Pós-Graduação em Ciência da Computação

ETL4NoSQL: Um Framework Programável para Extração, Transformação e Carga de Banco de Dados NoSQL

Por

Carine Calixto Aguena

Dissertação de Mestrado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE/2017

Carine Calixto Aguena

ETL4NoSQL: Um Framework Programável para Extração, Transformação e Carga de Banco de Dados NoSQL

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientadora: Prof. Valéria Cesário Times

RECIFE
2017

Eu dedico este trabalho à toda minha família, amigos e professores que me deram todo apoio necessário para alcançar meus objetivos.

Agradecimentos

À minha família, meus pais que sempre foram meu alicerce e estrutura de vida, ao meu irmão e cunhada pelos estímulos e conselhos, à minha sobrinha pelo carinho, ao meu amor por todo suporte e compreensão nos momentos difíceis e desafiantes que este trabalho proporcionou.

À Prof. Valéria, o meu reconhecimento pela oportunidade de realizar este trabalho ao lado de alguém paciente e que transpira sabedoria; meu respeito e admiração pela sua serenidade, capacidade de análise do perfil de seus alunos, e pelo seu dom no ensino da Ciência, inibindo sempre a vaidade em prol da simplicidade e eficiência.

Aos amigos, e todos que de alguma forma colaboraram nessa grande jornada, os precursores de tudo, que exemplificam a ética e competência profissionais, a dedicação e o aprimoramento contínuos, pelo incentivo e oportunidade de convívio.

E, finalmente, dedico este trabalho à Deus, que sempre me guia e impulsiona pelos caminhos que acalmam meu coração. Agradeço pela maravilhosa oportunidade de obter novos conhecimentos e encontrar pessoas maravilhosas.

Sempre que houver alternativas, tenha cuidado. Não opte pelo conveniente, pelo confortável, pelo respeitável, pelo socialmente aceitável, pelo honroso. Opte pelo que faz o seu coração vibrar. Opte pelo que gostaria de fazer, apesar de todas as consequências.

—OSHO

Resumo

A integração de dados e os processos de extração, transformação e carga de dados (ETL) são procedimentos cruciais para a criação de *data warehouses* (DW). Porém, os processos de ETL e integração de dados são habitualmente desenvolvidos para dados estruturados por modelos relacionais, que representam apenas uma pequena parte dos dados mantidos por muitas empresas. Dessa forma, existe uma demanda crescente para extrair, transformar e carregar dados estruturados por modelos de dados não relacionais e carregá-los em um repositório de dados unificado. Porém, devido a complexidade desses modelos de dados, Exitem vários desafios para a realização da extração, transformação e carga de dados organizados por modelos não relacionais que precisam considerar características específicas, como por exemplo, a heterogeneidade e distribuição dos dados, em um ambiente de extração, transformação e carga de dados.
Além disso, muitas empresas encontram dificuldades ao lidar com as ferramentas de ETL disponíveis no mercado, por causa muitas vezes da necessidade de integração destas ferramentas de ETL com sistemas legados. Aprender a lidar com essas ferramentas pode ser muito custoso em termos financeiro e de tempo, e por isso, muitas empresas acabam optando por desenvolver os seus processos utilizando uma linguagem de programação de propósito geral.
Portanto, neste trabalho propomos ETL4NoSQL, um *framework* programável, flexível e integrado para auxiliar a modelagem e execução de processos de ETL, que possibilita a extração, transformação e carga de dados estruturados em modelos de dados não relacionais. Apresentamos os componentes do *framework* ETL4NoSQL, bem como suas interfaces e funcionalidades. Ademais, realizamos um estudo experimental de *software*, que teve como objetivo verificar se o ETL4NoSQL é adequado para auxiliar no desenvolvimento de processos de ETL. O estudo consistiu na análise das ferramentas de ETL encontradas na literatura, com o propósito de caracterizá-las por meio da intersecção de suas funcionalidades no contexto comparativo entre elas. Finalmente, propomos um ambiente de implementação de ETL que permite o reuso e a extensão de interfaces de programação de ETL4NoSQL para desenvolver aplicações de ETL utilizando dois tipos diferentes de SGBDs NoSQL.

Palavras-chave: ETL, Frameworks, NoSQL, Data Warehouse, Estudo Experimental de Software

Abstract

Data integration and data extraction, transformation, and loading (ETL) processes are procedures for creating data warehouses (DW). However, ETL processes and data integration are usually developed for data structured by relational models, which represent only a small part of the data maintained by many companies. In this way, there is a growing demand to extract, transform, and load structured data from non relational data models and load them into a unified data repository. However, due to the complexity of the data models, there are several challenges for an extraction, transformation and loading of data organized by non relational models, requirements, specific characteristics, such as heterogeneity and distribution of data, in an environment of extraction, transformation and data loading. In addition, many companies are struggling to deal with ETL tools available in the market, often because of the need to integrate ETL tools with legacy systems. Learning how to handle these tools can be very costly in terms of time and financially, and so many companies end up opting to develop their processes using a general purpose programming language. Therefore, we propose ETL4NoSQL, a programmable, flexible framework and integration to help, model and execute ETL processes, which enables structured data extraction, transformation and loading in non relational data models. We present the components of the ETL4NoSQL framework, as well as its interfaces and functionalities. In addition, we conducted an experimental software study, which aimed to verify if ETL4NoSQL is suitable to assist in any development of ETL processes. The study consisted of the analysis of ETL tools found in the literature, with the purpose of characterizing them through the intersection of their functionalities in the comparative context between them. Finally, we propose an ETL implementation environment that allows the reuse and extension of ETL4NoSQL programming interfaces for ETL development applications, using two different types of NoSQL DBMSs.

Keywords: ETL, Frameworks, NoSQL, Data Warehouse, Experimental Software Study

Lista de Figuras

2.1 Inversão de controle em *framework*. (Adaptado de SOMMERVILLE (2013)) . 12
2.2 Exemplo da Ferramenta ARKTOS II em uso (Adaptado de VASSILIADIS et al. (2005)) 14
2.3 Fluxo de dados ETL no framework MapReduce (Adaptado de LIU; THOMSEN; PEDERSEN (2011)) 15
2.4 Arquitetura do CloudETL (Adaptado de LIU; THOMSEN; PEDERSEN (2013)) 16
2.5 Interface de configuração do P-ETL (Adaptado de BALA (2014)) 17

3.1 Modelo conceitual do ETL4NoSQL 23
3.2 Diagrama de Estado do ETL4NoSQL 25
3.3 Definição de interfaces do modelo de negócio do ETL4NoSQL 29
3.4 Especificação da arquitetura do componente de ETL4NoSQL 29
3.5 Diagrama de colaboração para conectar à base de fonte de dados 30
3.6 Diagrama de colaboração para verificar a estrutura de dados 30
3.7 Diagrama de colaboração para verificar se existe permissão de escrita na base de dados destino 30
3.8 Diagrama de colaboração para leitura dos dados da fonte 30
3.9 Diagrama de colaboração criação das operações de ETL 31
3.10 Diagrama de colaboração modelar as operações criadas 31
3.11 Diagrama de colaboração para gerenciar as operações 31
3.12 Diagrama de colaboração para processar as operações 31
3.13 Diagrama de especificação das interfaces de ETL4NoSQL 33
3.14 Tela do IDE LiClipse com a implementação da interface ETL4NoSQLMgr . . . 35
3.15 Tela do IDE LiClipse com a implementação da interface IDataMgr 36
3.16 Tela do IDE LiClipse com a implementação da interface IModelMgr 36
3.17 Tela do IDE LiClipse com a implementação da interface IOpMgr 37
3.18 Tela do IDE LiClipse com a implementação da interface IProcMgr 37

4.1 Questionário de Funcionalidades 47
4.2 Quantidade de Presença para cada funcionalidade 48
4.3 Níveis de utilidade para cada funcionalidade 48
4.4 Necessidade de melhoria para cada funcionalidade 49

5.1 Modelo Multidimensional da aplicação ETL4NoSQLMongoStar 58
5.2 Tela da aplicação ETL4NoSQLMongoStar 58
5.3 Modelo Multidimensional da aplicação ETL4NoSQLCassandraStar 59
5.4 Tela da aplicação ETL4NoSQLCassandraStar 60

Lista de Quadros

3.1 Requisitos do ETL4NoSQL 22
3.2 Modelo de Casos de Uso do ETL4NoSQL 24

4.1 Descrição da Instrumentação 42
4.2 Métricas 43
4.3 Questionário do Perfil da Ferramenta de ETL 46
4.4 Instrumentação para aplicar o questionário 46
4.5 Resultado do Perfil dos participantes 49
4.6 Legenda 49
4.7 Estatística Descritiva da Presença de Funcionalidades 50
4.8 Estatística Descritiva da Melhoria de Funcionalidades 50
4.9 Estatística Descritiva da Utilidade de Funcionalidades 50
4.10 Funcionalidade presente e parcialmente útil 51
4.11 Funcionalidade presente e é útil 51
4.12 Funcionalidade presente e necessita melhorar 51
4.13 Funcionalidade presente e não necessita melhoria 52
4.14 Quadro de resultado do valor observado de existência da funcionalidade 52
4.15 Quadro do resultado de χ^2 53
4.16 Funcionalidades não oferecidas pelo ETL4NoSQL 54

Lista de Acrônimos

APD	Área de Processamento de Dados
BDs NoSQL	Banco de Dados NoSQL
BI	Business Intelligence
DW	Data Warehouse
ETL	Extract, Transform and Load
JSON	JavaScript Object Notation
RDBMS	Relational Database Management System
SGBD	Sistema Gerenciador de Bancos de Dados
SQL	Structured Query Language
OCL	Object Constraint Language
UML	Unified Modeling Language
XML	eXtensible Markup Language
YAML	Yet Another Markup Language

Sumário

1 Introdução **1**
1.1 Contextualização . . . 2
1.2 Motivação . . . 3
1.3 Objetivos . . . 4
1.3.1 Objetivos Específicos . . . 5
1.4 Contribuições . . . 5
1.5 Organização do Trabalho . . . 5

2 Fundamentação Teórica **7**
2.1 Conceitos Básicos . . . 8
2.1.1 ETL . . . 8
2.1.2 Bancos de Dados NoSQL . . . 9
2.1.2.1 Banco de dados Orientados à Documentos . . . 9
2.1.2.2 Banco de dados Famílias de Colunas . . . 9
2.1.2.3 Banco de dados Baseado em Grafos . . . 10
2.1.2.4 Banco de dados Chave-Valor . . . 10
2.1.3 Desenvolvimento Baseado em Componentes e Frameworks . . . 10
2.1.4 Estudo Experimental de Software . . . 13
2.2 Trabalhos Correlatos . . . 13
2.2.1 ARKTOS II . . . 14
2.2.2 PygramETL . . . 14
2.2.3 ETLMR . . . 14
2.2.4 CloudETL . . . 15
2.2.5 P-ETL . . . 16
2.2.6 Big-ETL . . . 17
2.2.7 FramETL . . . 18
2.2.8 Outras Ferramentas . . . 18
2.2.8.1 Pentaho . . . 18
2.2.8.2 Talend Studio . . . 18
2.2.8.3 CloverETL . . . 19
2.2.8.4 Oracle Data Integrator (ODI) . . . 19
2.3 Considerações Finais . . . 19

3 O Framework ETL4NoSQL **20**
3.1 Requisitos de Software do ETL4NoSQL . . . 21
3.2 Modelagem do Domínio de ETL4NoSQL . . . 22

3.2.1 Modelo Conceitual 22
3.2.2 Modelo de Casos de Uso 23
3.2.3 Modelo Comportamental 23
3.3 Modelagem da Especificação do ETL4NoSQL 24
3.3.1 Identificação de Componentes 24
3.3.2 Interfaces de Sistemas 25
3.3.3 Interfaces de Negócio 28
3.3.4 Especificação da Arquitetura do Componente 28
3.4 Interação entre Componentes 29
3.4.1 Operações da interface de negócio 30
3.5 Especificação de Componentes 31
3.6 Ambiente de Implementação 34
3.7 Interfaces de Programação 34
3.8 Considerações Finais 37

4 Estudo Experimental de Software **39**
4.1 Objetivos do experimento 40
4.1.1 Objetivo da Medição 40
4.1.2 Objetivos do Estudo 40
4.1.3 Questões 40
4.2 Planejamento 41
4.2.1 Definição das Hipóteses 41
4.2.2 Descrição da instrumentação 42
4.2.3 Métricas 43
4.2.4 Seleção do contexto 43
4.2.5 Seleção dos indivíduos 44
4.2.6 Variáveis 44
4.2.7 Análise Qualitativa 44
4.2.8 Validade 45
4.3 Operação 45
4.3.1 Questionário do Perfil da Ferramenta de ETL 45
4.3.2 Questionário de Funcionalidades 46
4.3.3 Resultado do Estudo 48
4.4 Análise e interpretação dos resultados 49
4.4.1 Estatística descritiva 50
4.4.2 Aplicação do teste estatístico 50
4.4.2.1 Análise quantitativa 53
4.4.3 Análise qualitativa 54
4.4.4 Verificação das hipóteses 54

4.5 Considerações Finais 55

5 Avaliação **56**
5.1 Aplicações Baseadas no ETL4NoSQL 57
5.2 Aplicação ETL4NoSQLMongoStar 57
5.3 Aplicação ETL4NoSQLCassandraStar 59
5.4 Considerações Finais 60

6 Conclusão **61**
6.1 Principais Contribuições 62
6.2 Discussão 62
6.3 Trabalhos Futuros 63

Referências **64**

1

Introdução

Este capítulo contextualiza os principais assuntos abordados nesta dissertação, apresenta as motivações que levaram à escolha do tema, os objetivos gerais e específicos desta pesquisa, bem como a justificativa da investigação conduzida e suas principais contribuições.

1.1 Contextualização

Desde a década de 1970, com a criação do modelo relacional por Edgar Frank Codd, a estrutura de armazenamento adotada por muitos desenvolvedores de sistemas da área de tecnologia da informação têm se baseado nesse conceito. A maioria dos sistemas gerenciadores de banco de dados (SGBD) que possui aceitação no mercado fazem uso desse modelo, por exemplo o MySQL, Oracle e Microsoft SQL Server. Porém, os requisitos de dados para o desenvolvimento de ferramentas de *software* atual têm mudado significativamente, especialmente com o aumento das aplicações *Web* (NASHOLM (2012)). Este segmento de aplicações exige alta escalabilidade e vazão, e SGBD relacionais muitas vezes não conseguem atender satisfatoriamente os seus requisitos de dados. Como alternativa a isso, novas abordagens de SGBD utilizando o termo de SGBD NoSQL tornaram-se popular (SILVA (2016)).

O termo NoSQL é constantemente interpretado como "*Not Only SQL*" (não somente SQL), cujo SQL refere-se a linguagem disponibilizada pelos SGBD relacionais (NASHOLM (2012)). O propósito das abordagens NoSQL é oferecer alternativas onde os SGBD relacionais não apresentam um bom desempenho. Esse termo abrange diferentes tipos de sistemas. Em geral, SGBD NoSQL usam modelo de dados não-relacionais, com poucas definições de esquema, e geralmente, são executados em clusters. Alguns exemplos de SGBD NoSQL recentes são o Cassandra, MongoDB, Neo4J e o Riak (FOWLER; SADALAGE (2013)).

Muitas empresas coletam e armazenam milhares de gigabytes de dados por dia, onde a análise desses dados representa uma vantagem competitiva no mercado. Por isso, há uma grande necessidade de novas arquiteturas para o suporte à decisão (LIU; THOMSEN; PEDERSEN (2013)). Para isso, uma das formas bastante utilizada é a criação de um ambiente *data warehousing* responsável por providenciar informações estratégicas e esquematizadas a respeito do negócio (CHAUDHURI; DAYAL (1997)).

Segundo a definição de KIMBALL; ROSS (2002), *data warehouse* (DW) é uma coleção de dados voltada para o processo de suporte à decisão, orientada por assunto, integrada, variante no tempo e não volátil. Os dados de diferentes fontes de dados são processados e integrados em um *data warehouse* central através da Extração, Transformação e Carga (ETL) que é feita de maneira periódica. Os processos de ETL consistem em um conjunto de técnicas e ferramentas para transformar dados de múltiplas fontes de dados para fins de análise de negócio (SILVA (2016)). Ferramentas de ETL são sistemas de *software* responsáveis por extrair dados de diversas fontes de dados, transformar, customizar e inseri-los no *data warehouse*.

O projeto de ETL, ou seja, a criação dos seus processos, consome cerca de 70% dos recursos de implantação de um DW, pois desenvolver esse projeto é crítico e custoso, tendo em vista que gerar dados incorretos pode acarretar em más decisões. Porém, por algum tempo pouca importância foi dada ao processo de ETL pelo fato de ser visto somente como uma atividade de suporte aos projetos de DW. Apenas a partir do ano 2000, a comunidade acadêmica passou a dar mais importância ao tema (SILVA (2012)). Atualmente, ainda existem dificuldades ao lidar com

as soluções para ferramentas de ETL presentes na literatura. É comum que elas demonstrem mais importância aos SGBD relacionais. Pois, tradicionalmente o DW é implementado em um banco de dados relacional, onde o dado é armazenado nas tabelas fato e dimensões, que são descritas por um esquema em estrela (KIMBALL; ROSS (2002)). Por isso, para oferecer suporte aos sistemas que necessitem realizar os processos de ETL em banco de dados NoSQL, a proposta desse trabalho é especificar um *framework* programável, flexível e integrado para modelagem e execução de processos de ETL em banco de dados NoSQL.

1.2 Motivação

A integração de dados e os processos de ETL são procedimentos cruciais para a criação de *data warehouses*. Porém, esses procedimentos são tradicionalmente desenvolvidos para dados em modelos relacionais, que representam apenas uma pequena parte dos dados mantidos por muitas empresas (DARMONT et al. (2005), RUSSOM; MADSEN (2007), THOMSEN; PEDERSEN (2009)).

O uso generalizado da internet, web 2.0, redes sociais e sensores digitais produzem grandes volumes de dados. De fato, processos MapReduce (MR) são executados continuamente para tratar mais de vinte Petabytes de dados por dia (DEAN; GHEMAWAT (2008)). Esta explosão de dados é uma oportunidade para o surgimento de novas aplicações, como *Big Data Analytics* (BDA); mas é, ao mesmo tempo, um problema dado as capacidades limitadas das máquinas e das aplicações tradicionais.

Esse grande volume de dados é conhecido como "*Big Data*"e caracterizado pelos quatro "V": Volume, que implica a quantidade de dados que vão além das unidades usuais; a Velocidade com que esses dados são gerados e devem ser processados, a Variedade como diversidade de formatos e estruturas, e a Veracidade relacionada à precisão e confiabilidade dos dados. Além disso, novos paradigmas emergem, tais como *Cloud Computing* e *MapReduce* (MR) e novos modelos de dados são propostos para armazenamento de grandes volumes de dados, como o NoSQL (*Not Only SQL*) (DEAN; HAIHONG; DU (2011)).

Dessa forma, existe uma demanda crescente para extrair, transformar e carregar grande volume de dados, apresentados de forma variada, em modelos não relacionais em um ambiente de suporte à decisão. Contudo, devido a complexidade desses dados, novos desafios estão surgindo quando lidamos com suas características, como por exemplo a heterogeneidade e distribuição desses dados, no ambiente de extração, transformação e carga de dados (SALEM; BOUSSAïD; DARMONT (2012)).

Além disso, muitas empresas encontram dificuldades para lidar com as ferramentas de ETL disponíveis no mercado. Aprender a lidar com essas ferramentas pode ser muito custoso em termos financeiros e de tempo, e por isso, acabam optando desenvolver os seus processos por meio de uma linguagem de programação de propósito geral (AWAD; ABDULLAH; ALI (2011), MUÑOZ; MAZÓN; TRUJILLO (2009)).

