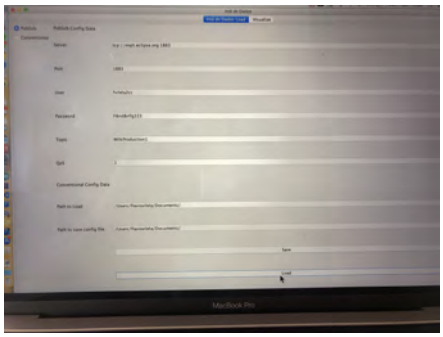
Figura 30 – ESP32 *Sender* em execução após receber os dados do ESP32 *Slave*



para que o Imã de Dados possa executar as demais tarefas e não mais depender de um servidor externo.

Figura 31 – Imã de Dados preparado para se conectar ao *broker*

(a) Computador no curral conectado à Internet.



(b) Configuração do Imã de Dados para se conectar ao *broker*.

A Figura 31 mostra o computador conectado à Internet no curral da fazenda (a), o qual mantém o Imã de Dados em execução. Além disso, é mostrado também o Imã de Dados em execução (b). Neste caso, é mostrada a aba de configuração para que a conexão com o

broker possa ser estabelecida. Para estabelecer essa conexão, é necessário informar os seguintes dados: caminho do servidor, número da porta de comunicação, nome de usuário e senha, o nome do tópico no qual deseja se conectar e qual a qualidade do serviço (campo QoS). Além dessas configurações, o usuário administrador pode gravar/alterar as configurações ou carregar configurações já salvas. Para isso, é necessário informar o caminho no qual se deseja ler o arquivo que armazena as configurações do Imã de Dados.

Algoritmo 5 Identificar *tags*

Entrada: Arquivo JSON recebido do *broker*
Saída: Arquivo JSON contendo os atributos originais e as *tags* associadas a cada atributo

```
início
    Conecta-se ao broker.
    Recebe uma string contendo os dados gerados.
    Cria um arquivo JSON contendo tais dados.
    Mantém o arquivo JSON temporariamente no Imã de Dados.
    Cria-se arquivo JSON temporário.
    para cada atributo do arquivo JSON recebido faça
        Conecta-se ao Repositório de Atributos e Tags.
        se existe tag associada ao atributo então
            Associa o nome do atributo à chave do JSON temporário.
            Associa o nome da tag ao valor da chave do JSON temporário.
            Adiciona o atributo chave-valor ao JSON temporário.
        fim
    fim
    Adiciona o JSON temporário ao JSON recebido.
    Retorna JSON recebido.
fim
```

Após esse processo de configuração e estabelecer conexão do Imã de Dados ao *broker*, o módulo Extração é responsável por realizar a primeira tarefa do Imã de Dados, isto é, identificar a *tag* associada a cada atributo. Essas tarefas são mostradas no Algoritmo 5. Após conectar-se ao *broker* e receber o arquivo JSON, para cada atributo do arquivo JSON obtido, é verificado junto ao Repositório de Atributos e Tags se existe uma *tag* configurada para esse atributo. Se existir, um novo atributo chamado Tags é criado dentro do arquivo JSON recebido. Esse atributo Tégs é composto por um outro arquivo JSON, o qual contém o nome do atributo como chave e o nome da *tag* como valor da chave. Nesse momento, percebe-se que já é possível ter todos os atributos correspondentes ao volume de leite juntamente com suas *tags*. Após esse processo, o arquivo JSON contendo os atributos e suas respectivas *tags* são mantidos temporariamente em memória primária para ser consumido pelo módulo Carga. Vale destacar que os passos mostrados no Algoritmo 5 podem ser visualizados de forma simplificada e abstrata na Figura 24 por meio da interação dos componentes Ambiente Operacional, Servidor e Extração.

O módulo Carga é responsável por receber o arquivo JSON modificado anteriormente contendo os atributos associados às *tags* e identificar qual DW tem interesse nos dados mantidos

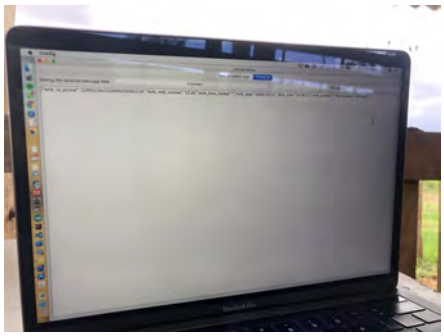
nos atributos associados às *tags*. Os passos para realizar essas tarefas são mostrados no Algoritmo 6. Inicialmente, para cada *tag* do arquivo JSON, é verificado junto ao Repositório de Interesse, se existe alguma fonte de dados que representa o DW interessada em dados mantidos nos atributos associados a *tag*. Após essa verificação, se não existir interessados, o processo é cancelado. Caso contrário, para cada DW identificado, deve-se executar os seguintes passos: 1) busca-se os metadados de conexão ao DW; 2) cria-se uma *string* de inserção no DW; 3) executa essa *string* de inserção no DW.

Algoritmo 6 Realizar o carregamento dos dados para o DW.

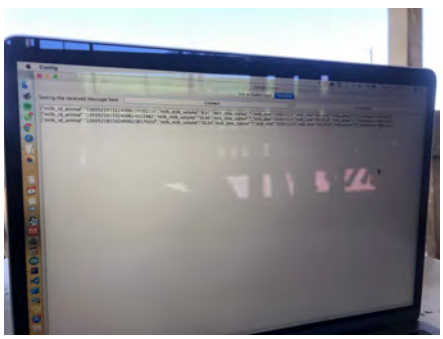
Entrada: Arquivo JSON modificado contendo as *tags* associadas aos atributos.
Saída: Dados armazenados no DW
início
  para cada *tag no arquivo JSON* **faça**
    Conecta ao Repositório de Interesses.
    se *existe alguma fonte de dados configurada para a tag* **então**
      Busca-se os metadados de conexão à fonte de dados identificada.
      Cria-se a *string* de inserção.
      Executa a *string* de inserção no DW.
    fim
  fim
  Adiciona o JSON temporário ao JSON recebido.
  Retorna JSON recebido.
fim

Vale destacar que no processo de busca de DW interessado na *tag*, identifica-se também o nome da tabela que irá armazenar os dados no DW. Logo, na construção da *string* de inserção, o nome da tabela é o nome da tabela configurada no Repositório de Metadados. Além disso, o nome dos atributos que compõem essa tabela serão obtidos dinamicamente a partir do arquivo JSON. Por fim, os valores dos atributos nessa *string* de inserção serão os mesmos valores dos atributos obtidos do arquivo JSON.

Figura 32 – Imã de Dados em execução



(a) Imã de Dados em execução.

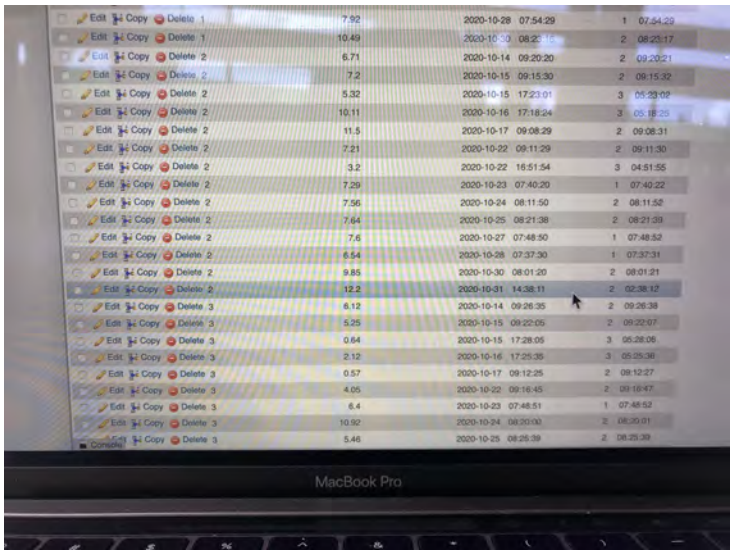


(b) Imã de Dados em execução com vários dados.

Após o ESP32 *Sender* enviar os dados do ambiente IdC para o *broker*, o Imã de Dados

finalizou a execução de todas as suas tarefas. Com isso, os dados gerados no ambiente IdC foram extraídos, tratados pelo Imã de Dados e armazenados no DW. A Figura 32 (a) mostra o Imã de Dados sendo executado, na qual é possível visualizar o arquivo JSON criado contendo os dados gerados no ambiente IdC. É possível visualizar o número do brinco da vaca, o volume de leite produzido, a data e a hora que ocorreu a extração dos dados, o período e o nome da entidade. Já na Figura 32 (b) é possível visualizar vários arquivos JSON criados. Cada linha indica um arquivo JSON contendo dados de produção de leite de diferentes vacas. Vale destacar que o Imã de Dados foi executado em um computador com as seguintes configurações: MacBook Pro, sistema operacional MacOS Catalina 10.15.7, 1.4 Intel Core i5, 8 GB de memória RAM.

Figura 33 – Dados armazenados no DW



Já a Figura 33 mostra os dados que foram gerados no ambiente IdC, tratados pelo Imã de Dados e armazenados no DW. Neste momento, percebe-se que o ciclo que compreende a geração dos dados no ambiente IdC e seu armazenamento no DW foi finalizado. Uma vez que os dados foram gerados em no ambiente IdC, o Imã de Dados foi capaz de detectar quais desses dados são de interesse ao usuário tomador de decisão, verificar qual base de dados tem interesse em armazenar esse dado e realizar a persistência do dado. Com isso, o usuário tomador de decisão tem disponível em tempo real os dados de interesse para tomada de decisão estratégica. No caso deste experimento no domínio de pecuária leiteira, o produtor e/ou veterinário tem disponível os dados de produção de leite em tempo real, a medida que tais dados são gerados na sala de ordenha.

Para acompanhar a geração desses dados em tempo real, paralelamente a execução do Imã de Dados, o produtor e/ou veterinário, por meio do aplicativo MVLTR, pode se conectar ao DW. Imediatamente após os dados ser persistidos pelo módulo de carregamento do Imã de Dados, esses dados de produção de leite são apresentados em tempo real no aplicativo. Esse fato mostra que, do ponto de vista da arquitetura, após as configurações do Imã de Dados, o mesmo é capaz de realizar suas tarefas de forma automática e em tempo real. Do ponto de vista

do produtor, tomador de decisão, os dados visualizados em tempo real por meio do aplicativo MVLTR auxiliam o processo de análise e tomada de decisão em tempo real. Isso se deve ao fato de que um valor de produção de leite de uma vaca individualmente que esteja fora do padrão esperado poderá ser visualizado em tempo real e uma tomada de decisão poderá ser feita imediatamente.

Figura 34 – Aplicativo MVLTR em execução

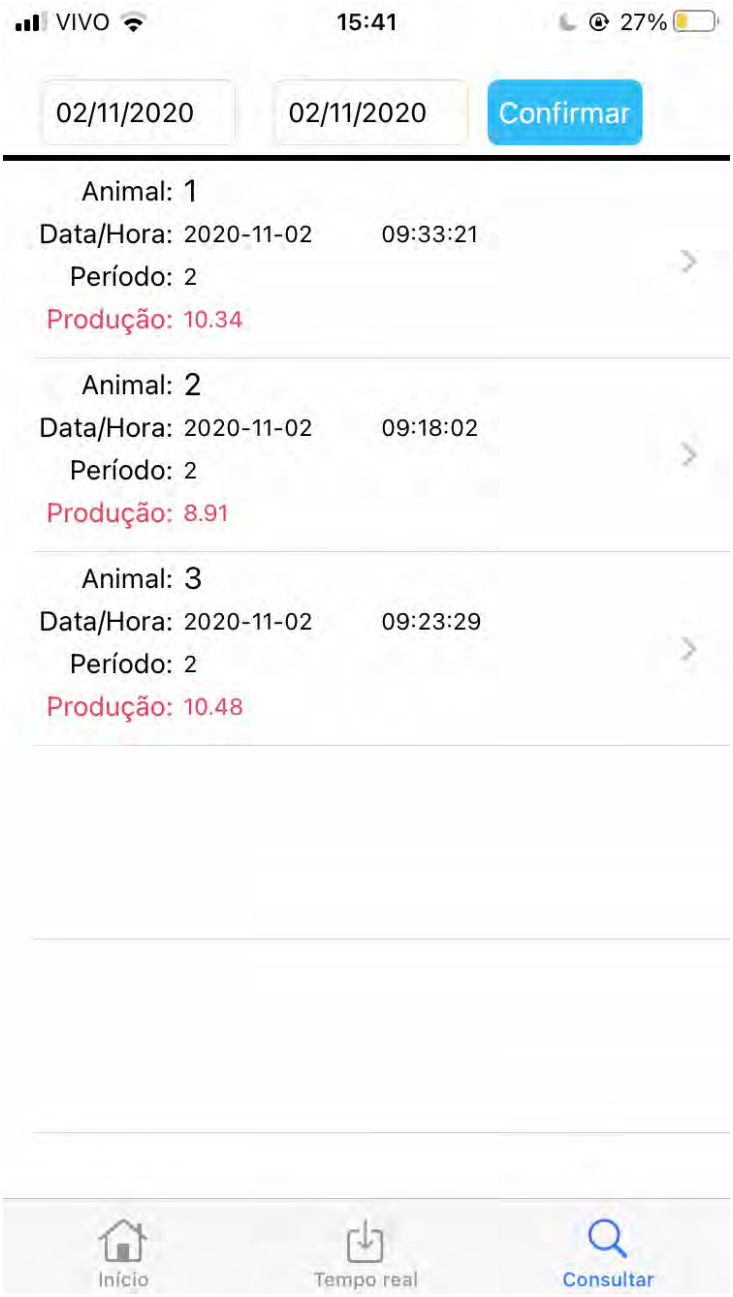
(a) Aplicativo MVLTR mostrando as pesagens de leite em tempo real.

(b) Aplicativo MVLTR mostrando as pesagens de leite de apenas uma vaca em um dia.

A Figura 34 mostra o aplicativo MVLTR em execução. A Figura 34 (a) mostra a aba Tempo Real ativa, na qual é possível acompanhar em tempo real o volume de leite. Neste exemplo, é possível observar o volume de leite do dia 31 de outubro de 2020, na qual foram utilizadas as vacas 1, 2 e 3. Por meio do aplicativo, é possível verificar os dados de volume de leite, tais como a hora da coleta, o período da coleta e o valor do volume de leite. Já na Figura 34 (b) é mostrado o detalhamento do volume de leite de uma vaca em específica. Por exemplo: a medida que as vacas são ordenhadas, seus dados de leite são mostrados no aplicativo (a). Contudo, caso a propriedade realize a coleta do volume de leite duas ou três vezes por dia e o produtor deseja observar o volume de leite de uma vaca específica naquele dia, é possível selecionar uma determinada vaca (a) e visualizar seus dados de produção de leite no dia atual (b).

A consultas foram feitas utilizando a linguagem SQL e executadas diretamente no DW. No caso da aba Tempo real, a consulta é executada utilizando a data atual como parâmetro. Já na Figura 35 é possível observar a aba Consultar. Por meio dessa aba, o usuário pode realizar a

Figura 35 – Consulta realizada pelo aplicativo MVLTR no DW



pesquisa por uma data específica. Neste exemplo, foi utilizada a data de 2 de novembro de 2020, na qual foi possível visualizar todas as vacas que foram ordenhadas naquele dia. Caso o usuário desejar, é possível selecionar uma vaca em específico e visualizar de forma detalhada, como mostrada na Figura 34 (b). Nesse caso, a consulta é realizada por data e por animal.

6.3.3 Considerações sobre os dados coletados do ambiente de pecuária leiteira ao executar o Imã de Dados

Após mostrar os equipamentos utilizados, como eles foram montados no ambiente de pecuária leiteira, explicado o processo de geração de dados no ambiente IdC e mostrado

o processo de configuração e execução do Imã de Dados, é importante mostrar os dados de produção de leite coletados pelo Imã de Dados. Além disso, é importante apresentar as análises para tomada de decisão que podem ser realizadas por parte do produtor e/ou veterinário com base nesses dados coletados. Essas análises servem para mostrar que, a partir dos dados coletados em tempo real, é possível que o produtor analise e tome as devidas decisões também em tempo real a partir da produção de leite individual de cada vaca.

Vale destacar algumas restrições identificadas na fazenda utilizada para realizar o experimento. Por consequência às restrições, alguns ajustes foram feitos para realizar integralmente o experimento: 1) de um total de aproximadamente 300 vacas em lactação, o proprietário cedeu apenas 3 vacas para o experimento. Além disso, foi separado um local próprio dentro do curral para realizar o experimento com essas vacas. Apesar desses fatos, foi possível coletar todos os dados necessários para realizar o experimento. Além disso, foi possível presenciar diversas situações reais, as quais as análises para tomada de decisão se tornaram também reais; 2) os dados foram coletados durante 14 dias, dos quais apenas 3 dias foi possível realizar a coleta do volume de leite em dois períodos (1 e 2, ou manhã e tarde). Isso se deve ao fato de que durante o período do ano no qual o experimento foi realizado, existe grande incidência de chuva durante a tarde e a noite. Esse fato impediu a montagem de parte dos equipamentos utilizados no experimento, tais como ESP32 *Sender* e o roteador de Internet. Assim, o experimento foi realizado em sua maioria no período 1 ou período 2, os quais compreende todo o período matutino. Além da grande incidência de chuvas, outro fator negativo da região é a falta de energia em períodos chuvosos. Esse fato também contribuiu para a não execução de todo o experimento em mais de um período.

Uma curiosidade sobre as vacas analisadas é que elas são consideradas de médio porte, isto é, sua produção de leite não está entre as melhores de todo rebanho, mas também não são as que menos produzem leite. Além disso, com relação a nutrição animal, quanto mais as vacas comem, mais elas produzem leite. Entretanto, qualquer alteração em seu *habitat* natural pode fazer com que sua produção de leite diminua, independentemente de sua alimentação. Esse aspecto foi constatado durante o experimento, por exemplo: no dia 15/10/2020 ocorreu a coleta de dados de produção de leite em dois períodos (manhã e tarde). As vacas 1 e 2 produziram uma quantidade de leite maior no período da manhã do que no período da tarde. Essa oscilação para baixo é considerada normal devido a própria natureza do animal. Contudo, a no mesmo dia, a vaca 3 teve sua produção praticamente zerada. Isso ocorreu por dois motivos: 1) essa vaca não é acostumada ir para o curral duas vezes no dia, mas sim apenas pela manhã. Isso fez com que gerasse um estresse no animal, fazendo que sua produção praticamente fosse zero; 2) quando a vaca entrou no curral a tarde, a vaca teve que ser forçada a entrar na sala de ordenha. Esse fato também contribuiu para aumentar seu estresse e consequentemente sua queda de produção de leite.

Já no dia 22/10/2020, no período 1 (manhã), todas as vacas sofreram uma queda em

sua produção de leite devido a um erro humano. Na noite anterior ao dia 22, um funcionário da fazenda esqueceu as vacas trancadas dentro do curral. Logo pela manhã, assim que foi começar a coleta de leite, as vacas estavam estressadas e desnutridas (pois as vacas passaram a noite toda presa sem água e comida). Esse fato impactou diretamente em sua produção de leite. Ainda no dia 22/10/2020, foi possível observar que apenas a vaca 2 teve sua coleta de volume de leite em dois períodos. Isso ocorreu pois, no momento em que foi iniciada a ordenha das demais vacas, houve uma queda de energia. Esse fato tornou impossível continuar com a coleta de dados das demais vacas.

No dia 28/10/2020, a vaca número 2 apresentou uma produção de leite menor que a esperado. Isso ocorreu pois, ao entrar na sala de ordenha, a mesma foi colocada de forma errada. Esse fato fez ela sair e entrar novamente na sala de ordenha. Como ela não está acostumada com esse comportamento inesperado, a mesma sofreu um estresse e teve uma queda em sua produção de leite naturalmente. Já no dia 31/10/2020, a produção de leite começou mais tarde do que o habitual. Isso ocorreu devido a falta de energia na propriedade. Esse fato fez com que as vacas continuassem a produzir leite naturalmente até que a energia fosse reestabelecida. Quando a energia voltou (já no período vespertino), a produção de leite no dia começou um pouco mais tarde. Esse fato fez com que houvesse uma variação positiva da produção de leite em todas as vacas. Por fim, no dia 03/11/2020, a vaca 1 produziu menos leite do que o esperado. Esse fato ocorreu devido a demora em iniciar sua coleta de leite e a mesma já estar preparada para realizar a coleta na sala de ordenha. O funcionário responsável pela ordenha estava fazendo outras atividades e o mesmo esqueceu de iniciar a coleta do leite. Essa demora fez com que o animal sofresse um estresse e não produziu o volume de leite esperado.

A partir dessas análises realizadas sobre os dados coletados, é possível observar que tais análises podem ser feitas em tempo real pelo produtor. Isso porque o Imã de Dados é capaz de extrair os dados do ambiente IdC e armazená-los no DW em tempo real independentemente do sensor que gerou o dado. Além disso, com o auxilio do aplicativo MVLTR, o produtor pode acompanhar a coleta do volume de leite também em tempo real. Assim, as análises para tomada de decisão sobre os dados de produção de leite pode ser feitas em tempo real. Por exemplo: caso ocorra uma queda acentuada na produção de leite de uma vaca de um período para outro ou de um dia para outro, o produtor pode intervir imediatamente e verificar no curral qual o motivo dessa queda na produção. Logo, alguma decisão pode ser tomada imediatamente após algum fato negativo ser constatado. Além disso, visto que os dados de produção de leite ficam armazenados no DW por um longo período de tempo, o histórico dos dados também pode ser consultado para detectar novos padrões de comportamento das vacas e assim favorecer a novos tipos de tomadas de decisões.

Vale destacar que, do ponto de vista dos requisitos de tempo real, isto é, desempenho, disponibilidade e escalabilidade, o baixo volume de dados coletados faz com que se torne inviável analisar rigorosamente tais requisitos de tempo real a partir do experimento com dados reais.

Dessa forma, a partir das restrições identificadas, o experimento com dados reais foi conduzido apenas para mostrar a viabilidade de se aplicar o Imã de Dados em um ambiente IdC no domínio de pecuária leiteira. Portanto, o experimento apresentado na seção 6.3.5 foi conduzido a partir de dados gerados sinteticamente, cujo o volume e a frequência na geração dos dados é maior que o experimento com dados reais. O objetivo do experimento com dados sintéticos é validar o Imã de Dados sob o ponto de vista dos requisitos de tempo real, isto é, validar a escalabilidade, disponibilidade e tempo de resposta.

6.3.4 Análise de desempenho do Imã de Dados com dados reais

Ao executar o Imã de Dados no ambiente de pecuária leiteira foi possível extrair os dados gerados pelo ambiente, identificar quais bases de dados tem interesse nesses dados e armazená-los em cada DW. Contudo, uma informação importante a ser analisada é em relação ao tempo de resposta. Por exemplo: quanto tempo é gasto para que o Imã de Dados receba os dados solicitados e armazene-os no DW?

Essa resposta pode ser obtida por meio do próprio esquema do arquivo JSON criado para compartilhar os dados do ambiente IdC e o DW. Nesse esquema, existem dois campos que representam tempo. O primeiro é o *milk_time*, o qual representa a hora na qual o ESP32 *Sender* enviou o arquivo JSON contendo os dados de produção de leite para o *broker*. Esse tempo é obtido por meio de acesso à Internet, outro motivo pelo qual o ESP32 *Sender* deve estar fixado em um local no qual é permitido acesso à Internet. Em outras palavras, assim que uma vaca entra na sala de ordenha, a mesma pode demorar 5, 10, 20 minutos ou mais para finalizar sua produção de leite. Contudo, assim que o sensor de média distância detecta sua saída da sala de ordenha, significa que sua produção de leite foi finalizada. Além disso, quer dizer que o DW deve receber esses dados imediatamente. Então, assim que o ESP32 *Sender* recebe esses dados de produção de leite, o mesmo integra a hora atual aos dados já existentes e envia para o *broker*.

Já o campo *time_arrive* indica a hora na qual o Imã de Dados gravou os dados de produção de leite no DW. Para isso, assim que o Imã de Dados recebe uma solicitação de gravação de dados em um DW, o mesmo integra a hora atual aos dados já recebidos do *broker*. Essa hora atual é obtida por meio da Internet, motivo pelo qual, assim como o ESP32 *Sender*, o Imã de Dados deve estar ativo em um computador com acesso à Internet. Dessa forma, é possível obter a hora em que os dados foram de fato armazenados no DW.

A partir desses dois campos que indicam tempo, é possível analisar quanto tempo levou para que um arquivo JSON seja gerado no ambiente IdC e seus dados sejam armazenados no DW. Esse tempo pode ser obtido subtraindo a hora em que os dados foram armazenados no DW e a hora que os dados saiu do ambiente IdC. Contudo, segundo Kleppmann, M. (KLEPPMANN, 2017), apenas analisar o tempo gasto para realizar uma operação não é suficiente para analisar o desempenho de uma aplicação. Segundo o autor, a média aritmética pode ser um bom ponto de partida para a análise de desempenho. Entretanto, outras duas variáveis devem ser analisadas,

tais como mediana e percentil (indicado pela letra *p*). Além disso, segundo o autor, para analisar essas duas variáveis, é necessário fazer os seguintes passos: 1) ordenar a lista de tempos em ordem crescente; 2) obter a média aritmética (soma todos os tempos e divide pela quantidade total).

Tabela 13 – Exemplo de tempo gasto para armazenar os dados do ambiente IdC no DW

Operação	**Nome**	**Tempo (s)**
1	Gravar dados do JSON 1	0
2	Gravar dados do JSON 2	1
3	Gravar dados do JSON 3	1
4	Gravar dados do JSON 4	1
5	Gravar dados do JSON 5	2
6	Gravar dados do JSON 6	2
7	Gravar dados do JSON 7	3
8	Gravar dados do JSON 8	3
9	Gravar dados do JSON 9	3
10	Gravar dados do JSON 10	3

A Tabela 13 mostra um exemplo de lista de tempo gasto para realizar a operação de gravar dados de um arquivo JSON (dados extraídos do ambiente IdC) no DW. Seguindo as recomendações de Kleppmann, M. (KLEPPMANN, 2017), a listagem está ordenada em ordem crescente de tempo gasto. Consequentemente, é possível obter, além da média, a mediana e o percentil desejado.

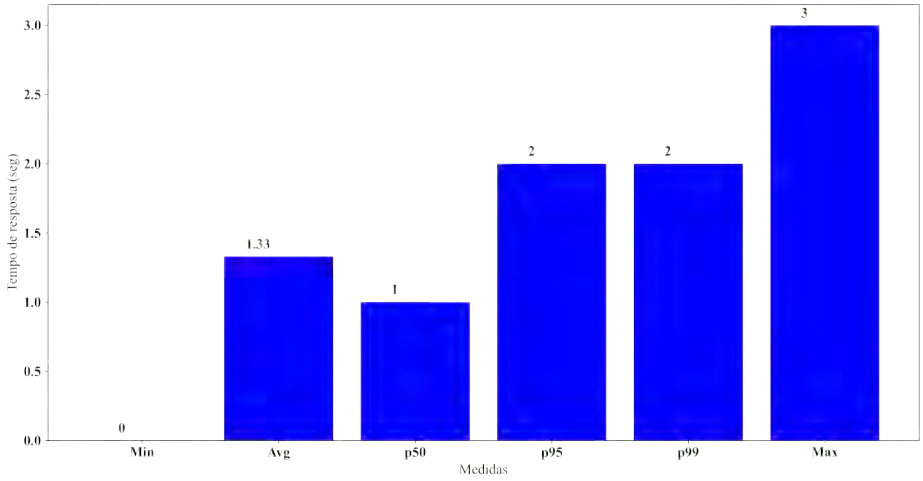
A partir dos dados exemplificados na Tabela 13, a media de tempo gasto para extrair os dados do ambiente IdC e armazená-los no DW foi de apenas 2,1 segundos. Essa média indica que o tempo gasto para realizar a operação é em torno de 2.1 segundos. Já a mediana é obtida por meio da listagem de todas as operações realizadas. A mediana é o valor do tempo que corresponde a operação que está na metade da listagem de todas as operações (p50). Na Tabela 13, a mediana é o valor do tempo da operação de número 5, isto é, 2 segundos.

Além da mediana, o percentil é importante para indicar outros valores de tempo gasto que estejam acima do valor da média e acima do valor do p50. Por exemplo: pode-se ter uma operação na qual gaste mais ou menos tempo para ser executada do que a média e a mediana. Além disso, visto que a listagem de tempo gasto está ordenada em ordem crescente de tempo gasto, ao analisar o percentil, é possível expandir a análise de desempenho para mostrar até que ponto a aplicação pode ser executada com um tempo desejado ou adequado. Assim, pode-se analisar o comportamento da aplicação sob vários pontos de vista. Normalmente, o percentil é obtido a partir do valor de tempo gasto que corresponde a operação 95% e 99% da lista de tempo gasto. Por exemplo, na Tabela 13, o percentil 95 (p95) é o valor do tempo gasto da operação 9, isto é, 3 segundos. Já o percentil 99 (p99) é o valor do tempo gasto da operação 10, isto é, também 3 segundos.

Além da média, da mediana e do percentil, é importante destacar qual o tempo mínimo e máximo gasto para realizar uma operação, ou seja, qual o tempo máximo gasto (quão devagar) para realizar a operação desejada ou qual o tempo mínimo (quão rápido) para realizar a operação. Por exemplo: na Tabela 13, é possível analisar que o tempo mínimo gasto corresponde a operação 1, isto é, praticamente zero segundos, ao passo que o tempo máximo corresponde as operações 7, 8, 9 e 10, isto é, no máximo 3 segundos.

A partir desses conceitos e exemplos apresentados, é importante mostrar qual foi o tempo mínimo, a média, a mediana, o p95, o p99, o tempo máximo com base no experimento realizado no ambiente de pecuária leiteira. Vale lembrar que as análises apresentadas a seguir são baseadas nos dados reais, os quais foram gerados pelo ambiente IdC de pecuária leiteira e armazenados no DW pelo Imã de Dados. Contudo, foram gerados apenas 47 registros de coleta de volume de leite. Esse valor é considerado pequeno para mostrar o comportamento do Imã de Dados em relação aos requisitos de tempo real. Portanto, as análises a seguir são apenas para mostrar a viabilidade de se aplicar o Imã de Dados no ambiente de pecuária leiteira e também servir como fonte de comparação com os resultados obtidos a partir do experimento com dados sintéticos.

Figura 36 – Gráfico de tempo gasto para realizar as operações



A Figura 36 mostra o gráfico no qual é possível analisar as medidas de tempo mínimo (Min), média aritmética (Avg), p50, p95, p99 e tempo máximo (Max). Vale destacar que os valores analisados são compostos por todos os registros coletados, e todos os valores são medidos em segundos. O valor do Max é aproximadamente 3 segundos e o valor do Min é aproximadamente 0 segundo. Isso quer dizer que dentre todos os registros coletados, o maior tempo de resposta 3 segundos e o menor tempo de resposta foi de aproximadamente 0 segundo. O valor da média foi de 1,33 segundos, isto é, a media de tempo de resposta do experimento foi de aproximadamente 1,33 segundos. Já os tempos do p50, p95 e p99 foram respectivamente e aproximadamente 1, 2 e 2 segundos. Isso quer dizer que dentre todos os registros coletados, o tempo de resposta da operação 23 foi aproximadamente de 1 segundo e das operações 44 e 46

foi de aproximadamente 2 segundos.

Após mostrar o tempo gasto para realizar a execução do Imã de Dados no ambiente de pecuária leiteira e analisar esse tempo sob diferentes perspectivas, é importante destacar que o valor do tempo de resposta foi obtido conforme explicado no início desta seção. Contudo, percebe-se que o tempo de resposta é um valor inteiro. Esse é um outro fato que afeta negativamente a análise de resultados dos requisitos de tempo real, além do fato do baixo volume de dados coletados. Isso se deve ao fato de que, após intensas pesquisas sobre como lidar com números a partir do microcontrolador ESP32, não foi possível encontrar um modo de permitir números decimais e considerá-los no tempo de resposta. Mesmo assim, tal fato negativo não impediu a condução do experimento para mostrar a viabilidade do Imã de Dados no ambiente de pecuária leiteira, visto que o experimento com dados sintéticos também foi conduzido para suprir esses aspectos negativos identificados no experimento com dados reais.

Após analisar o tempo de resposta sob várias perspectivas, percebe-se que o Imã de Dados realiza suas tarefas em um tempo adequado para o ambiente de pecuária leiteira. Segundo o proprietário da fazenda utilizada no experimento, atualmente, o tempo gasto para analisar o volume de leite produzido por uma vaca, assim como detectar algum comportamento inesperado do animal, leva dias ou até mesmo semanas para ser percebido e alguma ação ser tomada. A partir do ambiente IdC criado e da aplicação do Imã de Dados, foi possível obter os dados de produção de leite e disponibilizar esses dados para o pecuarista tomar alguma decisão em aproximadamente 0 segundo e em aproximadamente 3 segundos. De acordo com o próprio produtor, esse tempo de resposta é considerado além do ideal para o contexto de pecuária leiteira, visto que um tempo de resposta adequado para esse cenário é em torno de 5 a 10 minutos. Mesmo assim, caso esse tempo de resposta não seja respeitado, nenhuma falha ou dado catastrófico ocorrera com as vacas ou com a fazenda como um todo. A partir desse cenário, o conceito de *soft real-time* é considerado no contexto de pecuária leiteira e consequentemente na aplicação do Imã de Dados. Além disso, visto que o não cumprimento do *deadline* imposto pelo ambiente não implicará em perdas financeiras ou em descarte de animais, pode-se concluir que o Imã de Dados cumpre seu objetivo e viabilidade para ser aplicado em um ambiente IdC de pecuária leiteira.

É importante destacar que esse pequeno atraso entre a solicitação de algum serviço e sua execução pode ser encontrado em diferentes ambientes utilizados todos os dias atualmente e pode ser considerado normal. Por exemplo: 1) um documento no Google Docs pode demorar em torno de 1 a 2 segundos para ser salvo; 2) uma mensagem enviada de uma pessoa para outra no aplicativo WhatsApp pode demorar 1, 2, 3 ou mais segundos para ser entregue. Todos esses exemplos ocorrem em todo o mundo todos os dias. Esse atraso na execução da operação é considerado adequado e não inviabiliza a utilização de tais aplicativos. Isso ocorre devido a diversos fatores, tais como *hardware* disponível, infraestrutura de rede, conexão à Internet adequada para executar tais serviços ou outros fatores estruturais que afetam desde a solicitação até a entrega de um determinado serviço.