As pesquisas presentes na literatura sobre extração de dados em BDs NoSQL mostram que as ferramentas existentes no mercado propõem arquiteturas, metamodelos, aplicações e metodologias de modelagem para processos de ETL ((SILVA (2016), CHEVALIER et al. (2015), LIU; THOMSEN; PEDERSEN (2013)).No entanto, elas não apresentam um framework programável em um ambiente integrado, e ainda ignoram conceitos importantes sobre frameworks, tais como reuso, flexibilidade e outros aspectos como a variedade dos dados e a falta de uma linguagem comum, como o SQL.

Em geral, não é incomum observar que os especialistas em ETL utilizam interfaces textuais, enquanto que os não especialistas optem pelo uso de GUI. No entanto, mesmo que haja o envolvimento de gestores e outros membros não técnicos, a implementação final dos processos é realizada por especialistas, os quais são mais eficientes quando utilizam interfaces de programação textuais (SILVA (2012), MAZANEC; MACEK (2012)).

Portanto, o aumento do uso de SGBD com modelos de dados não relacionais baseados no paradigma NoSQL e a falta de uma ferramenta programável, flexível e integrada, independente de plataforma que dê suporte à extração, transformação e carga para esses SGBD é a grande motivação deste trabalho.

1.3 Objetivos

O objetivo principal desta pesquisa é especificar um *framework* programável, flexível e integrado para modelagem e execução de processos de ETL de bancos de dados NoSQL. Baseamos nossa proposta nos princípios de flexibilidade, extensibilidade, reuso e inversão de controle, conforme recomenda a literatura sobre frameworks (AWAD; ABDULLAH; ALI (2011), VASSILIADIS et al. (2005), FAYAD; SCHMIDT; E. (1999), FAYAD; SCHMIDT (1997), DARMONT et al. (2005)), além dos conceitos de desenvolvimento baseados em componentes apresentados na seção 2.1.3.

A arquitetura do ETL4NoSQL oferece uma interface de programação que contém elementos, tais como componentes de gerenciamento, leitura e escrita de dados, criação e execução de operações de ETL. O componente de gerenciamento é o responsável pelos fundamentos de ETL disponíveis na literatura (KIMBALL; CASERTA (2004)). Ele possibilita as aplicações baseadas no ETL4NoSQL reutilizar os componentes para modelar seus processos de ETL. Além disso, a flexibilidade do framework proposto permite que sejam criados outros componentes que encapsulam regras de transformação para áreas especificas de ETL. Um componente especializado permite a construção de processos de ETL que não seriam possíveis ou que exigiriam o uso conjunto de muitos componentes genéricos do framework. Para possibilitar a criação de uma interface integrada para especialização e modelagem de processos de ETL, utilizamos o desenvolvimento baseado em componentes. Isto propiciou a implementação do ETL4NoSQL em uma linguagem de programação de propósito geral. Os objetivos específicos são detalhados a seguir.

1.3.1 Objetivos Específicos

Um dos objetivos específicos desta dissertação é apresentar a especificação e modelagem dos componentes do framework ETL4NoSQL, bem como suas interfaces e funcionalidades. Outro objetivo deste trabalho consiste em realizar um estudo experimental de *software*, a fim de caracterizar as principais funcionalidades das ferramentas de ETL para BDs NoSQL. O estudo experimental tem como objetivo comparar o framework proposto neste trabalho e demonstrar suas vantagens e desvantagens em relação às ferramentas de ETL encontradas na literatura. Por fim, o nosso último objetivo é prover dois ambientes de ETL para facilitar a extração, transformação e carga de dados em DW modelados pelo esquema estrela, tendo em vista que este é o esquema de dados dimensional mais recomendado pela literatura (INMON (2002), KIMBALL; ROSS (2002)). No primeiro utilizamos o SGBD MongoDB com dados sintéticos de ranking de restaurantes, e o segundo ambiente, o SGBD escolhido foi o CassandraDB com dados sintéticos de localizações de táxis.

1.4 Contribuições

Uma das contribuições deste trabalho é o framework ETL4NoSQL. Ele permite extrair, transformar e carregar dados que estão armazenados em diversos SGBD NoSQL, ou até mesmo, repositórios de dados textuais e SGBD relacionais, em um único repositório. A vantagem ao utilizar o ETL4NoSQL é sua natureza programável, flexível e integrada que facilita a modelagem e execução dos processos de ETL em banco de dados NoSQL.

Outra contribuição desta pesquisa é apresentar, por meio de um estudo experimental *software* as principais funcionalidades de uma ferramenta de ETL, bem como possíveis melhorias, vantagens e desvantagens de acordo com os trabalhos correlatos encontrados na literatura.

Por fim, nossa última contribuição é oferecer duas aplicações de ETL, criadas a partir do ETL4NoSQL, utilizando domínios distintos baseados em dois sistemas NoSQL.

1.5 Organização do Trabalho

Este trabalho está organizado de acordo com a seguinte estrutura:

- **Fundamentação Teórica:** apresenta uma revisão de literatura sobre os principais assuntos abordados neste trabalho. São tratados temas a respeito de ETL, banco de dados NoSQL, frameworks, estudo experimental de *software* e descreve os trabalhos correlatos encontrados na literatura a respeito de ferramentas de ETL.
- **O Framework ETL4NoSQL:** descreve os requisitos, a arquitetura de *software* e implementação dos componentes do framework proposto.

- **Estudo Experimental de Software:** expõe o roteiro da experimentação de *software* para ferramentas de ETL. Define o objetivo, o planejamento, a operação e o resultado do estudo.
- **Avaliação:** descreve aplicações de ETL4NoSQL para dois domínios de naturezas distintas, a fim de ilustrar a reusabilidade e flexibilidade do ETL4NoSQL, avaliando a proposta desta dissertação.
- **Considerações Finais:** expressa as limitações e ameaças à validade do trabalho, considerações finais e sugere de trabalhos futuros.

2

Fundamentação Teórica

Neste capítulo, são apresentados os conceitos relacionados ao desenvolvimento desta pesquisa, bem como o embasamento teórico necessário para o entendimento do trabalho. Os assuntos abordados são: ETL, Banco de Dados NoSQL, *Frameworks*, Estudo Experimental de *Software* e trabalhos correlatos ao tema deste estudo.

2.1 Conceitos Básicos

Conceitos de ETL, banco de dados NoSQL, desenvolvimento baseado em componentes, *frameworks* e estudo experimental de *software* são essenciais para esta pesquisa. Dessa forma, os princípios básicos desses conceitos são apresentados nesta seção. Ainda que, estudo experimental de *software* não seja o tema principal deste trabalho, seus conceitos são essenciais para avaliar e medir nossa proposta sendo fundamental entendê-los.

2.1.1 ETL

ETL (Extração, Limpeza/Transformação e Carga - ETL) é conhecido na literatura por definir processos que permitem a extração e transformação de dados, centralizando-os numa base destino facilitando o gerenciamento e análise desses dados (KIMBALL; CASERTA (2004), RUD (2009)). O fluxo do processo de ETL inicia-se com extração dos dados a partir de uma fonte de dados, que podem ser arquivos textuais, banco de dados relacionais, banco de dados NoSQL, entre outros. Os dados são propagados para uma Área de Processamento de Dados (APD) onde são executadas a limpeza e transformação por meio de mecanismos de ETL definidos como agregação, junção, filtro, união, etc. Finalmente, os dados são carregados em estruturas que podem ser *data warehouses* ou repositórios analíticos (SILVA (2016), SILVA (2012), KIMBALL; CASERTA (2004)).

KIMBALL; CASERTA (2004) definem os processos de ETL em 4 macroprocessos, com 34 subsistemas que são listados a seguir.

a) Extração: Busca os dados dos sistemas de origem e grava na área de processamento de dados antes de qualquer alteração significativa. Esta etapa possui 3 subsistemas: *Data Profilling, Change Data Capture* e Sistema de Extração.

b) Limpeza e Transformação: Envia os dados de origem, por meio de várias etapas de processamento de ETL; melhora a qualidade dos dados recebidos da fonte de dados; mescla os dados de duas ou mais fontes de dados; cria dimensões; e aplica métricas. Esta etapa possui 5 subsistemas: *Data Cleasing System, Error Event Tracking, Deduplication, Data Conformance* e Criação de Dimensão de auditoria .

c) Entrega ou Carga: Estrutura fisicamente e carrega os dados conforme desejado em DWs ou repositórios analíticos. Esta etapa possui 13 subsistemas: *Slowly Changing Dimension (SCD) Manager, Surrogate Key Generator, Hierarchy Manager, Special Dimensions Manager, Fact Table Builders, Surrogate Key Pipeline, Multi-Valued Bridge Table Builder, Late Arriving Data Handler, Dimension Manager, Fact Table Provider, Aggregate Builder, OLAP Cube Builder, Data Propagation Manager*.

d) Gerenciamento: Gerencia os sistemas e processos relacionados ao ambiente ETL de forma coerente. Esta etapa possui 13 subsistemas: *Job Scheduler, Backup System,*

Recovery and Restart, Version Control, Version Migration, Workflow Monitor, Sorting, Lineage and Dependency, Problem Escalation, Paralleling and Pipelining, Security, Compliance Manager, Metadata Repository.

2.1.2 Sistemas de Bancos de Dados NoSQL

Sistemas de BD NoSQL consistem em sistemas projetados para armazenar grandes volumes de dados em modelos não relacionais disponibilizando estruturas e interfaces com acesso simplificado (LIMA; MELLO (2015)). Cada sistema de BD NoSQL possui um modelo de dados próprio, nos quais os modelos de dados mais conhecidos são divididos em quatro categorias: Chave-Valor, Orientado a Documentos, Famílias de Colunas e Baseado em Grafos (FOWLER; SADALAGE (2013), KAUR (2013)).

As principais características dos sistemas de banco de dados NoSQL são: distribuição, escalabilidade horizontal, gerenciamento de grande volume de dados, satisfaz propriedades do tipo BASE (Basicamente disponível, Estado leve, Eventualmente consistente) ao invés de ACID (Atomicidade, Consistência, Isolamento e Durabilidade), modelo de dados não relacional e não contempla SQL (FOWLER; SADALAGE (2013), NASHOLM (2012)).

2.1.2.1 Banco de dados Orientados à Documentos

Banco de dados orientados à documentos são capazes de armazenar documentos como dado. Estes documentos podem ser em qualquer formato como XML (eXtensible Markup Language), YAML (Yet Another Markup Language), JSON (JavaScript Object Notation), entre outros. Os documentos são agrupados na forma de coleções. Comparando com banco de dados relacional, as coleções são como tabelas e os documentos como os registros. Porém, a diferença entre eles é que cada registro na tabela do banco relacional tem o mesmo número de campos, enquanto que na coleção do banco de dados orientado à documentos podem ter campos completamente diferentes (KAUR (2013), FOWLER; SADALAGE (2013)).

Existem vários sistemas gerenciadores de banco de dados orientados à documentos disponíveis e os mais utilizados são MongoDB, CouchDB e o RavenDB (KAUR (2013)).

2.1.2.2 Banco de dados Famílias de Colunas

Banco de dados baseados em Famílias de Colunas são desenvolvidos para abranger três áreas: número enorme de colunas, a natureza esparsa dos dados e frequentes mudanças no esquema de dados. Os dados em famílias de colunas são armazenados em colunas de forma contínua, enquanto que em bancos de dados relacionais as linhas é que são contínuas. Essa mudança faz com que operações como agregação, suporte para *ad-hoc* e consultas dinâmicas se tornem mais eficientes (KAUR (2013), FOWLER; SADALAGE (2013)).

A maioria dos bancos de dados baseados em famílias de colunas são também compatíveis com o *framework* MapReduce, no qual acelera o processamento de enormes volumes de dados

pela distribuição do problema em um grande número de sistemas. Os SGBDs de família de colunas *open-source* mais populares são Hypertable, HBase e Cassandra (KAUR (2013)).

2.1.2.3 Banco de dados Baseado em Grafos

Bancos de dados baseado em grafos são como uma estrutura de rede contendo nós e arestas, onde as arestas interligam os nós representando a relação entre eles. Comparando com os banco de dados relacionais, o nó corresponde à tabela, a propriedade do nó à um atributo e as arestas são a relação entre os nós. Nos bancos de dados relacionais as consultas requerem atributos de mais de uma tabela resultando numa operação de junção, por outro lado, bancos de dados baseado em grafos são desenvolvidos para encontrar relações dentro de uma enorme quantidade de dados rapidamente, tendo em vista que não é preciso fazer junções, ao invés disso, ele fornece indexação livre de adjacência. Um exemplo de SGBD baseado em grafos é o Neo4j (KAUR (2013)).

2.1.2.4 Banco de dados Chave-Valor

Em bancos de dados Chave-Valor, os dados são organizados como uma associação de vetores de entrada consistindo em pares de chave-valor. Cada chave é única e usada para recuperar os valores associados a ele. Esses bancos de dados podem ser visualizados como um banco de dados relacional contendo múltiplas linhas e apenas duas colunas: chave e valor. Buscas baseadas em chaves resultam num baixo tempo de execução. Além disso, os valores podem ser qualquer coisa como objetos, hashes, entre outros (KAUR (2013)).

Os SGBDs Chave-Valor mais populares são Riak, Voldemort e Redis (KAUR (2013)).

2.1.3 Desenvolvimento Baseado em Componentes e Frameworks

A engenharia de *software* baseada em componentes é uma abordagem fundamentada em reuso para desenvolvimento de sistemas de *software*, ela envolve o processo de definição, implementação e integração ou composição de componentes independentes, não firmemente acoplados ao sistema. Os componentes são independentes, ou seja, não interferem na operação uns dos outros e se comunicam por meio de interfaces bem definidas, os detalhes de implementação são ocultados, de forma que as alterações de implementação não afetam o restante do sistema (SOMMERVILLE (2013)). Segundo SAMETINGER (1997), componentes são uma parte do sistema de *software* que podem ser identificados e reutilizados, onde descrevem ou executam funções específicas e possuem interfaces claras, documentação apropriada e a possibilidade de reuso bem definida. Ainda de acordo com o autor, um componente deve ser auto contido, identificável, funcional, possuir uma interface, ser documentado e ter uma condição de reuso.

Para CHEESMAN; DANIELS (2001), o processo de desenvolvimento baseado em componentes baseia-se na separação entre modelagem de domínio e modelagem de especificação.

A modelagem do domínio consiste no entendimento do contexto de um negócio ou situação, o seu propósito é compreender os conceitos do domínio, seus relacionamentos e suas tarefas. Os resultados da modelagem de domínio são os modelos de casos de uso, o modelo conceitual e o modelo de comportamento (SOUZA GIMENES; HUZITA (2005)).

Por outro lado, a modelagem da especificação de *software* é dividida em três etapas: a etapa de identificação dos componentes, onde produz uma especificação e arquitetura inicial; interação entre componentes, nesta etapa descobre-se as operações necessárias e aloca responsabilidades; e finalmente, a etapa de especificação de componentes, que cria uma especificação precisa das operações, interfaces e componentes.

O objetivo da modelagem de especificação é definir, em alto nível de abstração, os serviços oferecidos pelos componentes vistos como caixas pretas. É nela que a arquitetura é definida e os componentes especificados (SOUZA GIMENES; HUZITA (2005)).

Frameworks podem ser considerados aglomerados de *softwares*, onde estes são capazes de serem estendidos e adaptados para utilidades específicas (TALIGENT (1994)). PREE; SIKORA (1997), consideram que *frameworks* são aplicações semi-completas e que podem ser reutilizadas para especializar produtos de *software* customizados. SOMMERVILLE (2013), ressalta que *framework* é uma estrutura genérica estendida com o intuito de criar uma aplicação mais específica e SCHIMIDT; GOKHALE; NATARAJAN (2004) define como sendo um conjunto de artefatos de *software* (como classes, objetos e componentes) que colaboram para fornecer uma arquitetura reusável.

Os *frameworks* possibilitam a reusabilidade de projeto, bem como ao reúso de classes específicas, pois fornecem uma arquitetura de esqueleto para a aplicação, que é definida por classes de objetos e suas interações. As classes são reusadas diretamente e podem ser estendidas usando-se recursos, como a herança (SOMMERVILLE (2013)).

FAYAD; SCHMIDT (1997), separam os *frameworks* em três principais classes: de infraestrutura de sistema, de integração de *middleware* e de aplicações corporativas. *Frameworks* de infraestrutura de sistema apoiam o desenvolvimento de infraestruturas, como comunicações, interfaces de usuários e compiladores. Já os *frameworks* de integração de *middleware* são um conjunto de normas e classes de objetos associados que suportam componentes de comunicação e troca de informações. E finalmente, os *frameworks* de aplicações corporativas estão relacionados com domínios de aplicação específicos, como sistemas financeiros. Eles incorporam conhecimentos sobre o domínios de aplicações e apoiam o desenvolvimento para o usuário final.

Muitas vezes, os *frameworks* são implementações de padrões de projeto, como por exemplo o *framework* MVC (Model-View-Control). A natureza geral dos padrões e o uso de classes abstratas e concretas permitem a extensibilidade (SOMMERVILLE (2013)).

Para estender um *framework* não é necessário alterar o seu código, apenas é preciso adicionar classes concretas que herdam operações de classes abstratas. Ademais, há a possibilidade de definir *callbacks*, que são métodos chamados em resposta a eventos reconhecidos pelo *framework*.Esses métodos são reconhecido como 'inversão de controle' (SCHIMIDT; GOKHALE;

NATARAJAN (2004)). A figura 2.1 expressa o funcionalidade da inversão de controle. Os responsáveis pelo controle no sistema são os objetos do *framework*, ao invés de serem objetos específicos de aplicação. E em resposta aos eventos de interface do usuário, banco de dados, entre outros, esses objetos do *framework* invocam 'métodos *hook*' que, em seguida, são vinculados à funcionalidade fornecida ao usuário. A funcionalidade específica de aplicação responde ao evento de forma adequada. Por exemplo, um *framework* terá um método que lida com um toque em uma tecla a partir do ambiente. Esse método chama o método *hook*, que deve ser configurado para chamar os métodos de aplicação adequada para tratar o toque na tecla (SOMMERVILLE (2013)).

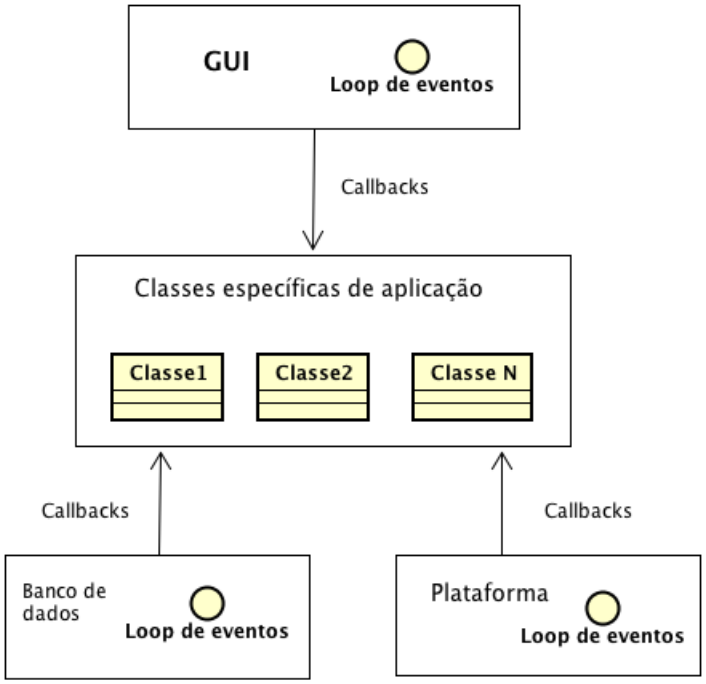

Figura 2.1 Inversão de controle em *framework*. (Adaptado de SOMMERVILLE (2013))

Para especificar o ETL4NoSQL utilizamos a metodologia de desenvolvimento baseada em componentes, pois esta metodologia é fundamentada no reuso e na integração de componentes independentes, cujo encaixa com a necessidade do ETL4NoSQL ser integrado apesar de muitos processos do fluxo de ETL ter funcionalidades independentes. É importante ressaltar também que o desenvolvimento baseado em componentes importa-se com o reuso que é uma característica fundamental para oferecer a flexibilidade necessária ao ETL4NoSQL.

No que diz respeito a *frameworks*, o ETL4NoSQL encaixa-se na categoria de aplicações corporativas, pois serve como base para aplicações de ETL, incorporando conhecimentos sobre a área de domínio para apoiar o desenvolvimento ao usuário final.

2.1.4 Estudo Experimental de Software

Esta dissertação considera a execução do estudo experimental de *software* para caracterizar, avaliar e propor melhorias ao *framework* ETL4NoSQL. O objetivo principal da aplicação do experimento é definir se o *framework* proposto é uma ferramenta adequada para auxiliar no desenvolvimento de processos de ETL em BDs NoSQL. Os participantes escolhidos foram as principais ferramentas de ETL encontradas na literatura. Os questionários utilizados para a coleta de dados são baseadas nos requisitos mínimos considerados pela literatura para ferramentas de ETL.

Segundo TRAVASSOS; GUROV; AMARAL (2002), a experimentação é o centro do processo científico, por meio dos experimentos que é possível verificar teorias, explorar fatores críticos e formular novas teorias. O autor reforça ainda a necessidade de avaliar novas invenções e sugestões em comparação com as existentes.

Para WOHLIN et al. (2000), existem quatro métodos relevantes para experimentação em Engenharia de *Software*: científico, de engenharia, experimental e analítico.

O paradigma indutivo, ou método científico, observa o mundo, pode ser utilizado quando se quer entender o processo, produto de *software* e ambiente. Ele mede e analisa, verifica as hipóteses do modelo ou teoria. Já o método de engenharia observa as soluções existentes, é uma abordagem baseada na melhoria evolutiva, modifica modelos de processos ou produtos de *softwares* existentes com propósito de melhorar os objetos de estudo. O método experimental é uma abordagem baseada na melhoria revolucionária. Ela sugere um modelo, não necessariamente baseado em um existente, aplica o método qualitativo e/ou quantitativo, faz a experimentação, analisa e repete o processo. Por fim, o método analítico sugere uma teoria formal, um método dedutivo que oferece uma base analítica para o desenvolvimento de modelos (TRAVASSOS; GUROV; AMARAL (2002)).

TRAVASSOS; GUROV; AMARAL (2002) sugere que a abordagem mais apropriada para a experimentação na área de Engenharia de *Software* seja o método experimental, pois considera a proposição e avaliação do modelo com os estudos experimentais.

Os principais objetivos relacionados à execução de um estudo experimental de *software* são: caracterização, avaliação, previsão, controle e melhoria a respeito de produtos, processos, recursos, modelos e teorias. Os elementos principais do experimento são: as variáveis, objetos, participantes, o contexto do experimento, hipóteses e o tipo de projeto do experimento.

2.2 Trabalhos Correlatos

Esta seção apresenta os principais *frameworks* correlatos a este trabalho encontrados na literatura, bem como os descreve demonstrando suas características, seus pontos positivos e negativos.

2.2.1 ARKTOS II

O principal objetivo do ARKTOS II é facilitar a modelagem dos processos de ETL, de forma que o usuário define a fonte dos dados e o destino, os participantes e o fluxo de dados do processo. Como ilustrado na figura 2.2, o usuário pode desenhar atributos e parâmetros, conectá-los ao seu esquema de dados, criar relacionamentos e desenhar arestas de um nó para outro de acordo com a arquitetura do grafo (VASSILIADIS et al. (2005)).

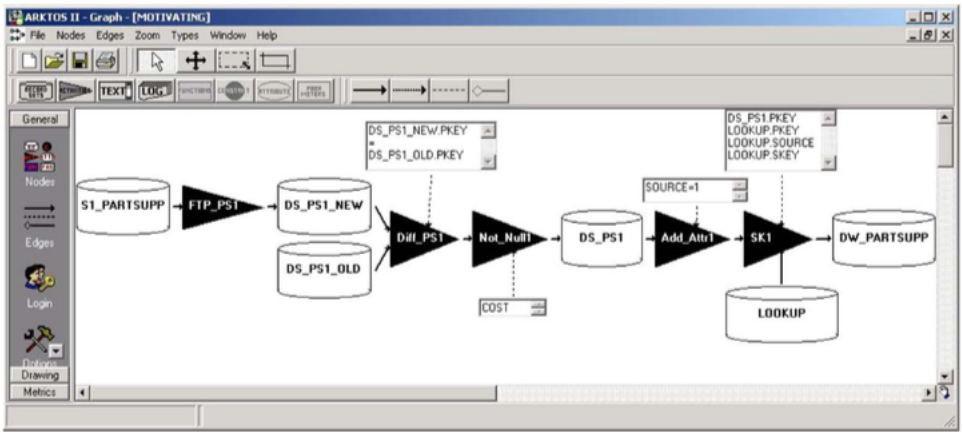

Figura 2.2 Exemplo da Ferramenta ARKTOS II em uso (Adaptado de VASSILIADIS et al. (2005))

A customização no ARKTOS II é oferecida pela reusabilidade de seus *templates*. Os processos são armazenados em um repositório implementado em um banco de dados relacional. Os autores do ARKTOS II ainda pretendem melhorar a ferramenta permitindo mais formatos de dados como XML.

2.2.2 PygramETL

PygramETL é um *framework* programável para desenvolvedores de ETL. Ele oferece a funcionalidade para desenvolver ETL demonstrando como deve-se iniciar um projeto. O propósito da ferramenta é facilitar a carga dos dados no DW gerenciado por banco de dados relacionais (SGBDs). Focando nos SGBDs relacionais como destino torna o desenvolvimento simples, não considerando outros tipos de estrutura de dados e a integração deles. Dessa forma, o PygramETL oferece suporte apenas para SGBDs relacionais (THOMSEN; PEDERSEN (2009)).

2.2.3 ETLMR

ETLMR é um framework de ETL que utiliza *MapReduce* para atingir escalabilidade. Ele suporta esquemas de DW como o esquema estrela, o *snowflake*, e o *slowly changing dimensions* (LIU; THOMSEN; PEDERSEN (2011)).

A figura 2.3 ilustra o fluxo de dados usando o ETLMR e MapReduce. O processamento da dimensão é feito em uma tarefa do MapReduce, e o processamento do fato é feito por outra

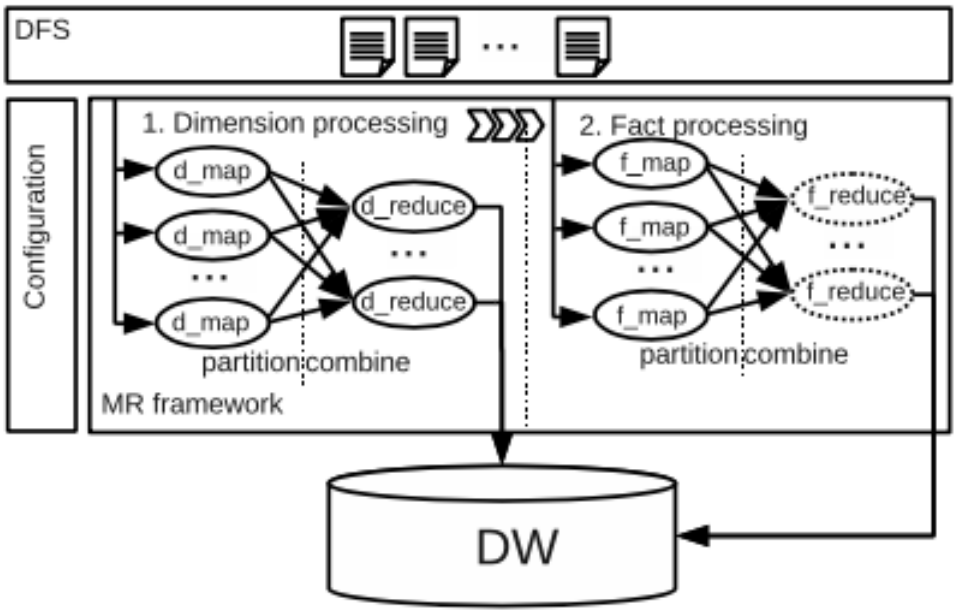

Figura 2.3 Fluxo de dados ETL no framework MapReduce (Adaptado de LIU; THOMSEN; PEDERSEN (2011))

tarefa MapReduce. A tarefa MapReduce gera um número de tarefas map/reduce paralelas para processar a dimensão ou o fato. Cada tarefa consiste em inúmeros passos, incluindo a leitura dos dados no sistema de arquivos distribuído (DFS - distributed file system), execução da função de mapeamento, particionamento, combinação do mapeamento de saída, execução da função reduce e escrita dos resultados (LIU; THOMSEN; PEDERSEN (2011)).

O ETLMR possui inúmeras contribuições, ele permite construir dimensões de ETL em alto nível processando os esquemas estrela, snowflake, SCDs e dimensões de dados intensivos. Pelo fato dele utilizar MapReduce, ele pode automaticamente processar mais de um nó enquanto ao mesmo tempo fornece a sincronização dos dados através dos nós. Além da escalabilidade, ele oferece alta tolerância à falhas, possui código aberto e é fácil de usar com um único arquivo de configuração executando todos os parâmetros.

O principal objetivo do ETLMR é otimizar o tempo de processamento dos processos de ETL por meio do *framework* MapReduce. Porém, não há nada a respeito de como auxiliar na modelagem dos processos de ETL.

2.2.4 CloudETL

O *framework* CloudETL é uma solução para processos de ETL que usa *Hadoop* para paralelizar os processos de ETL e *Hive* para processar os dados de forma distribuída. Para o CloudETL o *Hadoop* é a plataforma de execução dos processos de ETL e o *Hive* é o sistema de armazenamento. Conforme a figura 2.4, os componentes do CloudETL são as APIs (Interfaces de Programação de Aplicação), um conjunto de elementos para efetuar as transformações nos dados,

identificados como ETL *transformers*, e um gerenciador de tarefas que controla a execução das tarefas submetidas ao *Hadoop*.

O CloudETL fornece suporte de alto nível em ETL para construção de diferentes esquemas de DW, como esquema estrela, *snowflake* e SCD (*slowly changing dimensions*). Ele facilita a implementação de processos de ETL em paralelo e aumenta a produtividade do programador significativamente. Esta abordagem facilita as atualizações de SCDs em um ambiente distribuído (LIU; THOMSEN; PEDERSEN (2013)).

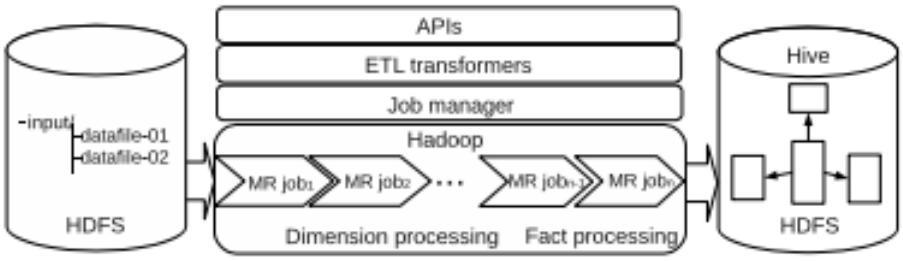

Figura 2.4 Arquitetura do CloudETL (Adaptado de LIU; THOMSEN; PEDERSEN (2013))

O CloudETL é uma alternativa quando o problema é o processamento de um grande volume de dados por possuir a propriedade de processamento distribuído, porém não oferece nenhum suporte para modelagem de processos de ETL ficando a cargo do programador ou da equipe responsável pelo projeto de DW.

2.2.5 P-ETL

P-ETL (Parallel - ETL) foi desenvolvido utilizando o *framework* Hadoop com o paradigma MapReduce. Ele oferece duas maneiras de ser configurado: por meio de uma GUI (*Graphical User Interface*) ou um arquivo de configuração XML. A figura 2.5 mostra a interface gráfica de configuração do P-ETL, ela é organizada em três abas: *Extract, Transform, Load*; e uma parte para parâmetros avançados.

O processo de ETL do *framework* inicia-se na aba *Extract*, as configurações fornecidas pelas outras abas dependem desta primeira, principalmente o formato dos dados da fonte de dados e sua estrutura. O primeiro passo da fase de extração é localizar a fonte de dados. O arquivo base do P-ETL é no formato "csv"(*Comma Separated Values*). Ele converte a fonte para o formato "csv"permitindo a entrada dos dados em vários formatos. Para acelerar a carga dos dados da fonte no HDFS (formato utilizado pelo *Hadoop*), o P-ETL permite o usuário comprimi-los. A respeito da partição, o usuário pode escolher o tipo de partição (*single, Round Robin, Round Robin by block*) e o número de dados por partição, além disso, ele pode configurar a extração pela quantidade de tuplas (por linhas ou blocos). A aba *Transform* permite o usuário escolher um lista de funções para transformação, cada função deve ser especificamente configurada (condições, expressões, entradas, etc.), assim, as funções são executadas na ordem que foram inseridas. E finalmente, a aba *Loading* permite configurar as tarefas de carga e incluir o destino dos dados

(*data warehouse, datamart, etc.*), os dados são comprimidos antes de serem carregados no HDFS e o separados no formato "csv"(BALA (2014)).

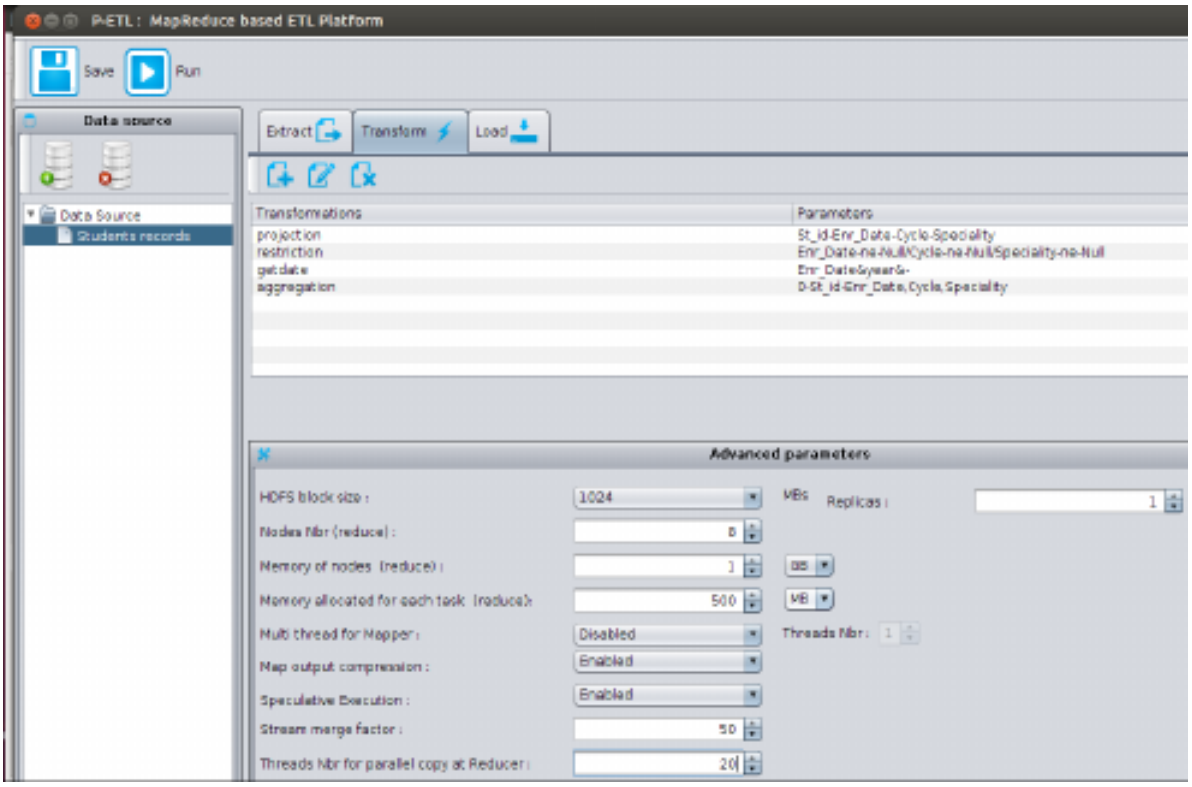

Figura 2.5 Interface de configuração do P-ETL (Adaptado de BALA (2014))

O P-ETL usa principalmente dois módulos do *framework* Apache Hadoop: (i) HDFS para o armazenamento distribuído e a alta vazão para o acesso aos dados das aplicações, e (ii) MapReduce para processar paralelamente. Futuramente, o P-ETL pretende adicionar outras funções de transformação para realizar processos mais complexos, e oferecer um ambiente na nuvem, mais precisamente, virtualizar e transformá-lo numa arquitetura orientada à serviço (SOA - *Service Oriented Architecture*) (BALA (2014)).

2.2.6 Big-ETL

BALA; BOUSSAID; ALIMAZIGHI (2015) propõe em seu trabalho uma abordagem chamada Big-ETL, no qual define que funcionalidades de ETL podem ser executadas em *cluster* utilizando o paradigma MapReduce (MR). O Big-ETL permite paralelizar e distribuir o processo de ETL em dois níveis: o processo de ETL (nível de granularidade maior - todo fluxo de ETL), e a funcionalidade de ETL (nível de granularidade menor - por exemplo, junção de tabelas); dessa forma, melhorando o desempenho. Para testar a abordagem proposta, o autor utilizou o P-ETL com intuito de melhorar a ferramenta definindo o processo de ETL (nível de granularidade maior), e a funcionalidade de ETL (nível de granularidade menor) como níveis para o experimento, demonstrando ser uma boa alternativa para melhorar o desempenho nos processos de ETL.

Futuramente os autores pretendem apresentar um *benchmark* no qual comparará quatro abordagens: processo de ETL centralizado; processo de ETL distribuído, Big-ETL e uma abordagem híbrida.

2.2.7 FramETL

O FramETL é um *framework* para desenvolvimento de aplicações ETL. Ele oferece um ambiente programável e integrado para modelagem e execução de processos de ETL utilizando uma linguagem de programação. O autor utilizou conceitos de *frameworks* como flexibilidade, extensibilidade, reuso e inversão de controle para o desenvolvimento do FramETL. Por meio desses conceitos, utilizando o *framework*, o autor aplicou sua solução para construções de duas aplicações de ETL. Porém, SILVA (2012) não fez uso de SGBDs NoSQL, pois o foco de sua ferramenta não era lidar com esses tipos de SGBDs.

2.2.8 Outras Ferramentas

Esta subseção apresenta outras ferramentas de ETL presentes no mercado e na literatura, mas que não possuem foco em SGBDs NoSQL apesar de darem algum tipo de suporte à eles.

2.2.8.1 Pentaho

Pentaho Data Integration (conhecido também por ***Kettle***) é uma ferramenta *open source* para aplicações de ETL. Ela é composta basicamente por quatro elementos: extração de diferentes fontes de dados, transporte de dados, transformação dos dados e carga em *data warehouse*. O *Kettle* pode ser implementado em um único nó, bem como na nuvem, ou em *cluster*. Ele pode carregar e processar *big data* de várias formas oferecendo flexibilidade e segurança (MALI; BOJEWAR (2015), INFORMATION (2017), INTEGRATION (2017)). Porém, por ser uma ferramenta genérica ela é de difícil customização, e muitas vezes é considerada de difícil utilização por seus usuários, além de ter partes de suas funcionalidades disponíveis apenas em edições comerciais.