6.3.5 Aplicação do Imã de Dados com dados sintéticos

Como já explicado ao longo da seção 6.3.4, o experimento com dados reais foi conduzido para validar a viabilidade de se aplicar o Imã de Dados em um ambiente real, ou seja, mostrar a correta execução pelo Imã de Dados de um *workflow* de ETL em tempo real. Contudo, o mesmo experimento não foi suficiente para validar três principais aspectos: 1) o comportamento do Imã de Dados a medida que aumenta a frequência e o volume de dados gerado; 2) os requisitos de tempo real ao se aplicar o Imã de Dados; 3) comparar o desempenho do Imã de Dados com a técnica de gatilho, considerada estado da arte no processo de extração de dados do processo ETL em tempo real. Caso esse experimento com dados sintéticos não fosse conduzido, as seguintes dúvidas poderiam surgir por parte do leitor: como é o desempenho do Imã de Dados ao aumentar de 3 para 100, 500, 1000 vacas enviando dados ao mesmo tempo? Como é o desempenho do Imã de Dados se a frequência de envio de dados for de alguns segundos? A partir do momento em que há um aumento linear no volume de dados gerado simultaneamente, como é a escalabilidade do Imã de Dados? O Imã de Dados consegue obter um bom desempenho com um grande volume de dados sendo gerado da mesma forma que se comporta com um pequeno volume de dados? O quão bom é o Imã de Dados em relação ao estado da arte em processo ETL em ambientes de *data warehousing* em tempo real?

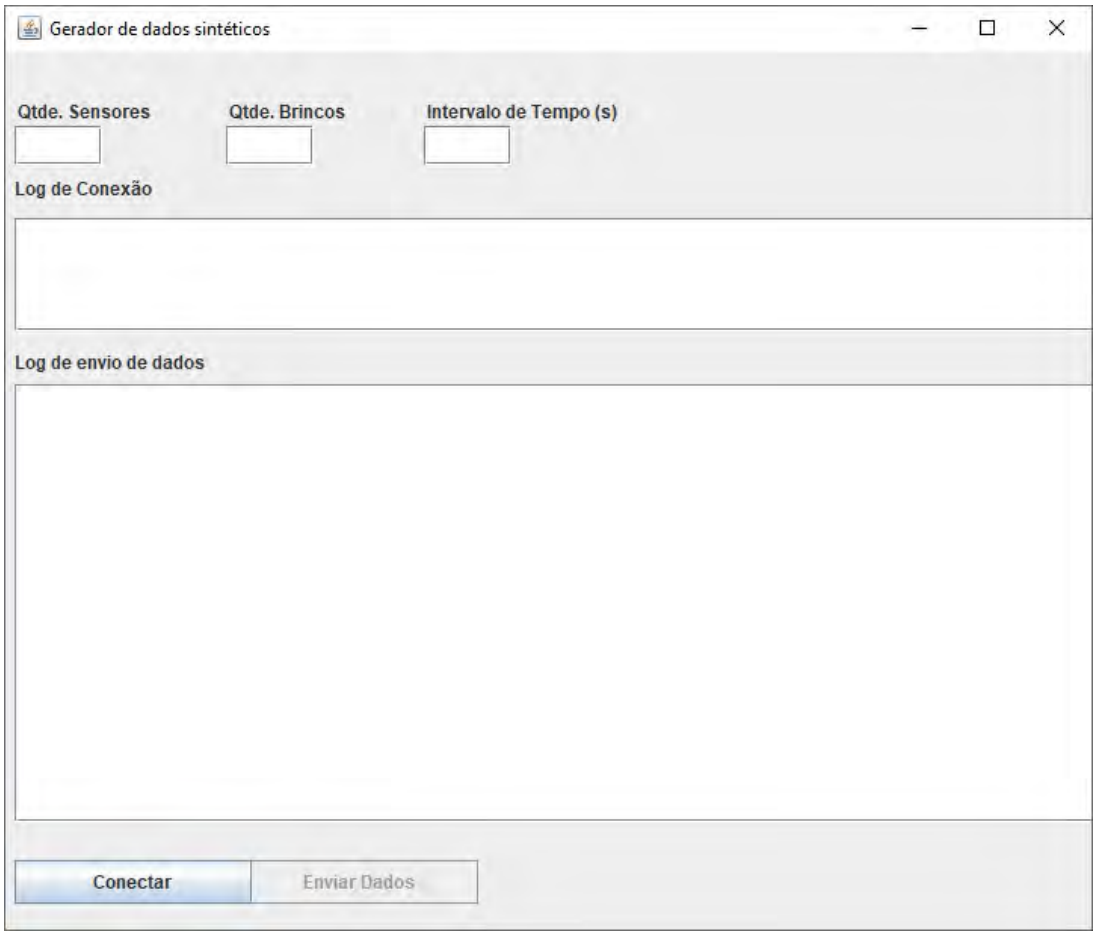
A partir desses e outros questionamentos que podem surgir, percebe-se que o experimento com dados reais realmente não foi suficiente para responder as dúvidas sobre como os requisitos de tempo real influenciam o Imã de Dados. Dessa forma, foi conduzido um experimento com dados sintéticos, cujo o objetivo foi validar os requisitos de tempo real do Imã de Dados, de modo que possa ser mostrado o comportamento do Imã de Dados ao aumentar o número de sensores, a frequência de geração dos dados e consequentemente o volume de dados gerado. É importante destacar que os dados sintéticos são dados gerados artificialmente (por meio de um *software*), mas compatíveis com dados reais. Os dados sintéticos são compostos pelos mesmos atributos (características) dos dados reais (conforme explicado na seção 6.3.1), mas com valores de campos gerados por um gerador de dados sintéticos. Esse fato faz com que seja possível manter as propriedades dos dados reais gerados sinteticamente, mas pode-se controlar a frequência e o volume de dados gerado simultaneamente.

6.3.5.1 Configuração do ambiente para gerar os dados sintéticos

Para gerar os dados sintéticos, foi desenvolvido um *software* chamado Gerador de Dados Sintético (GDS), como pode ser observado na Figura 37. A partir do GDS é possível configurar a quantidade de vacas sintéticas, a quantidade de sensores sintéticos e a frequência na qual os dados sintéticos são gerados. A partir dessa configuração, deve-se conectar ao *broker* do pub/sub e clicar no botão Enviar Dados. Dessa forma, os dados sintéticos são gerados automaticamente e infinitamente, ou até o usuário clicar no botão Desconectar para abortar o processo de geração dos dados. É importante destacar que no experimento com dados sintéticos, o GDS assume o

papel de ambiente operacional e é o responsável por gerar os dados do ambiente operacional sinteticamente.

Figura 37 – Gerador de dados sintéticos



O GDS foi desenvolvido com base na linguagem de programação Java e assim como no experimento com dados reais, foi utilizada a implementação Mosquitto como sistema pub/sub. Além disso, para ser possível gerar dados de N vacas simultaneamente, foi utilizada a técnica de programação paralela *ExecutorService*. O *ExecutorService* foi aplicado para simular o ambiente no qual há várias vacas enviando dados de leite paralelamente. Por exemplo: pode-se configurar a geração de dados de leite sintéticos de 50 vacas ao mesmo tempo. Logo, pode-se ter 50 *ExecutorService* gerando dados de leite sintéticos para cada vaca sintética simultaneamente. Vale destacar que o próprio *ExecutorService* é responsável por gerenciar a quantidade de processos paralelos criados e alocados em memória. Dessa forma, sempre que um processo paralelo estiver fora de uso, o *ExecutorService* desaloca-o da memória. Caso haja a necessidade de criar novos processos paralelos para lidar com o aumento no volume de dados gerado, o próprio *ExecutorService* cria os novos processos paralelos automaticamente.

Para criar o ambiente para gerar os dados sintéticos, foram utilizados dois computadores, um para executar o GDS e outro para executar o Imã de Dados. Esse ambiente foi necessário para simular o ambiente sintético mais próximo possível do ambiente real. Um computador gera

os dados sintéticos (ambiente operacional), envia os dados para o *broker* do pub/sub. Em outro computador, o Imã de Dados é executado e recebe esse dado gerado (ambiente informacional). Quando se executa esse ambiente, tem-se um ambiente de geração de dados sintéticos, o qual mantém as características do ambiente real, mas pode-se controlar o volume e frequência em que um dado é gerado.

O computador no qual o Imã de Dados foi executado tem as seguintes configurações: MacBook Pro, sistema operacional MacOS Catalina 10.15.7, 1.4 Intel Core i5, 8 GB de memória RAM. Já o computador no qual foi executado o GDS tem as seguintes configurações: Dell Inspiron 15R, sistema operacional Windows 10 64 bits, Intel Core i5, 8GB de memória RAM. Além dos dois computadores as seguintes tecnologias também foram utilizadas: banco de dados MySQL em nuvem para representar o DW, banco de dados MySQL local para representar o repositório de metadados, Eclipse Mosquitto como sistema pub/sub, sinal de internet de 10MB residencial compartilhada e roteador TP-Link TL-WR840N.

Assim como no experimento com dados reais, no experimento com dados sintéticos o Imã de Dados também deve ser configurado. As configurações são semelhantes às configurações com dados reais, porém os provedores de dados são diferentes, isto é, o GDS gera os dados sintéticos que serão enviados para o DW na nuvem. Dessa forma, é necessário configurar as *tags*, dados de interesse no gerador de dados (e não mais no arquivo JSON gerado pelo ambiente IdC), o DW de interesse, o DW e suas respectivas *tags* de interesse e os metadados de conexão com o DW.

Para realizar o experimento com a técnica de gatilho para permitir ser comparada com o Imã de Dados, foi necessário criar uma ferramenta ETL para atuar como processo ETL, a qual foi criada a partir da linguagem de programação Python. Essa ferramenta executa a fase de extração, isto é, busca os dados que foram gerados pelo gatilho e os envia para o DW. Além disso, foi necessário adaptar a ferramenta GDSG (chamada de GDS-Gatilho) para que a mesma possa gerar os dados sintéticos, enviar para um banco de dados local e consequentemente ser consumido pelo gatilho. Portanto, a seguinte configuração foi realizada: 1) o Imã de Dados foi executado no computador MacBook Pro, sistema operacional MacOS Catalina 10.15.7, 1.4 Intel Core i5, 8 GB de memória RAM; 2) o GDSG foi executado no computador Dell Inspiron 15R, sistema operacional Windows 10 64 bits, Intel Core i5, 8GB de memória RAM; 3) banco de dados MySQL em nuvem para representar o DW; 4) banco de dados MySQL local para armazenar os dados gerados pelo GDSG. Além disso, na tabela criada no banco de dados MySQL local foi criado um gatilho que envia os novos dados inseridos da tabela para a ferramenta de ETL; 5) ferramenta ETL; 6) sinal de internet de 10MB residencial compartilhada e roteador TP-Link TL-WR840N.

A partir das configurações realizadas no ambiente para gerar dados sintéticos, foi conduzido o experimento cujo o objetivo é validar os requisitos de tempo real a partir do aumento no número de sensores sintéticos e no aumento da frequência na geração de dados. As seções a

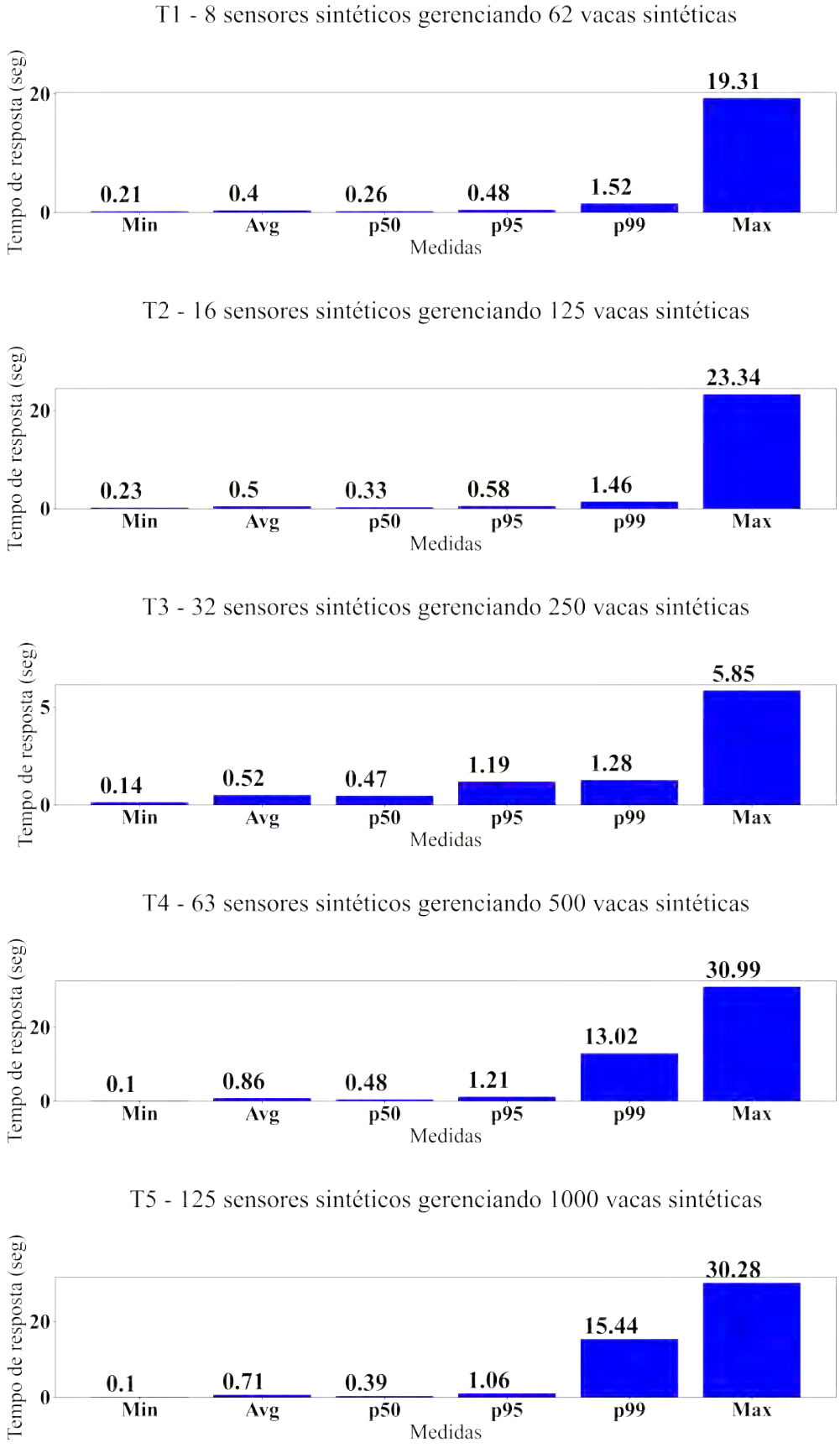
seguir retratam de forma detalhada cada teste realizado e a comparação entre o Imã de Dados e a técnica de gatilho.

6.3.5.2 Análise da escalabilidade do Imã de Dados ao aumentar o número de sensores sintéticos

O primeiro teste realizado teve por objetivo validar a escalabilidade do Imã de Dados com base no tempo de resposta à medida em que se aumentou o número de sensores sintéticos enviando dados simultaneamente. Para isso, os seguintes testes foram definidos no GDS: T1) 8 sensores sintéticos enviando dados de 62 vacas sintéticas simultaneamente; T2) 16 sensores sintéticos enviando dados de 125 vacas sintéticas simultaneamente; T3) 32 sensores sintéticos enviando dados de 250 vacas sintéticas simultaneamente; T4) 63 sensores sintéticos enviando dados de 500 vacas sintéticas simultaneamente; T5) 125 sensores sintéticos enviando dados de 1000 vacas sintéticas simultaneamente. Além disso, a frequência na geração de dados foi a cada 30 segundos. Vale destacar que cada um dos testes foi executado durante aproximadamente 1 hora, totalizando aproximadamente 5 horas de testes com essas configurações.

É importante destacar que essas configurações de teste definidas para o experimento sintético refletem o cenário atual das fazendas produtoras de leite. Atualmente, existem fazendas leiteiras com cerca de 600 vacas em lactação gerenciadas por até 8 sensores. Entretanto, apenas de considerar esse cenário, o experimento realizado foi além, isto é, foi considerado um pior caso de 125 sensores enviando dados de 1000 vacas simultaneamente.

Figura 38 – Tempo de resposta a medida em que aumenta o número de sensores sintéticos enviando dados simultaneamente



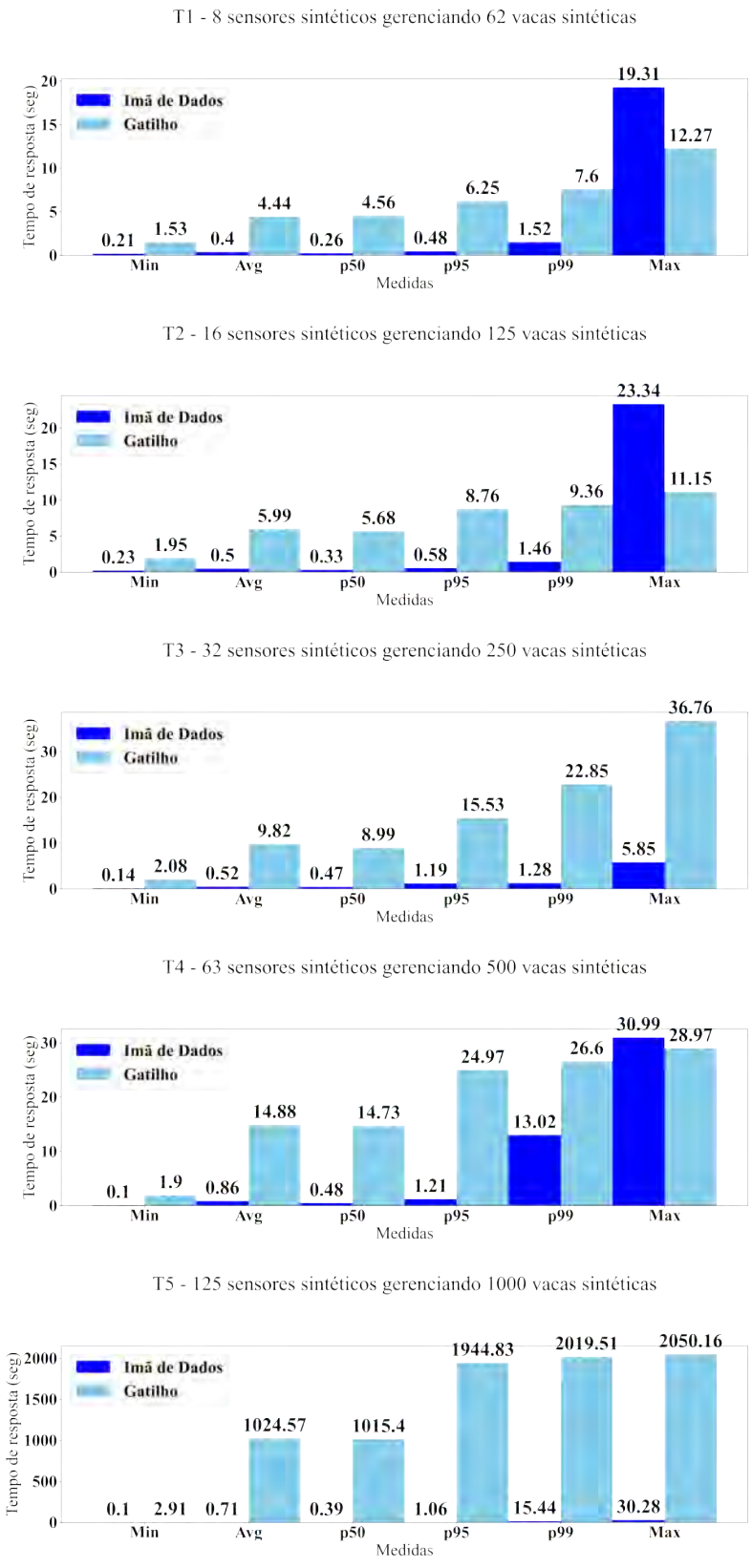
A Figura 38 mostra o resultado do tempo de resposta obtido pelo Imã de Dados à medida em que se aumentou o número de sensores sintéticos enviando dados simultaneamente. A figura é composta por 5 sub-figuras, cada qual representa o resultado do tempo de resposta de um teste definido e nomeado de acordo com o nome atribuído no início desta sub-seção.

Ao final dos 5 testes, foram coletados 27.668 registros de dados de leite sintéticos. Como pode-se notar, o valor de Max nos testes T1, T2, T4 e T5 permaneceu entre 19,31 e 30,99 segundos. Em percentual, esse valor representa apenas 0,57% de todos os dados coletados. Além disso, o valor de Max para o teste T3 foi de 5,85 segundos. Em relação ao valor de Min, em todos os testes esse valor permaneceu homogêneo e não sofreu uma variação impactante, isto é, o valor permaneceu entre 0,1 segundo e 0,23 segundos. Em percentual, esse conjunto de valores representa 11,2% de todos os valores coletados. Em relação ao valor de Avg, o maior valor obtido foi de 0,86 segundos, ao passo que o menor valor obtido foi de 0,40 segundos. Isso quer dizer que a medida em que aumentou o número de sensores, o tempo de resposta do Imã de Dados permaneceu estável. É importante dizer que esperava-se obter um crescimento linear no tempo de resposta e consequentemente no desempenho do Imã de Dados. Contudo, o que se pode observar é que à medida em que aumentou o número de sensores enviando dados simultaneamente e consequentemente o volume de dados, o desempenho do Imã de Dados permaneceu estável.

Em relação ao percentil, o valor de p50 e p95 permaneceu estável, isto é, esses valores permaneceram entre 0.26 segundos e 1,22 segundos. Isso quer dizer que 95% das operações teve o tempo de resposta máximo menor que 1,22 segundos. Em relação ao p99, para T1 o valor de p99 foi de 15,44 segundos e para T2 o valor foi de 13,02 segundos. Isso quer dizer que 99% das operações realizadas por esses dois testes teve seu tempo de resposta máximo abaixo de 15,44 segundos. Além disso, para os testes T3, T4 e T5, o valor de p99 foi 1,28 segundos, 1,46 segundos e 1,52 segundos respectivamente. Isso quer dizer que 99% das operações realizadas por esses três testes teve seu tempo de resposta máximo abaixo de 1,52 segundos. A variação no valor do p99 pode ser explicada devido ao *overhead* implícito na troca de dados em sistemas *on-line*. Particularmente, a oscilação no sinal de Internet é o motivo principal pelo qual ocorre a variação no valor de p99.

Após os testes apresentados, pode-se notar que o Imã de Dados permaneceu estável e com o tempo de resposta homogêneo ao longo do experimento, visto que o desvio padrão permaneceu estável para todos os testes. Além disso, alguns valores fora do padrão são relacionados a um número pequeno de operações que ocorreram em momentos em o sinal de Internet estava intermitente. Entretanto, assim como em todos os ambientes que trocam dados por meio da Internet, em alguns momentos é esperado alguma oscilação no sinal de Internet e consequentemente um aumento no tempo de resposta em operações pontuais. Esse fato pode ser notado na Figura 38 no valor de p99.

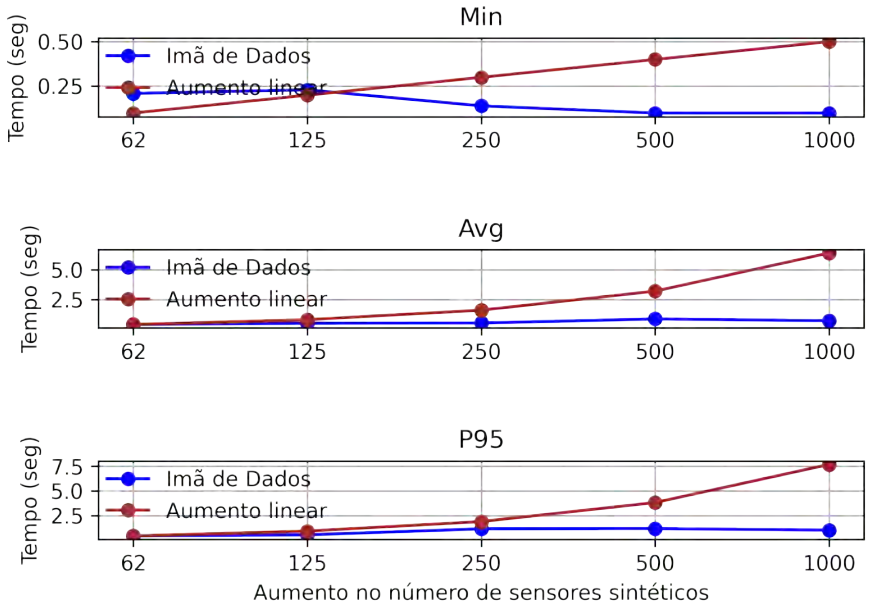
Figura 39 – Comparação do tempo de resposta entre o Imã de Dados e a técnica de gatilho a medida em que aumenta o número de sensores sintéticos enviando dados simultaneamente



A Figura 39 mostra a comparação do tempo de resposta entre o Imã de Dados e a técnica de gatilho a medida em que aumenta o número de sensores sintéticos enviando dados simultaneamente. A figura é composta por 5 sub-figuras, cada qual representa o resultado da comparação do tempo de resposta de um teste definido e nomeado de acordo com o nome atribuído no início desta sub-seção. Além disso, a cor azul escuro representa o Imã de Dados e a cor azul claro representa a técnica de gatilho.

A partir dos resultados obtidos após os testes realizados tanto com o Imã de Dados quanto com a técnica de gatilho, pode-se comparar o desempenho de ambos a partir do conceito de ganho de desempenho e perda de desempenho. Como pode-se notar na Figura 39, em todos os testes e em todas as medidas, o Imã de Dados obteve um ganho de desempenho sobre à técnica de gatilho. Em relação a média, o ganho de desempenho do Imã de Dados em relação ao gatilho variou entre 987% e 238,803%. Para o p50, o ganho variou entre 1,588% e 416,877%. Para o p95, o ganho variou entre 1,196% e 302,040%, e para o p99 o ganho variou entre 104% e 21,522%. Contudo, apenas para o valor de Max, o Imã de Dados obteve uma perda de desempenho para os testes T1, T2 e T4. Essa perda de desempenho variou entre 6% e 109%. Essa perda de desempenho para o valor de Max em alguns testes é devido ao *overhead* implícito imposto pelo próprio ambiente. Entretanto, essa perda de desempenho representa uma minúscula quantidade de registros coletados ao final do experimento.

Figura 40 – Comparação entre o Imã de Dados e o aumento linear esperado no tempo de resposta a medida em que aumenta o número de sensores sintéticos enviando dados simultaneamente



A Figura 40 mostra a comparação entre o Imã de Dados e o aumento linear esperado

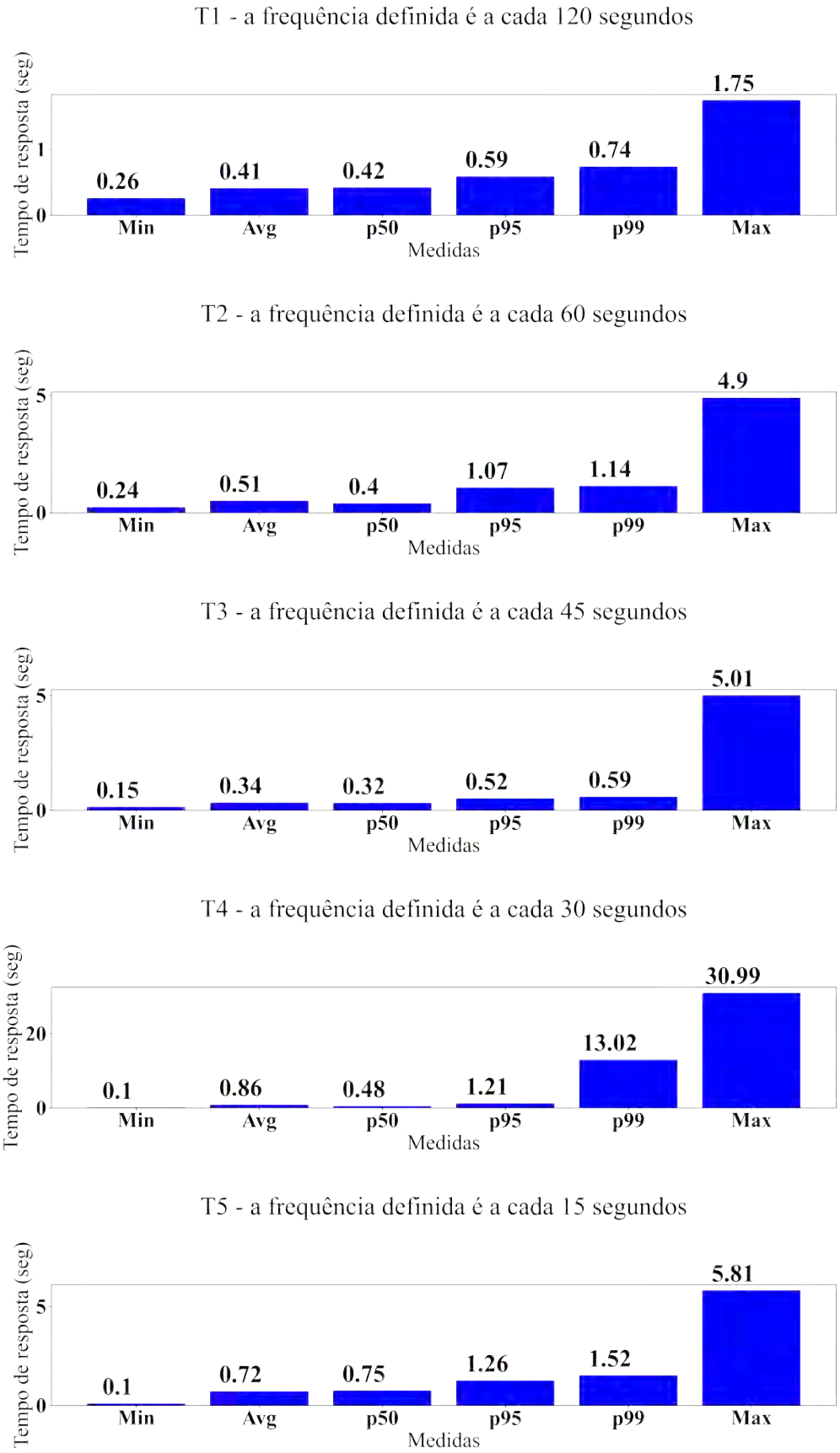
no tempo de resposta a medida em que aumenta o número de sensores sintéticos enviando dados simultaneamente. Como pode-se notar, a medida que aumenta a quantidade de sensores sintéticos enviando dados simultaneamente, o Imã de Dados apresenta um comportamento e tempo de resposta estável. Além disso, a medida que dobra-se a quantidade de sensores, era esperado que o tempo de resposta do Imã de Dados também dobrasse ou pelo menos um aumento linear no tempo de resposta. Isso devido ao comportamento da técnica de gatilho com a mesma configuração. Contudo, pode-se notar um ótimo comportamento e um pequeno aumento no tempo de resposta. Esse fato pode ser constatado a partir do valor das medidas da média, p50, p95 e p99 da Figura 38 e da Figura 39. Por outro lado, o gatilho apresentou uma grande mudança no comportamento e um maior tempo de resposta do que o Imã de Dados.

6.3.5.3 Análise da escalabilidade do Imã de Dados ao aumentar a frequência na geração de sintéticos

O segundo teste realizado teve por objetivo validar a escalabilidade do Imã de Dados com base no aumento da frequência na geração de dados sintéticos. Para isso, os seguintes testes foram definidos no GDS: T1) a frequência na geração dos dados é de 120 segundos; T2) a frequência na geração dos dados é de 60 segundos; T3) a frequência na geração dos dados é de 45 segundos; T4) a frequência na geração dos dados é de 30 segundos; T5) a frequência na geração dos dados é de 15 segundos. Além disso, foram fixados o número de sensores sintéticos enviando dados simultaneamente e o número de vacas sintéticas em 63 e 500 respectivamente. Vale destacar que cada um dos testes foi executado durante aproximadamente 1 hora, totalizando aproximadamente 5 horas de testes com essas configurações.

Assim como explicado na seção 6.3.5.2, essas configurações definidas no GDS refletem o cenário atual encontrado em fazendas produtoras de leite. Atualmente, existem fazendas que produzem leite de até 60 vacas em sequência, a partir de 8 sensores e com frequência de 5 a 10 minutos. Contudo, apesar de considerar o cenário atual, o experimento realizado foi além, ou seja, foi realizado testes com uma frequência maior do que a frequência encontrada no cenário atual.

Figura 41 – Tempo de resposta a medida em que aumenta a frequência na geração de dados



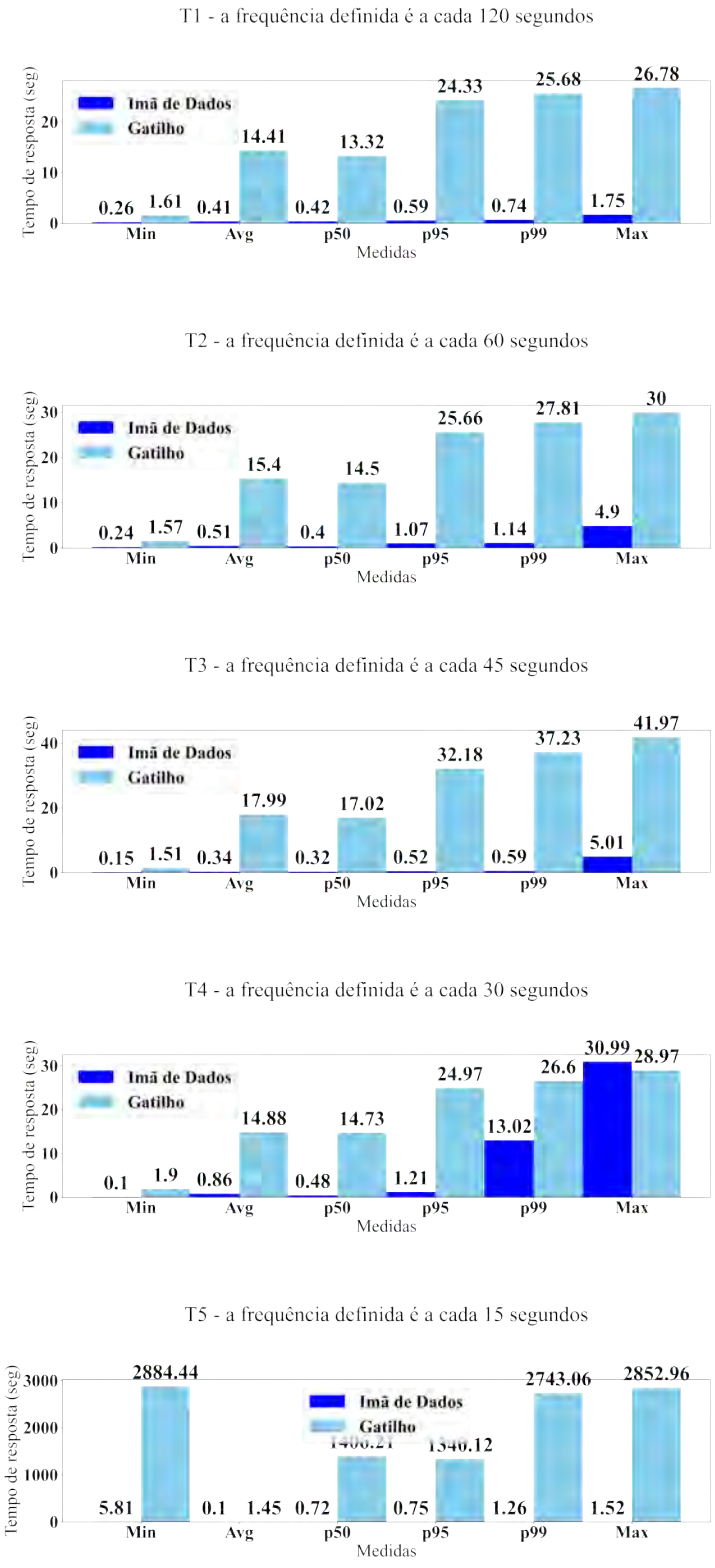
A Figura 41 mostra o tempo de resposta do Imã de Dados a medida que aumenta a frequência na geração de dados. A figura é composta por 5 sub-figuras, cada qual representa o resultado do tempo de resposta de um teste definido e nomeado de acordo com o nome atribuído no início desta sub-seção.