2.2.8.2 Talend Studio

Talend Open Studio é uma plataforma de integração de dados que possibilita processos de integração, seu monitoramento opera como um gerador de código, produzindo *scripts* de transformação. Ele possui um repositório de metadados no qual fornece os dados (definições e configurações relacionados a cada tarefa) para todos os seus componentes. O Talend Studio é comumente utilizado para migração de dados, sincronização ou replicação das bases de dados (MALI; BOJEWAR (2015), INFORMATION (2017)).

2.2.8.3 CloverETL

Clover é uma ferramenta ETL de código aberto considerada para transformação e integração, limpeza e distribuição de dados em aplicações, banco de dados e *data warehouses*. Ela é baseada em Java e pode ser utilizada em linha de comando e é independente de plataforma (MALI; BOJEWAR (2015)). Porém, por ter vários recursos sua curva de aprendizagem é alta e muitos desses recursos valiosos estão disponíveis apenas em sua edição comercial.

2.2.8.4 Oracle Data Integrator (ODI)

Oracle Data Integrator é uma plataforma de integração de dados que atende diversos requisitos de integração, desde grandes volumes de dados até o carregamento em *batch*. As bases de dados de origem e destino podem incluir base de dados relacionais, arquivos XML, tabelas *Hive, Hbase*, arquivos *HDFS*, entre outros. Os usuários podem inserir filtros, junções, agregações, e outros componentes de transformação (SILVA (2016)). Porém, o ODI é uma ferramenta comercial e não permite a customização de suas aplicações.

2.3 Considerações Finais

Este capítulo discorreu a respeito dos principais assuntos abordados nesta dissertação, bem como as ferramentas de ETL encontradas na literatura. A maioria delas foca no desempenho ao lidar com grandes volumes de dados e BDs NoSQL. A ferramenta P-ETL (BALA (2014)) apresentou um arquivo "csv"como alternativa para exportar diversos tipos de dados, porém não há um enfoque em BDs NoSQL. A abordagem PygramETL (THOMSEN; PEDERSEN (2009)) facilita a carga de dados, mas lida apenas com SGBDs relacionais. Outras ferramentas como o ETLMR, CloudETL, BigETL utilizam processamento paralelo e distribuído para facilitar a execução dos processos de ETL apenas, deixando a cargo do projetista de ETL a modelagem dos processos.

O capítulo seguinte irá apresentar os componentes de ETL4NoSQL, o *framework* programável, flexível e integrado proposto por esta dissertação como uma alternativa para sanar a dificuldade de modelar, executar e reutilizar processos de ETL em BDs NoSQL.

3

O Framework ETL4NoSQL

Neste capítulo são apresentados os conceitos do *framework* ETL4NoSQL, que consiste numa plataforma de *software* para desenvolvimento de aplicações de ETL, mais especificamente uma ferramenta que auxilia a construção de processos de ETL buscando apoiar a modelagem, reutilização e desempenho dos processos.

O ETL4NoSQL oferece um ambiente com componentes integrados para modelar processos de ETL e implementar funcionalidades utilizando uma linguagem de programação independente de uma GUI (*Graphical User Interface* - Interface Gráfica do Usuário).

Para a especificação do *framework* proposto neste trabalho, foram elencados os requisitos de *software* utilizando a abordagem de desenvolvimento baseado em componentes, fundamentada no estudo de CHEESMAN; DANIELS (2001). Neste estudo, temos a separação entre a modelagem do domínio e da especificação. A modelagem de domínio consiste na definição dos casos de uso, do modelo conceitual e do modelo comportamental. Já a modelagem de especificação é segmentada em três partes, a parte de identificação de componentes, de interação entre os componentes e a especificação de componentes.

A seguir, são detalhados os requisitos de *software*, os modelos de domínio, os modelos de especificação, as especificações dos componentes, o ambiente de implementação e as interfaces de programação do ETL4NoSQL.

3.1 Requisitos de Software do ETL4NoSQL

Requisitos de *software* são descrições de como o sistema deve se comportar, definidos durante as fases iniciais do desenvolvimento do sistema como uma especificação do que deveria ser implementado (SOMMERVILLE (2013)). Os requisitos podem ser divididos em funcionais e não funcionais, onde o primeiro descreve o que o sistema deve fazer, ou seja, as transformações a serem realizadas nas entradas de um sistema, a fim de que se produzam saídas, já o outro expressa as características que este *software* vai apresentar (SOMMERVILLE (2013)).

O ETL4NoSQL é um *framework* que tem como principal objetivo auxiliar na criação de aplicações de ETL ao se utilizar principalmente BDs NoSQL. Um sistema de *software* pode ter seus dados armazenados em BDs relacionais, que seguem o modelo entidade e relacionamento, ou não relacionais, no qual possui pouca definição de esquema, não seguem um modelo específico e são regularmente chamados de BDs NoSQL (FOWLER; SADALAGE (2013)). Os BDs NoSQL possuem quatro paradigmas frequentemente utilizados: Chave-Valor, Família de Colunas, Documentos e Grafo (FOWLER; SADALAGE (2013)).

Os SGBDs relacionais utilizam uma linguagem de gerenciamento de dados padrão conhecida por SQL (Structure Query Language) (FOWLER; SADALAGE (2013)), porém os SGBDs NoSQL não possuem uma linguagem em comum, como os SGBDs relacionais, cada SGBD NoSQL possui sua própria linguagem de gerenciamento de dados (FOWLER; SADALAGE (2013)). Por isso, é essencial que haja um componente que seja capaz de fazer a leitura diretamente da fonte de dados e um componente que também possa carregar esses dados diretamente no seu destino, independente do seu tipo, saber se é um arquivo texto, um arquivo XML, SGBD relacional, SGBD NoSQL, entre outros. Um componente tem como uma de suas definições ser uma unidade de *software* independente, que encapsula a sua implementação, e oferece serviços por meio de suas interfaces (SOUZA GIMENES; HUZITA (2005)).

Outra importante características ao especificar o uso do ETL4NoSQL são os processos de ETL, que possuem quatro etapas básicas: extração, limpeza/transformação e carga (KIMBALL; CASERTA (2004)). O fluxo do processo de ETL inicia-se com a extração dos dados a partir de uma fonte de dados. A começar da extração, é possível que um componente passe os dados para uma APD (Área de Processamento de Dados), onde é permitido modelar os dados executando processos de limpeza e transformação por meio de mecanismos (mecanismos de ETL) como de junção, filtro, união, agregação e outros. E finalmente, os dados são carregados em uma estrutura de dados destino.

Dessa forma, o ETL4NoSQL possui um componente que permite a leitura dos dados de diversos SGBDs NoSQL, de arquivos textuais, além dos SGBDs relacionais. Outro componente que permite a execução dos mecanismos de ETL, bem como faz uso de componentes para o gerenciamento da execução dos mecanismos, a construção da sequência dos processos de ETL e a escolha do tipo de processamento. O ETL4NoSQL é composto também de um componente que permite carregar diretamente os dados no destino independente do seu tipo. No quadro 3.1 é

apresentado os principais requisitos elencados do ETL4NoSQL. Definimos como importante as prioridades que são imprescindíveis para o desenvolvimento e funcionamento do *framework*, e desejável as funcionalidades que aprimoram o uso do *framework*, porém não interferem no seu principal objetivo.

Quadro 3.1 Requisitos do ETL4NoSQL

Funcionalidade	Requisito	Prioridade
Suporte à plataforma	Ser independente de plataforma.	Importante
Suporte à fonte	Ser capaz de ler diretamente da fonte de dados, independente do seu tipo, saber se é uma fonte SGBD relacional, arquivo de texto, XML ou NoSQL.	Importante
Suporte ao destino	Ser capaz de carregar diretamente os dado no destino, independente do seu tipo, saber se o destino é SGBD relacional, arquivo de texto, XML ou NoSQL.	Importante
Suporte à modelagem	Apoiar na extração de dados de múltiplas fontes de dados, na limpeza dos dados, na transformação, agregação, reorganização e operações de carga.	Importante
Paralelismo	Apoiar as operações de vários segmentos e execução em paralelo, internamente. A ferramenta deve ser capaz de distribuir tarefas entre múltiplos servidores.	Importante
Programável	Apoiar o agendamento de tarefas de ETL e ter suporte para programação em linha de comandos usando programação externa.	Importante
Reutilização	Apoiar a reutilização dos componentes do *framework* e da lógica das transformações para evitar a reescrita.	Importante
Apoio ao nível de *debugging*	Apoiar o tempo de execução e a limpeza da lógica de transformação. O usuário deve ser capaz de ver os dados antes e depois da transformação.	Desejável
Implementação	Suportar a capacidade de agrupar os objetos ETL e implementá-los em ambiente de teste ou de produção, sem a intervenção de um administrador de ETL.	Desejável
Garantia de Qualidade	Ser capaz de estabelecer processos, métricas e avaliações que possibilitem e garantam a qualidade de *software*.	Desejável

3.2 Modelagem do Domínio de ETL4NoSQL

A modelagem do domínio de ETL4NoSQL é apresentada a seguir por meio de seus três modelos: modelo conceitual, modelo de casos de uso e modelo de comportamento.

3.2.1 Modelo Conceitual

Os conceitos de entidades para aplicações de ETL identificadas para o ETL4NoSQL são: Fonte, Destino, Modelagem, Processamento, Operações, ProcessamentoDistribuído e ProcessamentoCentralizado. O modelo conceitual pode ser visualizado na figura 3.1.

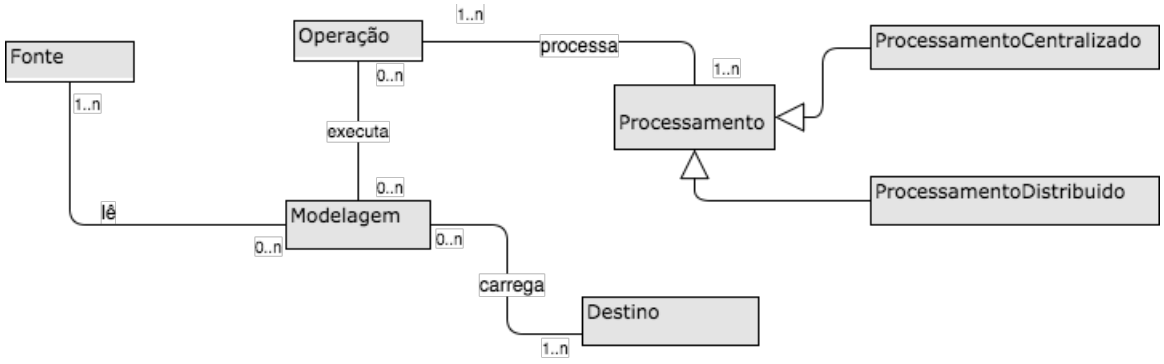

Figura 3.1 Modelo conceitual do ETL4NoSQL

A entidade Modelagem faz a leitura de uma ou mais Fontes de dados, executa nenhuma ou muitas Operações. As Operações por sua vez são processadas por um ou mais Processamentos de forma a serem Processamentos Centralizados ou Processamentos Distribuídos. E por fim, a Modelagem carrega o resultado das Operações processadas à um ou mais Destinos.

Na subseção seguinte é apresentado o modelo de casos de uso do ETL4NoSQL.

3.2.2 Modelo de Casos de Uso

Os casos de uso expressam as funcionalidades fundamentais ao desenvolvimento e uso do ETL4NoSQL. Eles permitem ao programador a visão do que é imprescindível ao implementar e determinar as interfaces de sistema e operações do *framework*.

Um conjunto de casos de uso foram identificados tais como: Ler fonte de dados, Escrever no destino, Modelar dados, Executar operação e Processar operações. O quadro 3.2 mostra a descrição sucinta de cada caso de uso.

Para finalizar a modelagem do domínio de ETL4NoSQL, a subseção seguinte apresenta o modelo comportamental do *framework* proposto neste trabalho.

3.2.3 Modelo Comportamental

Ao construir o modelo comportamental é possível identificar os conceitos com comportamentos mais relevantes para o negócio, bem como os estados e eventos que disparam as transições entre os estados (SOUZA GIMENES; HUZITA (2005)). Dessa forma, o diagrama de estados do ETL4NoSQL é apresentado na figura 3.2, nele podemos ver as transições de leitura da fonte de dados, validação e identificação dos dados, assim como o tratamento caso os dados não possam ser identificados. Subsequente a isso, podemos ver as transições do armazenamento dos dados para o processamento, a criação dos processos de ETL, a escolha da forma de processamento, execução das operações, e também o tratamento para as operações que não puderem ser executadas. Finalmente, a transição da carga dos dados pode ser feita na base de destino seguido da mensagem de tratamento, caso haja sucesso ou não na execução.

Quadro 3.2 Modelo de Casos de Uso do ETL4NoSQL

Nome: Ler fonte de dados **Objetivo:** Fazer a leitura de qualquer tipo de dados a partir de uma fonte de dados. **Pré-condição:** os parâmetros para permissão de conexão com a fonte de dados devem estar disponíveis. **Ação:** ler (Fonte)
Nome: Escrever no destino **Objetivo:** Fazer a escrita de qualquer tipo de dado a partir do modelo processado pelo ETL4NoSQL em uma base de dados de destino. **Pré-condição:** os parâmetros para permissão de conexão e escrita com o destino devem estar disponíveis. **Ação:** escrever (Destino)
Nome: Modelar dados **Objetivo:** Permitir a modelagem dos dados por meio de mecanismos de junção, filtro, união, agregação e outros. **Pré-condição:** os mecanismos de transformação e limpeza devem estar disponíveis para executar a modelagem. **Ação:** modelar (Modelagem, Operação)
Nome: Executar operação **Objetivo:** Armazenar, gerenciar e executar as operações criadas pela ação de modelar. **Pré-condição:** as operações devem ser criadas previamente pela ação de modelar. **Ação:** executar (Operação, Processamento)
Nome: Processar operações **Objetivo:** Processar as operações armazenadas de forma centralizada ou distribuída. **Pré-condição:** as operações precisam estar disponíveis para o processamento. **Ação:** processar (Processamento, Operação)

3.3 Modelagem da Especificação do ETL4NoSQL

A modelagem de especificação visa definir, em um nível alto de abstração, os serviços oferecidos pelos componentes (SOUZA GIMENES; HUZITA (2005)). Dessa forma, é possível determinar a arquitetura e especificar os seus componentes. É importante dar ênfase a especificação das interfaces, pois isso contribui para uma clara separação entre os componentes, e também, para assegurar o princípio de encapsulamento de dados e comportamento (SOUZA GIMENES; HUZITA (2005)). CHEESMAN; DANIELS (2001) divide a modelagem da especificação em três estágio, sendo assim, os estágios da modelagem de especificação do ETL4NoSQL são apresentados a seguir.

3.3.1 Identificação de Componentes

Seguindo o modelo de conceitos do negócio e do modelo de casos de uso foi possível identificar as interfaces para os componentes de negócio, as interfaces de sistema para os componentes de sistema e gerar a arquitetura de componentes inicial. As interfaces de negócio

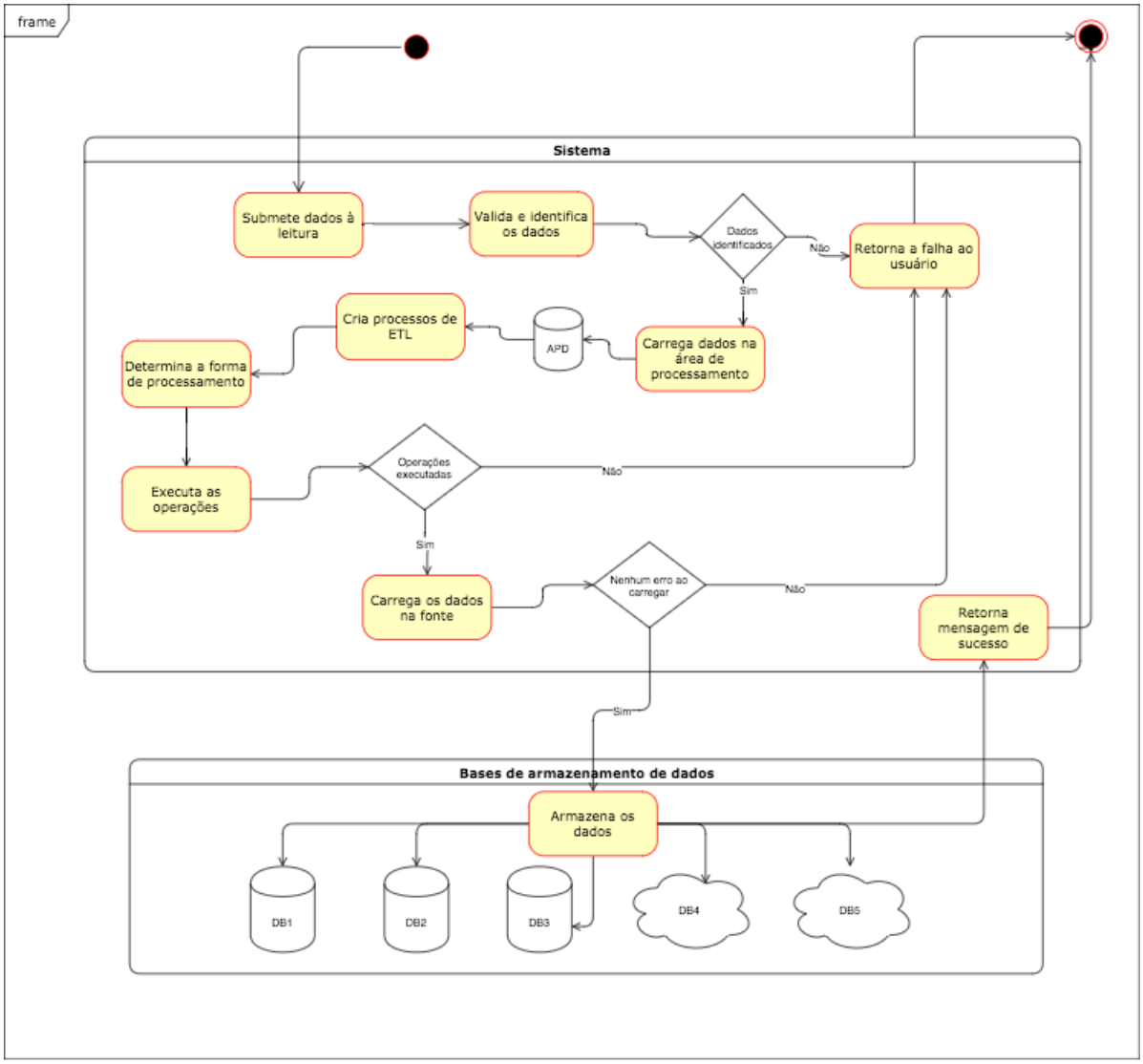

Figura 3.2 Diagrama de Estado do ETL4NoSQL

são reconhecidas por meio do modelo conceitual, as interfaces de sistema e operações a partir dos casos de uso.

3.3.2 Interfaces de Sistemas

As interfaces de sistemas do ETL4NoSQL identificadas, por meio do modelo de casos de uso apresentado no quadro 3.2, foram: IReadData, IWriteData, IModelData, IExeOp e IProcOp. Para especificar as interfaces foi utilizado OCL (Object Constraint Language) (WARMER; KLEPPE (1998)).

```
context System :: IReadData (source : Fonte)
pre:
Fonte.allInstances@includes(Fonte)
Fonte.connection = estabelecido
```

```
Fonte.allInstances@includes(Fonte)
Fonte.connection = retorna mensagem de erro
Fonte.allInstances@includes(Fonte)
Fonte.structureType = reconhecido
Fonte.allInstances@includes(Fonte)
Fonte.structureType = retorna mensagem de erro

post:
Fonte. allInstances@includes
(f: Fonte | not Fonte.allInstances@pre@includes (f) and
Os atributos do objeto f foram inicializados)
```

A partir da especificação foram identificadas as operações: Interface IReadData (Connection(connect) e structureType()).

```
context System :: IWriteData (load : Destino)
pre:
Destino.allInstances@includes(Destino)
Destino.connection = estabelecido
Destino.allInstances@includes(Destino)
Destino.connection = retorna  erro
Destino.allInstances@includes(Destino)
Destino.structureType = reconhecido
Destino.allInstances@includes(Destino)
Destino.structureType = retorna  erro
Destino.allInstances@includes(Destino)
Destino.structureType = permissao de escrita
Destino.allInstances@includes(Destino)
Destino.structureType = retorna  erro

post:
Fonte. allInstances@includes (d: Destino | not
Destino.allInstances@pre@includes (d) and
Os atributos do objeto d foram inicializados)
```

As operações identificadas foram: Interface IWriteData (connection(connect), structureType() e allowWrite()).

```
context System :: IModelData (model : Modelagem;
source: Fonte; operation: Operacao)
pre:
Fonte.allInstances@includes(Fonte)
Fonte.connection = estabelecido and
Fonte.structureType = reconhecido
```

```
Fonte.allInstances@includes(Destino)
Fonte.connection = retorna  erro
Operacao.allInstances@includes(Operacao)
Operacao.mecanismo = existe
Operacao.allInstances@includes(Operacao)
Operacao.mecanismo = retorna erro
Modelagem.allInstances@includes(Modelagem)
Modelagem.operacao(dados)
post:
Modelagem. allInstances@includes (m: Modelagem | not
Destino.allInstances@pre@includes (m) and
Os atributos do objeto m foram inicializados
m foi ligado ao objeto f
f.Fonte = Fonte
m foi ligado ao objeto o
o.Operacao = Operacao
Todas as operacoes de modelagem foram criadas e
armazenadas em um APD)
```

As operações identificadas foram: Interface IModelData (readData(CodFonte) e createOp(CodOperação, data)).

```
context System :: IExeOp (model : Modelagem; operation:
Operacao, processing: Processamento)
pre:
Modelagem.allInstances@includes(Modelagem)
Modelagem.operacoes = existem
Operaaoo.allInstances@includes(Operacao)
Operacao.executa = retorna  erro
Operacao.allInstances@includes(Operacao)
Operacao.manage(operacoes)
post:
Operacao. allInstances@includes (o: Operacao | not
Operacao.allInstances@pre@includes (o) and
Os atributos do objeto o foram inicializados
o foi ligado ao objeto m
m.Modelagem = Modelagem
o foi ligado ao objeto p
p.Processamento = Processamento
Todas as operacoes escalonadas e estao pronta para
o processamento)
```

As operações identificadas foram: Interface IExeOp (modelOperation(CodModelagem)

e operationManagement()).

```
context System :: IProcOp (operation: Operacao,
processing: Processamento)
pre:
Operacao.allInstances@includes(Operacao)
Operacao.toExec = pronto
Processamento.allInstances@includes(Processamento)
Processamento.typeProc(tipoProcessamento)
post:
Processamento. allInstances@includes (m: Processamento
| not Destino.allInstances@pre@includes (p) and
Os atributos do objeto p foram inicializados
p foi ligado ao objeto o
o.Operacao = Operacao
– Foi gerado o processamento das operacoes de acordo
com o tipo escolhido)
```

As operações identificadas foram: Interface IProcOp (process(CodOperação) e typeProc(tipoProcessamento)).

3.3.3 Interfaces de Negócio

Para definir as dependências no modelo, CHEESMAN; DANIELS (2001) identificam o conceito de tipos principais, sendo estes tipos que possuem existência independente. No ETL4NoSQL foi possível identificar quatro tipos principais: Fonte, Modelagem, Operação e Processamento. Para cada tipo principal foi possível definir uma interface de negócio, como é apresentado na figura 3.3.

Na figura 3.3 pudemos perceber a interação entre os componentes, porém essa interação será realizada por sua interface específica que será detalhada posteriormente na seção 3.4.

3.3.4 Especificação da Arquitetura do Componente

SOMMERVILLE (2013), define o projeto de arquitetura como um processo criativo em que se tenta organizar o sistema de acordo com os requisitos funcionais e não funcionais. Um estilo de arquitetura é um padrão de organização de sistema (SHAW; GARLAN (1996), SOMMERVILLE (2013)), como uma organização cliente-servidor ou uma arquitetura em camadas. Porém, a arquitetura não necessariamente utilizará apenas um estilo, a maioria dos sistemas de médio e grande porte utilizam vários estilos. Para SHAW; GARLAN (1996), há três questões a serem definidas na escolha do projeto de arquitetura, a primeira é a escolha da estrutura, cliente-servidor ou em camadas, que permita atender melhor aos requisitos. A segunda questão é a respeito da decomposição dos subsistemas em módulos ou em componentes. E por

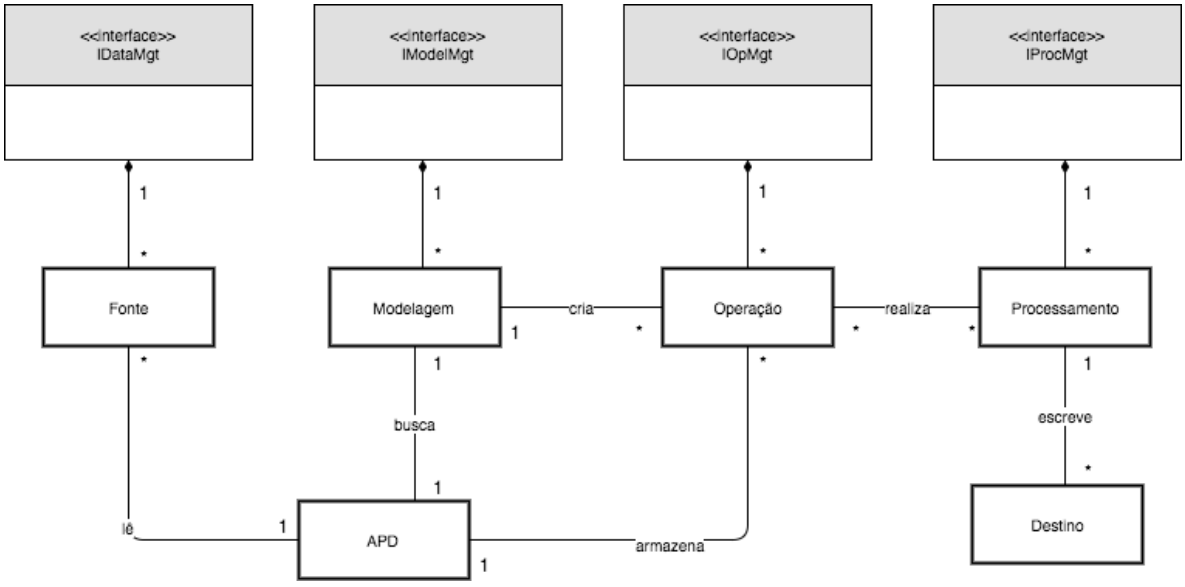

Figura 3.3 Definição de interfaces do modelo de negócio do ETL4NoSQL

fim, deve-se tomar a decisão de sobre como a execução dos subsistemas é controlada. A descrição da arquitetura pode ser representada graficamente utilizando modelos informais e notações como a UML (Unified Modeling Language) (CLEMENTS et al. (2002), SOMMERVILLE (2013)). Para o ETL4NoSQL, cada componente foi associado à sua interface de negócio identificada e uma interface de gerenciamento foi separada das outras interfaces de negócio, como pode ser visto na figura 3.4.

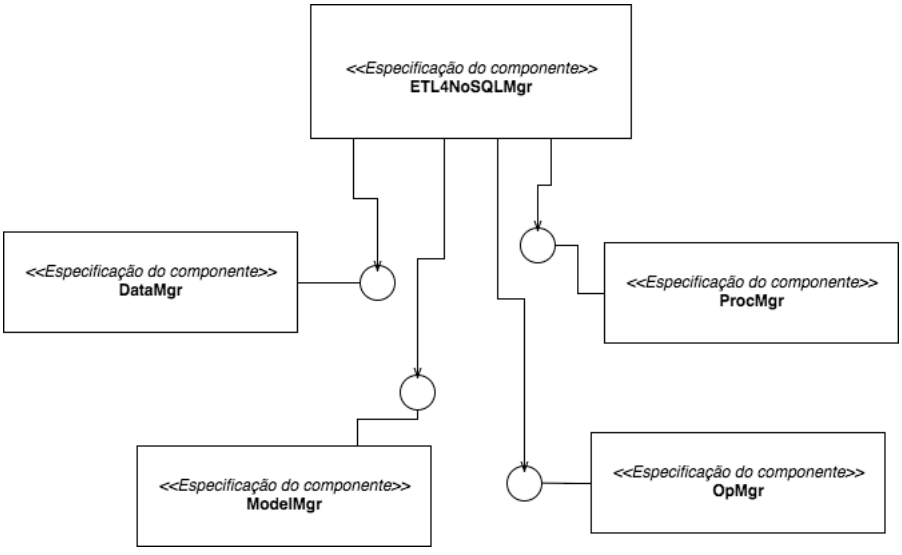

Figura 3.4 Especificação da arquitetura do componente de ETL4NoSQL

3.4 Interação entre Componentes

Esta etapa é detalhada, em termos de interações, utilizando diagramas de colaboração. Dessa forma, apresentamos os diagramas de colaboração para as interações do ETL4NoSQL na

subseção a seguir.

3.4.1 Operações da interface de negócio

As operações de negócio são identificadas quando as interações forem analisadas para cada interface do sistema. Analisando as pré e pós-condições das operações de interface do sistema identificadas anteriormente foi possível detalhar usando diagramas de colaboração as interações necessárias para efetuar as operações do ETL4NoSQL. Os diagramas de colaboração de cada operação podem ser visto nas figuras de 3.5 até 3.12.

A conexão com a fonte de dados é estabelecida (figura 3.5).

Figura 3.5 Diagrama de colaboração para conectar à base de fonte de dados

A estrutura de dados da fonte é reconhecida (figura 3.6).

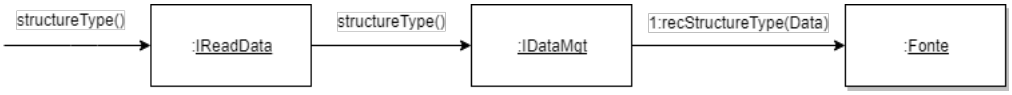

Figura 3.6 Diagrama de colaboração para verificar a estrutura de dados

A escrita na base de dados destino é permitida (figura 3.7).

Figura 3.7 Diagrama de colaboração para verificar se existe permissão de escrita na base de dados destino

A leitura dos dados da fonte é feita e armazenada na área de processamento de dados figura (3.8).

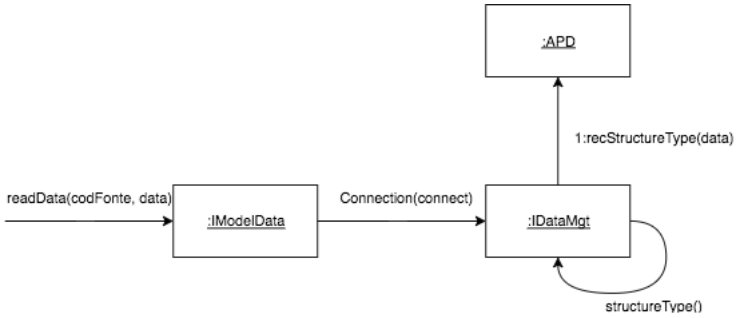

Figura 3.8 Diagrama de colaboração para leitura dos dados da fonte

Após a leitura dos dados da fonte é possível a criação das operações por meio dos mecanismos existentes (figura 3.9).

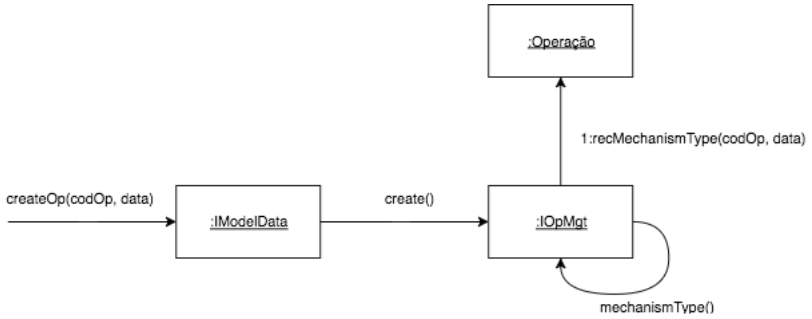

Figura 3.9 Diagrama de colaboração criação das operações de ETL

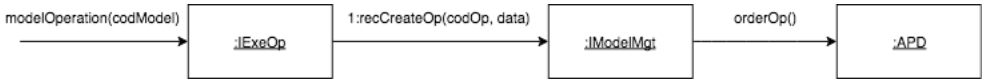

Figura 3.10 Diagrama de colaboração modelar as operações criadas

Com as operações criadas, deve-se colocá-las em ordem de execução (figura 3.10). É possível também que as operações sejam apagadas e alteradas (figura 3.11).

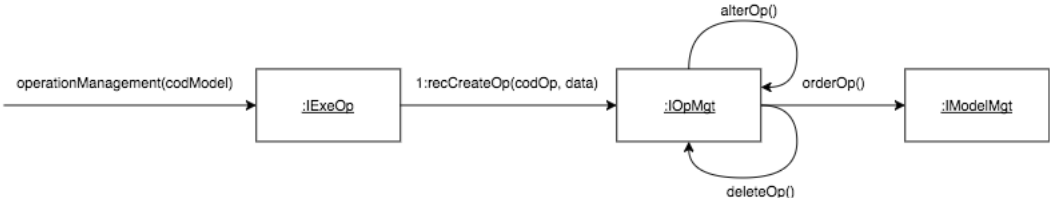

Figura 3.11 Diagrama de colaboração para gerenciar as operações

Com as operações criadas é possível processá-las de forma a escolher o tipo de processamento (centralizado ou distribuído) (figura 3.12).

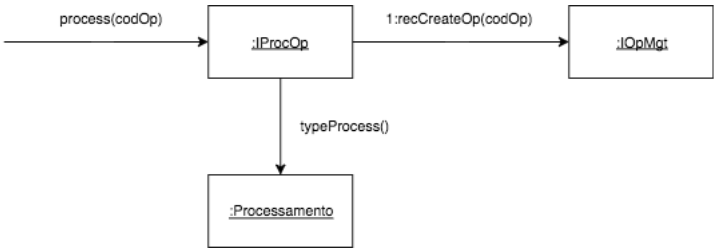

Figura 3.12 Diagrama de colaboração para processar as operações

3.5 Especificação de Componentes

Posteriormente à definição das interfaces de negócio, é possível detalhá-las. Para cada interface, as operações são especificadas com suas assinaturas, pré e pós-condições.

```
context IDataMgt :: connection(connect): Fonte
```

```
        pre:
        – Existe um parametro de conexao "connect"
        – Existe uma fonte para ser conectada
        post:
        – A conexao com a fonte foi estabelecida ou nao
        – Recebe uma mensagem de conexao bem sucedida ou erro

context IDataMgt :: strutuctureType():Fonte
        pre:
        – A conexao com a fonte foi bem sucedida
        post:
        – A estrutura de dados foi reconhecida ou nao
        – Retorna mensagem de sucesso ou erro

context IDataMgt :: allowWrite():Destino
        pre:
        – A conexao com o destino foi bem sucedida
        post:
        – A base de dados destino permite escrita ou nao
        – Retorna mensagem de sucesso ou erro

context IModelMgt :: readData(codFonte, data):APD
        pre:
        – A conexao com a fonte foi bem sucedida
        – A estrutura de dados foi reconhecida
        – Existe um parametro de busca "data" valido para
                estrutura de dados reconhecida
        post:
        – A leitura dos dados foi realizada ou nao
        – Retorna os dados em uma APD ou uma mensagem de erro

context IModelMgt :: createOp(codOp, data):Operacao
        pre:
        – Existe o mecanismo desejado para a criacao da operacao
        – Existe um parametro "data" com os dados a serem
        usados na operacao
        post:
        – A operacao foi criada ou nao
        – Retorna o codOp ou uma mensagem de erro

context IOpMgt :: modelOperation(codModel)
```

```
        pre:
        – Existem operacoes criadas na APD (codModel)
        post:
        – As operacoes sao ordenadas para execucao

context IOpMgt :: operationManagement(codModel)
        pre:
        – Existem operacoes criadas na APD (codModel)
        post:
        – As operacoes foram alteradas, apagadas ou nao
        – Retorna mensagem de sucesso ou erro

context IProcMgt :: process(codOp)
        pre:
        – Existe a operacao (codOp)
        – Foi escolhido o tipo de processamento
        (centralizado ou distribuido)
        post:
        – A operacao foi processada ou nao
        – Retorna mensagem de sucesso ou erro
```

Depois das operações terem sido especificadas, o diagrama de especificação das interfaces foi criado e está representado na figura 3.13. Essas especificações referem-se ao uso dos componentes. Elas representam o que o usuário precisa saber sobre os componentes.

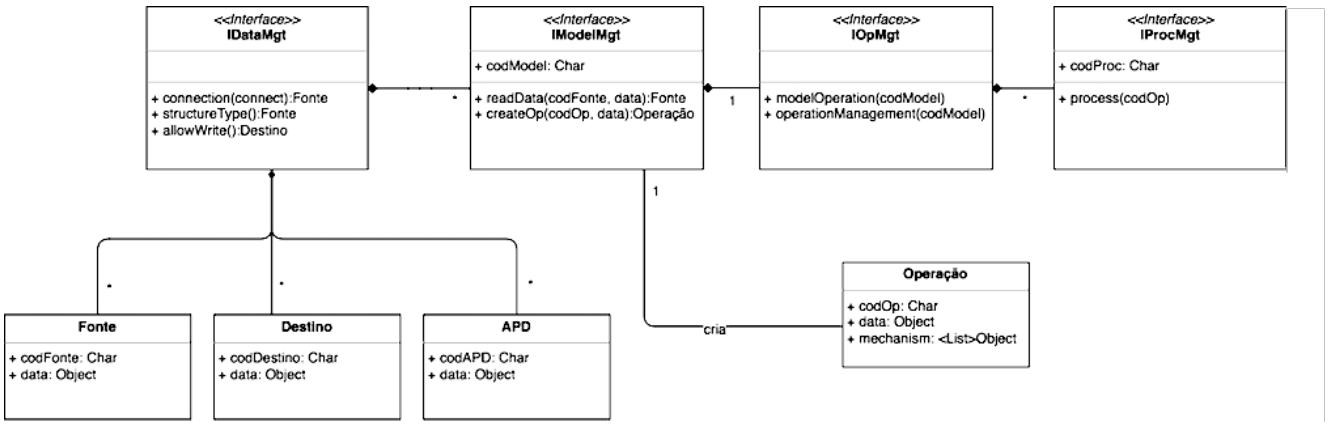

Figura 3.13 Diagrama de especificação das interfaces de ETL4NoSQL

Nas próximas seções mostraremos o ambiente de implementação utilizado para desenvolver o ETL4NoSQL, bem como suas interfaces de programação.

3.6 Ambiente de Implementação

Para a implementação do ETL4NoSQL utilizamos a linguagem de programação Python, pois ela segue o paradigma de orientação à objetos que é adequado para a proposta desta pesquisa. Além disso, Python tem uma sintaxe de fácil aprendizado e pode ser usada em diversas áreas, como Web e computação gráfica. Ela é uma linguagem de alto nível interpretada e também é um software livre. Devido a natureza do *framework* ser flexível, reusável e integrado faz com que a orientação à objetos torne-se um bom paradigma a ser aproveitado neste trabalho.