Ao final dos 5 testes, foram coletados 37.494 registros de dados de leite sintéticos. Como pode-se notar, o valor de Max para os testes T1, T3, T4 e T5 permaneceu entre 1,75 segundos e 5,81 segundos. Apenas o teste T4 apresentou um valor maior que 30,99 segundos. Em percentual de operações realizadas, esse valor representa apenas 0,52% de todos os dados coletados de todos os testes. Em relação ao valor de Min, em todos os testes esse valor se mostrou estável, isto é, permaneceu entre 0,1 segundo e 0,26 segundos. Em percentual de operações realizadas, esse valor representa 11,6% de todos os valores coletados de todos os testes. Em relação ao valor de Avg, o valor menor da média foi de 0,34 segundos, ao passo que o maior valor da média foi de 0,86 segundos. O valor da média mostra que o desempenho do Imã de Dados permaneceu estável durante todos os testes realizados.

Em relação ao percentil, apenas o teste T2 mostrou um valor de p99 acima de 1,6 segundos. Isso ocorreu devido ao *overhead* imposto pelo próprio ambiente e pelo sinal de Internet intermitente. Contudo, o valor de p50, p95 e p99 dos testes T1, T3, T4 e T5, e o valor de p50 e p95 do teste T2 permaneceu estável a medida em que aumenta a frequência na geração de dados. Em outras palavras, o valor máximo de tempo de resposta de 99% de todas as operações realizadas foi abaixo de 13,02 segundos, ao passo que o valor mínimo de tempo de resposta de 99% das operações foi de 0,59 segundos.

Apesar do fato do Imã de Dados obter no teste T2 o valor de Max igual a 30 segundos e o valor de p99 igual a 13,02 segundos, pode-se notar que o comportamento do Imã de Dados permaneceu estável e o tempo de resposta foi homogêneo ao longo do experimento, como pode ser notado nos valores do desvio padrão e variância de todos os testes. Além disso, esses valores fora do padrão são relacionados a um pequeno número de operações, as quais foram realizadas em um momento no qual houve oscilação no sinal de Internet e também devido ao próprio *overhead* imposto pelo ambiente. Além do mais, mesmo utilizando a Internet e um sistema pub/sub como meio de compartilhamento de dados entre ambientes, o Imã de Dados obteve um bom tempo de resposta, até abaixo do esperado.

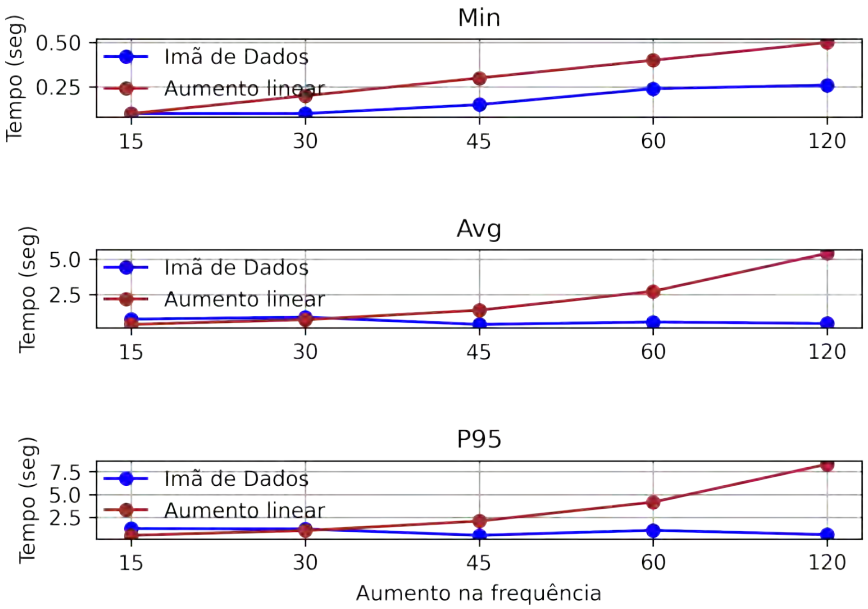
Figura 42 – Comparação do tempo de resposta entre o Imã de Dados e a técnica de gatilho a medida em que aumenta a frequência na geração de dados



A Figura 42 mostra a comparação do tempo de resposta entre o Imã de Dados e a técnica de gatilho a medida em que aumenta a frequência na geração de dados. A figura é composta por 5 sub-figuras, cada qual representa o resultado da comparação do tempo de resposta de um teste definido e nomeado de acordo com o nome atribuído ao teste no início desta sub-seção. Além disso, a cor azul escuro representa o Imã de Dados e a cor azul claro representa a técnica de gatilho.

A partir dos resultados mostrados na Figura 42, os quais analisam a influência do aumento da frequência na geração de dados no Imã de Dados, pode-se analisar a diferença entre o Imã de Dados e o gatilho a partir do ganho de desempenho e perda de desempenho. Para todos os testes e todas as medidas, o Imã de Dados obteve um ganho de desempenho em relação ao gatilho. Em relação a média, o ganho de desempenho do Imã de Dados em relação ao gatilho variou entre 1,626% e 320,756%. Para o p50, o ganho de desempenho variou entre 2,924% e 295,129%. Para o p95, o ganho de desempenho variou de 1,952% e 359,604%, e para o p99 o ganho de desempenho variou entre 104% e 310,804%. Contudo, apenas para o valor Max do teste T2, o Imã de Dados apresentou uma perda de desempenho de 6% em relação ao gatilho. Esse valor representa uma minúscula quantidade de operações realizadas devido a oscilação de Internet e ao próprio ambiente.

Figura 43 – Comparação entre o Imã de Dados e o aumento linear esperado no tempo de resposta a medida em que aumenta a frequência na geração de dados



A Figura 43 mostra a comparação entre o Imã de Dados e o aumento linear esperado no tempo de resposta a medida em que aumenta a frequência na geração de dados. Vale destacar que era esperado um aumento linear no tempo de resposta do Imã de Dados, visto que para a técnica

de gatilhos houve esse aumento a medida em que se dobra a quantidade de vacas e sensores sintéticos. Contudo, após os testes comparativos, pode-se notar que o Imã de Dados obteve ótimo desempenho e um tempo de resposta estável a medida em que aumenta a frequência na geração de dados. Esses fatos podem ser observados na Figura 41 a partir do valor das medidas de média, p50 e p95. Em relação ao valor de Min, mesmo no teste T5 em que a frequência na geração dos dados é de apenas 15 segundos, o Imã de Dados obteve ótimo desempenho e baixo tempo de resposta como no teste T1, em que a frequência dos dados é de 120 segundos. Por outro lado, a técnica de gatilho mostrou ter um aumento linear no tempo de resposta e consequentemente acima do tempo de resposta do Imã de Dados. Esse fato pode ser notado na Figura 41 a partir das medidas da média, p95 e p99. Além do mais, em todos os testes, a técnica de gatilho não apresentou um comportamento e um tempo de resposta nem próximo ao Imã de Dados.

6.4 Considerações finais

Neste capítulo foi apresentado os testes experimentos realizados com o Imã de Dados, tanto com dados reais quanto com dados sintéticos. Foi necessário realizar dois tipos de testes: o primeiro teste foi conduzido com dados reais, por meio do qual foi possível mostrar a viabilidade do Imã de Dados em ser aplicado em um ambiente no qual possui as características de um ambiente IdC. Os resultados obtidos mostraram que, do ponto de vista dos tomadores de decisão, é altamente importante a aplicação do Imã de Dados nesse ambiente real, pois os dados de pesagem de leite das vacas podem ser consumidos pelos produtores e/ou veterinários em tempo real e alguma ação preventiva pode ser tomada imediatamente. Do ponto de vista comercial, é importante a aplicação do Imã de Dados em um ambiente IdC de baixo custo, visto que as soluções modernas e atuais são muito caras e inviáveis para grande parte das fazendas produtoras de leite.

O segundo teste foi conduzido com dados sintéticos, por meio do qual foi possível analisar o Imã de Dados sob diferentes perspectivas e validar os requisitos de tempo real. Além disso, foi possível controlar o número de sensores, a frequência na geração dos dados e por conseguinte o volume de dados, ou seja, verificar a influência do aumento dessas características no desempenho do Imã de Dados. Além de validar os requisitos de tempo real, o segundo teste foi importante para comparar o desempenho do Imã de Dados com a técnica de gatilho, a qual é considerada o estado da arte do ponto de vista de extração de dados do processo ETL em tempo real em ambientes de *data warehousing*. Os resultados mostraram que o Imã de Dados conseguiu obter um bom tempo de resposta e garantir escalabilidade mesmo nos piores casos, isto é, com um volume de dados muito maior que o volume de dados previsto para um ambiente real. Além disso, os resultados mostraram que o desempenho, a escalabilidade e o tempo de resposta do Imã de Dados são melhores do que a técnica de gatilho.

Os resultados positivos alcançados pelo Imã de Dados foram possíveis devido os concei-

tos aplicados em seu projeto. A aplicação dos conceitos de não intrusivo e reativo, do conceito de *tag*, da ideia de desacoplar o processo ETL dos ambientes operacional e informacional, do conceito de paralelismo e do conceito de sistemas pub/sub fizeram com que o Imã de Dados alcançasse resultados positivos do ponto de vista de desempenho e escalabilidade. O trabalho em conjunto de todos esses componentes representam a forma de como o Imã de Dados conseguiu expressivos resultados no tempo de resposta, desempenho e escalabilidade. Além disso, o Imã de Dados representa uma mudança de paradigma ao se pensar e projetar o processo ETL em tempo real em ambientes de *data warehousing*.

No Capítulo 7, serão mostrados e discutidos os trabalhos relacionados ao tema deste trabalho.

Capítulo 7

TRABALHOS RELACIONADOS

Ao longo deste trabalho foi desenvolvida uma revisão sistemática (Apêndice A) para identificar o maior número de trabalhos relacionados a execução do processo ETL em tempo real em ambientes de *data warehousing*. Nesse sentido, foi possível identificar diversos produtos de pesquisa desenvolvidos, dentre eles, pode-se citar: métodos, técnicas, arquiteturas e estudos em geral. Cada tipo de trabalho aborda diferentes aspectos do processo ETL, por exemplo: 1) existem trabalhos que propõem estudos que comparam métodos já propostos na literatura para o processo ETL como um todo; 2) outros trabalhos propõem estudos de técnicas apenas para extração de dados; 3) alguns trabalhos propõem métodos para otimizar a extração e o carregamento de dados. Dessa forma, pode-se observar que cada trabalho propõe um tipo específico de abordagem, o que é útil para alcançar todos os aspectos envolvendo o processo ETL.

Portanto, para oferecer um melhor entendimento e organização deste trabalho, este capítulo foi dividido em três seções: na seção 7.1.1 serão apresentados os trabalhos cujo o foco é estudos em geral, isto é, apresentam um conteúdo que não envolve a proposta original de métodos, técnicas ou arquiteturas, mas sim a análise teórica do processo ETL sob diferentes perspectivas. Esses trabalhos basicamente realizam um *survey* das soluções propostas até o momento de sua publicação. Na seção 7.1.2 serão apresentadas as arquiteturas propostas na literatura que também abordam o mesmo problema de pesquisa. Na seção 7.2 será mostrada uma análise dos dados obtidos. Essa análise mostra as características dos trabalhos relacionados e suas diferenças com o Imã de Dados.

Vale a pena destacar que os trabalhos analisados durante toda essa seção foram obtidos por meio da revisão sistemática. Além disso, foram criadas questões de pesquisa no protocolo da revisão sistemática para auxiliar identificar os trabalhos relacionados. Dessa forma, essas questões de pesquisa serão respondidas ao longo da seção 7.2. Por fim, na seção 7.3 será mostrada uma análise final do resultado da revisão sistemática, assim como do estado da arte do processo ETL aplicado em ambientes de *data warehousing* em tempo real.

7.1 Trabalhos sobre o processo ETL em ambientes de data warehousing em tempo real baseado em abordagens tradicionais

7.1.1 Estudos

Vassiliadis e Simitsis (VASSILIADIS; SIMITSIS, 2009) citam a importância do surgimento do DW no contexto de negócios. A atualização de dados ocorria durante a janela de inatividade da aplicação, normalmente a cada vinte e quatro horas. Entretanto, os dados de negócios passaram a ser gerados por diferentes meios, sobretudo com o surgimento de sistemas *Web*. Assim, esses dados se tornaram úteis nas organizações para as tomadas de decisões de negócios, por exemplo: aumentar a quantidade de vendas, monitorar as compras dos clientes, oferecer serviços com base nas compras dos cliente, melhorar o controle de estoque e etc. Essa necessidade de tomadas de decisões resulta em uma necessidade de dados cada vez mais atualizados e acurados, isto é, as decisões devem ser tomadas com base nos dados gerados em tempo real e não com dados de dias anteriores.

Segundo os autores, diversas ferramentas foram propostas para resolver o problema de atualização de dados do DW. Essas ferramentas realizam o processo de atualização em aproximadamente quinze minutos. Dessa forma, sugere-se chamar essa solução de "em tempo quase real"e não "em tempo real". Entretanto, essas ferramentas foram desenvolvidas de maneira *ad-hoc*, sem seguir um padrão ou uma abordagem. Com isso, os autores desenvolveram um estudo cuja as contribuições foram: discutem os principais problemas do processo ETL convencional e citam os motivos que levam a necessidade de dados atualizados em tempo real. Além de proporem um estudo teórico sobre o tema, os autores descrevem uma arquitetura que permite executar o processo ETL em tempo real. A partir dessa arquitetura, os autores descrevem as atividades necessárias para permitir executar o processo ETL em tempo real.

A arquitetura proposta é composta pelo componente ***Source Flow Regulator*** (SFlowR), cuja a responsabilidade é identificar as modificações ocorridas nas fontes de dados de origens e encaminha-las para as fases posteriores. O componente ***Data Processing Flow Regulator*** (DPFlowR) é responsável por verificar quais fontes de dados estão prontas para transmitir dados. Uma vez feita essa verificação, os dados são encaminhados para o componente ***Data Processing Area*** (DPA), cuja a responsabilidade é manter os dados até que eles sejam transformados e limpos. Após esse processo, o componente ***Warehouse Flow Regulator*** (WFlowR) recebe os dados da DPA e, com base nas regras definidas pelos usuários, armazena-os no DW. Por fim, o ***data warehouse*** é composto pelas tabelas de fatos e dimensões, índices e visões materializadas para otimizar as consultas.

Além da arquitetura proposta, os autores expõem outras abordagens aplicadas para permitir integrar dados em tempo real. As abordagens estudadas foram: 1) ***Realtime partition***: consiste

em manter uma cópia da tabela de fatos, cuja a função é receber os dados de tempo real (dados dinâmicos). Com uma dada frequência, esses dados são integrados com os dados do DW (dados estáticos); 2) ***Enterprise Application Integration*** (EAI): com essa abordagem, é possível juntar as transações geradas por diferentes aplicações por meio de uma outra aplicação responsável por integrar essas aplicações; 3) ***Capture, Transform, Flow*** (CTF): essa abordagem é similar ao ETL, cuja a função é capturar os dados das fontes de dados operacionais, transforma-los e carrega-los para a partição de tempo real (dados dinâmicos); 4) ***Fast loading via Microbatch ETL***: essa abordagem auxilia o processo *Realtime partition*. Uma vez a aplicação se encontra ociosa, os dados da partição de tempo real são encaminhados para o DW; 5) ***On-demand reporting via Enterprise Information Integration*** (EII): essa abordagem representa virtualmente as fontes de dados heterogêneas em apenas uma fonte de dados homogênea. Por meio de consultas SQL, os dados são extraídos diretamente das fontes de dados de origem e apresentados ao usuário final.

Vale a pena destacar que, na arquitetura proposta, pode-se encontrar algumas questões relativas a cada uma das etapas que compõem a arquitetura. Dessa forma, ao final do trabalho, os autores expõem as questões técnicas sobre cada um dos componentes da arquitetura. Esse aspecto é importante pois permite entender como funciona cada etapa do processo ETL e quais os desafios superados para alcançar a execução de todas as etapas do processo ETL.

Penzlin et al. (PENZLIN et al., 2009) e Farooq e Sarwar (FAROOQ; SARWAR, 2010) citam a importância do DW para análises e tomadas de decisões de negócios. Entretanto, com a demanda cada vez maior de dados atualizados, o DW passou a ser atualizado numa frequência cada vez maior. Segundo Farooq e Sarwar (FAROOQ; SARWAR, 2010), aplicações compostas por sensores ou que detectam fraudes em cartão de crédito não permitem a latência com a qual o DW é atualizado no modo convencional. Contudo, Penzlin et al. (PENZLIN et al., 2009) dizem que determinadas aplicações não exigem necessariamente uma atualização em milésimos de segundos, ou seja, em diversos tipos de aplicações uma certa latência é permitida. Em todos os casos, essa demanda por dados atualizados resultou em dois desafios principais que devem ser superados: integrar dados em tempo real, isto é, a execução do processo ETL sempre que necessário (PENZLIN et al., 2009; FAROOQ; SARWAR, 2010), e a manutenção incremental do DW (FAROOQ; SARWAR, 2010). Por fim, Penzlin et al. (PENZLIN et al., 2009), além de fazer uma revisão da literatura sobre o estado da arte do processo ETL, citam os desafios futuros a serem superados e possíveis temas de pesquisa ao aplicar o processo ETL em ambientes de tempo real.

Já Farooq e Sarwar (FAROOQ; SARWAR, 2010) fizeram uma revisão da literatura sobre as tecnologias para se executar o processo ETL em tempo real. As técnicas identificadas são: ETL em tempo quase real, ETL com alimentação direta e ETL com armazenamento de dados em *cache*. Ambas as técnicas oferecem meios diferentes de permitir o processo ETL em tempo real, as quais já foram discutidas por Langseth (LANGSETH, 2004) e já comentadas na seção 4. Além disso, os autores analisaram duas técnicas para modelar o DW, que são modelagem

estruturada e semiestruturada. Por meio da implementação de um *framework*, os autores criaram um DW contendo dois modelos de dados: um relacional e outro baseado em XML. Para obter os dados operacionais, os autores utilizaram o mecanismo CDC, mas não especificaram qual técnica utilizada. Para permitir a atualização e consulta de dados no DW simultaneamente, os autores utilizaram um algoritmo de agendamento de tarefas baseada na técnica *First In First Out* (FIFO). Dessa forma, é possível definir prioridades e controlar a atualização e consulta dos dados.

Os testes realizados mostraram que, com o modelo semiestruturado, é possível obter um menor tempo de carregamento de dados utilizando a atualização incremental, ao passo que para o carregamento completo, ambas as abordagens apresentaram um tempo similar. Do ponto de vista de execução de consultas, o modelo semiestruturado obteve um melhor desempenho em relação ao modelo relacional. Entretanto, não foi possível identificar qual técnica CDC os autores utilizaram para extrair os dados operacionais. Esse aspecto afeta diretamente no desempenho e disponibilidade da proposta. Além disso, os autores não abordam questões de escalabilidade e disponibilidade.

Tank et al. (TANK et al., 2010) apresentam um estudo no qual é mostrado os componentes principais do processamento de dados no DW. Segundo os autores, as principais operações são: 1) junção: essa operação é responsável por integrar os dados de diferentes fontes de dados. Para isso, são aplicadas técnicas CDC que permite identificar e obter os dados operacionais; 2) agregação: essa operação é responsável por otimizar e permitir que consultas sejam executadas no DW com melhor desempenho.

O trabalho apresentado pelos autores se limita em mostrar apenas esses elementos citados. Além do mais, os autores não mostram a real proposta do trabalho, se é apresentar um estudo ou algum outro produto de pesquisa. Dessa forma, esse trabalho foi classificado como estudo, visto que o mesmo apresenta um conteúdo focado na parte teórica da fase de extração de dados do processo ETL. Além disso, nota-se que esse trabalho serviu como base para um outro trabalho do mesmo autor, o qual também foi apresentado nessa seção.

Kakish e Kraft (KAKISH; KRAFT, 2012) e Sabry e Ali (SABRY; ALI, 2014) argumentam que organizações de diversos setores de negócios devem se preocupar em utilizar dados de tempo real para as tomadas de decisões estratégicas, caso contrário perderão espaço para seus concorrentes. Isso se deve ao fato de que essas organizações geram dados a todo momento por meio de diversos meios. Além disso, essas informações são úteis para saber a real situação da empresa e realizar previsões de mercado. Dessa forma, o DW pode auxiliar o processo de armazenamento de dados históricos e de tempo real, bem como nas tomadas de decisões. Entretanto, as características convencionais do DW impedem sua utilização para o contexto em que os dados são gerados em tempo real. Assim, deve-se propor novos meios para que o DW seja atualizado em tempo real ou o mais rápido possível.

Com isso, Kakish e Kraft (KAKISH; KRAFT, 2012) desenvolveram um *survey*, no qual

fazem uma revisão do estado da arte de todas as fases do processo ETL, bem como sua evolução no tempo até ser possível aplicar o processo ETL em ambientes de tempo real. Além disso, expõem os atributos de qualidade desejáveis para os dados armazenados no DW. Juntamente com os estudos apresentados, os autores fazem algumas afirmações que podem auxiliar a entender como funciona a aplicação do processo ETL em ambientes de tempo real: 1) não é possível obter 100% de tempo real, pois questões técnicas podem impedir que os dados sejam atualizados em milésimos de segundos, ou seja, uma latência mínima é aceitável; 2) a fase de extração de dados é a mais crítica do processo ETL, visto que é essa fase a responsável por lidar com diversos tipos de fontes de dados. Por consequência dessa variedade de fontes de dados, acessar as fontes de dados e identificar os dados relevantes se torna uma tarefa custosa.

Por outro lado, Sabry e Ali (SABRY; ALI, 2014) apontam as principais características do DW, as formas de modelar suas tabelas e uma arquitetura básica que serve como base para explicação de todos os conceitos. Além do DW, os autores apresentam uma visão sobre o processo ETL convencional como um todo, assim como a evolução do processo. Por fim, os autores apontam algumas técnicas que podem ser aplicadas no processo ETL convencional para permitir que o processo ETL seja executado em tempo real.

Revathy et al. (S et al., 2013) argumentam que o DW foi criado para integrar dados originados de diferentes fontes de dados em uma fonte de dados homogênea. Entretanto, com a geração de dados cada vez mais em tempo real, surgiu a necessidade de novos estudos, sobretudo com o foco em *data stream*. Os autores definem *data stream* como uma sequência ordenada e contínua de dados. Com base nessas características, os autores mostram o conceito de *Data Stream Management System* (DSMS). O DSMS se assemelha ao *Database Management System* (DBMS), porém, tem o objetivo de gerenciar dados com essas características. Vários domínios são capazes de gerar *data stream*, tais como companhias que analisam tendências de mercado, sistemas de monitoramento e sistemas de monitoramento de trafego. Além disso, os autores apresentam os desafios que devem ser superados e as técnicas já existentes para permitir que o DW convencional possa gerenciar essas características dos dados.

Ferreira et al. (FERREIRA et al., 2013) argumentam que a demanda por dados cada vez mais atualizados fez com que o DW passasse a ser atualizado em tempo real. Entretanto, a aplicação do processo ETL convencional em atualizações em tempo real faz com que, entre diversos problemas, ocorra conflitos entre as consultas executadas no DW e o carregamento de dados operacionais, além de um aumento no tempo de execução de todo o processo. Com isso, algumas dúvidas quanto a aplicação do processo ETL em tempo real surgiram, tais como: 1) é possível alcançar a atualização em tempo quase real com a arquitetura do DW convencional? 2) Quais os fatores envolvidos para permitir a atualização em tempo real? 3) Como alcançar a atualização em tempo real?

Com base no contexto apresentado e nos questionamentos levantados, os autores desenvolveram uma avaliação experimental, cujo o objetivo foi analisar os fatores que influenciam na

execução do processo ETL em tempo real. Os fatores analisados foram: 1) método de carregamento de dados; 2) execução de consultas; 3) indexação; 4) integridade referencial; 5) atualização de visões materializadas; 6) particionamento de tabelas. A partir desses fatores, foram realizados vários experimentos para mostrar sua influência na execução do processo ETL em ambientes de tempo real.

Por fim, os autores concluem que, em ambientes de tempo real, o desempenho das consultas executadas no DW e o carregamento dos dados é bastante afetado, em comparação com o mesmo processo executado *offline*. Entretanto, os autores argumentam que o processo ETL pode ser executado em ambientes de tempo quase real se permitirem o particionamento de dados e aceitarem um período mínimo de inatividade do DW. Contudo, os autores não apresentam resultados com base em outras características de ambientes de tempo real, tais como escalabilidade e qualidade dos dados. Além disso, foi analisado um ambiente de tempo quase real, isto é, um ambiente que permite uma latência de aproximadamente cinco minutos. Em ambientes cuja a latência deve ser em segundos ou o mais rápido possível, tal como ambientes que lidam com sensores, esse tempo de inatividade não é permitido.

Wibowo, A. (WIBOWO, 2015) diz que o processo ETL convencional deve ser executado em um período em que o ambiente operacional e o ambiente informacional esteja fora de uso. Caso haja a necessidade de executar o processo ETL fora do período especificado, ambos os ambientes devem ser desativados. Esse aspecto se torna um grave problema em ambientes que devem operar sete dias por semana e vinte e quatro horas por dia.

Nesse cenário, surgiu o conceito de DW em tempo real, que consiste em, entre vários fatores, executar o processo ETL continuamente. Desse modo, os dados do DW sempre estarão atualizados. Entretanto, diversos problemas surgiram em cada fase do processo ETL ao executá-lo em ambientes de tempo real. Com isso, o autor propõe um estudo em que apontam os problemas identificados em cada fase do processo ETL, assim como as soluções já desenvolvidas. Segundo o autor, esse trabalho auxiliará a pesquisa de outros pesquisadores no futuro, pois permitirá identificar os problemas e soluções já desenvolvidas em cada fase do processo ETL.

A Tabela 14 mostra os problemas encontrados ao executar o processo ETL em ambientes de tempo real, assim como as possíveis soluções. Na fase de extração, os dados podem estar distribuídos em fontes de dados heterogêneas, o que torna um desafio acessar essas fontes de dados. Além disso, as fontes de dados do ambiente operacional servem a grupos de usuários que realizam operações de inserção e alteração nos dados. Esse fato resulta em uma sobrecarga nas fontes de dados ao ser acessadas para extrair os dados. Na fase de transformação, é necessário realizar operações com os dados armazenados nas tabelas de dimensões. Isso faz com que seja necessário uma fonte de dados extra para lidar com essas transformações. Entretanto, em um ambiente de tempo real, essa transformação deve ocorrer o mais rápido possível com um grande volume de dados envolvido. Como solução para esses problemas, deve-se armazenar os dados temporariamente em *cache*. Na fase de carregamento de dados, pode haver perda de desempenho

Tabela 14 – Problemas e soluções identificados em cada fase do processo ETL

Fase	**Problema**	**Solução**
Extração	Integrar fontes de dados heterogêneas	Aplicar técnicas CDC.
	Sobrecarga nas fontes de dados	Definir prioridades para os dados ou extrair os dados a partir de *log*.
Transformação	Sobrecarga nas tabelas de dimensões	Colocar os dados de dimensões em *cache*.
	Fonte de dados adicional para transformação	Usar a abordagem ELT.
Carga	Perda de desempenho	Armazenar dados de tempo real em tabelas temporárias.
	Inconsistência nas consultas	Utilizar a técnica *snapshot* em conjunto com a técnica RTDC.

Fonte: (WIBOWO, 2015)

nas consultas quando uma consulta é realizada ao mesmo tempo que ocorrem atualizações nos dados. Além disso, as consultas podem retornar dados inconsistentes, pois podem ocorrer atualizações nos dados transacionais e que ainda não foram atualizados no DW. Para diminuir esses problemas, deve-se armazenar os dados de tempo real em uma fonte de dados temporária. Além disso, deve-se aplicar a técnica *snapshot* em conjunto com a técnica RTDC (seção 4.3.1).

Gajanan Mane (NARENDRA, 2015) diz que o volume de dados cada vez maior e a necessidade cada vez maior por dados de negócios atualizados fez surgir o conceito de DW em tempo real. Companhias aéreas, assim como agencias governamentais devem ser capazes de analisar dados em tempo real para identificar potenciais ameaças. Com isso, é necessário que o DW receba os dados do ambiente operacional o mais rápido possível, para que as consultas analíticas possam sempre retornar os dados atualizados.

Com isso, o autor apresenta um estudo focado no mecanismo CDC e quais os benefícios proporcionados pelas técnicas quando aplicadas na extração de dados em tempo real. Além disso, o autor mostra uma arquitetura que é capaz de obter os dados operacionais em tempo quase real e disponibilizá-los no DW. Para isso, a arquitetura funciona da seguinte forma: 1) os dados operacionais são identificados por meio do mecanismo CDC, sobretudo pela técnica de *log*; 2) após isso, os dados são encaminhados para uma fila de dados; 3) com os dados armazenados temporariamente na fila de dados, é aplicado o processo de transformação e limpeza; 4) por fim, os dados são integrados ao DW. Além da arquitetura, o autor mostra as vantagens de se utilizar o mecanismo CDC em aplicações de tempo real, assim como suas principais características. Além disso, é mostrado também os problemas com os quais o processo ETL convencional deve ser capaz de lidar em ambientes de tempo real.

Entretanto, o autor não deixa claro qual o real objetivo do artigo, visto que foi mostrado vários aspectos sobre ETL em tempo real, mas não direcionado o objetivo do trabalho. A arquitetura proposta resolve apenas o problema de atualização de dados contínua e ignora outros requisitos do ambiente de tempo real, tais como escalabilidade, disponibilidade e baixa latência. Além disso, não há uma menção sobre tais requisitos do ambiente de tempo real, os quais não podem ser descartados ao propor uma solução ETL em tempo real.

Segundo Muddasir e Raghuveer (N; K, 2017), a grande demanda por dados que favoreçam as análises de dados de negócios, fez com que a atualização de dados no DW se tornasse obrigatório numa frequência cada vez maior. Entretanto, as técnicas ETL convencionais não são capazes de suprir as demandas exigidas por esses ambientes. Dessa forma, os autores propõem um estudo cujo o objetivo é expor técnicas disponíveis na literatura que permitem executar o processo ETL em tempo quase real.

O estudo foi conduzido por meio de três abordagens diferentes para auxiliar a execução do processo ETL em tempo quase real. As abordagens analisadas foram: abordagem baseada em metadados e abordagem baseada no método CDC. Na Abordagem baseada em metadados, os autores dividiram os metadados em duas categorias: 1) metadados de negócios, que tem a responsabilidade de definir as dimensões e fatos da modelagem multidimensional; 2) metadados técnicos, que tem a finalidade de definir a trajetória de um dado, desde o ambiente operacional até o DW. Além disso, a categoria metadados técnicos é dividida em outras três categorias: 1) metadados de tabelas de dados, que tem a função de armazenar os metadados das fontes de origem e destino, tais como tipos de dados, colunas, relacionamentos e etc.; 2) metadados de arquivo de dados, cujo o objetivo é armazenar os dados num formato padrão; 3) metadados de regras, cujo o objetivo é definir as regras de transformação, limpeza e mapeamento dos dados. As mudanças ocorridas no ambiente operacional são verificadas por meio de um *wrapper*. Se alguma mudança ocorrer, pode-se definir um carregamento completo ou incremental dos dados.

Já a abordagem baseada no método CDC foi utilizada para realizar o carregamento incremental de dados para o DW. Um dos trabalhos estudados pelos autores é baseado em evento. A partir de um evento (inserção, alteração, remoção), um *middleware* é responsável por capturar as mudanças ocorridas e encaminhar para uma lista de dados. Para realizar a atualização do DW, o conteúdo dessa lista é mesclado com os dados já armazenados no DW. Segundo os autores, apenas realizar a migração dos dados para o DW sem verificar sua integridade é perda de tempo. Com isso, eles expõem trabalhos que utilizam a técnica de *trigger* para disponibilizar os dados alterados do ambiente operacional. Além disso, outros trabalhos disponibilizam os dados alterados em arquivos XML. Com isso, as próximas tarefas do processo ETL podem ser executadas a partir dos dados alterados disponibilizados por esses meios.

De acordo com Sabtu et al. (SABTU et al., 2017), a aplicação do processo ETL convencional ou tradicional se torna efetivo em ambientes em que os dados são estruturados, bem definidos e as atualizações ocorrem numa frequência pré-determinada. Entretanto, a evolução

como os dados são gerados e a necessidade cada vez maior que esses dados sejam disponibilizados numa frequência cada vez maior, fez com que esse processo convencional se tornasse inadequado para o contexto atual. Além disso, atualmente, os dados podem ser gerados por meio de sensores, em tempo real e em diferentes formatos. Assim, o processo ETL deve lidar com ambientes cuja a disponibilidade da aplicação deve ser alta, permitir uma baixa latência na atualização dos dados e escalabilidade horizontal que permita o armazenamento de um grande volume de dados.

A partir desse contexto, os autores propõem um estudo em que apresentam os desafios encontrados em cada fase do processo ETL, assim como as soluções já disponíveis para superar esses desafios. Além disso, os autores apresentam alguns trabalhos disponíveis na literatura que abordagem as três características de ambientes em tempo real: alta disponibilidade, baixa latência e escalabilidade horizontal. E, para finalizar, os autores apresentam as soluções disponíveis para serem implementadas em ambientes com essas características.

Bouaziz et al. (BOUAZIZ et al., 2017) citam a importância do DW em tempo real para as tomadas de decisões no contexto atual em que os dados são gerados por diferentes meios e em tempo real. Eles definem o DW em tempo real como um sistema que representa as características e a real situação da organização. Diferente do DW convencional, o DW em tempo real mantém os dados sempre atualizados e permite armazenar dados no mesmo momento em que são produzidos pelo ambiente operacional. Dessa forma, o processo ETL deve ser executado várias vezes ao dia, permitindo assim que os tomadores de decisões sempre acessem dados atualizados.