Assim, a implementação do *framework* foi baseada nos princípios do *design* orientado à objetos de inversão de controle, onde-se determina que os módulos de alto nível não devem ser dependentes de módulos de baixo nível, e sim, de abstrações, ou seja, os detalhes devem depender das abstrações. Esse princípio sugere que dois módulos não devem ser ligados diretamente, pois devem estar desacoplados com uma camada de abstração entre eles. Para suprir esse princípio, o ETL4NoSQL possui interfaces como o ETL4NoSQLMgr, IDataMgr, IModelMgr, IOpMgr e IProcMgr (figura 3.4). Elas utilizam dos mesmos comportamentos para as diversas variações, porém aplicados de acordo com a especificidade de cada um. Outro princípio importante utilizado é o da segregação de interfaces onde os usuários não devem ser forçados a depender de interfaces que não necessitam, elas devem ser enxutas com métodos específicos para cada interface.

Na seção seguinte apresentaremos as interfaces de programação do ETL4NoSQL, como uma sugestão a ser utilizada para a implementação deste trabalho de dissertação.

3.7 Interfaces de Programação

O ETL4NoSQL possui cinco principais interfaces de programação (a arquitetura com as interfaces é apresentada na figura 3.4). O ETL4NoSQLMgr é a interface principal que interliga as demais interfaces, ela é a *controller* (controlador) do *framework*. Na figura 3.14 podemos ver a implementação da interface ETL4NoSQLMgr, nota-se que todas as outras quatro interfaces foram importadas nela (destaque 1 na figura 3.14). Nesta interface, também é realizada a execução dos processos de leitura e escrita das bases de dados fonte, destino e área de processamento de dados (destaque 2 na figura 3.14), cria a amostra dos dados a serem manipulados, as operações, realiza a execução das operações e as gerencia (destaque 3 na figura 3.14).

A interface IDataMgr pode ser vista na figura 3.15, ela é responsável pela conexão de todas as bases de dados, bem como determina o tipo de estrutura da base de dados, e também, informa se há permissão para escrita na base de dados desejada.

Na figura 3.16 é apresentada a interface IModelMgr, esta realiza a leitura dos dados da base armazenada pela interface IDataMgr e configurada pelo controlador ETL4NoSQLMgr. A interface IModelMgr também cria as operações de ETL e grava os dados na base de dados determinada no controlador. Note que nesta interface é possível criar novas operações de ETL, além da leitura e escrita, tornando o *framework* flexível para cada aplicação. Ademais, após as

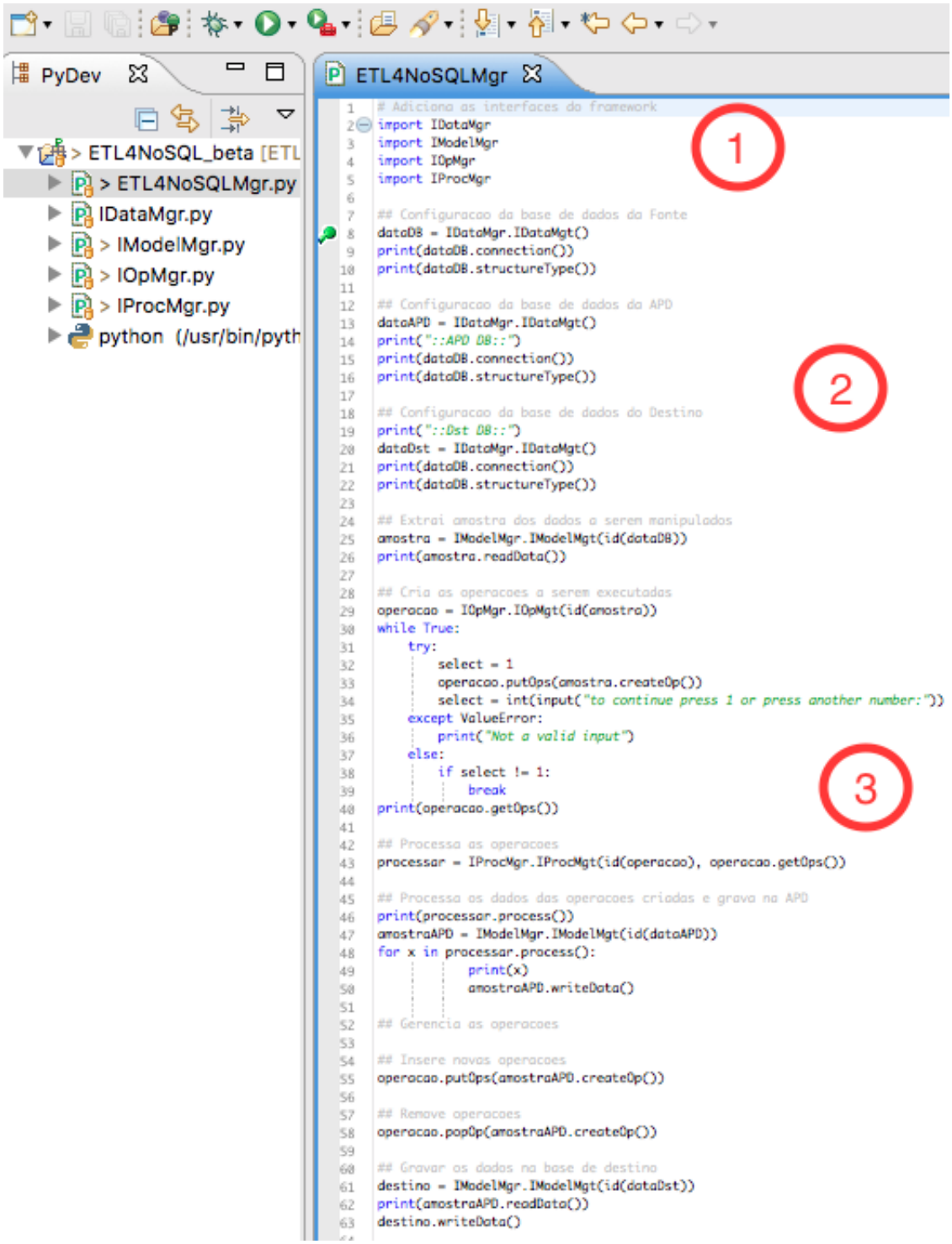

Figura 3.14 Tela do IDE LiClipse com a implementação da interface ETL4NoSQLMgr

operações serem criadas elas podem ser reutilizadas a qualquer momento facilitando o seu reuso.

As operações criadas pela interface IModelMgr são listadas na interface IOpMgr (figura 3.17). Ela gerencia a inserção, alteração e remoção das operações, bem como ordena a sequência de operações a serem executadas conforme desejável pelo projetista de ETL.

E finalmente, na figura 3.18, é apresentada a interface IProcMgr. Esta interface é responsável pela execução das operações listadas pela IOpMgr. Na IProcMgr também é possível

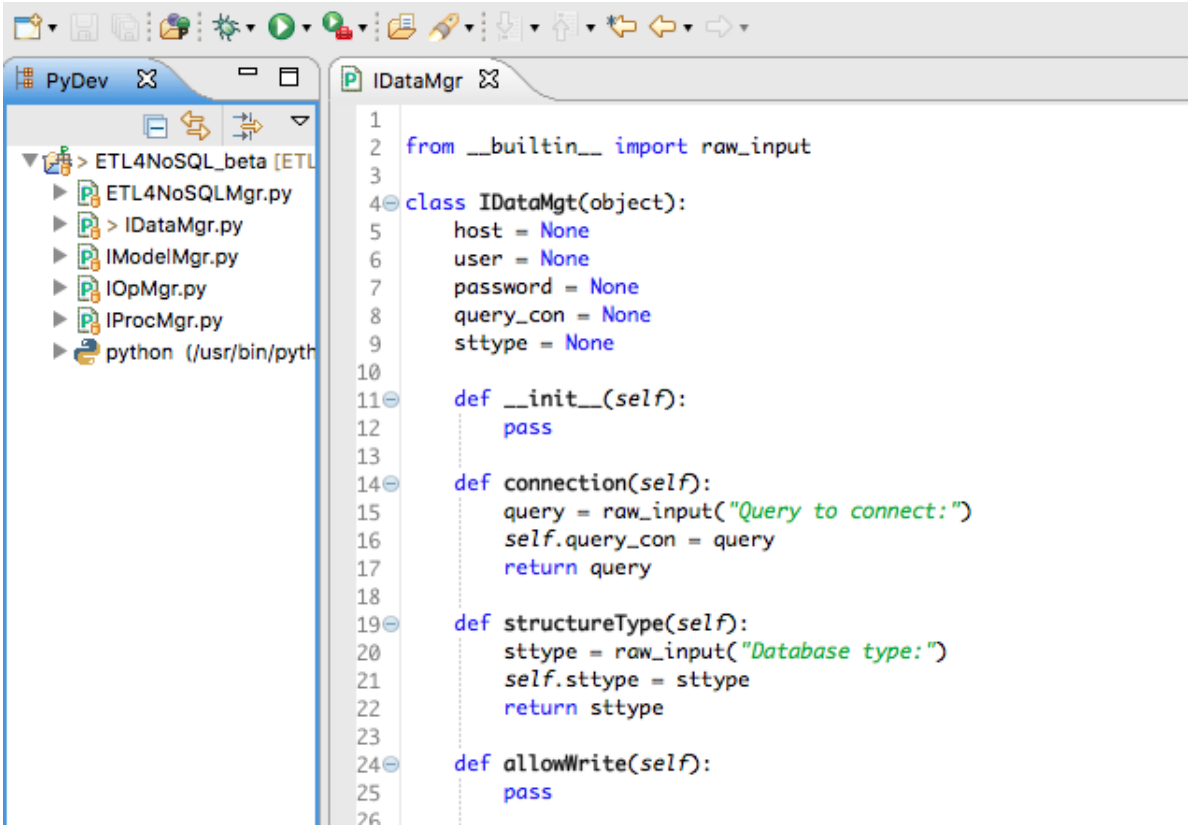

Figura 3.15 Tela do IDE LiClipse com a implementação da interface IDataMgr

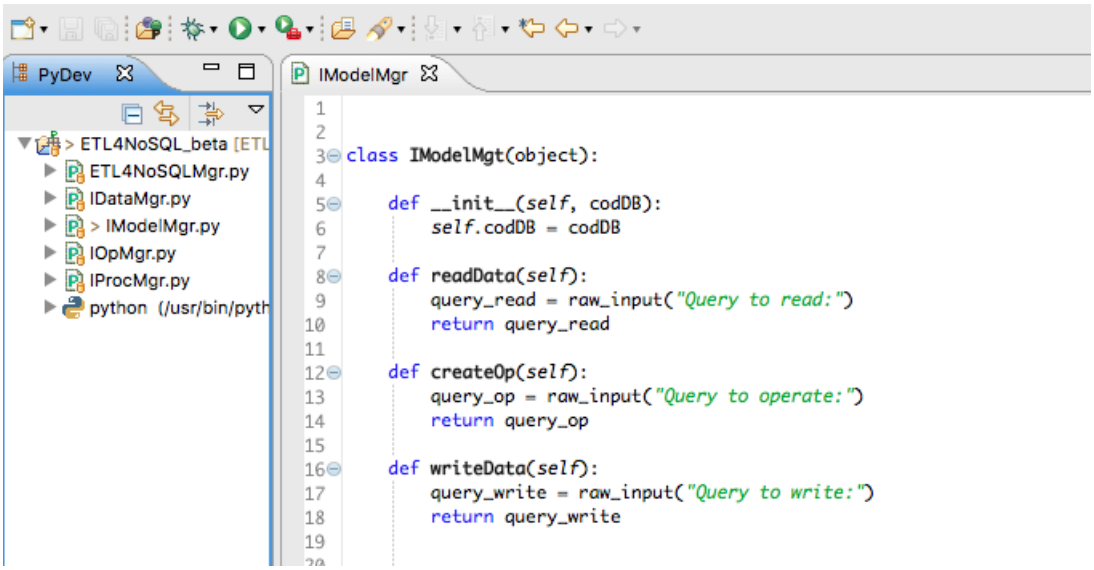

Figura 3.16 Tela do IDE LiClipse com a implementação da interface IModelMgr

determinar e implementar o tipo de processamento das operações: distribuído, centralizado e paralelo.

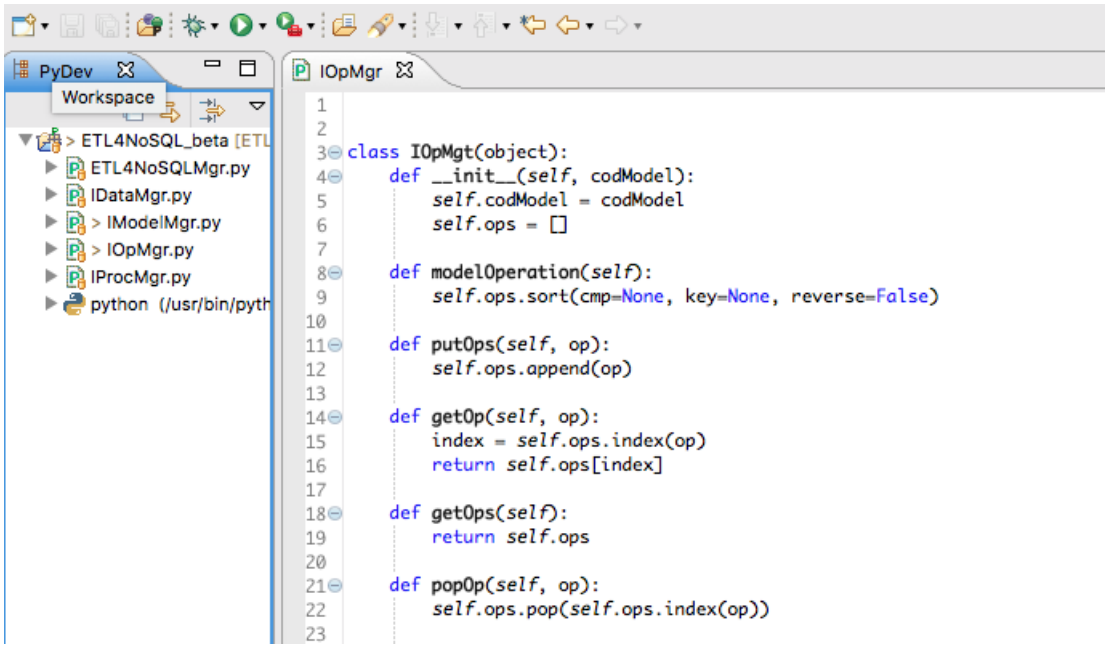

Figura 3.17 Tela do IDE LiClipse com a implementação da interface IOpMgr

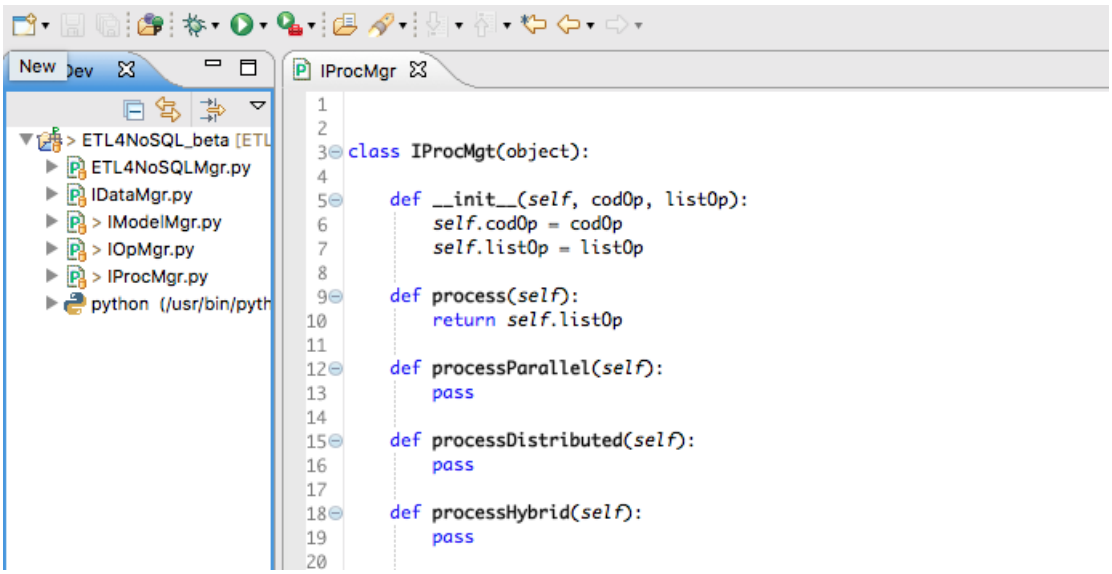

Figura 3.18 Tela do IDE LiClipse com a implementação da interface IProcMgr

3.8 Considerações Finais

Este capítulo descreveu o *framework* ETL4NoSQL. Para isso, foram apresentados os seus requisitos de desenvolvimento e exposta a sua modelagem do domínio por meio de seu modelo conceitual, de casos de uso e comportamental. Foram demonstradas a modelagem da especificação e identificado os componentes, as interfaces de sistemas e de negócio do *framework* proposto, além da especificação da arquitetura dos componentes do ETL4NoSQL, bem como suas interações. Mostramos como solução para implementação do *framework* um ambiente de implementação utilizando a linguagem orientada à objetos chamada Python, e também suas

interfaces de programação.

O próximo capítulo apresentará o estudo experimental de *software* que tem como objetivo definir se o ETL4NoSQL é adequado para desenvolver processos de ETL em BDs NoSQL e quais das suas funções são possíveis aprimorar.

4

Estudo Experimental de Software

Este capítulo provê o roteiro de experimentação de *software* para ferramentas de ETL. A Engenharia de *Software* Experimental tem como objetivo aprimorar métodos, técnicas e ferramentas de Engenharia de *Software* a partir de métodos experimentais (SOUZA et al. (2015)). As etapas definidas no processo de experimentação em Engenharia de *Software* proposto por SOUZA et al. (2015) consistem em etapas de definição, planejamento, operação, interpretação dos dados e empacotamento que serão melhor detalhadas nas seções a seguir.

4.1 Objetivos do experimento

O objetivo principal da aplicação deste experimento é definir se o *framework* proposto nesta dissertação é uma ferramenta adequada para auxiliar no desenvolvimento de processos de extração, transformação e carga de bancos de dados NoSQL.

4.1.1 Objetivo da Medição

Tendo como base as ferramentas de ETL existentes na literatura, o objetivo da medição é caracterizar:

1. Quais as principais funcionalidades que as ferramentas de ETL oferecem:
    (a) essas funcionalidades manipulam dados estruturados em modelos relacionais e não relacionais.
    (b) essas funcionalidades não manipulam BDs NoSQL.
2. Quais funcionalidades podem ser consideradas fundamentais para a produtividade na criação de processos de ETL:
    (a) quais necessitam manipular dados em grande escala.
    (b) quais não manipulam grande volume de dados.
3. Quais funcionalidades poderiam aprimorar as ferramentas de ETL.

4.1.2 Objetivos do Estudo

- Analisar as ferramentas de ETL;
- Com o propósito de caracterizar;
- Com respeito à intersecção das ferramentas de ETL existente;
- Do ponto de vista da literatura;
- No contexto comparativo entre as ferramentas mais conhecidas no mercado atual.

4.1.3 Questões

Q1. Existem funcionalidades listadas pelas ferramentas pesquisadas que não estão presentes no ETL4NoSQL?

Métrica: A lista de funcionalidades que não estão presentes no ETL4NoSQL.

Q2. Existem funcionalidades listadas pelas ferramentas pesquisadas que estão presentes no ETL4NoSQL, mas são consideradas pouco úteis (pouco útil nesta pesquisa é entendido como a funcionalidade ser pouco utilizada pelos processos de ETL)?

Métrica: A lista de funcionalidades que estão presentes nas ferramentas da literatura e no ETL4NoSQL e são consideradas pouco úteis.

Q3. Existem funcionalidades que estão presentes nas ferramentas de ETL da literatura e no ETL4NoSQL, consideradas úteis (nesta pesquisa, útil é entendido como constantemente requisitada pelos processos de ETL), e que poderiam ser melhoradas?

Métrica: A lista de funcionalidades que estão presentes nas ferramentas de ETL da literatura e no ETL4NoSQL, consideradas úteis e que necessitam melhorar.

Q4. Existem funcionalidades que estão presentes nas ferramentas de ETL da literatura que não estão presentes no ETL4NoSQL, mas que são consideradas úteis?

Métrica: A lista de funcionalidades que estão presentes nas ferramentas de ETL da literatura que não estão presentes no ETl4NoSQL e que são consideradas úteis.

4.2 Planejamento

Na etapa de planejamento são definidas as hipóteses do estudo, a descrição da instrumentação, as métricas, seleção do contexto e dos indivíduos, as variáveis, a análise qualitativa e a validade do experimento. Todas elas estão descritas nas subseções seguintes.

4.2.1 Definição das Hipóteses

Hipótese nula (H0): As funcionalidades oferecidas pelo ETL4NoSQL são similares às funcionalidades oferecidas pelas ferramentas presentes na literatura.

Fp - Funcionalidades do ETL4NoSQL;

Fl - Funcionalidades das ferramentas da literatura.

H0: Fl - (Fp $\cap$ Fl) = $\emptyset$

Hipótese alternativa (H1): A lista de funcionalidades oferecidas pelo ETL4NoSQL é diferente da lista de funcionalidades oferecidas pelas ferramentas presentes na literatura.

Fp - Funcionalidades do ETL4NoSQL;

Fl - Funcionalidades das ferramentas da literatura.

H1: Fl - (Fp $\cap$ Fl) $\neq \emptyset$

Hipótese alternativa (H2): Na lista de funcionalidades oferecidas pelo ETL4NoSQL e pelas ferramentas de ETL da literatura possuem funcionalidades consideradas pouco úteis.

Fpl - Funcionalidades presentes no ETL4NoSQL e nas ferramentas da literatura;

Fplu - Funcionalidades presentes no ETL4NoSQL e nas ferramentas da literatura que são consideradas úteis.

H2: Fpl - (Fpl $\cap$ Fplu) $\neq \emptyset$

Hipótese alternativa (H3): Na lista de funcionalidades oferecidas pelo ETL4NoSQL e pelas ferramentas de ETL da literatura possuem funcionalidades consideradas úteis, cujo necessita de melhorias.

Fplu - Funcionalidades presentes no ETL4NoSQL e nas ferramentas da literatura que são consideradas úteis;

Fplun - Funcionalidades presentes no ETL4NoSQL e nas ferramentas da literatura que são consideradas úteis, cujo necessita melhorias.

H3: Fplu - (Fplu $\cap$ Fplun) $\neq \emptyset$

Hipótese alternativa (H4): A lista de funcionalidades que estão presentes nas ferramentas da literatura e não estão presentes no ETL4NoSQL que são consideradas úteis.

Fpn - Funcionalidades de ETL4NoSQL não presentes (Fpn $\equiv$ Fl - (Fp $\cap$ Fl))

Fo - Funcionalidades não presentes no ETL4NoSQL, mas que são consideradas úteis.

H4: Fpn - (Fpn $\cap$ Fo) $\neq \emptyset$

4.2.2 Descrição da instrumentação

Para cada funcionalidade das ferramentas de ETL consideradas por este estudo oferecer a escolha no quadro 4.1:

Quadro 4.1 Descrição da Instrumentação

Presença da Funcionalidade (P)	Melhoria da Funcionalidade (M)	Utilidade da Funcionalidade (U)
1. Não está presente 2. Está presente parcialmente 3. Está presente	1. Necessita melhorar 2. Não há necessidade de melhoria 3. Pode melhorar, mas não há necessidade	1. É útil 2. Não é útil 3. É parcialmente útil

Para cada funcionalidade aplicar teste estatístico Qui-quadrado para definir:

- se pode considerar que essa funcionalidade é fornecida (P);
- se pode considerar que essa funcionalidade é constantemente utilizada pelos processos de ETL (U);

- se pode considerar que essa funcionalidade necessita de melhoria (M).

Resultado: N funcionalidades com valores (P; M; U) onde P - presença 0 - não presente; 1 - presente; U - utilidade 0 - não é útil; 1 - é útil; melhoria 0 - não necessita melhorar; 1 - necessita melhorar.

4.2.3 Métricas

Na tabela 4.2 são apresentadas as métricas utilizadas neste experimento.

Quadro 4.2 Métricas

N○	P	M	U	Descrição da Funcionalidade	Questões
1	0	0	0	Não está presente, não necessita melhorar, não é útil	Q1, Q4
2	0	0	1	Não está presente, não necessita melhorar, é útil	Q1, Q4
3	0	1	0	Não está presente, necessita melhorar, não é útil	N/A
4	0	1	1	Não está presente, necessita melhorar, é útil	Q1, Q4
5	1	0	0	Está presente, não necessita melhorar, não é útil	Q2
6	1	0	1	Está presente, não necessita melhorar, é útil	Q3
7	1	1	0	Está presente, necessita melhorar, não é útil	Q2
8	1	1	1	Está presente, necessita melhorar, é útil	Q3

4.2.4 Seleção do contexto

De acordo com Travassos (2002), o contexto pode ser caracterizado conforme quatro dimensões:

- o processo: on-line / off-line;
- os participantes: ferramentas de ETL;
- realidade: o problema real / modelado;
- generalidade: específico / geral.

Nosso estudo supõe o processo *off-line* porque as ferramentas não estão sendo testadas durante todo o tempo da utilização, mas em certo instante. Os participantes são as ferramentas de ETL encontradas na literatura. O estudo é modelado porque as funcionalidades das ferramentas não são caracterizadas durante a resolução do problema real, mas utilizando parâmetros subjetivos (ex. presença, utilidade e necessidade). As funcionalidades do ETL4NoSQL são comparadas com as ferramentas presentes na literatura, então, o contexto possui o caráter específico.

4.2.5 Seleção dos indivíduos

Como participantes para o estudo propõe-se utilizar as ferramentas encontradas na literatura. Assume-se que esses indivíduos estão presentes em diversos estudos realizados e avaliados no meio acadêmico.

Para a escolha das ferramentas utilizadas neste estudo foi levado em consideração a semelhança da finalidade do uso com a ferramenta proposta neste trabalho. Seria conveniente utilizar para o estudo ferramentas que tem o objetivo de auxiliar no desenvolvimento de processos de ETL em bancos de dados NoSQL. Dessa forma, a seleção dos indivíduos do estudo experimental baseou-se nas características das ferramentas de ETL existentes na literatura.

4.2.6 Variáveis

Variável independente: A lista de funcionalidades das ferramentas de ETL encontradas na literatura.

Variáveis dependentes:

1. A similaridade entre as funcionalidades oferecidas pela ferramenta proposta e as funcionalidades encontradas nas ferramentas da literatura.

   Pode receber os valores: Igual, quando todas as funcionalidades tem o valor PMU = { 1, X, X } (métricas 5-8); Diferente, quando todas as funcionalidades tem o valor PMU = { 0, X, X } (métricas 1-4) Similar, quando não se cumprem as condições de "Igual"e "Diferente". O grau de similaridade pode ser avaliado como: { 1, X, X } / { 0, X, X } + { 1, X, X } * 100%

2. A utilidade das funcionalidades similares.

   Mostra a parte útil das funcionalidades oferecidas pela ferramenta proposta: Parte útil: { 1, X, 1 } / { 1, X, X } * 100% Parte inútil: { 1, X, 0 } / { 1, X, X } * 100%

3. A melhoria das funcionalidades similares.

   Mostra a necessidade de melhoria nas funcionalidades oferecidas pela ferramenta proposta: Não necessita melhorar: { 1, 0, X } / { 1, X, X } * 100% Necessita melhorar: { 1, 0, X } / { 1, X, X } * 100%

4.2.7 Análise Qualitativa

Para analisar a informação referente às funcionalidades não oferecidas no ETL4NoSQL, mas que poderiam ser implementadas, propõe-se aplicar a análise qualitativa. Essa análise deve apresentar a lista de funcionalidades presentes nas ferramentas da literatura, que não estão presentes na ferramenta proposta, mas que são consideradas necessárias para facilitar a manipulação de dados. Assim, essa análise deve considerar funcionalidades com valor PMU = 0, X, X (métricas 1-4) e a opção ”É útil” para ”utilidade da funcionalidade”.

4.2.8 Validade

Validade interna: como mencionado na parte "Seleção dos indivíduos"para o estudo propõe-se utilizar ferramentas de ETL presentes na literatura, que são validadas pelo meio acadêmico. Assim, assume-se que elas são representativas para a população de ferramentas de ETL.

Além disso, para redução da influência dos fatores que não são interesse do nosso estudo e, portanto, para aumento da validade interna do estudo supõe-se utilizar dados das ferramentas mais populares da literatura, cuja a validação já tenha passado por diversas avaliações.

Validade de conclusão: para receber os valores da presença, utilidade e melhorias o teste binomial será utilizado. A verificação de hipótese será feita por meio de simples demonstração de presença ou não de funcionalidades nas listas que representam as variáveis independentes.

Validade de construção: esse estudo está caracterizado pela conformidade das funcionalidades listadas na ferramenta proposta com as funcionalidades reais necessárias para a utilização de ferramentas de ETL. As características das ferramentas de ETL presentes na literatura representa a lista de funcionalidades que uma ferramenta de ETL deve apresentar para mostrar o desempenho adequado do ponto de vista da literatura. As funcionalidades que tem o maior relacionamento com as ferramentas de ETL do ponto de vista dos pesquisadores foram escolhidas do conjunto total de funcionalidades das ferramentas de ETL presentes na literatura.

Validade externa: como foi mencionado nas partes "Seleção dos indivíduos"e "Validade interna"os participantes do estudo em geral podem ser considerados representativos para a população da literatura apresentada pela academia. Para avaliação do nível de importância das funcionalidades analisadas foi levada em consideração a frequência que a funcionalidade apareceu nas ferramentas da literatura.

Os materiais utilizados no estudo podem ser considerados representativos e "em tempo"para o problema sob análise, porque compõem-se das funcionalidades de ferramentas de ETL presentes na literatura atual.

4.3 Operação

A etapa de operação ocorre após a etapa de planejamento do estudo experimental. Nela é exercido o monitoramento do experimento para garantir que ele esteja ocorrendo conforme foi planejado (SOUZA et al. (2015)). Nesta seção serão apresentados os questionários do perfil da ferramenta de ETL e o de Funcionalidades.

4.3.1 Questionário do Perfil da Ferramenta de ETL

O quadro 4.3 mostra as questões usadas para definir o perfil das ferramentas utilizadas como indivíduos deste experimento.

Quadro 4.3 Questionário do Perfil da Ferramenta de ETL

Nome da ferramenta de ETL:	
Possui código aberto?	Sim ◯ Não ◯
Possui uma marca reconhecida no mercado?	Sim ◯ Não ◯
Tem como finalidade utilizar bancos de dados NoSQL?	Sim ◯ Não ◯
Possui interface gráfica?	Sim ◯ Não ◯
É programável?	Sim ◯ Não ◯
É integrada?	Sim ◯ Não ◯
Qual o tipo de processamento que a ferramenta executa?	Distribuído ◯ Centralizada ◯ Híbrido ◯
É extensível?	Sim ◯ Não ◯
Para qual finalidade a ferramenta procura auxiliar melhor os processos de ETL?	Modelagem ◯ Desempenho ◯ Ambos ◯

4.3.2 Questionário de Funcionalidades

Sob o ponto de vista das características das ferramentas e considerando a finalidade da ferramenta indicada acima, avalie as colunas correspondentes segundo as escalas abaixo, a presença, utilidade e melhorias quanto às funcionalidades das ferramentas apresentadas nos seus respectivos trabalhos de pesquisa, das funcionalidades listadas no questionário:

Quadro 4.4 Instrumentação para aplicar o questionário

Presença da Funcionalidade (P)	Melhoria da Funcionalidade (M)	Utilidade da Funcionalidade (U)
1. Não está presente 2. Está presente parcialmente 3. Está presente	1. Necessita melhorar 2. Não há necessidade de melhoria 3. Pode melhorar, mas não há necessidade	1. É útil 2. Não é útil 3. É parcialmente útil

N	Funcionalidade	Descrição	Presença			Utilidade			Melhoria		
			1	2	3	1	2	3	1	2	3
	Processo										
1	Suporte à plataforma	Ser independente de plataforma.									
2	Suporte à fonte	Ser capaz de ler diretamente da fonte de dados, independente do seu tipo, saber se é uma fonte RDBMS, arquivo de texto, XML ou NoSQL.									
3	Suporte ao destino	Ser capaz de carregar diretamente os dado no destino, independente do seu tipo, saber se o destino é RDBMS, arquivo de texto, XML ou NoSQL.									
4	Suporte à modelagem	Apoiar na extração de dados de múltiplas fontes, na limpeza dos dados, e na transformação, agregação, reorganização e operações de carga.									
5	Paralelismo	Apoiar as operações de vários segmentos e execução em paralelo, internamente. A ferramenta deve ser capaz de distribuir tarefas entre múltiplos servidores.									
6	Apoio ao nível de debugging	Apoiar o tempo de execução e a limpeza da lógica de transformação. O usuário deve ser capaz de ver os dados antes e depois da transformação.									
7	Programável	Apoiar o agendamento de tarefas de ETL e ter suporte para programação em linha de comandos usando programação externa.									
8	Implementação	Suportar a capacidade de agrupar os objetos ETL e implementá-los em ambiente de teste ou de produção, sem a intervenção de um administrador de ETL.									
9	Reutilização	Apoiar a reutilização dos componentes do framework e da lógica das transformações para evitar a reescrita.									
10	Garantia da qualidade	Ser capaz de estabelecer processos, métricas e avaliações que possibilitem e garantam a qualidade de software.									

Figura 4.1 Questionário de Funcionalidades

4.3.3 Resultado do Estudo

A figura 4.2 apresenta o gráfico da quantidade de presença para cada funcionalidade de acordo com cada ferramenta de ETL. Já a figura 4.3 mostra o nível do uso que as ferramentas fazem para cada funcionalidade e a figura 4.4 indica necessidade de melhoraria das funcionalidades em cada ferramenta.

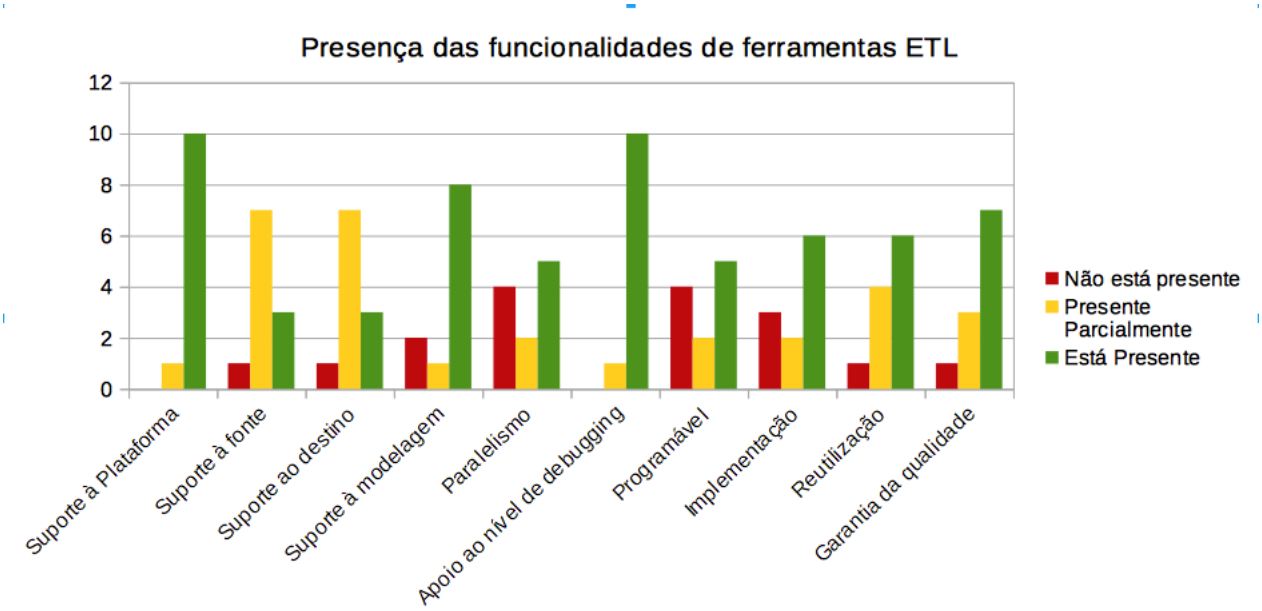

Figura 4.2 Quantidade de Presença para cada funcionalidade

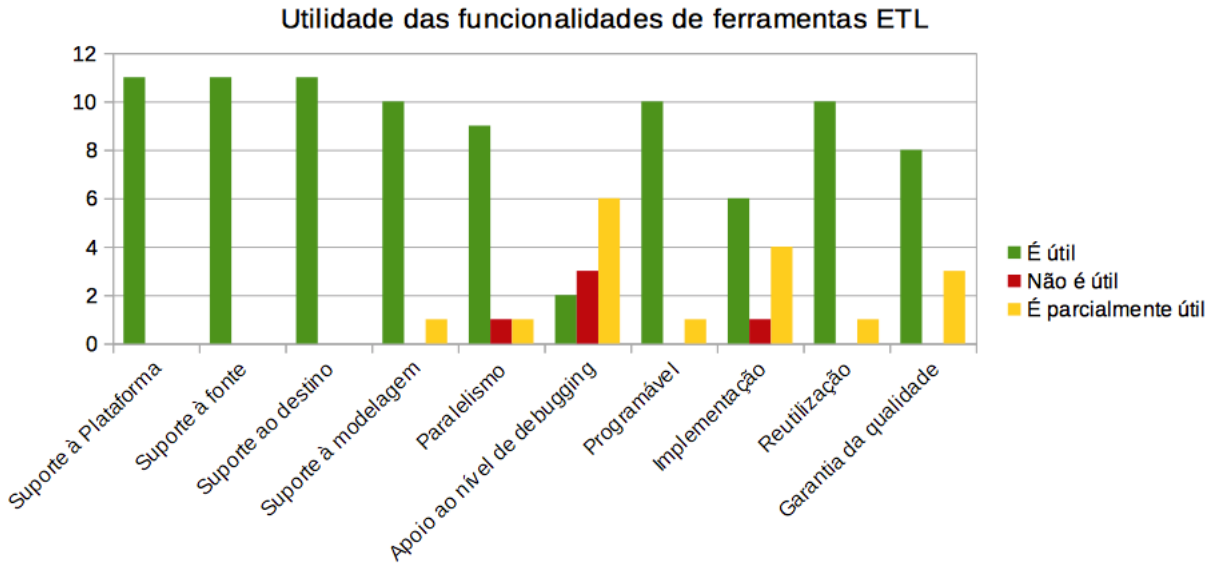

Figura 4.3 Níveis de utilidade para cada funcionalidade

O perfil dos participantes é apresentado no quadro 4.5, bem como a legenda para leitura é exibida no quadro 4.6.