A partir desse contexto, os autores desenvolveram um *survey* focado em três aspectos principais: 1) arquiteturas disponíveis na literatura que tratam do assunto; 2) as mudanças ocorridas no processo ETL convencional para permitir ser executado em tempo real; 3) integração de dados no DW em tempo real. Com isso, os autores apresentaram arquiteturas para permitir que o processo ETL seja executado no DW em tempo real e outros trabalhos que permitem a integração de dados em tempo real. Por fim, os autores expõem uma tabela, na qual contém uma comparação entre todos os trabalhos sob alguns critérios definidos pelos autores.

Phanikanth e Sudarsan (PHANIKANTH; SUDARSAN, 2017) argumentam o surgimento de diversos conceitos, tais como *Big Data* e *Data Stream* fez com que surgissem outras características nos dados gerados, tais como volume, velocidade e variedade. Com isso, fez com que o processo ETL se tornasse incapaz de ser executado em ambientes com essas características. Dessa forma, os autores apontam que as abordagens até então estudadas sobre o processo ETL em tempo real devem ser redefinidas para permitir o processamento de dados com tais características.

Com base nesse contexto, os autores desenvolveram uma pesquisa na qual apontam os trabalhos disponíveis na literatura que buscam executar o processo ETL com base em *data stream*. Em resumo, os autores concluem que as técnicas disponíveis atualmente resolvem parcialmente o problema de executar o processo ETL com base em *data stream*. Isso quer dizer que nenhum trabalho consegue resolver o problema abordando as três características dos dados.

Chandra, H. (CHANDRA, 2018) propõe uma análise dos métodos CDC em ambientes heterogêneos que apoiam o ambiente de *data warehousing* em tempo real. Segundo o autor, a maior preocupação ao construir o processo ETL é garantir o bom desempenho do processo, assim como garantir que a execução do processo ETL não interfira no desempenho das fontes de dados do ambiente operacional. Para isso, é necessário aplicar o método CDC que, por meio do qual, os dados do ambiente operacional são disponibilizados para o processo de extração de dados.

A partir desse contexto, o autor faz um estudo que mostra as principais técnicas CDC utilizadas em diferentes tipos de fonte de dados. As técnicas CDC analisadas foram: data e hora de modificação em cada registro, data e hora em arquivo externo, replicação, *trigger* e arquivo de *log*. Os tipos de fontes de dados sobre os quais foram aplicadas as técnicas CDC são: estrutura de arquivos, banco de dados hierárquico, banco de dados relacional e relacional binário. Para os arquivos, as técnicas CDC aplicadas foram data e hora de modificação em cada registro e data e hora em arquivo externo. Para o banco de dados hierárquico, as técnicas CDC aplicadas foram data e hora de modificação em cada registro, data e hora em arquivo externo e replicação. Para o banco de dados relacional, as técnicas aplicadas foram arquivos de *log*, replicação e *trigger*. Para o modelo relacional binário, as técnicas CDC aplicadas foram data e hora de modificação em cada registro, data e hora em arquivo externo e *trigger*.

Os testes realizados tiveram como objetivo avaliar o desempenho de cada técnica CDC aplicada em suas respectivas fontes de dados. Os resultados desse estudo apontam que a melhor técnica CDC depende do tipo de fonte de dados. Para os bancos de dados hierárquicos, a melhor opção é data e hora em arquivo externo. Para bancos de dados cuja a estrutura é relacional, a melhor técnica CDC a ser aplicada é *trigger*. E para bancos de dados cuja a estrutura é relacional binário, a melhor opção também é *trigger*.

A Tabela 15 mostra uma comparação entre os trabalhos fortemente relacionados cujo o objetivo foram estudos. Embora esta pesquisa não é focada em propor algum tipo de estudo, é essencial que se tenha conhecimento dos assuntos teóricos que cercam o processo ETL como um todo. Esse fato é importante pois permite identificar os conceitos, características e especificidades sobre o assunto. Além disso, esses trabalhos podem expor um resumo ou englobam outros trabalhos já desenvolvidos. Assim, é possível identificar o estado da arte com mais facilidade, além de identificar quais produtos de pesquisa foram inspiração para outros produtos de pesquisa.

A Tabela 15 mostra o Autor e Ano de cada trabalho. A coluna CDC indica se tal trabalho aborda o estudo de técnicas CDC. A coluna *Stream* indica se o trabalho aborda questões sobre *data stream*. A coluna E, T, L indicam, respectivamente, se o trabalho aborda a extração (E), a transformação (T) ou o carregamento (L). A coluna C indica se o trabalho aborda consultas OLAP (C). Essas colunas indicam qual(is) a(s) fase(s) do processo ETL abordada(s) no trabalho e se tal trabalho também busca realizar consultas OLAP. A coluna Nível indica se o trabalho

Tabela 15 – Tabela comparativa entre os estudos realizados

#	Autor	Ano	CDC	Stream	E	T	L	C	Nível	Requisitos RT
1	Vassiliadis e Simitsis	2009	X		X	X	X		C	
2	Penzlin et al.	2009			X	X	X		L	
3	Farooq e Sarwar	2009			X	X		X	L	
4	Tank et al.	2010	X		X				L	
5	Kakish e Kraft	2012	X		X	X	X		L	
6	Revathy et al.	2013	X	X	X		X		L	
7	Ferreira et al.	2013					X		L	
8	Sabry e Ali	2014	X	X	X	X	X		L	
9	Wibowo, A.	2015			X	X	X		L	
10	Gajanan Mane, N.	2015	X		X				L	
11	Muddasir e Raghuveer	2017	X		X				C,L	
12	Sabtu et al.	2017	X	X	X	X	X		L	X
13	Bouaziz et al.	2017			X	X	X		L	X
14	Phanikanth e Sudarsan	2017		X	X	X	X		L	X
15	Chandra, H.	2018	X		X				L	

Fonte: Próprio autor

propõe um estudo em nível conceitual (C), lógico (L) ou físico (F) do processo ETL. Por fim, a coluna Requisitos RT indica se o trabalho aborda o estudo dos requisitos do ambiente de tempo real.

Por meio da Tabela 15 é possível responder algumas das questões de pesquisa definidas no protocolo da revisão sistemática. Os trabalhos feitos pelos autores Sabry e Ali (SABRY; ALI, 2014) e Bouaziz et al. (BOUAZIZ et al., 2017) apresentam um levantamento bibliográfico acerca do processo ETL aplicado em ambientes de *data warehousing* em tempo real. Revathy et al. (S et al., 2013) também apresentam um *survey*, porém com foco em *data stream*. Contudo, Vassiliadis e Simitsis (VASSILIADIS; SIMITSIS, 2009), Wibowo, A. (WIBOWO, 2015), Sabtu et al. (SABTU et al., 2017) e Phanikanth e Sudarsan (PHANIKANTH; SUDARSAN, 2017) abordam um estudo bibliográfico que, embora não esteja explicito no texto, se trata de um *survey*, o qual engloba os desafios e as soluções já propostas na literatura para lidar com o processo ETL em ambientes de *data warehousing* em tempo real. Dessa forma, ambos os trabalhos podem responder as questões de pesquisa Questão 1 e Questão 3, e isoladamente o trabalho de

Vassiliadis e Simitsis (VASSILIADIS; SIMITSIS, 2009) pode responder a questão Questão 5, ambas descritas na Revisão Sistemática (Apêndice A). Além dessas questões, é possível notar que apenas um trabalho aborda a questão de executar consultas OLAP em ambientes de *data warehousing* em tempo real (Farooq e Sarwar (FAROOQ; SARWAR, 2010)). Dessa forma, é possível responder a questão de pesquisa Questão 6 descrita na Revisão Sistemática (Apêndice A). Além disso, é possível observar qual fase do processo ETL cada trabalho aborda. Logo, é possível responder a questão de pesquisa Questão 4 descrita na Revisão Sistemática (Apêndice A). Por fim, é possível responder a questão de pesquisa Questão 5 descrita na Revisão Sistemática (Apêndice A), pois é possível analisar o nível em que cada trabalho foi desenvolvido (coluna Nível). Vale ressaltar que essas classificações e respostas são referentes a estudos bibliográficos realizados. Dessa forma, nesse momento, ignora-se os trabalhos cujo objetivo foram desenvolver uma arquitetura, um método ou outro produto de pesquisa diferente de estudo, os quais serão analisados posteriormente neste capítulo.

7.1.2 Arquitetura

Embora todos os trabalhos estudados durante essa pesquisa são focados em apresentar arquiteturas, técnicas, métodos ou estudos, o trabalho de Bruckner et al. (BRUCKNER et al., 2002) descreve as tecnologias que permitem implementar um DW em tempo real na época de sua publicação, há algum tempo. Para representar essas tecnologias, os autores mostram uma arquitetura base que permite identificar os componentes e suas funções. Basicamente, a arquitetura é composta por três camadas: 1) fontes de dados, na qual são representadas as fontes de dados operacionais; 2) transformação: camada responsável por representar as transformações necessárias nos dados; 3) armazenamento: responsável por representar o DW e as ferramentas que recuperam os dados integrados. Além disso, a arquitetura é baseada na linguagem de programação Java, isto é, todas os módulos descritos são diretamente relacionados aos serviços disponíveis na linguagem, tais como conexão *Java Database Connectivity* (JDBC), *Java 2 Platform Enterprise Edition* (J2EE), *Java Message Service* (JMS) e outros serviços. Cada serviço é responsável por uma tarefa durante o processo ETL até o carregamento de dados no DW.

Vale destacar que a arquitetura apresentada mostra uma possível solução para DW em tempo real, porém em baixo nível, isto é, mostra detalhes de implementação, além de ser focado em uma linguagem específica. Contudo, os autores não mostraram testes da arquitetura proposta. Logo, não foi possível identificar se a arquitetura permite realizar todas as atividades e fornecer os requisitos de disponibilidade e escalabilidade. Entretanto, foi possível perceber que é possível apenas realizar atualização contínua.

Viana et al. (VIANA et al., 2005) dizem que o clima espacial é uma combinação das condições atuais do sol, vento, magnetosfera e outros elementos. Esses elementos exercem um papel fundamental no controle das de sistemas de monitoramento de espaçonaves. Além disso, esses dados climáticos poderiam ser utilizados por controladores de voos de naves espaciais

para fornecer informações preditivas e *online*. Contudo, tais dados são disponíveis em diferentes formatos, em fontes de dados externas aos sistemas de monitoramento. Portanto, coletar e transformar esses dados é uma tarefa essencial para permitir aos controladores um monitoramento e fornecer predições as espaçonaves.

Dessa forma, os autores desenvolveram uma arquitetura capaz de extrair os dados distribuídos em fontes de dados heterogêneas, transformá-los em um formato homogêneo e permitir consultas analíticas para tomadas de decisões sobre esses dados. A arquitetura proposta é composta por três componentes básicos: 1) **UDAP**: esse componente é responsável por recuperar todos os arquivos disponibilizados por agências de monitoramento do clima. Esses arquivos contém parâmetros cujo os valores são estruturados. Entretanto, a quantidade de parâmetros pode variar de acordo com cada processo de extração. Dessa forma, o componente também é responsável por viabilizar a extração de dados mesmo com a variedade de parâmetros que possa existir nos arquivos. Além disso, quando extraídos, os arquivos são mantidos localmente e temporariamente em um repositório para servir como valores de entrada para a próxima fase do processo; 2) **UDET**: esse componente é responsável por processar todos os arquivos recebidos do UDAP. Esses arquivos são compostos por dados estruturados e legíveis ao usuário. Os arquivos são processados por meio de operações definidas pelo usuário, de modo a disponibiliza-lo num formato homogêneo. Além disso, o UDET possui um mecanismo que, com o qual, os dados repetidos são ignorados em cada processo de transformação dos dados. Após esse processo, os dados são disponibilizados em diferentes formatos, dependendo das configurações do usuário, e prontos para serem armazenados na fonte de dados de destino; 3) **UDOB**: esse componente é composto por uma fonte de dados relacional, sendo responsável por manter, temporariamente, os dados processados pelo UDET. Com base nesses dados temporários e em configurações definidas pelo usuário, os dados são encaminhados para o DW ou para fontes de dados operacionais.

Embora a arquitetura proposta seja capaz de extrair dados de arquivos e transformá-los em um formato homogêneo, os autores focaram apenas na fase de extração e transformação dos dados em um formato homogêneo. Dessa forma, não foi possível analisar se a arquitetura permite resolver outros tipos de transformações, tais como homônimos, sinônimos e limpeza de dados. Além disso, os autores mostram detalhadamente apenas um componente da arquitetura. Dessa forma, não foi possível analisar a arquitetura como um todo. Além do mais, o tempo gasto para executar o processo mostrado no trabalho pode levar aproximadamente cinco minutos. Esse tempo pode não ser aceitável e determinados ambientes nos quais os dados são exigidos numa latência menor. Outro aspecto negativo é que os autores não apresentaram testes para validar a proposta, não sendo possível verificar com clareza se é possível obter escalabilidade com a arquitetura proposta.

Naeem et al. (NAEEM et al., 2008) citam a importância de executar o processo ETL para atualização de dados do DW. Entretanto, no modo convencional, as fontes de dados do ambiente operacional devem estar *offline*. Esse tempo de inatividade se torna inaceitável em determinados

contextos, em que os dados devem ser obtidos com uma latência mínima. Além disso, os autores dizem que, no processo ETL convencional, tanto os dados que sofrem alterações constantemente (dados transacionais, materializado no DW na tabela de fatos) quanto os dados que não sofrem alterações (dados mestres, materializado no DW nas tabelas de dimensões) são obtidos pelo processo. Esse fato faz com o que o processo ETL se torne ainda mais demorado, visto que o volume de dados manipulado é maior.

A partir desse contexto, os autores desenvolveram uma arquitetura capaz de extrair os dados operacionais em tempo quase real. Além disso, a arquitetura é capaz de manipular os dados transacionais (dados que serão consolidados na tabela de fatos do DW) sempre que houver alguma alteração e os dados mestre (dados das tabelas de dimensões do DW) apenas quando os mesmos sofrerem alterações no ambiente operacional. A arquitetura proposta é composta pelos seguintes componentes: 1) *middleware*: responsável por capturar as alterações dos dados ocorridas no ambiente operacional; 2) roteador de conteúdo: é responsável por receber os dados operacionais. Após isso, esse componente identifica os dados transacionais e os dados mestre. Então, os dados mestre são armazenados em um repositório próprio, enquanto os dados transacionais são encaminhados para o componente enriquecimento de conteúdo; 3) enriquecimento de conteúdo: é responsável por integrar os dados transacionais e os dados mestre armazenados em sua fonte de dados. Segundo os autores, esse é o componente principal da arquitetura, pois, por meio do qual, não é necessário sincronizar os dados mestre com o DW a cada transação. A sincronização só ocorre quando há alterações nos dados mestre nas fontes de dados operacionais; 4) verificador: esse componente é responsável por receber os dados transformados do componente enriquecimento de conteúdo e armazená-los no DW. Essa atualização ocorre assim que a transformação é finalizada.

Segundo os autores, com a arquitetura baseada em eventos, em que cada atualização nos dados operacionais é considerado um evento, é possível disponibilizar os dados no DW numa baixa latência. Entretanto, os autores não realizaram testes para avaliar a arquitetura proposta. Inclusive, como a arquitetura é iniciada a cada alteração no ambiente operacional, não é possível avaliar seu desempenho se houver um grande volume de atualizações. Além disso, não é possível analisar outros aspectos importantes de ambientes de tempo real, tais como escalabilidade e disponibilidade.

Chieu e Zeng (CHIEU; ZENG, 2008) citam a importância de sistemas de monitoramento de dados utilizarem dados gerados em tempo real para permitir o acompanhamento de informações referentes a negócios. Entretanto, as ferramentas de extração de dados não permitem a obtenção de dados em tempo real. Esse fato ocorre pois tais ferramentas são executadas, no melhor caso, uma vez por dia. Essa característica inviabiliza sua utilização.

Dessa forma, os autores apresentam uma arquitetura chamada *Capture-Transform-Update* (CTU) que é possível obter os dados operacionais sob demanda. A arquitetura é composta pelos seguintes componentes: 1) **processo ETL**: nessa abordagem, o processo ETL é responsável por

extrair os dados históricos do ambiente operacional e armazená-los no DW, em um conjunto de tabelas próprio para os dados históricos; 2) **CTU**: esse componente é responsável por capturar os dados operacionais por meio de *trigger*. Além disso, é utilizado um campo *timestamp* para identificar os dados mais recentes a serem extraídos. Após a captura, os dados sofrem uma transformação a fim de serem disponibilizados no formato da fonte de dados de destino. Por fim, os dados resultantes dessa transformação são armazenados no DW em um conjunto de tabelas próprio para os dados de tempo real. Quando solicitada uma consulta, a integração de dados entre as duas fontes de dados ocorre por meio de *views* do próprio banco de dados.

Embora a arquitetura proposta permita realizar a extração de dados em tempo real, esse processo pode se tornar custoso quando um grande volume de dados for analisado. Isso se deve ao fato de que, na fase de transformação, o componente CTU executa outras consultas nas fontes de dados de destino para buscar informações das fontes. Esse fato claramente induz a problemas relacionados a desempenho e conflitos entre inserção e consultas. Outro aspecto negativo apontado pelos próprios autores são o desempenho e a consistência dos dados em consultas. Isso se deve ao fato de que a arquitetura não foi projetada para lidar com dados em *cache*. Logo, toda consulta terá de ser executada sempre no DW. Além disso, os autores não realizaram testes para validar a escalabilidade da arquitetura proposta.

Thomsen et al. (THOMSEN et al., 2008) discutem a importância de manter os dados do DW atualizados. Entretanto, essa atualização se torna custosa, visto que, com um grande volume de dados, o tempo de atualização se torna cada vez maior. Dessa forma, os autores apresentam uma arquitetura que permite realizar consultas sob demanda a partir de dados de tempo real, assim como dados armazenados no DW, além de garantir a consistência e acurácia dos resultados.

A arquitetura proposta é composta por três camadas: 1) ***data producer***: essa camada é situada nas fontes de dados operacionais e tem a responsabilidade de gerar e encaminhar os dados operacionais para o *catalyst*. Por meio dessa camada, é possível executar duas operações: 1) *insert*: essa operação recebe os dados operacionais e os mantém em *cache* para servirem às outras fases do processo; 2) *commit*: essa operação recebe os dados operacionais e os encaminha diretamente para o DW. Essas operações podem ser executadas em momentos distintos e são definidos pelo usuário; 2) ***catalyst***: tem a responsabilidade de manter temporariamente os dados operacionais e controlar seu armazenamento no DW. Além disso, remove os dados armazenados em *cache* quando os mesmos já estão armazenados no DW. Cada registro mantido temporariamente no *catalyst* tem um ID próprio e sequencial, além de um atributo *timestamp*. Esse fato faz com que torne fácil recuperar os dados mais recentes e identificar os registros que foram carregamentos num mesmo momento. Quando uma consulta é solicitada, o *catalyst* é capaz de obter os dados por meio do *cache* realizado pela operação de *insert*, pelo seu próprio *cache* e pelo DW; 3) ***data consumer***: esse componente é responsável por obter os dados do *catalyst* para o DW, assim como garantir a acurácia dos dados. Para obter esses dados, é utilizado o nome da tabela temporária na

qual os dados são mantidos no *catalyst*, o valor para o campo *timestamp* e o intervalo de ID que compreende os registros solicitados.

Embora a arquitetura apresentada pelos autores permite realizar consultas sob demanda, os testes realizados mostram que há uma sobrecarga e perda de desempenho ao executar consultas em ambas as fontes de dados. Além disso, mesmo que a proposta foi executar o processo ETL sob demanda, não foi possível analisar como a arquitetura se comporta caso receba várias demandas de consulta ao mesmo tempo. Além disso, não possível analisar o desempenho e a escalabilidade com um grande volume de solicitações.

Santos e Bernardino (SANTOS; BERNARDINO, 2008), Santos et al. (SANTOS et al., 2011) e Cuzzocrea et al. (CUZZOCREA et al., 2014) destacam o DW no contexto de análise de negócios e tomadas de decisões. Entretanto, em domínios tais como comércio eletrônico, bolsa de valores, telecomunicações e saúde, a demanda por dados atualizados se tornou cada vez maior, sobretudo pela forma de como os dados que alimentam esses sistemas são gerados. Isso quer dizer que esses domínios necessitam de dados atualizados continuamente numa latência mínima. Outra regra fortemente identificada para esses domínios é a disponibilidade da aplicação, isto é, os usuários que geram os dados devem ser capazes de gerar os dados ao mesmo tempo que outros usuários fazem consulta sobre esses dados em tempo real. Por meio da abordagem tradicional de atualização do DW, é impossível que essas características sejam respeitadas, pois a atualização ocorre num período em que a aplicação esteja ociosa justamente para não ocorrer conflitos entre a atualização e consulta. Por outro lado, os dados consultados podem não estar atualizados com os dados gerados em tempo real.

Dessa forma, Santos e Bernardino (SANTOS; BERNARDINO, 2008) propõem uma arquitetura que permite atualizar e realizar consultas simultaneamente no DW. A abordagem proposta consiste em criar tabelas temporárias cujo esquema é igual ao do DW. Entretanto, essas tabelas temporárias não contém índices, chaves primárias, chaves estrangeiras ou qualquer tipo de restrição. Segundo os autores, esse fato favorece o desempenho da aplicação no processo de inserção de dados e no processo de integração posterior. Assumindo que o processo de extração e transformação já foram executados, os dados são armazenados na Área de Tratamento de Dados por meio de uma *trigger*. Então, esses dados são encaminhados para as tabelas temporárias continuamente por meio de instruções SQL. Para realizar consultas, os dados podem ser retornados das tabelas temporárias e do DW por meio do operador SQL *UNION* ou retornados de apenas uma das bases de dados. Após um período de tempo, ou um limite de tuplas nas tabelas temporárias, ou um limite de espaço de armazenamento, os dados das tabelas temporárias são sincronizados com o DW e as tabelas temporárias são recriadas.

Já Santos et al. (SANTOS et al., 2011) propõem a implementação da arquitetura desenvolvida por Santos e Bernardino (SANTOS; BERNARDINO, 2008) em continuidade ao trabalho. Além da implementação, os autores apresentaram outros três componentes que compõem a arquitetura: 1) ***Real-Time Data Warehouse Manager***: responsável por gerenciar as ferramentas

de bancos de dados e monitorar a execução do processo de carregamento de dados; 2) ***Real-Time Data Warehouse Loader***: responsável por executar os procedimentos de atualização de dados nas tabelas temporárias e no DW; 3) ***Real-Time Data Warehouse Query Executor***: responsável por manipular as consultas dos usuários, redirecionando as consultas para as tabelas temporárias e/ou para o DW.

Segundo Santos e Bernardino (SANTOS; BERNARDINO, 2008), o período que compreende a sincronização das tabelas temporárias e do DW é o único momento em que o método se torna inacessível aos usuários e não permite executar consultas no DW. Além disso, Santos et al. (SANTOS et al., 2011) destacam que um ponto negativo é o alto custo com *hardware* que deve ser gasto para permitir que o método seja executado. Entretanto, segundo os autores, esse custo é aceitável, visto os benefícios proporcionados pelo DW atualizado em tempo real. Entretanto, é possível notar que há um período em que o DW se torna inativo, o que afeta diretamente na disponibilidade da aplicação. Além disso, os autores abordam apenas a fase de carregamento de dados do processo ETL, além de não estudar outros fatores do ambiente de tempo real, tal como escalabilidade.

Shi et al. (SHI et al., 2008) argumentam que em um ambiente de *data warehousing* em tempo real não é possível utilizar o processo ETL convencional, devido as suas características de aplicação. Dessa forma, os autores dizem que o ponto principal para permitir que o processo ETL seja executado em tempo real é a extração de dados. Com isso, os autores desenvolveram um *framework* que com o qual é possível realizar a extração de dados em tempo real por meio da técnica CDC de *log*. Além de permitir a extração de dados em tempo real, o *framework* controla o carregamento, agendamento de tarefas a serem executadas e o consumo de recursos de *hardware*. Além disso, utilizando a técnica CDC de *log*, não causa impacto nas fontes de dados de origem. Dessa forma, segundo os autores, é possível obter um bom desempenho em sua execução.

Embora fique evidente que o *framework* pode realizar a atualização de dados contínua sem causar impacto nas fontes de dados de origem, não é possível essa afirmação com um grande volume de dados e nem com fontes de dados heterogêneas. Além disso, não é possível avaliar o *framework* sob o ponto de vista de escalabilidade e disponibilidade.

Javed e Nawaz (JAVED; NAWAZ, 2010) dizem que a atualização do DW no modo convencional se torna inadequada para ambientes em que é necessário dados atualizados o mais rápido possível para as tomadas de decisões. Além disso, devido a heterogeneidade das fontes de dados operacionais, a atualização de dados em tempo real se torna inadequada com as técnicas disponíveis.

Dado esse contexto, os autores mostram uma arquitetura capaz de atualizar em tempo real os dados que sofreram modificações no ambiente operacional. Dessa forma, é possível que os dados modificados sejam acessados em tempo real, enquanto o restante dos dados podem ser atualizados no modo convencional. A extração dos dados modificados ocorre por meio de uma técnica CDC. Contudo, os autores não citam qual a técnica utilizada. Após a extração de dados,

os dados são mantidos em arquivos, os quais mantém os dados para serem carregados em tempo real e o restante dos dados para serem carregados no modo *offline*. Entretanto, os autores não explicam como ocorre o processo de transformação e carregamento.

Embora os resultados mostram que a arquitetura proporciona um melhor desempenho na atualização dos dados operacionais para o DW, não foi possível analisar aspectos como escalabilidade, disponibilidade e atualização contínua. Além disso, segundo os próprios autores, a arquitetura proposta permite apenas a extração dos dados, não sendo possível realizar a transformação e carregamento dos dados.

Zuters (ZUTERS, 2011) cita a importância do DW para ambientes de tomadas de decisões estratégicas. No modo convencional, sua atualização ocorre num período pré-definido e num momento em que não há consultas sendo executadas. Entretanto, as tomadas de decisões passaram a ser tomadas com uma frequência cada vez maior e com dados gerados em tempo real. Para realizar essa atualização em um tempo quase real, surgiram diversas abordagens com as quais se tornou possível obter dados numa frequência menor do que era realizado no passado. Entretanto, a frequência de atualização disponibilizada por essas abordagens ainda era inadequada.

Com isso, o autor propôs uma abordagem chamada *Multi-stage Trickle and Flip*. Essa abordagem é baseada na abordagem *Trickle and Flip* proposta por Langseth, J. (LANGSETH, 2004) sendo composta pelo que os autores chamaram de *staging tables*, *realtime data D*, *staging tables D*, *realtime data H*, *staging tables H* e DW. Vale destacar que *realtime data D* e *realtime data H* são partições criadas para o DW e tem a função de armazenar os dados de tempo real produzidos no dia atual e na última hora respectivamente. Os componentes *staging tables D* e *staging tables H* tem a responsabilidade de receber os dados advindos do *staging tables* e serem mantidos para possíveis transformações ou cargas do dia atual ou da última hora respectivamente. Segundo o autor, os dados da *staging tables D* são sincronizados com o *realtime data D* de hora em hora. Já os dados da *staging tables H* são sincronizados com o *realtime data H* de cinco em cinco minutos. De acordo com o autor, os dados do dia atual e da última hora armazenados em áreas separadas e fora da partição, favorece ao carregamento contínuo dos dados do ambiente operacional e deixa as partições livres para consulta. Por fim, é possível executar uma consulta contendo as seguintes combinações: 1) somente DW; 2) DW e *realtime data D*; 3) DW, *realtime data D* e *realtime data H*.

Segundo o autor, a abordagem proposta resolve o problema de conflitos entre o carregamento de dados do ambiente operacional e as consultas no DW e permite a obtenção de dados analíticos atualizados numa frequência menor. Entretanto, o trabalho não aborda outros problemas que podem existir em ambientes de tempo real, tais como disponibilidade, escalabilidade, integridade e qualidade dos dados.

Zhou et al. (ZHOU et al., 2011) dizem que o DW em tempo real é uma extensão do DW convencional, mas com recursos adicionais que permitem que os dados do ambiente operacional

sejam disponíveis o mais rápido possível. Dessa forma, o DW em tempo real é uma combinação de dois elementos: dados de tempo real gerados pelo ambiente operacional e o DW. Para realizar a integração de dados, é necessário executar o processo ETL de forma contínua. Entretanto, identificar os dados a serem atualizados é a tarefa mais custosa de todo o processo. Segundo os autores, os dados de tempo real pode ser extraídos por meio de mensagens, *triggers*, arquivo de *log* do banco de dados ou por meio de *snapshot*.

Com isso, os autores propõem uma estratégia, baseado na técnica CDC de *log*, que permite extrair os dados do ambiente operacional e disponibiliza-los para as fases posteriores do processo ETL, tais como transformação, limpeza e carregamento. O método proposto funciona da seguinte forma: a cada operação de inserção, alteração e remoção realizada nas fontes de dados operacionais, os dados envolvidos na operação são armazenados em um arquivo de *log*. Nas operações de inserção e remoção, são armazenados os metadados das fontes de origem. Na operação de alteração, são armazenados duas tuplas: uma para informar os dados antes da alteração e outra para informar os dados após a alteração. Por meio de um *listening*, é realizada uma operação de consulta no arquivo de *log* em tempo real, no qual pode ser consultado por ordem de entrada dos registros ou por ordem de data. Dessa forma, os dados alterados são disponibilizados com maior flexibilidade ao para o restante das fases do processo ETL.

Embora a estratégia proposta permita extrair dados em tempo real, não foi possível avaliar os aspectos de ambiente de tempo real, tais como disponibilidade, baixa latência e escalabilidade. Além disso, não possível avaliar o desempenho da proposta com base em algum experimento.

YiChuan e Yao (YICHUAN; YAO, 2012) dizem que o DW convencional surgiu para resolver o problema de consultas analíticas e tomadas de decisões. Entretanto, essa forma de lidar com os dados não é adequada para domínios nos quais os dados devem ser tratados em tempo real. Dessa forma, surgiu o DW em tempo real, em que os autores definem como uma extensão do DW convencional, adicionada a capacidade de lidar com dados em tempo real. Contudo, os autores dizem que o principal problema identificado nesse ambiente é o fato das atualizações serem realizadas ao mesmo tempo que consultas são executadas.

Dessa forma, os autores desenvolveram uma arquitetura que permite que as atualizações sejam executadas ao mesmo tempo que consultas são realizadas. Para isso, os autores nomeiam os dados gerados em tempo real de dados dinâmicos e os dados armazenados no DW de dados estáticos. Inicialmente, o processo ETL é dividido em duas partes: 1) ETL periódico: é responsável por armazenar os dados operacionais no DW baseado no modo convencional, isto é, após um período pré-definido pela organização; 2) ETL em tempo real: após aplicar o mecanismo CDC, os dados operacionais são encaminhados para uma área de dados denominada de área de dados de tempo real e mantidos em *cache*. Na área de dados de tempo real, os dados são separados em duas partes: 1) partição 1: mantém os dados de tempo real para serem consultados; 2) partição 2: utilizado para receber novos dados operacionais. Dessa forma, evita-se conflitos de atualização e consulta sendo executados simultaneamente. Ainda na área de dados de tempo real,

podem haver vários *caches*, um para cada ciclo de atualização dos dados, por exemplo: segundo os autores, a arquitetura pode manter dados de 0 minutos, 10 minutos, 30 minutos, 60 minutos e 240 minutos. Dessa forma, se uma consulta necessitar de dados de uma hora, a arquitetura é capaz de posicionar a execução da consulta no devido *cache*. Por fim, os dados da área de dados de tempo real são sincronizados com o DW por meio do processo ETL convencional após a definição do usuário.

Embora a arquitetura proposta resolva o problema de atualização de dados contínua no DW, além de permitir bom desempenho nas consultas, os autores não deixam claro as questões de disponibilidade e escalabilidade. Além disso, os autores não mostram de forma detalhada como ocorre a sincronização dos dados da área de dados de tempo real com o DW e a atualização direta do DW com os dados operacionais. Esses dois aspectos implicam diretamente no desempenho e escalabilidade da arquitetura proposta.

MadeSukarsa et al. (MADESUKARSA et al., 2012) citam a importância do DW em tempo real em organizações que necessitam de análise de dados em tempo real. Além disso, a aplicação do mecanismo CDC auxilia na obtenção dos dados operacionais e torna o processo ETL mais eficiente, visto que o volume de dados manipulados é menor. Entretanto, os autores dizem que as abordagens para executar o processo ETL em tempo real pode causar problemas de comunicação, visto que todo o processo é executado numa camada entre o ambiente operacional e o DW. Além disso, o processo de transformação e carregamento é executado no ambiente do DW, o que causa problemas de desempenho.

Dessa forma, os autores propõem uma abordagem definida como *Capture-Transform-Load* (CTL), que consiste em executar todo o processo de extração e transformação no ambiente operacional, isto é, cada dispositivo responsável por gerar dados é responsável também por lidar com o processo de extração e transformação. Basicamente, a a abordagem segue o seguinte fluxo: 1) é feito um mapeamento dos dados do ambiente operacional com os dados do DW. Esse mapeamento é utilizado pela fase de carregamento para identificar qual tabela do DW é relacionada a tabela do ambiente operacional. 2) a cada inserção, alteração ou remoção executada, uma base de dados temporária recebe essas mudanças por meio de uma *trigger*; 3) por meio de agendador, os dados armazenados localmente são mesclados com os dados já armazenados no DW e o mesmo é sincronizado.

Segundo os autores, essa abordagem diminui os problemas de comunicação entre o ambiente operacional e o DW durante o processo de sincronização de dados, visto que a extração e transformação ocorre no ambiente operacional. Além disso, a utilização de *trigger* diminui a latência com que os dados alterados são disponibilizados. Entretanto, os autores não explicam alguns aspectos essenciais para o ambiente de tempo real: 1) como ocorre o processo de consulta dos dados em tempo real, ou seja, se uma consulta pode retornar dados do DW e das fontes de dados operacionais; 2) quais procedimentos tomar se ocorrer alguma anomalia com algum computador do ambiente operacional; 3) se alguma anomalia ocorrer com algum computador do

ambiente operacional, como resolver os problemas de disponibilidade e escalabilidade.

Tank, D. (TANK, 2012) expõe a importância de *Business Intelligence* e DW no mundo de negócios. Segundo o autor, é possível oferecer serviços que com os quais permite descobrir as áreas em que podem haver mais lucros ou danos, identificar tendências e comportamentos dos clientes, identificar áreas com uma tendência maior de lucros e etc. Entretanto, com a mudança na forma de gerar dados, isto é, cada vez mais os dados de negócios são gerados em tempo real, é necessário que esses dados sejam refletidos em tempo real no ambiente de DW.

O DW em tempo real surgiu como forma de resolver parte desse problema identificado em ambientes que geram dados em tempo real. Entretanto, se tornou necessário o estudo de novas técnicas ETL para permitir tal atualização, visto que o processo ETL tradicional não é adequado para esse tipo de atualização de dados. Além disso, ao estudar esse tema, o autor sugere quatro requisitos de ambientes de tempo real que devem ser respeitados ao propor uma solução baseada em tempo real, que são: desempenho, escalabilidade, disponibilidade e dados atualizados.

A partir desse contexto, o autor mostra a aplicação do mecanismo CDC e sua importância ao integra-lo com o processo ETL. Segundo o autor, ao aplicar alguma técnica CDC é possível obter melhor desempenho e consequentemente uma menor latência na atualização de dados, visto que o volume de dados a ser manipulado é menor. Entretanto, o autor não deixa claro qual sua real proposta de trabalho, bem como o que realmente pretende resolver, visto que no título e durante todo o trabalho, ele induz que apresentará uma arquitetura capaz de atualizar dados em tempo real e, no final, apenas mostra uma fundamentação superficial do mecanismo CDC.

Jain et al. (JAIN et al., 2012) mostram que várias áreas de negócios passaram lidar com dados em tempo real como um modo de fornecer mais segurança a seus clientes, por exemplo: companhias aéreas passaram a analisar e detectar passageiros suspeitos em seus voos, bancos passaram a analisar dados em tempo real de modo a realizar auditorias em contas e monitorar possíveis roubos, organizações de diferentes setores passaram a realizar previsões de lucros e custos com base em dados obtidos em tempo real. Todos esses setores de negócios que lidam com dados em tempo real devem permitir que os dados sejam disponibilizados aos tomadores de decisões também em tempo real. Isso faz com que as tomadas de decisões sejam realizadas com base nos dados atuais da empresa. Entretanto, o processo ETL convencional é inadequado para lidar com esse tipo de ambiente. Os dados são atualizados numa frequência em que as tomadas de decisões são realizadas com dados desatualizados. Além disso, todas as atividades que devem ser executadas se tornam custosas ao realiza-las na frequência desejada.

Com base nesse contexto, os autores desenvolveram um método segundo o qual é possível atualizar os dados mais importantes (dados críticos) em tempo real e os dados com menos importância (dados não críticos) no modo convencional. Segundo os autores, os dados críticos são os dados cujo seu valor pode representar um impacto negativo nas tomadas de decisões. Além disso, é considerado como dados críticos um dado cujo seu valor sofreu várias

atualizações com tempo. Esses dados são obtidos por meio de um arquivo de *log* e armazenados em filas de dados, uma para cada tipo de dado. Em seguida, os dados críticos são armazenados em tempo real em tabelas temporárias contendo o mesmo esquema do DW, ao passo que os dados não críticos são mantidos para serem sincronizados com o DW no modo convencional.

Embora os autores deixam claro que é possível realizar atualizações no DW em tempo real, eles não deixam claros questões em relação escalabilidade e disponibilidade. Além disso, é possível observar que o método proposto será executado duas vezes, uma para cada tipo de dado. Nesse caso, se torna uma problema em casos de disponibilidade da aplicação, assim como no consumo de recursos, visto que o método será executado duas vezes em momentos distintos.

Xue et al. (XUE et al., 2012) citam que os dados de medição de energia elétrica poderiam auxiliar, de certa forma, a diminuir o consumo de energia nas regiões em que o consumo e a escassez de energia é alto. Embora dados de consumo de energia elétrica são disponíveis, esses dados não são utilizados para fornecer um monitoramento de consumo em tempo real. Com isso, os autores apresentam uma arquitetura capaz de executar o processo ETL a cada duas horas. Para isso, os autores utilizaram a técnica *Trickle and Flip* proposta por (LANGSETH, 2004).

Embora a arquitetura apresentada permite executar o processo ETL em tempo quase real e aplica técnicas propostas na literatura para tal, a latência de duas horas pode ser inaceitável para outros tipos de organizações que necessitam de dados numa latência menor. Além disso, os autores não deixam claro como ocorre o processo de extração, transformação e carregamento dos dados, além de não abordarem os requisitos de ambientes de tempo real, tais como disponibilidade e escalabilidade.

Jia et al. (JIA et al., 2013) argumentam que no cenário atual de negócios, os dados para tomadas de decisões devem estar o mais atualizado possível. Contudo, embora o DW tenha se tornado uma ferramenta essencial para as tomadas de decisões, sua atualização convencional se tornou um problema para obter esses dados atualizados. Dessa forma, é necessário que o DW seja atualizado o mais rápido possível e permita executar consultas em paralelo a atualização.

Dessa forma, os autores desenvolveram uma arquitetura que com a qual é possível atualizar os dados operacionais continuamente no DW e permitir que consultas sejam executadas ao mesmo tempo. Inicialmente, os autores aplicaram a técnica CDC de *log*. Segundo eles, o *log* permite criar uma independência das fontes de dados de origem e, dessa forma, permite obter um melhor desempenho. A partir disso, os dados são extraídos do arquivo de *log* de forma incremental. Além disso, o DW é dividido em duas partes: 1) armazenamento histórico: mantém os dados históricos, isto é, representa o DW; 2) armazenamento de tempo real: armazena continuamente os dados operacionais a partir do arquivo de *log*. Ambas as fontes de dados são sincronizadas regularmente de acordo com as configurações definidas pelo usuário.

Além da extração e armazenamento dos dados em tempo real, os autores garantem dois aspectos fundamentais dos dados: 1) consistência dos dados: após serem extraídos, os dados são

colocados em uma fila de dados contendo a tabela de origem na qual o dado sofreu alteração, a alteração ocorrida, a hora em que a alteração ocorreu e o local na transação. Dessa forma, mesmo que ocorra algum problema que impeça a sincronização dos dados no momento, é possível sincronizar os dados em outro momento sem perder a consistência dos dados; 2) atualização contínua: por meio de filtros definidos pelo usuário, é possível definir os dados que não são necessários para análise. Dessa forma, diminui o volume de dados a ser processado, o que permite um melhor desempenho na fase de carregamento.

Embora os autores mostrem que a arquitetura proposta permite realizar atualização de dados contínua, não foi possível identificar se a arquitetura garante a disponibilidade e escalabilidade. Além disso, os autores não mostram como é feita a sincronização dos dados históricos e de tempo real. Dessa forma, não foi possível identificar o quão custoso é, qual momento e como ocorre a sincronização dos dados.

Obali et al. (OBALI et al., 2013) citam a importância da utilização do DW em armazenar dados originados de diferentes fontes de dados heterogêneas. Entretanto, a forma como os dados passaram a ser gerados, cada vez mais em tempo real, mudou a forma de atualização desses dados. A atualização dos dados passou a ser cada vez mais em tempo real, visto que é necessário refletir no DW as mudanças ocorridas no ambiente operacional. Entretanto, o processo ETL convencional deve ser executado *off-line*. Esse tempo de espera na atualização é inadequado para domínios que necessitam de dados em tempo real.

Nesse contexto, os autores desenvolveram uma arquitetura com a qual é possível ter acesso a dados em tempo real originados de diferentes fontes de dados. Além disso, é possível obter dados de tempo real e dados históricos numa mesma consulta. A arquitetura proposta é composta pelos seguintes componentes: 1) ***Web Service Client* (WS Client)**: um *Web Service* responsável por obter os dados alterados do ambiente operacional por meio de técnicas CDC, tais como *trigger* e *log*; 2) ***Web Service Provider* (WS Provider)**: outro *Web Service* responsável por receber os dados do WS *Client* e adicionar num outro componente chamado *Real Time Partition*; 3) ***Real Time Partition***: este componente é responsável por manter os dados de tempo real. Ao final de um período estabelecido (normalmente ao final do dia), os dados de tempo real são integrados e armazenados com os dados já armazenados no DW; 4) ***Real Time Data Integration***: este componente é responsável por integrar dados históricos e dados de tempo real. Quando uma consulta é solicitada, este componente identifica em qual fonte de dados estão os dados solicitados. Além disso, ele é capaz de acessar a fonte de dados na qual estão os dados solicitados.

Embora a arquitetura proposta pelos autores permite a extração e consulta de dados em tempo real, os autores não fizeram experimentos para avaliar a disponibilidade, desempenho e escalabilidade. Além disso, não foi possível analisar os detalhes de como ocorre a extração, transformação e o carregamento dos dados em tempo real.

Halenar, R. (HALENAR, 2013) argumenta que os usuários tomadores de decisões

necessitam cada vez mais de dados atualizados. Esse fato é caracterizado pelos dados serem gerados com uma frequência cada vez maior e novas fontes de dados surgirem. A própria *Web* pode ser considerada uma fonte de dados, pois, por meio dela, pode-se obter dados sobre qualquer domínio desejado. Os tomadores de decisões de organizações como vendas *online*, monitoramento de rodovias, monitoramento de estoque e outras, devem realizar suas tarefas a partir de dados gerados em tempo real, ou seja, nesse tipo de domínios não é permitida a latência semanal, diária ou de uma hora. Os dados devem ser atualizados em tempo real ou o mais rápido possível.

Com isso, o autor apresenta uma arquitetura que é capaz de executar o processo ETL em tempo quase real. A arquitetura é composta pelos seguintes componentes: 1) **fontes de dados**; 2) **armazenamento temporário**; 3) **área de processamento de dados**; 4) **dados exportados**; 5) ***data warehouse***; 6) **arquivo de *flags***. Inicialmente, os dados operacionais são extraídos de uma fonte de dados cuja a extensão dos arquivos é *Microsoft Database File* e exportados para arquivos XLS na **área de processamento de dados**. Após esse processo, os dados são transformados de modo a se apresentarem com suas referências, valores de chaves, ordenação e agrupamento. Além disso, o **arquivo de *flags*** contém regras e procedimentos definidos anteriormente pelo usuário. Dessa forma, os dados sofrem também uma transformação com base nas regras definidas nesse arquivo. Por fim, os dados transformados são carregados para DW e um *flag* na tabela de fatos indica se o tupla sofreu alteração quando foi executada a transformação pelo **arquivo de *flags*** (*flag* 1) ou se o tupla não sofreu alteração (*flag* 0).

Segundo o autor, a utilização de *flags* auxilia no processo de verificação do nível de confiança dos dados consultados. Se um dado, no processo de transformação, sofre transformações com base nas regras do arquivo de *flags*, logo, seu nível de confiança diminui. Dessa forma, esse mecanismo pode favorecer a confiança do usuário nos dados consultados. Entretanto, o autor não deixa claro como é executado cada uma das etapas do processo ETL. Com isso, não foi possível verificar qual a frequência em que o processo ETL é executado, assim como características do ambiente no qual o trabalho foi aplicado. Da mesma forma, também não foi possível analisar alguns requisitos do ambiente de tempo real, tais como disponibilidade, escalabilidade, baixa latência, assim como não foi possível verificar a qualidade dos dados após a execução do processo ETL.

Freudenreich et al. (FREUDENREICH et al., 2013) citam a importância do processo ETL na obtenção de dados operacionais e as devidas transformações ocorridas com os dados. Entretanto, esse é processo estritamente acoplado, ou seja, as fases devem ser realizadas uma após a outra e não há meios de flexibiliza-las. Essa desvantagem do processo ETL convencional resulta em outras desvantagens, tais como disponibilidade, processamento desnecessário de dados, escalabilidade limitada. O desacoplamento das fases do processo ETL implica em executar o processo ETL de acordo com as demandas de cada organização, além de fornecer consultas com dados atualizados sob demanda.

A partir desse contexto, os autores propõem uma técnica na qual utilizam a abordagem ELT que consiste em armazenar os dados operacionais em um local que os autores chamam de rascunho e é localizado conjuntamente com o DW. Os autores utilizam visões, procedimentos e operações SQL para realizar as transformações e apresentação dos dados. Segundo os autores, essa abordagem permite realizar consultas sob demanda e não é necessária a fase de carregamento, pois com a definição das visões, os dados são disponibilizados quando necessário. Contudo, os autores não enfocam e nem deixam claro se a técnica proposta é capaz de resolver os problemas de disponibilidade e baixa latência. Além disso, pode-se observar que a técnica proposta resolve apenas a questão da escalabilidade dos dados. Por fim, a técnica proposta, assim como um protótipo, está em fase inicial de desenvolvimento.

Valencio et al. (VALENCIO et al., 2014) argumentam que, atualmente, as tomadas de decisões não podem ser tomadas com dados desatualizados, com o risco dessas decisões serem tomadas com base em dados que não dizem a real situação de uma organização. Contudo, o processo ETL é executado de modo *offline*, o que impossibilita a extração de dados numa frequência maior. Além disso, os autores apontam a fase de extração de dados a mais custosa de todo o processo ETL. Com isso, os autores apresentam uma arquitetura capaz de extrair os dados operacionais em tempo real, ao passo que as fases de transformação e carregamento estão fora do escopo do trabalho.

A arquitetura proposta é baseada na técnica CDC *trigger*. A cada operação (inserção, alteração e remoção) executada nas tabelas selecionadas no ambiente operacional, uma *trigger* é executada. Essa *trigger* é executada na mesma transação da operação executada no ambiente operacional. Segundo os autores, esse aspecto faz com que não ocorra perda ou conflito nos dados extraídos. Após a execução, os dados obtidos pela *trigger* são armazenados temporariamente em uma tabela chamada tabela de *log*. Essa tabela se assemelha ao *log* do próprio banco de dados, mas contém outras informações que serão utilizadas pelas outras fases do processo ETL, como por exemplo: tipo da operação executada, id que representa a transação que gerou a alteração, id que representa o identificador único da tupla e outras informações. Segundo os autores, essa abordagem facilita outros processos que necessitam desses dados, tais como *backup*, recuperação e replicação de dados.

Embora os testes focados em escalabilidade mostram que o tempo de execução é considerado aceitável mas não dentro do esperado, os autores entendem que a proposta é mais útil em outros processos que necessitam desses dados (por exemplo, *backup*). Esses processos também necessitam de dados extraídos e, com isso, outros tipos de serviços podem ser oferecidos. Além disso, a proposta é altamente dependente da fonte de dados de origem. Esse aspecto faz com que a disponibilidade da aplicação seja afetada, assim como a dependência se torne maior.

Cuzzocrea et al. (CUZZOCREA et al., 2014) acrescentam à Santos e Bernardino (SANTOS; BERNARDINO, 2008) e Santos et al. (SANTOS et al., 2011) o fato que índices e restrições tornam lento o processo de atualização de dados e, para contornar esse problema, o DW deve

ser atualizado no modo *offline*. Dessa forma, os autores propõem uma abordagem que permite atualizar os dados do DW em tempo real e, ao mesmo tempo, permitir consultas sobre os dados. Nesse sentido, os autores focam apenas na fase de carregamento e consulta de dados, ignorando as fases de extração e transformação.

A abordagem proposta consiste em separar os dados do DW em dois grupos, dinâmico e estático. Os dados dinâmicos são os dados de tempo real gerados pelo ambiente operacional. Os dados estáticos são os dados históricos armazenados no DW. No mesmo sentido, a abordagem é composta por dois componentes principais: 1) ***static* DW (S-DW)**: é a representação do DW e é responsável por armazenar os dados históricos; 2) ***Dynamic* DW (D-DW)**: é composto por tabelas cujo o esquema é o mesmo do S-DW. Entretanto, não contém índices e demais restrições. Segundo os autores, esse fato favorece a eficiência da atualização e consulta dos dados. Além disso, esse componente é responsável por armazenar continuamente os dados de tempo real. As consultas solicitadas podem ser realizadas em ambas as fontes de dados. A própria abordagem é responsável por direcionar a consulta para as devidas fontes de dados. Os dados no componente D-DW são mantidos diariamente e, ao final do período definido, as tabelas recriadas.

Segundo os autores, a abordagem proposta permite ser executada para permitir atualização de dados e consulta de dados em tempo real. Entretanto, os autores não deixam claro como ocorre a sincronização dos dados do componente D-DW para o S-DW e como a disponibilidade da aplicação é afetada nesse processo. Além disso, os autores não abordam aspectos tais como escalabilidade e disponibilidade.

Lebdaoui et al. (LEBDAOUI et al., 2014) citam a importância do processo de integração de dados na era de *Big Data*. Os dados de negócios são gerados em tempo real e armazenados no DW para análises de mercado tomadas de decisões. Contudo, várias abordagens foram propostas na literatura em busca de permitir que atualização de dados em tempo real. Entretanto, essas propostas não focam na integridade dos dados respeitando as regras de seguranças já estabelecidas.

Com isso, os autores propõem uma abordagem chamada *Divide-Join Data Integration* (DJ-DI), com a qual é possível integrar os dados alterados no ambiente operacional em tempo real baseado no conceito de dividir o volume de dados alterados e armazena-los em uma tabela de fatos duplicada. De acordo com os experimentos realizados, é possível obter bom desempenho na integração dos dados e garantir a integridade dos dados. Entretanto, os autores não deixam claro outros aspectos relevantes ao ambiente de tempo real, tais como escalabilidade e disponibilidade. Além disso, não deixam claro quais as técnicas CDC aplicadas na abordagem.

Mao et al. (MAO et al., 2014) e Li e Mao (LI; MAO, 2015) citam a importância do ambiente OLTP e OLAP nas tomadas de decisões de negócios. O ambiente OLTP é responsável por gerenciar as transações dos usuários que geram dados em tempo real. Por outro lado, o ambiente OLAP é focado na tomada de decisão, ou seja, um grupo de usuários responsável por executar consultas e analisar dados de forma estratégica. Esse fato causa um impacto no DW,

pois os usuários OLTP podem gerar dados ao mesmo que os usuários OLAP executam consulta. Dessa forma, ocorre o que os autores chamam de contenção de consulta.

Dessa forma, os autores propõem uma arquitetura capaz de permitir que consultas OLAP sejam realizadas ao mesmo tempo que ocorre a atualização de dados operacionais no DW. A arquitetura é composta por três módulos principais: 1) *historical* ETL: este componente é responsável por armazenar os dados considerados históricos do ambiente operacional diretamente no DW; 2) *real-time* ETL: esse componente é responsável por detectar os dados alterados no ambiente operacional por meio de técnicas CDC e armazenar numa área chamada *Dynamic Storage Area* (DSA); 3) DSA: este componente é responsável por armazenar temporariamente os dados de tempo real. Os dados armazenados nesse componente são armazenados temporariamente em arquivos de dados, os quais representam uma fonte de dados operacional. Logo, os dados de uma mesma fonte de dados são agrupados num mesmo arquivo. Esses arquivos de dados são criados dinamicamente sempre que um novo dado é inserido no DSA ou destruídos dinamicamente sempre que os dados forem sincronizados com o DW. Além disso, é possível criar vários arquivos de dados dinamicamente, em que cada arquivo é ligado uns aos outros por meio de um *link*. Dessa forma, é possível obter várias versões de um mesmo dado em tempo real. Ao ser solicitada uma consulta, a arquitetura é capaz de identificar em qual arquivo de dados estão armazenados os dados solicitados. Além disso, é possível identificar os dados mais recentes por meio de um atributo *timestamp*. Por fim, a partir de configurações definidas pelo usuário, ocorre a sincronização de dados do DSA para o DW. Após a sincronização de dados, o arquivo de dados assim como seus respectivos *links* são removidos do DSA.

Com base na arquitetura proposta, os autores afirmam que é possível resolver o problema de contenção de consulta, além de resolver também o problema de acurácia dos dados. Além disso, foi possível analisar que a arquitetura proposta permite obter um melhor desempenho e tempo de resposta nas consultas realizadas. Entretanto, aspectos escalabilidade e disponibilidade não foram analisados pelos autores. Além disso, o volume de dados analisado foi baixo, ou seja, não foi possível afirmar se a proposta é eficiente se um grande volume de dados for aplicado.

Guo et al. (GUO et al., 2015) definem o processo ETL como uma série de etapas para obter dados do ambiente operacional, independentemente da origem dos dados. Por esse motivo, é o processo mais custoso e demorado ao desenvolver um DW. Entretanto, a execução do processo ETL tem um baixo desempenho, pois deve haver a Área de Tratamento de Dados responsável por armazenar temporariamente os dados. Esse processo faz com que as operações de escrita em disco torne o processo custoso. Além disso, o processo ETL pode ser executado várias vezes ao dia ou em tempo real. Esse fato faz com que sua execução se torne ainda mais custosa e cara.

No processo ETL convencional, a utilização da Área de Tratamento de Dados resolve o problema da heterogeneidade dos dados, mas ao aplicá-lá torna o processo custoso devido ao tempo de execução. Além disso, tanto o processo ETL convencional quanto o DW focam em manter os dados redundantes, isto é, armazenar os dados no ambiente operacional e no

DW. Esse aspecto faz com que também não seja considerada a reusabilidade dos dados para consultas que são realizadas com frequência. Assim, os autores apontam algumas falhas que podem ser identificadas no processo ETL convencional: 1) o tempo de execução é inadequado; 2) redundância de dados pode levar a problemas de espaço de armazenamento; 3) por não considerar a reusabilidade, resulta em executar todo o processo ETL repetidas vezes.

A partir desse contexto, os autores propõem uma nova abordagem chamada *Transform-Extract-Load* (TEL). O objetivo principal é extrair os dados para o DW sob demanda, executando primeiramente a fase de transformação. A abordagem proposta não inclui a Área de Tratamento de Dados que, segundos os autores, é a etapa que mais custosa do processo ETL convencional. Ao invés disso, os autores propõem uma abordagem que é composta por três camadas: 1) **camada física:** essa camada é composta por um conjunto de tabelas físicas, cujo os dados são originados de fontes de dados homogêneos ou heterogêneos; 2) **camada virtual:** essa camada é o componente principal da abordagem proposta, cuja a responsabilidade é manter o relacionamento entre as fontes de dados operacionais e o DW. O relacionamento entre essas fontes ocorre por meio de regras de mapeamento feitas pelo usuário na fase de transformação. Essa camada virtual elimina os problemas de heterogeneidade das fontes de dados e leitura e escrita em disco, além de não armazenar dados fisicamente; 3) **camada efetiva:** essa camada é responsável por interagir com o usuário que possuem esquema e dados. O esquema dessa camada é herdado da camada virtual, e os dados são o resultado do conteúdo da camada física.

Por meio dessa abordagem, os resultados das consultas obtidas a partir da camada virtual podem ter diferentes latências. Quanto maior o volume de dados solicitados, maior será o tempo de resposta de uma consulta. Nesse sentido, por meio de um *benchmark*, avaliaram o desempenho e o tempo de resposta das consultas. Segundo os autores, em determinados domínios, a arquitetura pode proporcionar bom desempenho. Contudo, em outros contextos (não apontados quais), a arquitetura deve ser remodelada. Além disso, não foi possível avaliar outros elementos de ambientes de tempo real, tais como disponibilidade e escalabilidade.

Martins et al. (MARTINS et al., 2016) argumentam que o processo ETL é responsável por alimentar o DW independente do cenário em que é aplicado. Esses cenários podem envolver pequeno ou grande volume de dados. Em ambos os cenários, o processo ETL deve se comportar da mesma forma e fornecer bom desempenho. Contudo, as soluções ETL atuais não fornecem tais recursos automaticamente. Dessa forma, os autores desenvolveram uma arquitetura capaz de automaticamente fornecer escalabilidade horizontal em um ambiente de *data warehousing*.

Foram realizados testes em ambientes *offline* e em tempo real. Os resultados mostraram que, sem a abordagem proposta, a medida que o volume de dados aumenta, a escalabilidade é afetada e os dados a serem extraídos e transformados são descartados. Por outro lado, com a abordagem proposta, é possível obter melhor desempenho em ambos os ambientes. A arquitetura proposta se comportou adequadamente quando um grande volume de dados (aproximadamente 500MB) é processado pelo processo ETL em um tempo de aproximadamente 2 segundos.

Embora os resultados mostraram que é possível obter uma escalabilidade horizontal para um ambiente de tempo real, não foi possível analisar o tempo de resposta ao realizar consultas utilizando a arquitetura proposta. Além do tempo de resposta, não foi possível avaliar outro requisito essencial que é a disponibilidade. Além desses fatores que envolvem um ambiente de tempo real, não foi possível analisar como ocorre o processo de extração e transformação dos dados, visto que os autores apenas mostraram o que a arquitetura faz, mas não como ela faz.

Figueiras et al. (FIGUEIRAS et al., 2017a) dizem que com o aumento da utilização de *smartphones*, sensores e demais dispositivos conectados à Internet, diversos setores de negócios sofreram mudanças na forma de como lidar com esses dados. Especificamente, em Portugal, os sistemas inteligentes de transporte passaram a lidar com essa variedade e volume de dados produzidos por esses dispositivos. Por meio desses sistemas é possível analisar o tráfego das principais estradas em tempo real, assim como oferecer predições sobre tais estradas. Além disso, em rodovias que tem pedágios, o preço pode variar de acordo com a quantidade de viagens feitas por uma pessoa ou de acordo com a hora do dia. Contudo, o principal desafio é extrair e processar esses dados em tempo real e consequentemente oferecer serviços que permitam diminuir o congestionamento nas estradas.

Nesse sentido, os autores propuzeram uma arquitetura que, por meio da qual é possível mudar o preço do pedágio em dinamicamente. Com isso, é possível que os motoristas façam uso de outras estradas e rodovias de acordo com as predições feitas pelo sistema. Para realizar a mudança nos preços, a arquitetura considera os seguintes fatores: condições em tempo real das redes rodoviárias, segurança das rodovias, custo de manutenção, congestionamento, condições meteorológicas e eventos de tráfego. Além disso, é utilizado predições do fluxo de tráfego com base no histórico dos dados.

Para auxiliar no processo, a arquitetura utiliza algoritmos de aprendizagem de máquina para fornecer predições das rodovias em tempo real. Os dados de tempo real necessários para fornecer as predições são extraídos dos sensores fixados nas rodovias e sensores nas praças de pedágios e armazenados na base de dados MongoDB. Além disso, são utilizados os dados correspondentes ao histórico de tráfegos que também são armazenados no MongoDB. Contudo, os autores não descrevem se tal base de dados representa um DW.

Vale a pena destacar que essa abordagem proposta é originada de outros dois trabalhos, que são dos autores Guerreiro et al. (GUERREIRO et al., 2016) e Figueiras et al. (FIGUEIRAS et al., 2017b). Em Guerreiro et al. (GUERREIRO et al., 2016), os autores mostram a arquitetura proposta, seus componentes e a validação feita para a arquitetura. Contudo, não foi possível analisar se a arquitetura lida com requisitos de ambiente de tempo real, tais como escalabilidade e disponibilidade. Já em Figueiras et al. (FIGUEIRAS et al., 2017b), os autores mostram como ocorrem a obtenção e a preparação dos dados antes de serem armazenados no MongoDB. Além do mais, os autores afirmam que a arquitetura permite lidar com vários formatos e volume de dados, asseguram a eficiência na transformação e armazenamento dos dados e

garantem a interoperabilidade dos dados. Além disso, descrevem as tecnologias utilizadas para o desenvolvimento da arquitetura, que são Apache Spark, MongoDB e arquivos em diferentes formatos. Entretanto, os autores mostram apenas como ocorre a limpeza dos dados. É mostrado, por exemplo, como ocorre a limpeza de um campo contendo uma data fora do padrão adotado. Contudo, não foi possível analisar se a arquitetura permitir realizar a extração e integração dos dados em tempo real. Tanto que os autores explicam sobre sensores e como os dados são gerados nas rodovias, mas a fase de extração de dados é mostrada com arquivos CSV já importados para a arquitetura. Ou seja, não é mostrado como ocorre a extração desses dados em tempo real.

Muddasir e Raghuveer (MUDDASIR; RAGHUVEER, 2017) citam a importância de tornar o DW o mais atualizado possível devido a grande demanda por dados de tempo real para tomadas de decisões. Para isso, o processo ETL deve ser executado numa frequência maior do que é executado atualmente. Entretanto, ao executar o processo ETL em tempo real, pode ocorrer uma sobrecarga nas fontes de dados do ambiente operacional, que já servem um grupo de usuários inserem dados, e também uma sobrecarga no DW que já serve a outro grupo de usuários focados em análise de dados.

Para solucionar esse problema, diversas pesquisas vem sendo realizadas com o objetivo de permitir o processo ETL seja executado numa frequência maior do que o convencional. Com base nessas pesquisas, os autores desenvolveram uma arquitetura capaz de executar o processo ETL em tempo quase real, que por meio da qual é possível obter dados históricos e dados de tempo real sem haver sobrecarga nas fontes de dados envolvidas. Para o desenvolvimento da arquitetura, os autores utilizaram: 1) arquivo de *log*: a técnica CDC de *log* foi aplicada para obter os dados operacionais, isto é, sempre que algum dado de tempo real é solicitado, estes são obtidos por meio da fase de extração de dados e capturados no arquivo de *log*; 2) técnica *Direct Trickle Feed*: os dados operacionais são obtidos do arquivo de *log* e armazenados diretamente no DW; 3) junção de dados para consulta: se o volume de dados atinge um determinado limite no arquivo de *log*, estes são encaminhados diretamente para o DW. Posteriormente, os dados podem ser consultados diretamente no DW. Se o volume de dados no *log* não atingir o limitado definido, mas houver dados importantes para o usuário, os dados do arquivo de *log* são integrados com os dados do DW e apresentados em uma consulta por meio da operação *union* da linguagem SQL.

Segundo os autores, a arquitetura proposta permite obter consultas contendo dados históricos e dados de tempo real sem sobrecarga nas fontes de dados envolvidas. Isso se deve ao fato dos dados de tempo real serem obtidos por meio do arquivo de *log*. Isso faz com que crie uma independência do ambiente operacional e o DW. Entretanto, a partir dos testes realizados, não foi possível analisar o comportamento da arquitetura com um grande volume de dados, visto que os testes foram executados com um volume pequeno de dados. Além disso, não foi possível avaliar outros requisitos de ambientes de tempo real, tais como disponibilidade, escalabilidade e baixa latência.

Biswas et al. (BISWAS et al., 2019) argumentam que o processo ETL é um requisito

essencial para ambientes de *data warehousing*. As ferramentas ETL existentes são baseadas em *Graphical User Inserface* (GUI), as quais permitem a interação do usuário por meio de uma interface gráfica. Por meio dessas ferramentas é possível criar o cenário de ETL e favorecer os processos essenciais. Entretanto, a desvantagem dessas ferramentas é criar cenários específicos devido a limitação de componentes gráficos. Para isso, é possível utilizar ferramentas baseadas em código, as quais permitem criar cenários personalizados de acordo com a necessidade do usuário.

Dessa forma, os autores desenvolveram um trabalho cujo objetivos são: 1) mostrar uma visão geral de ferramentas ETL baseadas em código; 2) apresentar a abordagem desenvolvida que permite extrair e transformar os dados operacionais em tempo real e armazenar no DW. Na fase de extração de dados, os autores utilizaram a técnica CDC de *snapshot*. Dessa forma, é possível obter apenas os dados operacionais alterados e favorecer a atualização incremental dos dados. Na fase de transformação de dados, os autores mostram apenas como ocorre a eliminação de valores duplicados. Contudo, os autores não abordaram a fase de carregamento dos dados.

Por meio da abordagem proposta pelos autores, foi possível analisar que o desempenho com o carregamento incremental é melhor do que o carregamento completo. Além disso, foi possível obter uma visão geral das ferramentas ETL baseadas em código. Entretanto, não foi possível analisar a escalabilidade e latência da abordagem proposta. Além do mais, os autores consideram apenas fontes de dados relacionais como fonte de dados, logo, não foi possível analisar a heterogeneidade do ambiente operacional. Além desses fatores, também não foi possível analisar outros tipos de transformações, tais como correção de homônimos, correção de sinônimos e limpeza de dados inconsistentes.

Godinho et al. (GODINHO et al., 2019) afirmam que o volume de dados produzidos na área de imagens médicas é adequado com a definição de *Big Data*. O volume de dados é grande, a variedade de dados é cada vez maior e a tomada de decisão médica a partir dessas características se torna cada vez maior difícil com as ferramentas disponíveis. Além disso, existem bancos de dados médicos (o autor chama de *Picture Archive and Communications Systems (PACS))* que unem os dados das organizações médicas e fornecem um grande e poderoso conjunto de dados de imagens médicas. Além disso, essas imagens são compostas por metadados, os quais auxiliam no processo de identificar os dados para a tomada de decisão médica.

Com isso, os autores desenvolveram um *framework* baseado no conceito do processo ETL capaz de extrair os dados imagens médicas dos bancos de dados PACS em tempo real e mostrar ao usuário, sem a necessidade de utilizar um DW como meio de armazenamento, visto que os dados são extraídos dos PACS. Uma vez extraído, os dados passam por uma fase de limpeza, a qual é composta por regras de transformação, que são: detecção de campos em branco, valores esperados, preenchimento de campos predefinidos, normalização de campos e outras transformações. Após esse processo, os dados são apresentados ao usuário por meio de visões.

Apesar do fato dos autores dizerem que fazem a extração dos dados do PACS e mostram

como ocorre a transformação desses dados, os autores não mostraram quais as técnicas utilizadas para extrair esses dados e nem mostraram se os dados foram extraídos pelas técnicas tradicionais do processo ETL ou por meio de *select* diretamente no banco de dados PACS. Além disso, os autores não mostraram os requisitos de desempenho, disponibilidade e escalabilidade ao se aplicar a solução com um grande volume de dados gerados simultaneamente.

Machado et al. (MACHADO et al., 2019) dizem que o processo ETL é um procedimento vital a ser adotado pelas organizações em busca de dados para tomada de decisão estratégica. Atualmente, os dados estão disponíveis em ambientes heterogêneos, os quais são produzidos em diferentes intervalos de tempo por diferentes tipos de usuários e dispositivos. Esses fatos tornam o processo ETL ainda mais desafiador devido as características tradicionais e as técnicas, métodos e arquiteturas até então propostas até então.

Dessa forma, os autores desenvolveram uma arquitetura chamada DOD-ETL, a partir da qual é possível realizar o processo ETL em tempo real a partir da aplicação das seguintes técnicas: a técnica CDC de *log* foi aplicada para permitir extrair os dados operacionais; foi utilizado *cluster* de computadores e armazenamento de dados em memória para permitir o particionamento de dados de forma distribuída, fator este que favorece ao desempenho. Além disso, foram utilizadas as ferramentas Kafka, Beam e Spark *Streaming*.

Os testes experimentais foram executados a partir de um ambiente real e sintético, gerando o mesmo volume de dados para ser processado pelo processo ETL por meio do Spark *Streaming* com e sem o DOD-ETL. Os testes mostraram que a partir da utilização do DOD-ETL, foi possível obter um ganho de desempenho de aproximadamente 10 vezes melhor do que sem a utilização do DOD-ETL. Além disso, os testes experimentais consideram os requisitos de disponibilidade, escalabilidade e tempo de resposta.

Embora os resultados obtidos a partir dos testes experimentais tenham sido satisfatórios e mostrado o funcionamento em um ambiente real, o trabalho não considera o fato do ambiente operacional produzir dados por meio de sensores, fato que pode tornar a execução do processo ETL ainda mais crítico. Além disso, não considera o fato do ambiente operacional produzir dados por meio de fontes de dados heterogêneas, isto é, foi testada apenas com dados produzidos a partir de um banco de dados relacional. Além do mais, a solução desenvolvida é intrusiva, isto é, depende de elementos do banco de dados do ambiente operacional (neste caso, arquivo de *log*) e não reativa, não considera o conceito de *tag* para permitir o desacoplamento do processo ETL e o ambiente operacional. Por fim, ao realizar os testes experimentais, os autores consideraram apenas o tempo de resposta como medida de desempenho. Contudo, como já foi explicado no Capítulo 6, é importante aplicar outras medidas para avaliar por completo a escalabilidade e desempenho da solução.

Muddasir e Raghuveer (N; K, 2020) dizem que as anomalias em consultas são aqueles dados que podem aparecer em uma determinada consulta, mas não ter sua representação no DW. Ao contrário, uma anomalia pode ser um dado que está armazenado no DW, mas não aparece

em uma determinada consulta. Segundos os autores, isso ocorre devido as características de um ambiente heterogêneo, o qual possuem várias fontes de dados com diferentes formatos. Além disso, o processo de atualização do DW pode ocorrer em paralelo e falhas na atualização podem ocorrer.

Uma alternativa para resolver esse problema é permitir realizar consultas no DW só após o processo ETL finalizar a execução de suas tarefas. Contudo, isso pode afetar diretamente o tempo total de resposta de uma consulta, visto que as tarefas do processo ETL pode levar tempo para ser finalizada. Além disso, quando espera-se o processo ETL ser finalizado para executar uma consulta, obtém-se um tempo de consulta maior, porém o resultado da consulta é sem anomalias. Por outro lado, ao executar consultas em paralelo ao processo ETL, ganha-se em tempo de resposta, porém o resultado da consulta pode ser composto por anomalias.

Dessa forma, os autores propuseram uma arquitetura capaz de reduzir o tempo de resposta a uma consulta. A arquitetura foi baseada no trabalho desenvolvido por Santos e Bernardino (SANTOS; BERNARDINO, 2008), a partir da qual é possível realizar o carregamento contínuo para o DW. A ideia base da proposta é criar uma réplica das tabelas do DW, porém sem restrições, índices e/ou qualquer outro elemento que possa interferir no desempenho. Assim, o processo de carregamento incremental de dados deve ocorrer nas tabelas réplica do DW e não diretamente no DW. Dessa forma, uma consulta é executada cujo os dados atualizados são obtidos a partir da réplica do DW.

Embora os autores tenham desenvolvido o trabalho e mostrado que é possível realizar consultas em paralelo a atualização de dados e obter um ganho de no tempo de resposta, os autores não mostraram a viabilidade em aplicar essa arquitetura em um ambiente real. Além disso, os autores não mostraram qual é o comportamento ao executar a arquitetura com dados gerados em uma maior frequência. Além do mais, os autores aplicaram técnicas intrusivas em sua solução, fator este que implica diretamente no desempenho a medida que se aumenta o volume e frequência dos dados.

Thulasiram e Ramaiah (THULASIRAM; RAMAIAH, 2020) argumentam que o processo ETL surgiu para auxiliar o processo de tomada de decisão, ao armazenar os dados do ambiente operacional no DW e, por sua vez, servir às consultas analíticas. Contudo, as mudanças ocorridas no ambiente operacional devem ser encaminhas para o DW a fim de permitir que as consultas sempre mostrei os resultados atualizados. Dessa forma, os autores destacam as técnicas CDC como um meio de manter o DW sempre atualizado.

Dessa forma, os autores propuseram uma arquitetura baseada no carregamento incremental e nas técnicas CDC de replicação e *timestamp* para permitir que os dados do ambiente operacional sejam atualizados no DW sempre sofrerem alterações. A partir de um indicativo de *Insert* (I), *Delete* (D) ou *Update* (U), é possível saber a origem de um determinado dado. Além disso, a partir da técnica de *timestamp*, é possível obter o última versão de um dado atualizado ou removido.

Embora tenham proposto esse método, os autores não deixam claro qual a real proposta da solução, qual o contexto no qual a solução pode ser aplicada, como a solução foi validada e como os dados são extraídos e carregados no DW. Além disso, as ilustrações não apresentam de forma clara qual a real proposta da solução. Contudo, pode-se inferir que os autores utilizaram a técnica de replicação conjuntamente às ferramentas *Informatica Power Center* e *Attunity Replication* para aplicar a solução proposta e otimizar a sincronização das fontes de dados e o DW.

7.2 Análise dos resultados

Após apresentar as arquiteturas propostas na literatura que buscam resolver o problema de pesquisa, nesta seção será apresentada uma análise de todos os trabalhos. Essa análise tem o objetivo de comparar as arquiteturas propostas com este trabalho de pesquisa. Além disso, será mostrada uma classificação, na qual mostra as características de cada trabalho, assim como os principais elementos (repositório utilizado, fases do processo ETL aplicados e etc.) que os compõem.

Tabela 16 – Tabela comparativa entre os trabalhos fortemente relacionados cujo o foco foi propor algum produto de pesquisa diferente de estudo e o Imã de Dados.

#	Autor	Ano	E	T	L	C	Repositório	Nível TR	Stream	Prop.
1	Bruckner et al.	2002	X	X	X		DW	NRT		Arq.
2	Viana et al.	2005	X	X			DW	NRT		Arq.
3	Naeem et al.	2008	X		X		DW	NRT		Arq.
4	Chieu, T. e Zeng, L.	2008	X		X		DW	NRT		Arq.
5	Thomsen et al.	2008	X		X	X	DW	OND		Arq.
6	Santos e Bernar-dino	2008			X		DW	RT		Metod.
7	Shi et al.	2008	X				DW	RT		Met.
8	Majeed et al.	2010	X				DW	RT	X	Arq.

Continua na próxima página

										Continuação
#	**Autor**	**Ano**	**E**	**T**	**L**	**C**	**Repositório**	**Nível TR**	**Stream**	**Prop.**
9	Javed, M. e Nawaz, A.	2010	X				DW	RT		Arq.
10	Zuters, J.	2011			X		DW	NRT		Estr.
11	Zhou et al.	2011	X				DW	RT		Estr.
12	Santos et al.	2011			X		DW	RT		Metod.
13	YiChuan e Yao	2012	X		X	X	DW	RT		Arq.
14	MadeSukarsa et al.	2012	X	X	X		DW	NRT		Arq.
15	Tank, D.	2012	X				DW	RT		Estr.
16	Jain et al.	2012	X		X		DW	NRT		Arq.
17	Xue et al.	2012	X		X		DW	NRT		Apl.
18	Jia et al.	2013	X		X		DW	RT		Arq.
19	Obali et al.	2013	X		X	X	DW	NRT		Arq.
20	Halenar, R.	2013		X			DW	NRT		Estr.
21	Freudenreich et al.	2013	E	L	T		DW	OND	X	Arq.
22	Valêncio et al.	2013	X				DW	RT		Arq.
23	Cuzzocrea et al.	2014			X		DW	NRT		Metod.
24	Lebdaoui et al.	2014	X		X	X	DW	RT		Metod.
25	Mao et al.	2014	X		X		DW	RT		Arq.
26	Li e Mao	2015	X		X		DW	RT		Arq.
27	Guo et al.	2015	T	E	L	X	DW	OND		Arq.
28	Martins et al.	2016	X	X	X	X	DW	OND		Arq.
29	Figueiras et al.	2017	X	X			Mongo DB	NRT		Metod.
										Continua na próxima página

										Continuação
#	**Autor**	**Ano**	**E**	**T**	**L**	**C**	**Repositório**	**Nível TR**	**Stream**	**Prop.**
30	Muddasir and Raghuveer	2017	X		X		DW	NRT		Arq.
31	Biswas et al.	2019	X				DW	RT		Alg.
32	Godinho et al.	2019	X			X	DW	RT		Arq.
33	Machado et al.	2019	X	X	X		DW	NRT		Arq.
34	Muddasir and Raghuveer	2020			X	X	DW	RT		Arq.
35	Thulasiram and Ramaiah	2020			X		DW	NRT		Arq.
36	Vilela et al.	2021	X		X	X	DW	RT		Arq.

A Tabela 16 mostra uma comparação de todos os trabalhos fortemente relacionados ao assunto desta pesquisa de Doutorado. Essa classificação tem o objetivo de mostrar em quais fases do processo ETL o trabalho é focado, qual nível de tempo real proporcionado, assim como a real proposta do trabalho. As colunas Autor e Ano serve como um indicador de cada trabalho, mostrando o nome do(s) autor(es) e o ano de publicação. As colunas E (Extração), T (Transformação), L (Carregamento) e C (Consulta), indicam quais fases o trabalho investiga. A coluna Repositório indica qual repositório de dados o trabalho aborda. Ou seja, se o autor utilizou DW (DW) como fonte de armazenamento de dados ou outra fonte de dados. A coluna Nível TR (Nível de Tempo Real) indica se a proposta lida com o processo ETL em tempo real (TR), em tempo quase real (NRT) ou sob demanda (OND). A coluna Stream indica se o trabalho lida com *data stream*. Por fim, a coluna Prop. (Proposta) mostra qual foi a proposta de trabalho do autor. Isto é, se o autor propôs uma arquitetura (Arq.), uma metodologia (Metod.), um método (Met.), uma estratégia (Estr.), uma aplicação (Apl.) ou um algoritmo (Alg.).

É importante destacar que grande parte dos trabalhos lidam com o processo ETL utilizando o DW como repositório dos dados. Contudo, alguns trabalhos utilizam o banco de dados NoSQL MongoDB para armazenar os dados. Além disso, parte dos trabalhos assumem executar

o processo ETL em tempo real, enquanto outros realizam o processo em tempo quase real. Além disso, esses trabalhos que utilizam o MongoDB como meio de armazenamento, aplicam técnicas de aprendizado de máquina (e não consultas OLAP) para disponibilizar os dados para análise e tomada de decisão.

Por meio da Tabela 16, é possível responder algumas questões de pesquisa definidas no protocolo da revisão sistemática. Dessa vez, as respostas tem o objetivo de identificar questões práticas, isto é, responder quais trabalhos desenvolveram uma arquitetura e quais fases do processo ETL foram abordadas. Ao todo, foram identificados 24 trabalhos cuja a proposta foi arquitetura. Além disso, é possível analisar quais as fases do processo ETL são abordadas em cada trabalho e se tal trabalho aborda a questão de consultas OLAP. Dessa forma, é possível responder as questões de pesquisa Questão 2 e Questão 6. Isoladamente, o trabalho de Machado et al. (MACHADO et al., 2019) responde as questões de pesquisa Questão 9 e Questão 10. Contudo, ao final da análise dos trabalhos listados na Tabela 16, foi possível notar que as questões de pesquisa Questão 7 e Questão 8 não foram respondidas, visto que não há trabalhos que abordam o conceito de *tag*, não intrusivo e reativo.

Tabela 17 – Tabela comparativa entre os trabalhos fortemente relacionados cujo o foco foi propor algum produto de pesquisa diferente de estudo e o Imã de Dados.

#	Autor	Ano	Tipo CDC	DTF	TF	ERTDC	RTP	EP	Outro
1	Bruckner et al.	2002							
2	Viana et al.	2005							
3	Naeem et al.	2008	Trigger				X		Middleware Fila
4	Chieu, T. e Zeng, L.	2008	Trigger Timestamp				X		
5	Thomsen et al.	2008	Wrapper						Memória, Paralelismo
6	Santos e Bernardino	2008					X		
7	Shi et al.	2008	Log						Fila
8	Majeed et al.	2010							Stream P.

Continua na próxima página

Continuação

#	Autor	Ano	Tipo CDC	DTF	TF	ERTDC	RTP	EP	Outro
9	Javed, M. e Nawaz, A.	2010	Timestamp					X	
10	Zuters, J.	2011			X		X		
11	Zhou et al.	2011	Log						
12	Santos et al.	2011					X		
13	YiChuan e Yao	2012	Trigger			X	X		Fila
14	MadeSukarsa et al.	2012	Trigger						
15	Tank, D.	2012							
16	Jain et al.	2012	Log				X	X	Fila
17	Xue et al.	2012	Log		X				
18	Jia et al.	2013	Log		X		X		Fila Comandos SQL
19	Obali et al.	2013	Log		X		X		
20	Halenar, R.	2013							
21	Freudenreich et al.	2013	E	L	T		X		
22	Valêncio et al.	2013	Trigger					X	Comandos SQL
23	Cuzzocrea et al.	2014			X		X		Comandos SQL
24	Lebdaoui et al.	2014	Trigger				X	X	Comandos SQL
25	Mao et al.	2014	Trigger			X	X		Arq.
26	Li e Mao	2015	Trigger			X	X		Arq.
27	Guo et al.	2015	T	E	L				
28	Martins et al.	2016	Log			X			
29	Figueiras et al.	2017					X		

Continua na próxima página

									Continuação
#	**Autor**	**Ano**	**Tipo CDC**	**DTF**	**TF**	**ERTDC**	**RTP**	**EP**	**Outro**
30	Muddasir and Raghuveer	2017	Log	X					Comandos SQL
31	Biswas et al.	2019	Snapshot	X					
32	Godinho et al.	2019							Comandos SQL
33	Machado et al.	2019	Log				X		
34	Muddasir and Raghuveer	2020				X			
35	Thulasiram and Ramaiah	2020							Replicação e *timestamp*
36	Vilela et al.	2021							Sistema pub/sub, *tag*, paralelismo

A Tabela 17 mostra uma comparação dos trabalhos fortemente relacionados com o Imã de Dados. Dessa vez, essa comparação tem o objetivo de identificar quais as técnicas aplicadas para permitir que o processo ETL pudesse ser executado em tempo real. As colunas Autor e Ano representam, respectivamente, o nome do autor e o ano no qual o trabalho foi publicado. A coluna Tipo CDC indica qual a técnica CDC aplicada no trabalho. Em alguns trabalhos, o autor não deixou explicito qual a técnica aplicada. Logo, o valor da coluna foi deixada em branco. A coluna DTF (*Direct Trickle Feed*), TF (*Trickle and Flip*), ERTDC (*External Real Time Data Cache*) e (*Real Time Partition*) indicam qual técnica utilizada para permitir atualização e consulta de dados em tempo real. A coluna EP (Extração Parcial) indica se o trabalho, na fase de extração, propõe extrair apenas os dados essenciais para responder uma dada consulta (dados críticos) ou propõe extrair dados críticos e não críticos. Por fim, a coluna Outro indica alguma outra técnica aplicada conjuntamente com as técnicas anteriormente citadas e que ajuda a caracterizar o trabalho.

É importante destacar que grande parte dos trabalhos utilizam *trigger* e arquivos de *log*

como técnica CDC para captura de dados em tempo real. Além disso, a maioria dos trabalhos utilizam a técnica de *Realtime Partition* como meio de separar os dados de tempo real e os dados históricos, isto é, os dados de tempo real são armazenados em uma partição separada ao DW. Isso permite que a atualização de dados seja executada ao mesmo tempo em que as consultas são realizadas. Outro fato importante é que alguns trabalhos utilizam fila ou arquivo de dados como meio de armazenar dados temporariamente durante a execução do processo ETL.

Tabela 18 – Tabela comparativa entre os trabalhos fortemente relacionados que desenvolveram uma arquitetura, do ponto de vista dos requisitos de tempo real, e o Imã de Dados.

#	Autor	Ano	Disp.	TR	Esc.	Big Data
1	Bruckner et al.	2002		X		
2	Viana et al.	2005	X		X	
3	Naeem et al.	2008		X		
4	Chieu, T. e Zeng, L.	2008				
5	Majeed et al.	2010				
6	Javed, M. e Nawaz, A.	2010		X		
7	YiChuan e Yao	2012		X		
8	MadeSukarsa et al.	2012		X		
9	Jain et al.	2012		X		
10	Jia et al.	2013		X		
11	Obali et al.	2013		X		
12	Freudenreich et al.	2013			X	
13	Valêncio et al.	2013			X	
14	Mao et al.	2014		X		
15	Li e Mao	2015		X		
16	Guo et al.	2015				

Continua na próxima página

						Continuação
#	**Autor**	**Ano**	**Disp.**	**TR**	**Esc.**	**Big Data**
17	Martins et al.	2016			X	
18	Muddasir and Raghuveer	2017		X		
19	Godinho et al.	2019		X		
20	Machado et al.	2019	X	X	X	
21	Muddasir and Raghuveer	2020		X		
22	Thulasiram and Ramaiah	2020		X		
23	Vilela et al.	2020	X	X	X	

A Tabela 18 mostra uma comparação de todos os trabalhos fortemente relacionados os quais propuseram uma arquitetura, e o Imã de Dados. Dessa vez, essa classificação visa comparar os trabalhos com os requisitos de ambiente de tempo real. As colunas Autor e Ano mostram, respectivamente, o nome do autor e o ano no qual o trabalho foi publicado. A coluna Disp. (Disponibilidade) indica se a arquitetura considera o requisito de disponibilidade. A coluna TR (Tempo de Resposta) indica se é considerado o baixo tempo de resposta no processo de extração ou consulta de dados no DW. A coluna Esc. (Escalabilidade) indica se a arquitetura permite ser escalável. Por fim, a coluna *Big Data* indica se a arquitetura lida com dados gerados com as características de *Big Data*.

Tabela 19 – Tabela comparativa entre os trabalhos fortemente relacionados que desenvolveram uma arquitetura e as principais características que as diferem do Imã de Dados.

#	**Autor**	**Ano**	**Int.**	**Reat.**	**Tag**	**Par.**	**Des.**
1	Bruckner et al.	2002	X				

Continua na próxima página

							Continuação
#	**Autor**	**Ano**	**Int.**	**Reat.**	**Tag**	**Par.**	**Des.**
2	Viana et al.	2005	X				
3	Naeem et al.	2008	X				
4	Chieu, T. e Zeng, L.	2008	X				
5	Majeed et al.	2010	X				
6	Javed, M. e Nawaz, A.	2010	X				
7	YiChuan e Yao	2012	X				
8	MadeSukarsa et al.	2012	X				
9	Jain et al.	2012	X				
10	Jia et al.	2013	X				
11	Obali et al.	2013	X				
12	Freudenreich et al.	2013	X				
13	Valêncio et al.	2013	X				
14	Mao et al.	2014	X				
15	Li e Mao	2015	X				
16	Guo et al.	2015	X				
17	Martins et al.	2015	X				
18	Muddasir and Raghuveer	2017	X				
19	Godinho et al.	2019	X				
20	Machado et al.	2019	X			X	
							Continua na próxima página

							Continuação
#	**Autor**	**Ano**	**Int.**	**Reat.**	**Tag**	**Par.**	**Des.**
21	Muddasir and Raghuveer	2020	X				
22	Thulasiram and Ramaiah	2020	X				
23	Vilela et al.	2021		X	X	X	X

A Tabela 19 mostra uma comparação entre os trabalhos fortemente relacionados que desenvolveram uma arquitetura e o Imã de Dados. Dessa vez, o objetivo da tabela é mostrar os principais recursos os quais foram utilizados no projeto do Imã de Dados e que os fazem diferenciar dos trabalhos fortemente relacionados. As colunas Autor e Ano mostram, respectivamente, o nome do autor e o ano no qual o trabalho foi publicado. A coluna Int. (Intrusivo), mostra se o trabalho aplica o conceito de não intrusivo. A coluna Reat. (Reativo) mostra se o trabalho considera o conceito de reativo. A coluna *Tag* indica se o trabalho utiliza o conceito de *tags*. A coluna Par. (Paralelismo) indica se o trabalho considera o conceito de paralelismo em seu desenvolvimento. Por fim, a coluna Des. (Desacoplamento) indica se o trabalho permite o desacoplamento do processo ETL do ambiente operacional.

Ao analisar a Tabela 19, percebe-se que o Imã de Dados difere dos trabalhos fortemente relacionados, pois o mesmo foi projetado com os recursos mostrados na tabela. Esses recursos fazem com o Imã de Dados tenha um melhor desempenho na execução das tarefas as quais foi projetado para executar. Além do desempenho, o Imã de Dados permite um melhor desempenho do ponto de vista operacional, pois, uma vez configurado pelo usuário administrador, o Imã de Dados consegue executar suas tarefas de forma reativa e sem utilizar técnicas intrusivas para buscar os dados de interesse, tais como gatilhos, arquivos de *log* ou outra técnica CDC. Como consequência, o Imã de Dados obtém um melhor tempo de resposta e escalabilidade, como pode ser analisado no Capítulo 6, na seção 6.3.5, na qual aborda-se os resultados dos testes experimentais.

7.3 Análise dos trabalhos relacionados e estado da arte do processo ETL

Ao final da execução da revisão sistemática, foi possível obter todos os trabalhos relacionados ao tema de pesquisa, responder as questões de pesquisa definidas em seu protocolo, identificar todos os conceitos, técnicas, ferramentas, abordagens em geral e as arquiteturas já propostas na literatura. Além disso, foi possível comparar todos os trabalhos com requisitos que permitem identificar o que cada trabalho se propõe a resolver. Além do mais, foi possível identificar o estado atual do processo ETL em cada uma das fases. Dessa forma, os trabalhos que compõem o estado da arte em cada uma das fases do processo ETL serão apresentados a seguir.

7.3.1 Extração de dados

De forma geral, o processo de extração de dados pode ser realizado de duas formas: 1) carregamento completo, em que ocorre o carregamento de todos os dados operacionais para o DW; 2) carregamento incremental, em que apenas os dados modificados do ambiente operacional são encaminhados para o DW. Segundo Jain et al. (JAIN et al., 2012) e Tank et al. (TANK et al., 2010), ao aplicar algum método CDC, é possível obter o carregamento incremental de dados. Esse fato faz com que o volume de dados operacional seja menor e, como consequência, ocorre um melhor desempenho no processo de extração de dados. Dessa forma, assume-se que o carregamento incremental de dados é o processo considerado como ponto inicial de discussão sobre a de extração de dados.

Chandra, H. (CHANDRA, 2018) desenvolveu um trabalho no qual testou diferentes tipos de fontes de dados juntamente com diferentes tipos de métodos CDC. Os tipos de fontes de dados testados foram: estrutura baseada em *flat file*, estrutura baseada em modelo hierárquico, estrutura de dados baseada em rede, estrutura de dados baseada no modelo relacional e estrutura de dados baseada no modelo relacional binário. Os métodos CDC testados foram: data da modificação no registro (atributo *timestamp* na tupla), *timestamp* no arquivo, replicação, *trigger* e arquivos de *log*. Os testes realizados foram conduzidos em três etapas: a primeira etapa foi a execução de consultas. A segunda etapa foi inserir os dados alterados para uma tabela temporária por meio dos métodos de *log* e *trigger*. Por fim, a terceira etapa foi executar os comandos *delete, insert* e *update* para as tabelas de fato envolvidas. O objetivo dos testes foi avaliar o percentual de desempenho dos métodos CDC nas estruturas de dados citadas. Além disso, quanto maior o percentual de desempenho, mais adequado é o método CDC para tal estrutura de dados do ponto de vista de desempenho em sua execução.

Como conclusão do trabalho, o autor diz que o método CDC adequado é diferente para cada tipo de fonte de dados. Para uma estrutura baseada em *flat file*, o método CDC mais adequado é o *timestamp* no arquivo, cujo o percentual de desempenho foi de 43.32%. Para uma estrutura baseada em modelo hierárquico, a melhor opção também é *timestamp* no arquivo, cujo

o percentual de desempenho foi de 72.84%. Para a estrutura de dados baseada em rede, o melhor método CDC avaliado foi também *timestamp* no arquivo, cujo o percentual de desempenho foi de 75.25%. Para a estrutura de dados baseada no modelo relacional, o método CDC mais adequado foi *trigger*, cujo o percentual de desempenho foi de 69.98%. Por fim, para a estrutura de dados baseada no modelo relacional binário, o melhor método CDC testado foi *trigger*, cujo o percentual de desempenho foi de 79.94%.

Por meio desse trabalho, foi possível analisar os principais métodos CDC aplicados no processo de extração de dados. Além disso, foi possível analisar o método CDC mais adequado de acordo com o contexto que se deseja aplicar e analisar o que já se propôs até o momento sobre extração de dados operacional. Contudo, o autor não cita outros trabalhos que, embora não tenham analisado de forma profunda os métodos CDC, também discutiram sobre o assunto, tais como Muddasir e Raghuveer (N; K, 2017), Muddasir e Raghuveer (MUDDASIR; RAGHUVEER, 2017), Jain et al. (JAIN et al., 2012), Tank et al. (TANK et al., 2010) e Vassiliadis e Simitsis (VASSILIADIS; SIMITSIS, 2009). Além do mais, Valencio et al. (VALENCIO et al., 2014) desenvolveu um trabalho no qual realizou a extração de dados por meio de *trigger* e realizou experimentos para validar sua proposta. Contudo, não foi citado também nesse trabalho de Chandra, H.

7.3.2 Transformação

Sabe-se que a fase de transformação é destinada ao tratamento dos dados, de modo a armazená-los no DW em um formato homogêneo, íntegro e consistente. Além disso, algumas das transformações ocorridas nessa fase são: correção de valores incorretos ou nulos, eliminação de valores duplicados, integração de esquema e instância, agregações, junções e demais operações para limpar e integralizar os dados. Além do mais, como pode-se notar, se torna necessária a intervenção do usuário em vários momentos, pois somente ele tem a capacidade de tomar alguma decisão de transformação. Em alguns casos, de acordo com a configuração de metadados, o processo de transformação pode, com o tempo, ser executado de forma automática.

Dado esse contexto e após a leitura dos trabalhos correlatos, foi possível notar uma carência de propostas para a fase de transformação. Alguns trabalhos, tais como Viana et al. (VIANA et al., 2005), MadeSukarsa et al. (MADESUKARSA et al., 2012), Halenar R. (HALENAR, 2013) e Biswas et al. (BISWAS et al., 2019), abordam o processo de transformação superficialmente, não apresentando de forma clara como ocorre o processo. Entretanto, apenas Viana et al. (VIANA et al., 2005) mostra como ocorre a configuração de metadados para permitir realizar a transformação. Além disso, Biswas et al. (BISWAS et al., 2019) mostra apenas um algoritmo de como ocorre o processo de eliminar dados duplicados. Entretanto, existem na literatura muitos trabalhos que tratam da deduplicação e outras inconsistências nos dados, tais como os trabalhos encontrados em (BILKE et al., 2005; BLEIHOLDER; NAUMANN, 2008; MARCELLO et al., 2009; CECCHIN et al., 2010). Além do mais, ele deixa claro em sua

conclusão que, como trabalhos futuros, pretende-se agregar outras transformações avançadas, tais como agregação e junção.

Um fato importante é que grande parte dos trabalhos correlatos tratam apenas as fases de extração e/ou carregamento do processo ETL. Já a fase de transformação é mantida fora do escopo dos trabalhos. Além disso, foi possível notar que muitas das propostas referentes a transformação de dados são realizadas apenas a nível de códigos fornecidos pela linguagem SQL, isto é, diretamente no banco de dados. Esse fato faz com que se torne impossível obter os dados curados da fase de transformação. Isso se deve ao fato de que, para realizar a transformação, é necessária a definição de metadados que definem a transformação e, em diversos casos, é necessária a intervenção do usuário. Logo, apenas utilizando a linguagem SQL se torna insuficiente para suprir a demanda por dados curados no processo de transformação.

De forma geral, pode-se definir o estado da arte do ponto de vista da transformação por meio dos trabalhos apresentados nessa subseção. Contudo, é importante destacar a carência de trabalhos que abordam tal fase. Como consequência dessa falta de estudos aprofundados na fase de transformação, pode-se dizer que existem muitos temas de pesquisa e várias lacunas ainda em aberto.

7.3.3 Carregamento

A partir do desenvolvimento da revisão sistemática, foi possível perceber que Bouaziz et al. (BOUAZIZ et al., 2017) desenvolveram um *survey* que faz referência ao *survey* desenvolvido por Wibowo, A. (WIBOWO, 2015), que, por sua vez, faz referência ao trabalho feito por Langseth, J. (LANGSETH, 2004). Nesse cenário, percebe-se que o trabalho feito por Langseth, J. (LANGSETH, 2004) serve como referência para os trabalhos mais atuais. Isso se deve ao fato que as soluções para os problemas de carregamento de dados apontadas por Langseth, J. (LANGSETH, 2004) são destacadas e referenciadas em grande parte dos trabalhos mais recentes, tais como nos trabalhos de Santos e Bernardino (SANTOS; BERNARDINO, 2008), Cuzzocrea et al. (CUZZOCREA et al., 2014), YiChuan e Yao (YICHUAN; YAO, 2012) e Li e Mao (LI; MAO, 2015). Portanto, Langseth, J. (LANGSETH, 2004) pode ser considerados estado da arte da fase de carregamento do processo ETL.

Além de Langseth, J. (LANGSETH, 2004), no trabalho desenvolvimento por Machado et al. (MACHADO et al., 2019) foram aplicadas técnicas avançadas para executar o processo ETL. Tais técnicas são paralelismo e *buffer* conjuntamente com sistemas pub/sub. Dessa forma, esse trabalho pode ser considerado o mais próximo ao Imã de Dados e também o estado da arte da fase de carregamento do processo ETL. Langseth, J. (LANGSETH, 2004) propõe como os dados devem ser carregados para o DW, enquanto Machado et al. (MACHADO et al., 2019) considera a utilização de tecnologias avançadas para o processo ETL no compartilhamento de dados entre ambientes heterogêneos (as diferenças entre essa proposta e o Imã de Dados foram explicadas na seção 7.1.2).

7.3.4 Consulta

Basicamente, uma consulta é realizada no DW ao mesmo tempo em que os dados operacionais são atualizados. Esse fato faz com que ocorra um problema chamado de contenção de consulta, ou seja, ocorre o conflito de atualização de dados e consultas sendo realizadas ao mesmo tempo. Contudo, Santos e Bernardino (SANTOS; BERNARDINO, 2008) desenvolveram uma estratégia que consiste em criar réplicas das tabelas do DW. Essas réplicas são criadas sem nenhum tipo de restrição e relacionamentos. Além disso, elas recebem continuamente os dados operacionais e permitem que as consultas sejam executadas em paralelo às atualizações de dados.

Após o desenvolvimento desse trabalho, diversos autores surgiram com trabalhos que se baseiam nessa proposta. Por exemplo: Cuzzocrea et al. (CUZZOCREA et al., 2014) e YiChuan e Yao (YICHUAN; YAO, 2012) desenvolveram uma proposta baseada em *multi-level caches*. Tais propostas são compostas por dois componentes básicos chamados de *Static Data* e *Dynamic Data*. Basicamente, ambos os componentes possuem a mesma estrutura mas são projetados com objetivos diferentes. O componente *Static Data* recebe os dados históricos e permite que consultas sejam executadas. Já o *Dynamic Data* recebe os dados atuais (diários) em tempo real sempre que tal dado sofrer alterações no ambiente operacional. Percebe-se que ambas as propostas são baseadas em um mesmo mecanismo, ou seja, a ideia base é a mesma para os três trabalhos já citados. Contudo, uma curiosidade negativa que envolve esses trabalhos é que Cuzzocrea et al. (CUZZOCREA et al., 2014) não cita o trabalho desenvolvido por YiChuan e Yao (YICHUAN; YAO, 2012). Eles citam apenas Santos e Bernardino (SANTOS; BERNARDINO, 2008). Além disso, YiChuan e Yao (YICHUAN; YAO, 2012) não citam o trabalho desenvolvido por Santos e Bernardino (SANTOS; BERNARDINO, 2008).

Li e Mao (LI; MAO, 2015) desenvolveram um trabalho baseado em uma área de armazenamento temporário externa ao DW, com a qual é possível lidar com consultas executadas sobre dados históricos e dados de tempo real. Segundo os autores, a utilização de uma área de armazenamento externa para armazenamento de dados auxilia o processo dinâmico de atualização e dados. Além disso, a proposta consiste em receber os dados operacionais e alocá-los dinamicamente na área de armazenamento temporário. Esse processo ocorre pelo fato da área de armazenamento temporário ser limitada em espaço. Logo, é possível distribuir os dados em diferentes bases de dados e, quando sincronizados com o DW, tais dados são removidos automaticamente da área de armazenamento temporário. Além do mais, é possível obter consultas a partir dos dados armazenados na área de armazenamento temporário.

Embora existam quatro trabalhos que buscam lidam com o problema de contenção de consultas, a proposta desenvolvida por Li e Mao (LI; MAO, 2015) pode ser considerada a solução mais atual para o problema. Isso se deve ao fato dessa solução envolver outros fatores que podem contribuir positivamente com o desempenho em tempo real, tais como a área de armazenamento externa para lidar com dados de tempo real e dois processos ETL, um para lidar com dados históricos e outro para lidar com dados de tempo real. Além disso, por meio dos experimentos

realizados, foi possível observar um melhor desempenho ao executar consultas OLAP a medida que o volume de dados aumenta, assim como um melhor desempenho quando a frequência de atualização contínua no DW aumenta. Entretanto, uma característica negativa é que Li e Mao (LI; MAO, 2015) não citam nenhum dos trabalhos mencionados nessa seção. Esse fato faz com que se torne difícil definir qual a melhor abordagem para lidar com o problema de contenção de consultas, visto que ambas as abordagens não foram testadas e avaliadas entre si.

Vale a pena destacar que ambos os trabalhos carecem de experimentos para validar a disponibilidade, tempo de resposta e a escalabilidade de tal proposta. Tais requisitos são essenciais para ambientes de tempo real e devem ser descritos de modo a permitir avaliar se tal proposta cumpri com tais requisitos. Dessa forma, considera-se que o trabalho definido como estado da arte do ponto de vista de consulta possui falhas em sua validação para ambientes de tempo real.

7.3.5 Comentários finais sobre os trabalhos relacionados

Após a leitura de todos os trabalhos fortemente relacionados ao tema de pesquisa, além de identificar qual o estado da arte sobre o processo ETL aplicado em ambientes de *data warehousing* em tempo real, foi possível analisar os procedimentos metodológicos adotados pelos autores ao desenvolver seus trabalhos e analisar como os autores chegaram nos resultados apresentados.

De forma geral, alguns pontos positivos puderam ser analisados: 1) alguns trabalhos antigos, tais como o de Langseth, J. (LANGSETH, 2004) e Vassiliadis e Simitsis (VASSILIADIS; SIMITSIS, 2009) servem como base e são citados por muitos dos trabalhos mais recentes; 2) Grande parte dos trabalhos, após apresentarem suas propostas, expõem seus trabalhos futuros e lacunas ainda bem aberto. Isso faz com que outros pesquisadores possam iniciar suas pesquisas e investigações a partir daquele ponto já estudado. Contudo, alguns pontos negativos também puderam ser analisados, tais como: 1) todos os trabalhos relacionados não se comparam entre si, apenas se citam como um trabalho correlato ou como um exemplo motivacional. Além do mais, a proposta feita por um autor não é testada e validada contra uma proposta desenvolvida por outro autor; 2) existem trabalhos mais antigos que não são conhecidos pelos trabalhos mais recentes; 3) todos os trabalhos não validam experimentalmente suas propostas. 4) não foi possível identificar nenhum trabalho publicado em periódico, apenas trabalhos publicados em conferências.

Com relação ao item 3, os trabalhos Javed e Nawaz (JAVED; NAWAZ, 2010), YiChuan e Yao (YICHUAN; YAO, 2012) realizaram experimentos analisando o desempenho da arquitetura desenvolvida contra si mesmo. MadeSukarsa et al. (MADESUKARSA et al., 2012) realizou um experimento que mostra o objetivo para o qual foi desenvolvido. Valencio et al. (VALENCIO et al., 2014) validou sua proposta por meio de um experimento cujo objetivo foi analisar o desempenho da proposta contra si mesmo e mostrar que a proposta cumpre com o objetivo para o qual foi desenvolvido. Mao et al. (MAO et al., 2014) e Li e Mao (LI; MAO, 2015)

validaram suas propostas por meio de um experimento cujo objetivo foi avaliar o desempenho da arquitetura proposta e também avaliar o desempenho de consultas OLAP. Ambos os experimentos foram testados contra si mesmo. Guo et al. (GUO et al., 2015) analisou o tempo de resposta e desempenho da proposta contra si mesmo. Guerreiro et al. (GUERREIRO et al., 2016) avaliaram sua proposta por meio de um experimento usando o processo ETL tradicional e a arquitetura proposta. Contudo, os autores não fazem nenhuma menção sobre qual trabalho ou quais métodos foram considerados durante os experimentos. Por fim, Muddasir e Raghuveer (MUDDASIR; RAGHUVEER, 2017) realizaram experimentos cujo o objetivo foi apenas mostrar o funcionamento da arquitetura proposta.

Como conclusão geral sobre os trabalhos relacionados e o Imã de Dados, pode-se dizer que o Imã de Dados é uma arquitetura altamente importante para organizações que desejam tomar decisões com dados atualizados em tempo real. Além disso, o Imã de Dados difere das outras abordagens devido aos novos conceitos introduzidos eu seu projeto, os quais fazem com que o Imã de Dados possa obter melhor desempenho e escalabilidade, mesmo quando há um grande volume de dados sendo gerado no ambiente operacional e são requisitados no DW.

É importante destacar que esse tema de pesquisa possui muitas possibilidades de pesquisa em aberto, visto que muitos autores apresentam apenas uma ideia inicial, mas não procuraram resolver um problema como um todo e tampouco deram continuidade nas investigações. Além disso, as soluções já desenvolvidas não são testadas e validadas entre si, o que se torna impossível, metodologicamente, avaliar qual a melhor solução para lidar com cada fase do processo ETL. Dessa forma, é possível utilizar as propostas já disponíveis na literatura como base de conhecimento e impulsionar o processo ETL com base nos conceitos aplicados no Imã de Dados.

Para encerrar este trabalho, o Anexo A mostra a revisão sistemática feita ao longo deste trabalho. O desenvolvimento da revisão sistemática foi altamente importante para desenvolver este trabalho, pois foi possível obter o maior número de trabalhos relacionados ao tema da pesquisa. Além disso, foi possível identificar os conceitos, características, problemas já resolvidos, problemas não resolvidos, pontos positivos e negativos sobre o tema, a partir dos trabalhos identificados na revisão sistemática.

Capítulo 8
CONCLUSÃO

Após ser apresentada a contextualização, motivação, problema e objetivo no Capítulo 1, a fundamentação teórica dos principais conceitos acerca do tema de pesquisa no Capítulo 2, os principais conceitos do processo ETL no Capítulo 3, as características de um ambiente de *data warehousing* em tempo real no Capítulo 4, a apresentação e detalhamento da arquitetura do Imã de Dados no Capítulo 5, os testes experimentais no Capítulo 6 e os trabalhos relacionados no Capítulo 7, este capítulo apresenta a conclusão desta pesquisa de Doutorado e a indicação de trabalhos futuros. Além disso, neste capítulo será discutida a confirmação das hipóteses definidas na seção 1.5.

Esta tese apresentou detalhadamente o Imã de Dados, uma nova e inovadora arquitetura para realizar o processo ETL em tempo real em ambientes de *data warehousing*. O Imã de Dados introduz o conceito de *tags*, as propriedades não intrusiva e reativa e a utilização de sistemas *Publish/Subscribe* ao se projetar o processo ETL. Além disso, visto que o conceito de tempo real não foi profundamente discutido nos trabalhos relacionados, o Imã de Dados mostra uma maneira inovadora de pensar sobre o conceito de tempo real por meio de uma perspectiva de *soft real-time*. Além do mais, este trabalho definiu o conceito de tempo real corretamente e discutiu sua aplicabilidade e viabilidade para a maioria das aplicações que fazem uso de processos ETL em tempo real.

Ao aplicar a propriedade não intrusiva, o Imã de Dados não acessa as fontes de dados operacionais para extrair os dados de interesse. Portanto, o Imã de Dados não faz uso de *wrappers*, gatilhos, arquivos de *log* ou arquivos delta. Em vez disso, o Imã de Dados foi projetado para receber dados de interesse de fontes de dados operacionais. Para tanto, o Imã de Dados faz uso da propriedade chamada *Tag*. Basicamente, uma *tag* é um indicador de que um item de dados de fontes de dados operacionais será usado e armazenado em um *data warehouse* para auxiliar no processo de tomada de decisão. Portanto, esta propriedade permite desacoplar o processo ETL e as fontes de dados operacionais e por consequência, permite obter um ganho de desempenho para o processo ETL, pois a fase de extração torna-se menos custosa para ser realizada, uma vez que o Imã de Dados não precisa lidar com a forma de acesso e a heterogeneidade das fontes de dados operacionais.

Ao aplicar a propriedade reativa, o Imã de Dados reage a um evento de inserção nas fontes de dados operacionais e recebe os dados. Para isso, o Imã de Dados faz uso de um sistema pub/sub, o qual permite o compartilhamento de dados entre ambientes heterogêneos. Portanto, o Imã de Dados pode receber o item de dados de fontes de dados operacionais, identificar o item de dados e sua *tag*, identificar os *data warehouses* interessados em armazenar dados atribuídos à *tag* e armazenar automaticamente os dados nos *data warehouses* de destino. Para tanto, o Imã de Dados faz uso de metadados que permitem identificar as *tags*, itens de dados de origem e os *data warehouses* de destino.

A partir dos testes experimentais com dados reais foi possível validar a viabilidade de aplicar o Imã de Dados no domínio da pecuária digital, mais especificamente em uma fazenda leiteira. Neste experimento, o Imã de Dados apresentou bom desempenho e mostrou viabilidade para realizar processos ETL em tempo real. Além disso, a partir dos testes experimentais realizados com dados sintéticos foi possível validar os requisitos de tempo real. O Imã de Dados garantiu disponibilidade, escalabilidade com o aumento do volume de dados e obteve baixo tempo de resposta em sua execução.

Ainda com relação aos experimentos, foi possível comprovar as hipóteses definidas na seção 1.5. Por meio dos testes experimentais com dados sintéticos, foi possível observar que o Imã de Dados obteve um grande ganho de desempenho em relação a técnica tradicional de gatilho, comumente utilizada em projetos de processo ETL em ambientes de *data warehousing*. A utilização das propriedades *tag*, não intrusiva e reativa e de um sistema pub/sub garantiu o desacoplamento entre o processo ETL e as fontes de dados operacionais. A aplicação dessas propriedades fez com que a execução do processo ETL fosse menos custosa, visto que não é necessário lidar com a heterogeneidade dos dados e com o acesso direto às fontes de dados operacionais. Além disso, a aplicação dessas propriedades favorece a redução do *overhead* nas fontes de dados operacionais. Essas propriedades conjuntamente com as técnicas de *buffer* e paralelismo permitiu obter um maior ganho de desempenho e garantir a escalabilidade do processo ETL.

Todas essas características do Imã de Dados representam e comprovam a maneira inovadora de se pensar sobre processos ETL em tempo real em ambientes de *data warehousing*. Nesse sentido, é possível desacoplar o processo ETL e as fontes de dados operacionais e executá-lo de forma não intrusiva e reativa, o que por sua vez pode efetivamente obter um grande ganho de desempenho na execução do processo ETL. Além disso, o Imã de Dados se tornou uma mudança de paradigma ao se pensar em projetos de processo ETL em tempo real em ambientes de *data warehousing*.

8.1 Trabalhos futuros

Os trabalhos futuros após o desenvolvimento desta pesquisa de doutorado incluem:

- Permitir que o Imã de Dados filtre os dados críticos e não críticos no processo de validação. Nesse sentido, espera-se priorizar a validação de dados críticos e postergue a validação de dados não críticos, pois no projeto atual do Imã de Dados não há distinção entre dados críticos e não críticos.
- Pretende-se investigar o impacto da fase de transformação no processo de ETL em tempo real.
- Outro aspecto a ser investigado é o tratamento distinto de dados em tempo real e dados históricos em repositórios separados, de modo a possibilitar a execução do Imã de Dados e as consultas OLAP paralelamente.

8.2 Produções científicas

Antes de encerrar este trabalho, é importante destacar as produções bibliográficas e técnicas desenvolvidas ao longo de toda essa pesquisa de Doutorado.

8.2.1 Artigos aceitos em eventos Qualis A4

- Título - An Innovative Method to Extract Data in a Real-time Data Warehousing Environment
  Evento - 9th International Conference on Computational Science and Engineering (CSE)
- Título - A Novel Solution to Perform Real-Time ETL Process based on Non-Intrusive and Reactive Concepts
  Evento - 2021 International Conference on Computational Science and Computational Intelligence (CSCI)
- Título - A Survey of Real-time ETL Process Applied to Data Warehousing Environments
  Evento - 19th International Conference on Information Technology: New Generations (ITNG)

8.2.2 Artigo sob revisão em periódico Qualis A4

- Periódico - Heliyon
  Vale destacar que este artigo já se encontra indexado na base de dados do Heliyon.

8.2.3 Artigos que serão escritos

- Artigo da implementação do Imã de Dados a ser publicado no periódico MethodX.

- Artigo sobre os dados produzidos pelos experimentos a ser publicado no periódico Data in Brief.

8.2.4 Produção técnica

- Monitoramento de Volume de Leite em Tempo Real (MVLTR) - Software para monitorar o volume de leite em tempo real.
- Gerador de Dados Sintéticos (GDS) - Gerador de dados criado para gerar os dados sintéticos no experimento com dados sintéticos.
- Gerador de Dados Sintéticos com Gatilho (GDSG) - Gerador de dados criado para gerar os dados sintéticos e serem consumidos por um gatilho.
- Ambiente IdC para pecuária leiteira - Ambiente IdC criado para detectar a entrada da vaca na sala de ordenha, coletar os dados de leite produzidos, detectar a saída da vaca da sala de ordenha e enviar os dados produzidos e integrados para o *broker*.
- Imã de Dados - Arquitetura não intrusiva e reativa para realizar o processo ETL em tempo real em ambientes de *data warehousing*.

Apêndice A

PROCESSO ETL EXECUTADO EM UM AMBIENTE DE DATA WAREHOUSING EM TEMPO REAL: REVISÃO SISTEMÁTICA

A.1 Considerações iniciais

O processo ETL aplicado em ambientes de *data warehousing* tem sido objeto de estudo de vários pesquisadores ao longo dos anos. Entretanto, esses estudos não são direcionados para ambientes cujos dados são gerados em tempo real. Dessa forma, é necessário realizar um levantamento do estado da arte sobre a execução do processo ETL em ambientes de *data warehousing* em tempo real. Esse levantamento do estado da arte foi feito por meio de uma revisão sistemática, a partir da qual foi possível identificar os estudos sobre o assunto, assim como as propostas já realizadas de arquiteturas, metodologias, métodos, técnicas, ferramentas, APIs (*Application Program Interfaces*) e tecnologias que lidam com o desafio de atender os requisitos de ambientes de *data warehousing* em tempo real. Esta pesquisa enfocará especificamente na etapa de ETL. Assim, foi possível também identificar os trabalhos que investigaram parcialmente (subconjunto das etapas E, T e L) ou em sua totalidade o processo ETL (todas as etapas E, T e L).

A.2 Revisão sistemática

A revisão sistemática tem o objetivo de identificar, interpretar e avaliar todos os aspectos relevantes sobre um determinado tema de pesquisa. A revisão sistemática pode ser realizada para: 1) identificar as lacunas existentes em um tema de pesquisa e propor novas investigações; 2) fornecer uma fundamentação teórica para um determinado tema de pesquisa; 3) sumarizar as evidências acerca de tecnologias (KITCHENHAM; CHARTERS, 2007).

O processo de revisão sistemática é guiado por um protocolo, o qual é composto por atividades que definem seu rigor e como a revisão deverá ser conduzida. Após a definição do

protocolo, é realizada a execução do protocolo a fim de obter o maior número de trabalhos relacionados ao tema de pesquisa de interesse. Por fim, é feita uma filtragem dos trabalhos retornados com o propósito de selecionar apenas os trabalhos relevantes ao tema de pesquisa.

Por meio do protocolo definido para a revisão, o leitor pode avaliar o rigor e completude com as quais os trabalhos, produto resultante da revisão sistemática, foram considerados relevantes ou não relevantes. Dessa forma, torna-se possível replicar o processo de revisão sistemática em pesquisas futuras para o mesmo tema de pesquisa (KITCHENHAM; CHARTERS, 2007).

O processo de revisão sistemática foi realizado com o auxílio de duas ferramentas: 1) Parsifal: essa ferramenta fornece meios de gerenciar todas as fases que envolve o processo de revisão, além de disponibilizar dados estatísticos sobre o resultado final da revisão sistemática; 2) Mendeley: essa ferramenta auxilia no gerenciamento das referências bibliográficas.

A.2.1 Planejamento

Na fase de planejamento ocorre a definição do protocolo, o qual é composto por elementos que servirão como guia para todo o processo de revisão sistemática. No protocolo define-se o escopo da revisão sistemática com o propósito de especificar como ocorrerão as demais fases do processo. O protocolo é definido por meio dos objetivos, das questões de pesquisa, das fontes utilizadas, dos idiomas considerados, das palavras-chave e dos critérios de inclusão e exclusão de trabalhos. Essas definições são essenciais para definir o escopo do trabalho, assim como para obter o maior número de trabalhos relevantes.

A.2.1.1 Objetivos

Os principais objetivos desta revisão sistemática foram:

- Identificar arquiteturas, estratégias, técnicas, metodologias, métodos, tecnologias e estudos em geral que investigaram o processo ETL aplicados especificamente em ambientes de *data warehousing* em tempo real.
- Identificar trabalhos nos quais o processo ETL é executado em ambientes diferentes de *data warehousing*.

A.2.1.2 Questões de pesquisa

As questões de pesquisa devem ser criadas a partir dos objetivos. Sua finalidade é servir como base para identificar os trabalhos que discutem e respondem essas questões. Se algum trabalho responder alguma dessas questões, considera-se então que esse trabalho tem relação com os objetivos da pesquisa. Dessa forma, as questões de pesquisa criadas, todas aplicadas no contexto de ambientes de *data warehousing* em tempo real, são:

Questão 1 : Quais trabalhos realizaram levantamento bibliográfico (*survey*) de soluções para ambientes de *data warehousing* em tempo real?

Questão 2 : Quais trabalhos realizaram a proposta de arquitetura para ambientes de *data warehousing* em tempo real?

Questão 3 : Quais trabalhos investigaram o processo ETL para ambientes de *data warehousing* em tempo real?

Questão 4 : Dentre os trabalhos que investigaram o processo ETL, quais etapas (extração, transformação e carga) cada trabalho de fato abordou propondo uma solução?

Questão 5 : Quais trabalhos investigaram o projeto do *data warehouse* para ambientes de *data warehousing* em tempo real em nível conceitual, lógico ou físico?

Questão 6 : Quais trabalhos investigaram o processamento de consultas OLAP para ambientes de *data warehousing* em tempo real?

Questão 7 : Quais trabalhos utilizaram o conceito de não intrusivo ao aplicar o processo ETL em ambientes de *data warehousing* em tempo real?

Questão 8 : Quais trabalhos utilizaram o conceito de reativo ao aplicar o processo ETL em ambientes de *data warehousing* em tempo real?

Questão 9 : Quais trabalhos utilizaram o conceito de paralelismo ao aplicar o processo ETL em um ambiente de *data warehousing* em tempo real?

Questão 10 : Quais trabalhos adotaram um sistema pub/sub como meio de compartilhamento de dados entre o ambiente operacional e o ambiente de *data warehousing* em tempo real?

A.2.1.3 Fontes de busca

Para definir as fontes de busca da revisão sistemática deve-se considerar alguns critérios, tais como: cobertura, conteúdo atualizado e disponibilidade. Sobre a cobertura, só serão consideradas as fontes que retornarem uma quantidade relevante de trabalhos. Mesmo assim, o conteúdo retornado deve ser recente e corresponder ao tema de pesquisa. Além disso, só serão consideradas as fontes de busca que permitirem acesso na íntegra aos seus trabalhos.

Dessa forma, as fontes consideradas para a revisão sistemática foram:

- IEEEXplore Digital Library;
- ACM Digital Lybrary;
- Scopus;
- Web Of Science.

- Google Scholar.

A.2.1.4 Idiomas dos trabalhos

Os idiomas definidos para a revisão sistemática são Inglês e Português. O Inglês deve ser considerado pois é um idioma internacionalmente aceito em trabalhos científicos. Além disso, na área de computação, quase a totalidade dos trabalhos científicos são escritos em Inglês. O Português é considerado pois sua exclusão eliminaria todos os trabalhos científicos de pesquisadores brasileiros.

A.2.1.5 Palavras-chave

As palavras-chave são aplicadas nas bases de dados a fim de retornar os trabalhos que se relacionam com elas. Com isso, com base nos objetivos e fontes de busca definidos, foram definidos dois termos principais: tempo real e ETL. Além disso, é necessário definir os sinônimos desses termos, de modo a retornar outros trabalhos que se relacionam com os termos. Dessa forma, as palavras-chave consideradas para a revisão sistemática, juntamente com seus respectivos sinônimos, são:

- Tempo Real: *realtime*, *zero-latency*, RTDW.
- ETL: *Extract-Transform*, *Extracting-Transforming*, ETL.

Percebe-se que o termo *data warehousing* foi desconsiderado dos termos propostos. Isso se deve ao fato de que um dos objetivos desta revisão sistemática é identificar evidências que apontam que o processo ETL pode ser aplicado em outros contextos. Portanto, se incluir a expressão *data warehousing* e seus sinônimos, a pesquisa será limitada apenas para esse ambiente. Além disso, outros termos foram desconsiderados, tais como *near* e *on-demand*. Isso se deve ao fato de que, para esta pesquisa, espera-se encontrar trabalhos cujo foco seja essencialmente em tempo real.

A.2.1.6 Critérios de inclusão e exclusão de trabalhos

A última etapa da fase de planejamento e definição do protocolo é determinar como deve ser a seleção dos trabalhos relevantes. Para isso, é necessário estabelecer os critérios para considerar um determinado trabalho relevante ou não relevante. A definição dos critérios garante que os trabalhos considerados relevantes correspondem apenas aos objetivos da pesquisa.

A Tabela 20 mostra os critérios definidos para considerar a relevância dos trabalhos retornados. A coluna Sigla representa um código que será utilizado ao longo deste trabalho para indicar a relevância de cada trabalho retornado. A coluna Critério corresponde a descrição de cada critério definido. A coluna Grau de Relevância indica o quão relevante é o trabalho que

Tabela 20 – Critérios de inclusão de trabalhos da revisão sistemática

Critérios de Inclusão		
Sigla	**Critério**	**Grau de Relevância**
I-1	Trabalhos que apresentam arquiteturas, estratégias, técnicas, metodologias, tecnologias e estudos em geral sobre o processo ETL aplicado em ambientes de *data warehousing* em tempo real.	Fortemente
I-2	Trabalhos que apresentam arquiteturas, estratégias, técnicas, metodologias, tecnologias e estudos em geral sobre o processo ETL em tempo real aplicado em ambientes diferentes de *data warehousing*.	Fortemente
I-3	Trabalhos que apresentam arquiteturas, estratégias, técnicas, metodologias, tecnologias e estudos em geral sobre o processo ETL aplicado em ambientes *data warehousing* convencional.	Fracamente

se enquadrar no critério definido. Se um determinado trabalho se enquadrar nos critérios I-1 ou I-2, o trabalho será considerado como fortemente relevante. Se algum trabalho se enquadrar no critério I-3, o trabalho será considerado fracamente relevante.

Tabela 21 – Critério de exclusão de trabalhos da revisão sistemática

Critérios de Exclusão		
Sigla	**Critério**	**Grau de Relevância**
E-1	Qualquer trabalho que não corresponda aos critérios de inclusão.	Descarte
E-2	Trabalhos que abordam otimização do processo ETL.	Descarte

A Tabela 21 mostra o critério definido para considerar os trabalhos não relevantes. Qualquer trabalho que se enquadrar no critério E-1 da Tabela 21, isto é, que não corresponder a nenhum dos critérios definidos na Tabela 20, será considerado excluído ou não relevante.

A.2.2 Execução

A fase de execução consiste em aplicar os termos nas fontes de dados definidas e obter o maior número de trabalhos relevantes sobre o tema de pesquisa. O protocolo previamente definido deve servir como guia para todas as etapas dessa fase. Vale a pena destacar que a revisão sistemática foi executada ao longo de todo o ano de 2019, parte de 2020 e no final de 2021. Em 2019, ano da primeira execução, diversos trabalhos foram recuperados e analisados. Em meados de 2020, a revisão sistemática foi executada novamente e nenhum trabalho relevante foi identificado. Contudo, em dezembro de 2021, a revisão sistemática foi executada e 5 novos trabalhos foram recuperados e analisados.

A.2.2.1 Construção da *string* de busca

A *string* de busca é composta pelas palavras-chave previamente definidas no protocolo. Além disso, deve-se utilizar o operador "AND", para expressar termos diferentes, ou o operador "OR", para agrupar termos sinônimos ou que representam técnicas diferentes. Dessa forma, a *string* de busca ficou assim definida:

"(“RTDW” OR "Real-Time"OR "Real Time"OR "RealTime"OR "Zero-Latency"OR "Zero Latency") AND ("ETL"OR "Extract-Transform"OR "Extract Transform"OR "Extract, Transform"OR "Extracting-Transforming"OR "Extracting Transforming"OR "Extracting, Transforming")"

A.2.2.2 Execução das consultas

Ao executar uma consulta, é necessário indicar em qual parte do texto espera-se encontrar as palavras-chave definidas. Ou seja, é necessário indicar se os termos devem aparecer no título, no resumo, no corpo do texto ou em outras partes do texto. Além disso, cada fonte de busca tem um formato próprio para executar a *string* de busca. Dessa forma, é necessário adequá-la para o formato de cada fonte de busca. Assim, a *string* de busca dessa revisão sistemática considera apenas o título e o resumo dos trabalhos e ficou assim definida para cada fonte de busca:

- **IEEEXplore Digital Library:**

  (("Document Title":"RTDW"OR "Document Title":"Real-Time"OR "Document Title":"Real Time"OR "Document Title":"RealTime"OR "Document Title":"Zero-Latency"OR "Document Title":"Zero Latency") AND ("Document Title":"ETL"OR "Document Title":"Extract-Transform"OR "Document Title":"Extract Transform"OR "Document Title":"Extract, Transform"OR "Document Title":"Extracting-Transforming"OR "Document Title":"Extracting Transforming"OR "Document Title":"Extracting, Transforming")) OR (("Abstract":"RTDW"OR "Abstract":"Real-Time"OR "Abstract":"Real Time"OR "Abstract":"RealTime"OR "Abstract":"Zero-Latency"OR "Abstract":"Zero Latency") AND ("Abstract":"ETL"OR "Abstract":"Extract-Transform"OR "Abstract":"Extract Transform"OR "Abstract":"Extract, Transform"OR "Abstract":"Extracting-Transforming"OR "Abstract":"Extracting Transforming"OR "Abstract":"Extracting, Transforming"))

- **ACM Digital Library:**

  Title:(("RTDW"OR"Real-Time"OR "Real Time"OR "RealTime"OR "Zero-Latency"OR "Zero Latency") AND ("ETL"OR "Extract-Transform"OR "Extract Transform"OR "Extract, Transform"OR "Extracting-Transforming"OR "Extracting Transforming"OR "Extracting, Transforming")) OR Abstract:(("RTDW"OR "Real-Time"OR "Real Time"OR "Real-Time"OR "Zero-Latency"OR "Zero Latency") AND ("ETL"OR "Extract-Transform"OR

"Extract Transform"OR "Extract, Transform"OR "Extracting-Transforming"OR "Extracting Transforming"OR "Extracting, Transforming"))

- **Scopus:**

  (Title ("RTDW") OR Title ("Real-Time") OR Title ("Real Time") OR Title ("RealTime") OR Title ("Zero-Latency") OR Title ("Zero Latency") AND Title ("ETL") OR Title ("Extract-Transform") OR Title ("Extract Transform") OR Title ("Extract, Transform") OR Title ("Extracting-Transforming") OR Title ("Extracting Transforming") OR Title ("Extracting, Transforming")) OR (ABS ("RTDW") OR ABS ("Real-Time") OR ABS ("Real Time") OR ABS ("RealTime") OR ABS ("Zero-Latency") OR ABS ("Zero Latency") AND ABS ("ETL") OR ABS ("Extract-Transform") OR ABS ("Extract, Transform") OR ABS ("Extract Transform") OR ABS ("Extracting-Transforming") OR ABS ("Extracting, Transforming") OR ABS ("Extracting Transforming")) AND LANGUAGE ("English") AND SUBJAREA ("COMP") AND (LIMIT-TO (DOCTYPE , "ar") OR LIMIT-TO (DOCTYPE , "ch"))

- **Web Of Science:**

  (((TI=(“RTDW” OR "Real-Time"OR "Real Time"OR "RealTime"OR "Zero-Latency"OR "Zero Latency")) AND TI=("ETL"OR "Extract-Transform"OR "Extract Transform"OR "Extract, Transform"OR "Extracting-Transforming"OR "Extracting Transforming"OR "Extracting, Transforming")) OR AB=(“RTDW” OR "Real-Time"OR "Real Time"OR "RealTime"OR "Zero-Latency"OR "Zero Latency")) AND AB=("ETL"OR "Extract-Transform"OR "Extract Transform"OR "Extract, Transform"OR "Extracting-Transforming"OR "Extracting Transforming"OR "Extracting, Transforming")

- **Google Scholar:**

  Realtime ETL.

Vale destacar que a fonte de dados Google Scholar não permite a customização da *string* de busca. Dessa forma, não foi possível construir a *string* de busca contendo termos e os operadores *OR* ou *AND*. Além disso, não foi possível fazer com que a busca fosse realizada sobre o resumo de cada trabalho. Portanto, foram considerados apenas os termos citados, pois assim foram retornados todos os trabalhos relacionados com *realtime* e ETL em ambientes de *data warehousing* ou em outros tipos de ambientes baseando-se apenas no título. Caso o título seja relevante, o resumo do trabalho foi analisado para a verificação final de seu grau de relevância. Outro detalhe importante dessa fonte de busca é que, por se tratar de um grande volume de trabalhos retornados, foram considerados apenas as cinquenta primeiras páginas da busca.

A.2.3 Seleção dos estudos

A fase de seleção dos trabalhos consiste em filtrar os trabalhos obtidos na fase de execução, a fim de selecionar apenas os trabalhos que correspondem aos objetivos da revisão sistemática. Além disso, essa filtragem ocorre com base nos critérios de inclusão e exclusão de trabalhos definidos nas Tabelas 20 e 21 respectivamente. Essa fase foi dividida em quatro etapas:

A.2.3.1 Processo de importação

Os trabalhos obtidos por meio da execução das consultas, juntamente com seu respectivo resumo, foram extraídos para o formato *Bibtex* e importados para a ferramenta Parsifal.

A Tabela 22 mostra a quantidade de trabalhos obtidos após a realização da primeira etapa. A fonte de dados IEEEXplore obteve 57 trabalhos, sendo a maior quantidade de trabalhos dentre todas as fontes de dados utilizadas. A fonte de dados *Web Of Science* obteve 37 trabalhos, a fonte de dados ACM obteve 19 trabalhos, a fonte de dados Scopus obteve 32 trabalhos e a fonte de dados Google Scholar obteve 36 trabalhos. Em resumo, a execução da *string* de busca nas fontes de busca resultou em 181 trabalhos.

Tabela 22 – Total de trabalhos retornados para cada fonte de busca pesquisada.

Trabalhos Retornados	
Fonte	**Qtde**
IEEEXplore *Digital Library*	57
Web Of Science	37
ACM *Digital Lybrary*	19
Scopus	32
Google Scholar	36
Total	**181**

A.2.3.2 Processo de limpeza de trabalhos

Após a etapa de importação, foi executado o processo de limpeza de trabalhos. Esse processo consiste em remover os trabalhos duplicados, ou seja, várias fontes de dados retornaram o mesmo trabalho. Neste caso, deve-se considerar apenas um desses trabalhos.

A Tabela 23 mostra a quantidade de trabalhos obtidos após a limpeza de trabalhos. O total de trabalhos retornados foi de 181 trabalhos, sendo que 25 trabalhos foram classificados como repetidos. Portanto, ao final do processo de limpeza de trabalhos, foi possível obter um total de 156 trabalhos. Vale a pena destacar que essa quantidade de trabalhos foi obtida por meio da ferramenta Parsifal. A própria ferramenta identifica os trabalhos duplicados e retorna a quantidade e quais foram os trabalhos.

Tabela 23 – Total de trabalhos após o processo de limpeza de trabalhos.

Total de trabalhos após o processo de limpeza			
Fonte	**Qtde**	**Duplicados**	**Total**
IEEEXplore *Digital Library*	57		
Web Of Science	37		
ACM *Digital Lybrary*	19		
Scopus	32		
Google Scholar	36		
Totalização	181	25	**156**

A.2.3.3 Processo de seleção inicial

A seleção inicial dos trabalhos ocorreu por meio da leitura do título e resumo dos mesmos, baseando-se nos critérios de inclusão e exclusão de trabalhos definidos na seção A.2.1.6. Além disso, alguns trabalhos não apresentaram uma explicação clara do resumo e que permitisse sua classificação pelos critérios de inclusão. Nesses casos, foi feita uma leitura da introdução e conclusão de cada trabalho.

Tabela 24 – Total de trabalhos relevantes e descartados para cada fonte de dados pesquisada.

Trabalhos Relevantes			
Fonte	**Total**	**Forte Rel.**	**Descarte**
IEEEXplore *Digital Library*	57	17	40
Web Of Science	37	6	31
ACM *Digital Lybrary*	19	3	11
Scopus	32	9	23
Google Scholar	36	25	11
Total	**181**	**60**	**121**

A Tabela 24 apresenta a quantidade de trabalhos obtidos após a realização do processo de seleção inicial. Os trabalhos considerados como Fortemente Relacionados foram 60 trabalhos. Os trabalhos classificados como Descarte foram 121 trabalhos. Vale a pena destacar que os trabalhos descartados são trabalhos classificados conforme o critério de exclusão de trabalhos apresentado na Tabela 21 e os trabalhos classificados como Fracamente Relevante, conforme o critério apresentado na Tabela 20.

A.2.3.4 Processo de análise e extração de dados

O processo de análise e extração de dados corresponde a leitura completa e detalhada dos trabalhos considerados fortemente relevantes. A análise detalhada dos trabalhos relacionados deve ser realizada para identificar os principais elementos que correspondem aos objetivos deste trabalho de Doutorado. Além disso, por meio da leitura detalhada é possível responder a todas as questões de pesquisa definidas no protocolo. Dessa forma, espera-se envolver todos os pontos relacionados ao tema de pesquisa. Vale destacar que nessa seção serão apresentados apenas os

trabalhos definidos como fortemente relacionados. Dessa forma, os trabalhos descartados não serão contemplados.

A Tabela 25 mostra o resumo dos trabalhos fortemente relevantes que foram resultado do processo de seleção inicial. Além disso, essa tabela corresponde aos trabalhos indicados na coluna Forte Rel. da Tabela 24, porém, de forma detalhada. As colunas Título e Autor correspondem, respectivamente, o título e o autor do trabalho. A coluna Ano corresponde ao ano em que o trabalho foi publicado. A coluna Critério representa qual o critério adotado para o trabalho ser considerado relevante, com base na seção A.2.1.6.

Trabalhos para fase de análise e extração				
Id	**Título**	**Autor**	**Ano**	**Critério**
IEEE				
1	Analysis of Change Data Capture Method in Heterogeneous Data Sources to Support RTDW	Chandra, H.	2018	I-1
2	User Interface Support for a Big ETL Data Processing Pipeline	Figueiras et al.	2017	I-2
3	A Big Data Perspective of Current ETL Techniques	Phanikanth and Sudarsan	2017	I-2
4	Study of Methods to Achieve Near Real Time ETL	Muddasir and Raghuveer	2017	I-1
5	CDC and Union based near real time ETL	Muddasir and Raghuveer	2017	I-1
6	The Challenges of Extract, Transform and Loading (ETL) System Implementation For Near Real-Time Environment	Sabtu et al.	2017	I-1
7	An architecture for big data processing on intelligent transportation systems. An application scenario on highway traffic flows	Guerreiro et al.	2016	I-1
8	Problems and available solutions on the stage of EXTRACT, TRANSFORM, and Loading in near REAL-TIME data warehousing (a literature study)	Wibowo, A	2015	I-1
9	Real-Time Data ETL Framework for Big Real-Time Data Analysis	Li and Mao	2015	I-1

Continua na próxima página

				Continuação
Id	**Título**	**Autor**	**Ano**	**Critério**
10	Dynamic Mirror Based Real-Time Query ContentionSolution for Support Big Real-Time Data Analysis	Mao et al.	2014	I-1
11	Real Time Delta Extraction Based on Triggers to Support Data Warehousing	Valêncio et al.	2013	I-1
12	Measurement of the energy real-time data warehouse system design and Implementation	Hong-ye et al.	2012	I-1
13	Efficient Data Streams Processing in the Real Time Data Warehouse	Majeed et al.	2010	I-1
14	Data load distribution by semi real time data warehouse	Javed, M. e Nawaz, A.	2010	I-1
15	An Event-Based Near Real-Time Data Integration Architecture	Naeem et al.	2008	I-1
16	Real-time performance monitoring for an enterprise information management system	Chieu and Zeng	2008	I-1
17	RiTE: Providing On-Demand Data for Right-Time Data Warehousing	Thomsen et al.	2008	I-1
Web Of Science				
18	An adaptive and real-time based architecture for financial data integration	Fikri et al.	2019	I-1
19	DOD-ETL: distributed on-demand ETL for near real-time business intelligence	Machado et al.	2019	I-1
20	An online-processing critical patient monitoring system- an interoperability overview	Portela et al.	2017	I-1
21	An automated real-time integration and interoperability framework for bioinformatics	Lopes, P. e Oliveira, J.	2015	I-1
22	On-Demand ELT Architecture for Right-Time BI: Extending the Vision	Freudenreich et al.	2013	I-1
23	A real time data extraction, transformation and loading solution for semi-structured text files	Viana et al.	2005	I-2

Continua na próxima página

				Continuação
Id	**Título**	**Autor**	**Ano**	**Critério**
ACM Digital Lybrary				
24	Real-time ETL in Striim	Sen et al.	2018	I-1
25	Near real-time with traditional data warehouse architectures: Factors and How-to	Ferreira et al.	2013	I-1
26	Real-time data warehousing for business intelligence	Farooq and Sarwar	2010	I-1
Scopus				
27	A Novel Approach to Handle Huge Data for Refreshment Anomalies in Near Real-Time ETL Applications	Shaw et al.	2020	I-1
28	Real Time Data Warehouse Updates Through Extraction-Transformation-Loading Process Using Change Data Capture Method	Thulasiram, S., Ramaiah, N.	2020	I-1
29	Efficient incremental loading in ETL processing for real-time data integration	Biswas et al.	2019	I-1
30	Study of meta-data enrichment methods to achieve near real time ETL	Mohammed Muddasir, N. e Raghuveer, K.	2019	I-1
31	Issues and handy solutions addressed at every stage in real time data warehousing, I.E. ETL (Extraction, transformation & loading)	Wani, A. e Raina, B.	2019	I-1
32	ETL Framework for Real-Time Business Intelligence over Medical Imaging Repositories	Godinho et al.	2019	I-1
33	The challenges of extract, transform and load (ETL) for data integration in near real-time environment	Sabtu et al.	2017	I-1
34	AScale: Big/Small Data ETL and Real-Time Data Freshness	Martins et al.	2016	I-1
35	On-Demand ELT Architecture for Right-Time BI	Freudenreich et al.	2013	I-1
Google Scholar				

Continua na próxima página

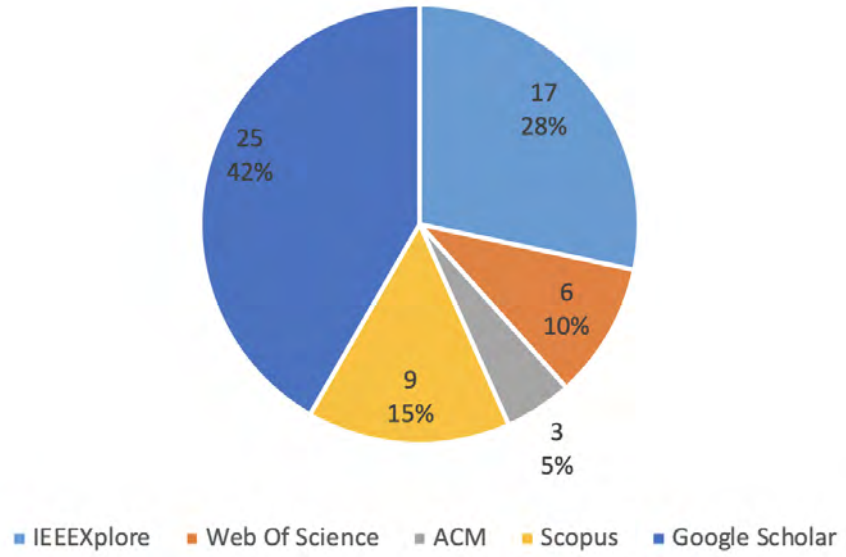
				Continuação
Id	**Título**	**Autor**	**Ano**	**Critério**
36	From traditional data warehouse to real time data warehouse	Bouaziz et al.	2017	I-1
37	Near Real-Time Data Warehousing Using ETL(Extract, Transform and Load) Tools	Gajanan et al.	2015	I-1
38	A New ETL Approach Based on Data Virtualization	Guo et al.	2015	I-1
39	Real-Time data ETL framework for big real-time data analysis	Li e Mao	2015	I-1
40	Enhancing Traditional Data Warehousing Architectures with Real-Time Capabilities	Cuzzocrea et al.	2014	I-1
41	A Survey of Real-Time Data Warehouse and ETL	Sabry e Ali	2014	I-1
42	An integration adaptation for real-time datawarehousing	Lebdaoui et al.	2014	I-1
43	Research on real time data warehouse architecture	Jia et al.	2013	I-1
44	A real time data warehouse approach for data processing	Obali et al.	2013	I-1
45	Real Time ETL Improvement	Halenar, R.	2013	I-1
46	From Data Warehouses to Streaming Warehouses: A Survey on the Challenges for Real-Time Data Warehousing and Available Solutions	Revathy et al.	2013	I-1
47	Research of Real-time Data Warehouse Storage Strategy Based on Multi-level Caches	YiChuan e Yao	2012	I-1
48	Change Data Capture on OLTP Staging Area for Nearly Real Time Data Warehouse Base on Database Trigger	MadeSukarsa et al.	2012	I-1
49	Reducing ETL Load Times by a New Data Integration Approach for Real-time Business Intelligence	Tank, D.	2012	I-1
50	ETL Evolution for Real-Time Data Warehousing	Kakish e Kraft	2012	I-1

Continua na próxima página

				Continuação
Id	**Título**	**Autor**	**Ano**	**Critério**
51	Refreshing Datawarehouse in Near Real-Time	Jain et al.	2012	I-1
52	Near real-time data warehousing with multi-stage trickle and flip	Zuters, J.	2011	I-1
53	An ETL strategy for real-time data warehouse	Zhou et al.	2011	I-1
54	24/7 real-time data warehousing: A tool for continuous actionable knowledge	Santos et al.	2011	I-1
55	Current state and future challenges of real-time ETL	Penzlin et al.	2009	I-1
56	Near Real Time ETL	Vassiliadis e Simitsis	2008	I-1
57	Real-time data warehouse loading methodology	Santos e Bernardino	2008	I-1
58	Study on Log-Based Change Data Capture and Handling Mechanism in Real-Time Data Warehouse	Shi et al.	2008	I-1
59	The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning Conforming, and Delivering Data	Kimball et al.	2004	I-1
60	Striving towards Near Real-Time Data Integration for Data Warehouses	Bruckner et al.	2002	I-1

Tabela 25 – Trabalhos relevantes para o processo de seleção final

Figura 44 – Total de trabalhos relevantes



A Figura 44 mostra um gráfico no qual aponta a quantidade e porcentagem dos trabalhos retornados considerados fortemente relacionados. A fonte de dados IEEE obteve 17 trabalhos relevantes, o que equivale a 28% de todos os trabalhos classificados como relevantes. A fonte de dados Google Scholar retornou 25 trabalhos relevantes, equivalendo a 42% dos trabalhos. Já as fontes *Web Of Science*, ACM e Scopus obtiveram, respectivamente, 6 trabalhos relevantes (10%), 3 trabalhos relevantes (5%) e 9 trabalhos relevantes (15%). Dessa forma, nota-se que a fonte de dados Google Scholar, embora não tenha retornado a maior quantidade de trabalhos, obteve o maior percentual de trabalhos fortemente relevantes. Além disso, os 60 trabalhos classificados como relevantes equivalem a 38,46% de todos os 156 trabalhos retornados pela revisão sistemática.

A.3 Considerações finais

Após o término da Revisão Sistemática, foi possível identificar todos os trabalhos que buscam propor soluções para o processo ETL em ambientes de *data warehousing* em tempo real. A partir da leitura de todos os trabalhos considerados fortemente relacionados ao tema de pesquisa, foi possível obter uma análise criteriosa sobre o estado da arte do processo ETL aplicado em ambientes de *data warehousing* de tempo real. Além disso, foi possível observar os pontos negativos e limitações de cada trabalho, assim como do tema de pesquisa como um todo. Por se tratar de um grande volume de trabalhos retornados, os pontos negativos, as limitações e as comparações técnicas entre cada trabalho relacionado e o Imã de Dados são amplamente detalhadas e discutidas no Capítulo 7, o qual retrata os trabalhos fortemente relacionados.

REFERÊNCIAS

AKKAOUI, Z. E.; ZIMANYI, E. Defining ETL worfklows using BPMN and BPEL. n. January 2015, p. 41, 2009. Citado na página 77.

BILKE, A.; BLEIHOLDER, J.; BÖHM, C.; DRABA, K.; NAUMANN, F.; WEIS, M. Automatic data fusion with HumMer. In: *VLDB 2005 - Proceedings of 31st International Conference on Very Large Data Bases*. [S.l.: s.n.], 2005. v. 3, p. 1251–1254. ISBN 1595931546. Citado na página 190.

BISWAS, N.; SARKAR, A.; MONDAL, K. C. Efficient incremental loading in ETL processing for real-time data integration. *Innovations in Systems and Software Engineering*, Springer London, 2019. ISSN 16145054. Disponível em: <https://doi.org/10.1007/s11334-019-00344-4>. Citado 2 vezes nas páginas 175 e 190.

BLEIHOLDER, J.; NAUMANN, F. Data Fusion. *ACM Comput. Surv. Article*, v. 41, n. 1, 2008. Citado na página 190.

BOKADE, M. B.; SDHANDE, S.; VYAVAHARE, H. R.; SCHOLAR, P. G. Framework Of Change Data Capture And Real Time Data Warehouse. 2013. Citado 3 vezes nas páginas 55, 56 e 58.

BOUAZIZ, S.; NABLI, A.; GARGOURI, F. From Traditional Data Warehouse To Real Time. *Intelligent Systems Design and Applications*, 2017. Citado 5 vezes nas páginas 62, 63, 154, 156 e 191.

BRUCKNER, R. M.; LIST, B.; SCHIEFER, J. Striving towards Near Real-Time Data Integration for Data Warehouses. *Lncs*, v. 2454, p. 317–326, 2002. Disponível em: <http://www.ifs.tuwien.ac.at/{~}bruckner/pubs/dawak2002{_}data{_}integra>. Citado na página 157.

CECCHIN, F.; De Aguiar Ciferri, C. D.; HARA, C. S. XML data fusion. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. [S.l.: s.n.], 2010. v. 6263 LNCS, p. 297–308. ISBN 3642151043. ISSN 03029743. Citado na página 190.

CHANDRA, H. Analysis of Change Data Capture Method in Heterogeneous Data Sources to Support RTDW. In: *2018 4th International Conference on Computer and Information Sciences (ICCOINS)*. [s.n.], 2018. p. 1–6. ISBN 978-1-5386-4744-8. Disponível em: <https://ieeexplore.ieee.org/document/8510574/>. Citado 4 vezes nas páginas 20, 23, 155 e 189.

Chetan Dwarkani, M.; Ganesh Ram, R.; JAGANNATHAN, S.; PRIYATHARSHINI, R. Smart farming system using sensors for agricultural task automation. *Proceedings - 2015 IEEE International Conference on Technological Innovations in ICT for Agriculture and Rural Development, TIAR 2015*, IEEE, n. Tiar, p. 49–53, 2015. Citado na página 37.

CHIEU, T. C.; ZENG, L. Real-time performance monitoring for an enterprise information management system. In: *IEEE International Conference on e-Business Engineering, ICEBE'08 - Workshops: AiR'08, EM2I'08, SOAIC'08, SOKM'08, BIMA'08, DKEEE'08*. [S.l.: s.n.], 2008. ISBN 9780769533957. Citado na página 159.

CIFERRI, C. D. d. A. *Distribuição dos dados em ambientes de data warehousing: o Sistema WebD 2W e algoritmos voltados à fragmentação horizontal dos dados*. 263 p. Tese (Doutorado) — Universidade Federal de Pernambuco, 2002. Citado na página 20.

COOPERSTOCK, J. R.; TANIKOSHI, K.; BEIRNE, G.; NARINE, T.; BUXTON, W. Evolution of a reactive environment. *Conference on Human Factors in Computing Systems - Proceedings*, v. 1, p. 170–177, 1995. Citado 2 vezes nas páginas 38 e 39.

CURRY, E. Message-Oriented Middleware. *Middleware for Communications*, p. 1–28, 2005. Citado 2 vezes nas páginas 41 e 43.

CUZZOCREA, A.; FERREIRA, N.; FURTADO, P. Enhancing Traditional Data Warehousing Architectures with Real-Time Capabilities. *ISMIS*, p. 456–465, 2014. Citado 4 vezes nas páginas 161, 170, 191 e 192.

DOBBELAERE, P.; ESMAILI, K. S. Industry paper: Kafka versus RabbitMQ: A comparative study of two industry reference publish/subscribe implementations. *DEBS 2017 - Proceedings of the 11th ACM International Conference on Distributed Event-Based Systems*, p. 227–238, 2017. Citado 2 vezes nas páginas 42 e 47.

DOSSOT, D. *RabbitMQ Essentials*. [s.n.], 2014. 1–182 p. ISBN 9781783983209. Disponível em: <https://www.packtpub.com/application-development/rabbitmq-essentials>. Citado na página 40.

ELLIS, B. *Real-Time Analytics - Techniques to analyse e visualize streaming data*. [S.l.: s.n.], 2014. Citado 4 vezes nas páginas 27, 29, 30 e 31.

FAROOQ, F.; SARWAR, S. M. Real-time data warehousing for business intelligence. In: *Proceedings of the 8th International Conference on Frontiers of Information Technology - FIT '10*. New York, New York, USA: ACM Press, 2010. p. 1–7. ISBN 9781450303422. Disponível em: <http://portal.acm.org/citation.cfm?doid=1943628.1943666>. Citado 2 vezes nas páginas 148 e 157.

FERREIRA, N.; MARTINS, P.; FURTADO, P. Near real-time with traditional data warehouse Architectures: factor and how to. *Proceedings of the 17th International Database Engineering & Applications Symposium on - IDEAS '13*, p. 68–75, 2013. Disponível em: <http://dl.acm.org/citation.cfm?doid=2513591.2513650>. Citado na página 150.

FIGUEIRAS, P.; COSTA, R.; GUERREIRO, G.; ANTUNES, H.; ROSA, A.; JARDIM-GONÇALVES, R.; ENG, D. D. User Interface Support for a Big ETL Data Processing Pipeline. p. 1437–1444, 2017. Citado 2 vezes nas páginas 22 e 62.

FIGUEIRAS, P.; COSTA, R.; GUERREIRO, G.; ANTUNES, H.; ROSA, A.; JARDIM-GONCALVES, R. User interface support for a big ETL data processing pipeline an application scenario on highway toll charging models. In: *2017 International Conference on Engineering, Technology and Innovation: Engineering, Technology and Innovation Management Beyond 2020: New Challenges, New Approaches, ICE/ITMC 2017 - Proceedings*. [S.l.: s.n.], 2017. ISBN 9781538607749. ISSN 1552-4973. Citado na página 174.

FIGUEIRAS, P.; GUERREIRO, G.; COSTA, R.; BRADESKO, L.; STOJANOVIC, N.; JARDIM-GONÇALVES, R. Big data harmonization for intelligent mobility: A dynamic toll-charging scenario. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 10034 LNCS, p. 76–86, 2017. ISSN 16113349. Citado na página 174.

FREUDENREICH, T.; FURTADO, P.; KONCILIA, C.; THIELE, M.; WAAS, F.; WREMBEL, R. An on-demand ELT architecture for real-time BI. *Lecture Notes in Business Information Processing*, v. 154, p. 50–59, 2013. ISSN 18651348. Citado 2 vezes nas páginas 66 e 169.

FUENTES, S.; VIEJO, C. G.; CULLEN, B.; TONGSON, E.; CHAUHAN, S.; DUNSHEA, F. Artificial intelligence applied to a robotic dairy farm to model milk productivity and quality based on cow data and daily environmental parameters. *Sensors*, v. 20, 05 2020. Citado 2 vezes nas páginas 22 e 62.

GODINHO, T. M.; LEBRE, R.; ALMEIDA, J. R.; COSTA, C. ETL Framework for Real-Time Business Intelligence over Medical Imaging Repositories. *Journal of Digital Imaging*, Journal of Digital Imaging, 2019. ISSN 1618727X. Citado na página 176.

GUERREIRO, G.; FIGUEIRAS, P.; SILVA, R.; COSTA, R.; JARDIM-GONCALVES, R. An architecture for big data processing on intelligent transportation systems. An application scenario on highway traffic flows. *2016 IEEE 8th International Conference on Intelligent Systems, IS 2016 - Proceedings*, p. 65–71, 2016. ISSN 1860949X. Citado 2 vezes nas páginas 174 e 194.

GUO, S. S.; YUAN, Z. M.; SUN, A. B.; YUE, Q. A New ETL Approach Based on Data Virtualization. *Journal of Computer Science and Technology*, v. 30, n. 2, p. 311–323, 2015. ISSN 10009000. Citado 2 vezes nas páginas 172 e 194.

HALENAR, R. Real Time ETL Improvement. *International Journal of Computer Theory and Engineering*, v. 4, n. 3, p. 405–409, 2013. ISSN 17938201. Citado 2 vezes nas páginas 168 e 190.

JAIN, T.; S, R.; SALUJA, S. Refreshing Datawarehouse in Near Real-Time. *International Journal of Computer Applications*, v. 46, n. 18, p. 975–8887, 2012. Disponível em: <https://pdfs.semanticscholar.org/0c8c/ac4020f70b414085752f4c753e81eb8050be.pdf>. Citado 7 vezes nas páginas 22, 56, 58, 62, 166, 189 e 190.

JAVED, M. Y.; NAWAZ, A. Data load distribution by semi real time data warehouse. *2nd International Conference on Computer and Network Technology, ICCNT 2010*, p. 556–560, 2010. Citado 2 vezes nas páginas 162 e 193.

JIA, R.; XU, S.; PENG, C. Research on real time data warehouse architecture. *Communications in Computer and Information Science*, v. 392 PART I, p. 333–342, 2013. ISSN 18650929. Citado na página 167.

KAKISH, K.; KRAFT, T. A. ETL Evolution for Real-Time Data Warehousing. *Conference on Information Systems Applied Research*, v. 5, n. 2214, 2012. Disponível em: <www.aitp-edsig.org>. Citado 4 vezes nas páginas 21, 23, 60 e 149.

KIMBALL, R.; CASERTA, J.; KIMBALL, R.; CASERTA, J. *The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning Conforming, and Delivering Data*. [S.l.: s.n.], 2004. 526 p. ISSN 1098-6596. ISBN 8126505540. Citado 6 vezes nas páginas 21, 55, 56, 58, 60 e 62.

KITCHENHAM, B.; CHARTERS, S. Guidelines for performing Systematic Literature Reviews in Software Engineering. *Technical report*, v. 2, n. 3, 2007. ISSN 00010782. Citado 2 vezes nas páginas 199 e 200.

KLEPPMANN, M. *Designing Data-Intensive Applications*. [S.l.: s.n.], 2017. ISBN 9781449373320. Citado 3 vezes nas páginas 30, 127 e 128.

KOPETZ, H. *Real-time Systems*. Boston: Springer, Boston, MA, 2002. 347 p. ISBN 079239657X. Citado 2 vezes nas páginas 24 e 27.

KULATUNGA, C.; SHALLOO, L.; DONNELLY, W.; ROBSON, E.; IVANOV, S. Opportunistic Wireless Networking for Smart Dairy Farming. *IT Professional*, v. 19, n. 2, p. 16–23, 2017. ISSN 15209202. Citado 3 vezes nas páginas 22, 36 e 62.

LANGSETH, J. Real-Time Data Warehousing: Challenges and Solutions. 2004. Citado 11 vezes nas páginas 23, 62, 66, 68, 70, 71, 148, 163, 167, 191 e 193.

LEBDAOUI, I.; ORHANOU, G.; ELHAJJI, S. An integration adaptation for real-time data warehousing. *International Journal of Software Engineering and its Applications*, v. 8, n. 11, p. 115–128, 2014. ISSN 17389984. Citado na página 171.

LI, X.; MAO, Y. Real-Time data ETL framework for big real-time data analysis. In: *2015 IEEE International Conference on Information and Automation, ICIA 2015 - In conjunction with 2015 IEEE International Conference on Automation and Logistics*. Lijiang, China: [s.n.], 2015. p. 1289–1294. ISBN 9781467391047. Citado 6 vezes nas páginas 22, 23, 171, 191, 192 e 193.

LIU, J. *Real-Time Applications*. [S.l.: s.n.], 2000. 409 p. Citado 4 vezes nas páginas 24, 27, 28 e 30.

MACHADO, G. V.; CUNHA, Í.; PEREIRA, A. C.; OLIVEIRA, L. B. DOD-ETL: distributed on-demand ETL for near real-time business intelligence. *Journal of Internet Services and Applications*, Journal of Internet Services and Applications, v. 10, n. 1, p. 1–15, 2019. ISSN 18690238. Citado 3 vezes nas páginas 177, 182 e 191.

MADESUKARSA, I.; Wayan Wisswani, N.; K. Gd. Darma Putra, I.; LINAWATI, L. Change Data Capture on OLTP Staging Area for Nearly Real Time Data Warehouse Base on Database Trigger. *International Journal of Computer Applications*, v. 52, n. 11, p. 32–37, 2012. Citado 3 vezes nas páginas 165, 190 e 193.

MAJEED, R. W.; RÖHRIG, R. Automated realtime data import for the i2b2 clinical data warehouse: Introducing the HL7 ETL cell. *Studies in Health Technology and Informatics*, v. 180, n. August, p. 270–274, 2012. ISSN 18798365. Citado na página 21.

MAO, Y.; MIN, W.; WANG, J.; JIA, B.; JIE, Q. Dynamic mirror based real-time query contention solution for support big real-time data analysis. *Proceedings of 2nd International Conference on Information Technology and Electronic Commerce, ICITEC 2014*, p. 229–233, 2014. Citado 2 vezes nas páginas 171 e 193.

MARCELLO, C.; STROPARO, C.; SILVA, E.; HARA, C. S. XFusion: Uma Ferramenta para Fusão e Limpeza de Dados XML. In: *Anais do Simpósio Brasileiro de Banco de Dados - Sessão de Demos*. [s.n.], 2009. Disponível em: <http://www.inf.ufpr.br/carmem/pub/sbbd09XFusion.pdf>. Citado na página 190.

MARTINS, P.; ABBASI, M.; FURTADO, P. AScale: Big/Small Data ETL and Real-Time Data Freshness. *Communications in Computer and Information Science*, v. 613, p. 315–327, 2016. ISSN 18650929. Citado na página 173.

MESITI, M.; FERRARI, L.; VALTOLINA, S.; LICARI, G.; GALLIANI, G.; DAO, M.; ZETTSU, K. StreamLoader: An event-driven ETL system for the on-line processing of heterogeneous sensor data. *Advances in Database Technology - EDBT*, v. 2016-March, p. 628–631, 2016. ISSN 23672005. Citado 2 vezes nas páginas 22 e 62.

MUDDASIR, M.; RAGHUVEER, K. CDC and Union based near real time ETL. In: *2nd International Conference On Emerging Computation and Information Technologies (ICECIT)*. [S.l.: s.n.], 2017. p. 1–5. Citado 8 vezes nas páginas 20, 21, 22, 23, 65, 175, 190 e 194.

MUKHERJEE, R.; KAR, P. A comparative review of data warehousing ETL tools with new trends and industry insight. In: *Proceedings - 7th IEEE International Advanced Computing Conference, IACC 2017*. [S.l.: s.n.], 2017. ISBN 9781509015603. ISSN 0353-9466. Citado 3 vezes nas páginas 19, 20 e 21.

N, M.; K, R. Study of Methods to Achieve Near Real Time ETL. In: *2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC)*. [S.l.: s.n.], 2017. p. 436–441. ISBN 9781538632437. Citado 4 vezes nas páginas 21, 23, 153 e 190.

N, M. M.; K, D. R. *A Novel Approach to Handle Huge Data for Refreshment Anomalies in Near Real-Time ETL Applications - Pág. 545*. [S.l.: s.n.], 2020. v. 1154. ISSN 21945365. ISBN 9789811540318. Citado na página 177.

NAEEM, M. A.; DOBBIE, G.; WEBER, G. An event-based near real-time data integration architecture. In: *Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOC*. [S.l.: s.n.], 2008. p. 401–404. ISBN 978-1-4244-4485-4. ISSN 15417719. Citado na página 158.

NARENDRA, G. M. Near Real-Time Data Warehousing Using ETL(Extract, Transform and Load) Tools. *International Journal of Advanced Research in Computer Engineering & Technology*, v. 4, n. 5, 2015. Citado na página 152.

OBALI, M.; DURSUN, B.; ERDEM, Z.; GORUR, A. K. A real time data warehouse approach for data processing. In: *2013 21st Signal Processing and Communications Applications Conference (SIU)*. IEEE, 2013. p. 1–4. ISBN 978-1-4673-5563-6. Disponível em: <http://ieeexplore.ieee.org/document/6531245/>. Citado na página 168.

ONICA, E.; FELBER, P.; MERCIER, H.; RIVIÈRE, E. Confidentiality-preserving publish/subscribe: A survey. *ACM Computing Surveys*, v. 49, n. 2, p. 1–41, 2016. ISSN 15577341. Citado 2 vezes nas páginas 40 e 46.

ONYALO, N.; KANDIE, H.; NJUKI, J. The Internet of Things, Progress Report for Africa: A Survey. *International Journal of Computer Science and Software Engineering*, v. 4, n. 9, p. 2409–4285, 2015. ISSN 13873326. Disponível em: <www.IJCSSE.org>. Citado 2 vezes nas páginas 33 e 34.

PENZLIN, F.; SCHINK, H.; BECKER, C.; KRUG, M.; DREILING, A. Current state and future challenges of real-time ETL. *Computer*, n. June, p. 6–10, 2009. Citado na página 148.

PHANIKANTH, K. V.; SUDARSAN, S. D. A big data perspective of current ETL techniques. *Proceedings - 2016 3rd International Conference on Advances in Computing, Communication and Engineering, ICACCE 2016*, p. 330–334, 2017. ISSN 00189219. Citado 5 vezes nas páginas 21, 22, 62, 154 e 156.

PLENTZ, P. D. M. *Mecanismos de Previsão de Perda de Deadline para Sistemas Baseados em Threads Distribuídas Tempo Real*. 210 p. Tese (Doutorado), 2008. Citado 2 vezes nas páginas 24 e 27.

RYU, M.; YUN, J.; MIAO, T.; AHN, I. Y.; CHOI, S. C.; KIM, J. Design and implementation of a connected farm for smart farming system. *2015 IEEE SENSORS - Proceedings*, p. 1–4, 2015. Citado 2 vezes nas páginas 22 e 62.

S, R.; BALAJI, S.; KARTHIKEYAN, N. K. From Data Warehouses to Streaming Warehouses: A Survey on the Challenges for Real-Time Data Warehousing and Available Solutions. *International Journal of Computer Applications*, v. 81, n. 2, p. 15–18, 2013. Citado 2 vezes nas páginas 150 e 156.

SABRY, F.; ALI, E. A Survey of Real-Time Data Warehouse and ETL. *International Journal of Scientific & Engineering Research*, v. 5, n. 7, 2014. ISSN 2229-5518. Disponível em: <http://www.ijser.org>. Citado 8 vezes nas páginas 20, 23, 52, 60, 65, 149, 150 e 156.

SABTU, A.; AZMI, N. F. M.; SJARIF, N. N. A.; ISMAIL, S. A.; YUSOP, O. M.; SARKAN, H.; CHUPRAT, S. The challenges of extract, transform and load (ETL) for data integration in near real-time environment. *Journal of Theoretical and Applied Information Technology*, v. 95, n. 22, p. 6314–6322, 2017. ISSN 18173195. Citado 5 vezes nas páginas 22, 24, 30, 31 e 62.

SABTU, A.; AZMI, N. F. M.; SJARIF, N. N. A.; ISMAIL, S. A.; YUSOP, O. M.; SARKAN, H.; CHUPRAT, S. The challenges of Extract, Transform and Loading (ETL) system implementation for near real-time environment. *International Conference on Research and Innovation in Information Systems, ICRIIS*, p. 3–7, 2017. ISSN 23248157. Citado 4 vezes nas páginas 22, 62, 153 e 156.

SANTOS, R. J.; BERNARDINO, J. Real-time data warehouse loading methodology. In: *Proceedings of the 2008 international symposium on Database engineering & applications - IDEAS '08*. [S.l.: s.n.], 2008. ISBN 9781605581880. Citado 6 vezes nas páginas 161, 162, 170, 178, 191 e 192.

SANTOS, R. J.; BERNARDINO, J.; VIEIRA, M. 24/7 Real-Time Data Warehousing: A Tool for Continuous Actionable Knowledge. In: *2011 IEEE 35th Annual Computer Software and Applications Conference*. IEEE, 2011. p. 279–288. ISBN 978-1-4577-0544-1. Disponível em: <http://ieeexplore.ieee.org/document/6032354/>. Citado 4 vezes nas páginas 63, 161, 162 e 170.

SHI, J.; BAO, Y.; LENG, F.; YU, G. Study on Log-Based Change Data Capture and Handling Mechanism in Real-Time Data Warehouse. In: *2008 International Conference on Computer Science and Software Engineering*. IEEE, 2008. p. 478–481. ISBN 978-0-7695-3336-0. Disponível em: <http://ieeexplore.ieee.org/document/4722662/>. Citado na página 162.

SRIVASTAVA, S.; SINGH, M.; GUPTA, S. Wireless Sensor Network: A Survey. *2018 International Conference on Automation and Computational Engineering, ICACE 2018*, v. 38, p. 159–163, 2019. Citado na página 35.

TANK, D. Reducing ETL Load Times by a New Data Integration Approach for Real-time Business Intelligence (PDF Download Available). *International Journal of Engineering Innovation & Research*, n. October, p. 2277–5668, 2012. Disponível em: <https://www.researchgate.net/publication/308953446{_}Reducing{_}ETL{_}Load{_}Times{_}by{_}a{_}New{_}Data{_}Integration{_}Approach{_}fo>. Citado na página 166.

TANK, D. M.; GANATRA, A.; KOSTA, Y. P.; BHENSDADIA, C. K. Speeding ETL processing in data warehouses using high-performance joins for changed data capture (CDC). *Proceedings - 2nd International Conference on Advances in Recent Technologies in Communication and Computing, ARTCom 2010*, n. Cdc, p. 365–368, 2010. Citado 3 vezes nas páginas 149, 189 e 190.

TARKOMA, S. *Publish/Subscribe Systems*. [S.l.: s.n.], 2012. ISBN 9781119951544. Citado 7 vezes nas páginas 40, 41, 42, 44, 46, 47 e 48.

THOMSEN, C.; PEDERSEN, T. B.; LEHNER, W. RiTE: Providing On-Demand Data for Right-Time Data Warehousing. *ICDE 2008*, v. 00, p. 456–465, 2008. Citado na página 160.

THULASIRAM, S.; RAMAIAH, N. *Real Time Data Warehouse Updates Through Extraction-Transformation-Loading Process Using Change Data Capture Method*. [S.l.: s.n.], 2020. v. 44. 552–560 p. ISSN 23674520. ISBN 9783030370503. Citado na página 178.

TOSHEV, M. *Learning RabbitMQ*. Birmingham, UK: Packt Publishing, 2015. ISBN 9781783984565. Citado 2 vezes nas páginas 39 e 40.

VAISMAN, A.; ZIMÁNYI, E. *Data Warehouse Systems - Design and Implementation*. Berlin, Heidelberg: Springer, 2014. 625 p. ISSN 09505849. ISBN 978-3-642-54654-9. Disponível em: <http://link.springer.com/10.1007/978-3-642-54655-6>. Citado na página 63.

VALENCIO, C. R.; MARIOTO, M. H.; ZAFALON, G. F. D.; MACHADO, J. M.; MOMENTE, J. C. Real time delta extraction based on triggers to support data warehousing. *Parallel and Distributed Computing, Applications and Technologies, PDCAT Proceedings*, p. 293–297, 2014. Citado 4 vezes nas páginas 65, 170, 190 e 193.

VASSILIADIS, P.; SIMITSIS, A. *Near Real Time ETL*. [S.l.: s.n.], 2009. v. 3. 1–38 p. ISBN 9780387874302. Citado 6 vezes nas páginas 51, 147, 156, 157, 190 e 193.

VIANA, N.; RAMINHOS, R.; MOURA-PIRES, J. A real time data extraction, transformation and loading solution for semi-structured text files. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 3808 LNCS, p. 383–394, 2005. ISSN 03029743. Citado 2 vezes nas páginas 157 e 190.

WADHWA, R. A Pub / Sub based Architecture to Support Public Healthcare Data Exchange. 2015. Citado na página 42.

WALKENBACH, J. *Kafka the Definitive Guide*. [S.l.: s.n.], 2010. 338–340 p. ISSN 1098-6596. ISBN 9781491936160. Citado na página 41.

WHITMORE, A.; AGARWAL, A.; Da Xu, L. The Internet of Things — A survey of topics and trends. *Information Systems Frontiers*, v. 17, n. 2, p. 261–274, 2015. ISSN 15729419. Citado na página 33.

WIBOWO, A. Problems and available solutions on the stage of Extract, Transform, and Loading in near real-time data warehousing (a literature study). *2015 International Seminar on Intelligent Technology and Its Applications, ISITIA 2015 - Proceeding*, p. 345–349, 2015. Citado 6 vezes nas páginas 21, 66, 151, 152, 156 e 191.

XU, L. D.; HE, W.; LI, S. Internet of things in industries: A survey. *IEEE Transactions on Industrial Informatics*, v. 10, n. 4, p. 2233–2243, 2014. ISSN 15513203. Citado na página 33.

XUE, H. Y.; YANG, Q.; LI, A. G. Measurement of the energy real-time data warehouse system design and Implementation. In: *Proceedings - 2012 International Conference on Computer Science and Service System, CSSS 2012*. [S.l.: s.n.], 2012. p. 2087–2090. ISBN 9780769547190. Citado na página 167.

YADRANJIAGHDAM, B.; POOL, N.; TABRIZI, N. A survey on real-time big data analytics: Applications and tools. In: *Proceedings - 2016 International Conference on Computational Science and Computational Intelligence, CSCI 2016*. [S.l.: s.n.], 2017. p. 404–409. ISBN 9781509055104. ISSN 1479-3261. Citado na página 19.

YICHUAN, S.; YAO, X. Research of Real-time Data Warehouse Storage Strategy Based on Multi-level Caches. *Physics Procedia*, Elsevier Srl, v. 25, p. 2315–2321, 2012. ISSN 18753892. Disponível em: <http://dx.doi.org/10.1016/j.phpro.2012.03.390>. Citado 4 vezes nas páginas 164, 191, 192 e 193.

YICK, J.; MUKHERJEE, B.; GHOSAL, D. Wireless sensor network survey. *Computer Networks*, v. 52, n. 12, p. 2292–2330, 2008. ISSN 13891286. Citado 2 vezes nas páginas 35 e 36.

ZAMORA-IZQUIERDO, M. A.; SANTA, J.; MARTÍNEZ, J. A.; MARTÍNEZ, V.; SKARMETA, A. F. Smart farming IoT platform based on edge and cloud computing. *Biosystems Engineering*, v. 177, p. 4–17, 2019. ISSN 15375110. Citado 3 vezes nas páginas 22, 36 e 62.

ZHOU, H.; YANG, D.; XU, Y. An ETL strategy for real-time data warehouse. *Advances in Intelligent and Soft Computing*, v. 124, p. 329–336, 2011. ISSN 18675662. Citado na página 163.

ZUTERS, J. Near real-time data warehousing with multi-stage trickle and flip. In: *Lecture Notes in Business Information Processing*. [S.l.: s.n.], 2011. ISBN 9783642245107. ISSN 18651348. Citado na página 163.