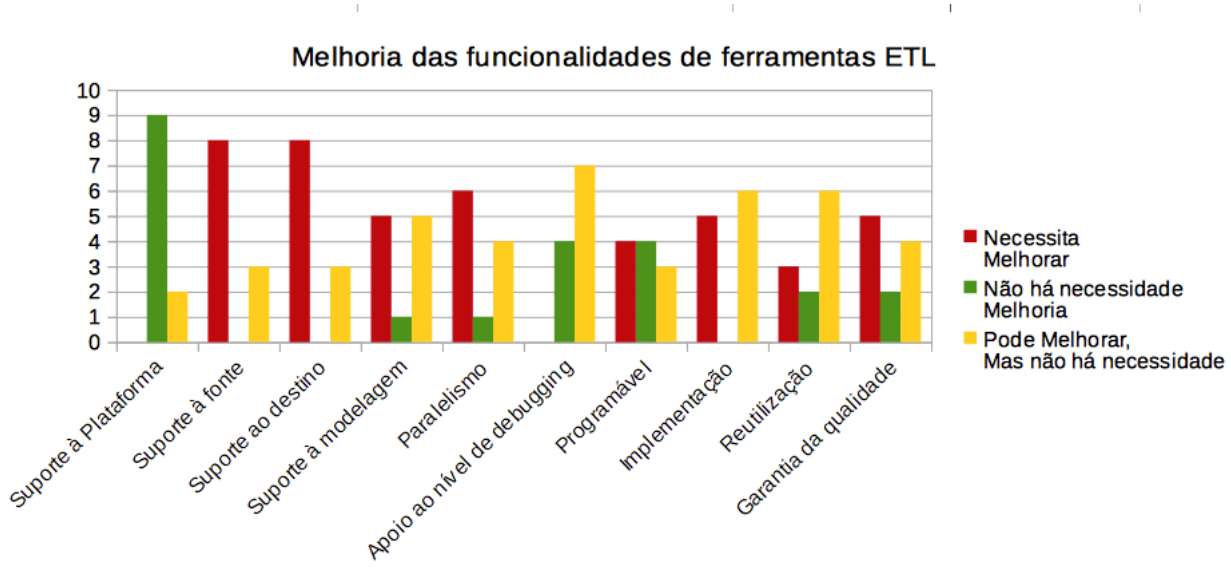

Figura 4.4 Necessidade de melhoria para cada funcionalidade

Das onze ferramentas de ETL apresentadas neste estudo dez possuem código aberto, sete não possui uma marca reconhecida no mercado, nenhuma foi desenvolvida para suportar BDs NoSQL, seis possuem uma GUI, apenas quatro não são programáveis, duas não são integradas, nove delas oferece alguma alternativa para o processamento distribuído, um pouco mais da metade são extensíveis e a finalidade da maioria é modelagem tentando aliar ao desempenho.

Quadro 4.5 Resultado do Perfil dos participantes

Ferramenta	Código Fonte	Marca de mercado	Para NoSQL	GUI	Programável	Integrada	Processamento	Extensível	Finalidade
PygramETL	1	2	2	2	1	1	2	1	2
ARKTOSII	1	2	2	1	2	1	3	2	1
Big-ETL	1	2	3	2	1	2	1	1	3
ETLMR	1	2	2	2	1	1	1	1	2
CloudETL	1	2	3	2	1	1	1	1	2
P-ETL	1	2	3	1	2	1	3	2	3
FramETL	1	2	2	2	1	1	2	1	1
Talend Studio	1	1	3	1	1	1	3	2	3
Pentaho Kettle	1	1	3	1	2	1	3	2	1
CloverETL	1	1	3	1	1	2	3	2	3
Oracle (ODI)	2	1	3	1	2	1	3	2	3

Quadro 4.6 Legenda

Código Fonte	Marca de mercado	Para NoSQL	GUI	Programável	Integrada	Processamento	Extensível	Finalidade
1-Aberto	1-Reconhecido	1-Sim	1-Possui	1-Sim	1-Sim	1-Distribuído	1-Sim	1-Modelagem
2-Fechado	2-Não reconhecido	2-Não	2-Não possui	2-Não	2-Não	2-Centralizado	2-Não	2-Desempenho
		3-Possui, mas não é o foco				3-Híbrido		3-Ambos

4.4 Análise e interpretação dos resultados

Nesta seção é apresentada a análise e interpretação dos resultados conforme as regras estatísticas escolhidas para serem aplicadas neste estudo.

4.4.1 Estatística descritiva

As medidas de tendência central como os valores "Presença", "Melhoria"e "Utilidade"são da escala ordinal, por isso é possível definir as métricas de "moda"e "mediana". As métricas de presença estão no quadro 4.7, as métricas de utilidade são apresentadas no quadro 4.9 e as métricas de melhoria podem ser vistas no quadro 4.8.

Quadro 4.7 Estatística Descritiva da Presença de Funcionalidades

Presença										
Funcionalidades	1	2	3	4	5	6	7	8	9	10
Mediana	3	2	2	3	2	3	2	3	3	3
Moda	3	2	2	3	3	3	3	3	3	3

Quadro 4.8 Estatística Descritiva da Melhoria de Funcionalidades

Melhoria										
Funcionalidades	1	2	3	4	5	6	7	8	9	10
Mediana	2	1	1	2	1	3	2	3	3	2
Moda	2	1	1	{1,3}	1	3	{1,2}	3	3	1

Quadro 4.9 Estatística Descritiva da Utilidade de Funcionalidades

Utilidade										
Funcionalidades	1	2	3	4	5	6	7	8	9	10
Mediana	1	1	1	1	1	3	1	1	1	1
Moda	1	1	1	1	1	3	1	1	1	1

Considerando as respostas recebidas durante o estudo e utilizando os resultados de estatística descritiva podemos dividir as perguntas em 4 grupos. O primeiro grupo são as funcionalidades que estão presente, mas são parcialmente úteis; já o segundo grupo são as funcionalidades presentes e úteis; o terceiro grupo consiste na presença da funcionalidade com necessidade de melhoria; e finalmente, o quarto grupo que representa as funcionalidades que estão presentes e não necessitam de melhoria. Todos os grupos podem ser analisados nos quadros 4.10, 4.11, 4.12 e 4.13.

Os valores na tabela significam P - presente:não presente; M - necessita melhoria:não necessita melhoria; U - útil:inútil.

4.4.2 Aplicação do teste estatístico

Para cada funcionalidade foi aplicado o teste de associação Qui-quadrado, com o objetivo de verificar se existe associação entre a ferramenta e a presença da sua funcionalidade. O teste do qui-quadrado pode ser usado em pesquisas de amostras independentes com a variável de resposta qualitativa (categórica) (BARBETTA (2012)). O quadro 4.14 mostra os resultados da quantidade de funcionalidades presentes nas ferramentas do estudo.

Quadro 4.10 Funcionalidade presente e parcialmente útil

N.	Funcionalidade	P	M	U	Característica
6	Apoio ao nível de debugging	10:1	0:11	8:3	- A funcionalidade existe na maioria das ferramentas, mas não dão enfoque à ela. - É possível comparar os dados da fonte e destino mesmo sem *debugging*. - Pode ser útil para evitar o trabalho de fazer a comparação da fonte com o destino, além da possibilidade de obter uma resposta em tempo de execução.

Quadro 4.11 Funcionalidade presente e é útil

N	Funcionalidade	P	M	U	Características
2	Suporte à fonte	10:1	8:3	11:0	- As ferramentas oferecem algum suporte em relação à leitura e carga da fonte e destino dos dados, porém apenas lidam com SGBDs relacionais. - Deve haver um enfoque nas novas soluções de SGBDs como os SGBDs NoSQL. - Algumas ferramentas não oferecem a opção para utilizar linha de comando, que é considerado importante para ter maior liberdade na implementação dos processos de ETL. -Algumas ferramentas, não dão importância ao processamento em paralelo, porém em se tratando de grandes volumes de dados, o processamento em paralelo é fundamental para possibilitar a execução de processos de ETL.
3	Suporte ao destino	10:1	8:3	11:0	
5	Paralelismo	7:4	6:5	10:1	
7	Programável	7:4	4:7	11:0	

Quadro 4.12 Funcionalidade presente e necessita melhorar

N	Funcionalidade	P	M	U	Características
4	Suporte à modelagem	9:2	5:6	11:0	- A maioria das ferramentas dão suporte à modelagem dos processos de ETL, porém algumas focam apenas no desempenho necessitando melhoria no processo de modelagem. - Poucas ferramentas dão importância para garantir qualidade aos processos de ETL.
10	Garantia de qualidade	10:1	5:6	11:0	

Dessa forma, calculamos o valor esperado para cada funcionalidade através da fórmula de frequências esperadas:

Quadro 4.13 Funcionalidade presente e não necessita melhoria

N	Funcionalidade	P	M	U	Características
1	Suporte à plataforma	11:0	2:9	11:0	- As ferramentas demonstraram ser independentes de plataforma por terem características como código compilável e serem programáveis. - Criar processos de ETL demonstrou ser complexo, necessitando de algum tipo de experiência na área. Dessa forma, não é de fundamental importância que haja suporte à implementação para usuários não administradores. - A maioria das ferramentas, por serem de código aberto e programáveis, oferecem a possibilidade de reutilização de código ou algumas até mesmo oferecem possibilidade de reusar os processos de ETL já criados.
8	Implementação	8:3	5:6	10:1	
9	Reutilização	10:1	3:8	11:0	

Quadro 4.14 Quadro de resultado do valor observado de existência da funcionalidade

Funcionalidade	1	2	3	4	5	6	7	8	9	10	Total
Existe	10	3	3	8	5	10	5	6	6	7	63
Não existe	1	8	8	3	6	1	6	5	5	4	47
Total	11	11	11	11	11	11	11	11	11	11	110

```
E = (total da linha) x (total da coluna) / (total geral)

E = 63 x 11 / 110 = 6,3

E = 47 x 11 / 110 = 4,7
```

Para a frequência das funcionalidades existentes o valor esperado é E = 6,3 e para frequência não existentes o valor esperado é E = 4,7.

A fórmula estatística para o teste qui-quadrado é definida por:

$$\chi^2 = \sum \frac{(O-E)\hat{}2}{E}$$

Onde:

A = soma se estende a todas as células da tabela de contingência;

O = representa a frequência observada na célula; e

E = representa a frequência esperada na célula.

Somando os valores das parcelas do χ^2, temos:

Quadro 4.15 Quadro do resultado de χ^2

Funcionalidade	1	2	3	4	5	6	7	8	9	10
Existe	2,17	1,72	1,72	0,45	0,26	2,17	0,26	0,014	0,014	0,077
Não existe	2,91	2,31	2,31	0,61	0,36	2,91	0,36	0,02	0,02	0,1

$$\chi^2 = 20,765$$
$$com$$
$$gl = (l-1).(c-1) = (2-1)(10-1) = 9$$

Pela tabela de distribuição qui-quadrado, verificamos que a probabilidade de significância ρ é inferior a 0,01. Então, para qualquer nível usual de significância ($\alpha = 0,05$), o teste detecta associação entre as funcionalidades e as ferramentas (pois, $\rho < \alpha$). Em outras palavras, o teste qui-quadrado mostrou que as ferramentas em estudo são diferentes quanto às suas funcionalidades.

4.4.2.1 Análise quantitativa

Para verificar a similaridade entre as funcionalidades existentes nas ferramentas de ETL na literatura e as funcionalidades do ETL4NoSQL é necessário calcular o valor de variável dependente.

1. A mediana dos conjuntos de funcionalidades existentes nas ferramentas de ETL da literatura e as funcionalidades do ETL4NoSQL não podem ser consideradas iguais (a quantidade de funcionalidades com valor PMU { 1, X, X } 7 < 10), nem diferentes (a quantidade de funcionalidades com valor PMU { 0, X, X } 3 < 10). Assim, precisamos calcular o grau de similaridade segundo a fórmula da variável dependente 1:

   grau de similaridade = 7 / 10 * 100% = 70%

2. Para verificar a utilidade das funcionalidades similares, é necessário calcular o valor da variável dependente 2:

   parte útil das funcionalidades similares = 6 / 7 * 100% =85,71%

   parte inútil das funcionalidades similares = 1 / 7 * 100% = 14,29%

3. Para verificar a necessidade de melhoria das funcionalidades similares é necessário calcular o valor de variável dependente 3:

   parte que necessita de melhoria das funcionalidades similares = 3 / 7 * 100% = 42,85%

   parte que não necessita de melhoria das funcionalidades similares = 4 / 7 * 100% = 57,15%

4.4.3 Análise qualitativa

Para verificar as funcionalidades que não estão presentes no ETL4NoSQL neste trabalho de pesquisa, mas que são consideradas úteis para melhoria dos processos de ETL, foi feita a análise qualitativa apresentada no quadro 4.16.

Quadro 4.16 Funcionalidades não oferecidas pelo ETL4NoSQL

Funcionalidade	6	8	10
Quantidade total de ferramentas que não possuem ou possuem parcialmente a funcionalidade	4	5	4
Quantidade de ferramentas que considera útil	2	6	9

Assim, podemos concluir que as funcionalidades 8 (implementação) e 10 (garantia de qualidade) despertam interesse à serem implantadas nas ferramentas de ETL. Isso pode ser explicado pelas características e aplicabilidade que as funcionalidades oferecem. As funcionalidades de implementação e garantia de qualidade, provavelmente, não estão presentes em geral ou são oferecidas de forma parcial, pois não são o foco de boa parte das ferramentas de ETL, em razão de seus usuários possuírem um certo grau de conhecimento a respeito de programação e criação de processos de ETL. Porém, essas funcionalidades melhorariam o desempenho geral dos processos de ETL facilitando o uso por usuários que não possuem conhecimento em linguagens de programação.

4.4.4 Verificação das hipóteses

Para verificar o hipótese nula (H0) precisamos responder a questão Q1 (Existem funcionalidades listadas pelas ferramentas pesquisadas que não estão presentes no ETL4NoSQL?) utilizando a métrica M2. O resultado do teste Qui-quadrado mostra que 3 das 10 funcionalidades analisadas não podem ser consideradas como oferecidas pela ferramenta proposta neste trabalho. Assim, podemos concluir que a lista de funcionalidades oferecidas pelas ferramentas de ETL da literatura é diferente das funcionalidades oferecidas pelo ETL4NoSQL, ou seja, "As funcionalidades oferecidas pelo ETL4NoSQL são diferentes das funcionalidades oferecidas pelas ferramentas da literatura"(hipótese alternativa H1).

Podemos dizer ainda que existe uma funcionalidade considerada pouco útil (funcionalidades que são pouco utilizadas nos processos de ETL) para as ferramentas de ETL analisadas (hipótese alternativa H2), pois respondendo a questão Q2 utilizando a métrica M5, o resultado do teste Qui-quadrado mostra que a funcionalidade 6 (Apoio ao nível de *debugging*) é considerada pouco útil para a realização de processos de ETL.

Por conseguinte, para funcionalidades presentes no ETL4NoSQL e nas ferramentas da literatura, que são úteis e necessitam melhorias (hipótese H3), respondendo a questão Q3 com a métrica M8 o resultado de Qui-quadrado mostra que 3 funcionalidades necessitam melhoria (2 - suporte à fonte, 3 - suporte ao destino, 5 - paralelismo) e outras 3 podem melhorar, mas não tem

necessidade (6 - apoio ao nível de *debugging*, 8 - implementação, 9 - reutilização).

Finalmente, para realizar conclusões relevantes sobre a hipótese alternativa (H4) foi feita a análise qualitativa. Os resultados da análise apresentaram 2 funcionalidades (8 - Implementação e 10 - Garantia de qualidade) não presentes no ETL4NoSQL e que são úteis. Assim, podemos concluir que existem funcionalidades não presentes no ETL4NoSQL, mas que são úteis e poderiam ser implementadas futuramente como melhoria para a ferramenta.

4.5 Considerações Finais

Este capítulo apresentou o estudo experimental de *software* com o objetivo principal de definir se o *framework* proposto por esta dissertação é adequado para auxiliar no desenvolvimento de processos de ETL em BDs NoSQL. Os resultados do experimento, tendo como base as ferramentas de ETL encontradas na literatura, indicaram que o *framework* proposto possui uma quantidade significativa de grau de similaridade com as ferramentas encontradas na literatura, porém o estudo conclui também que existem funcionalidades úteis e não implementadas que poderiam ser melhoradas futuramente.

O capítulo seguinte apresentará a avaliação da nossa proposta onde estenderemos o ETL4NoSQL para criar duas aplicações de ETL utilizando SGBDs NoSQL. O ojetivo dessa avaliação é ilustrar a reusabilidade e flexibilidade do ETL4NoSQL e sua aplicabilidade em BDs NoSQL.

5

Avaliação

Este capítulo fornece aplicações exemplos baseadas no ETL4NoSQL em dois domínios de naturezas distintas, utilizando SGBDs NoSQL para os dados de entrada. O desenvolvimento dessas aplicações pretende ilustrar a reusabilidade e flexibilidade do ETL4NoSQL em aplicações de ETL, que apresentam requisitos distintos de inserção e transformação de dados, e assim, avaliar o trabalho proposto neste documento.

Nas seções seguintes, são apresentadas as aplicações utilizadas como exemplo para a avaliação do nosso trabalho. Demonstraremos a implementação de duas aplicações de ETL estendidas do ETL4NoSQL, aplicando processos para extração, transformação e carga de dados modelando-os no esquema estrela, utilizando os dados de entrada armazenados no SBGD MongoDB e SGBD Cassandra.

5.1 Aplicações Baseadas no ETL4NoSQL

Para avaliar o trabalho descrito neste documento, implementamos duas aplicações a partir do ETL4NoSQL. Elas são baseadas em dados sintéticos que podem ser encontrados em MONGODB (2017) e ZHENG (2017).

Nossa motivação para escolha das duas aplicações foi para demonstrar a flexibilidade da ferramenta proposta ao lidar com SGBDs NoSQL de diferentes paradigmas, além dos SGBDs escolhidos serem bastante utilizados atualmente. Outra motivação foi facilitar a modelagem e carga desses dados no esquema estrela em DW.

Para a implementação, instanciamos a interface de leitura de dados (IDataMgr) e criamos as operações através da interface de operação (IOpMgr) de forma a modelar os dados, usando a interface de modelagem (IModelMgr), no esquema estrela. Finalmente, pudemos processar os dados por meio da interface de processamento (IProcMgr) e carregá-los em um arquivo de saída com o formato JSON.

Dessa forma, o uso das interfaces do *framework* ETL4NoSQL auxilia o projetista de ETL na modelagem dos processos a partir de seu reuso, além de permitir adequar os processos de acordo com o seu domínio por meio da flexibilização das interfaces dos componentes do ETL4NoSQL.

Na seção seguinte, exibiremos o desenvolvimento da aplicação ETL4NoSQLMongoStar, que consiste em uma aplicação de ETL baseada no ETL4NoSQL utilizando o SGBD MongoDB como fonte de dados e o esquema estrela como modelo de dados para saída dos dados.

5.2 Aplicação ETL4NoSQLMongoStar

Para o desenvolvimento deste exemplo de aplicação utilizamos o SGBD MongoDB e uma base de dados que armazena a avaliação dos clientes de vários tipos de restaurantes. A seguir apresentamos a estrutura da base de dados utilizada nesta aplicação.

```
{
        "address": {
                "building": "1007",
                "coord": [ -73.856077, 40.848447 ],
                "street": "Morris␣Park␣Ave",
                "zipcode": "10462"
        },
        "borough": "Bronx",
        "cuisine": "Bakery",
        "grades": [
                { "date": { "$date": 1393804800000 }, "grade": "A", "score": 2 },
                { "date": { "$date": 1378857600000 }, "grade": "A", "score": 6 },
                { "date": { "$date": 1358985600000 }, "grade": "A", "score": 10 },
                { "date": { "$date": 1322006400000 }, "grade": "A", "score": 9 },
                { "date": { "$date": 1299715200000 }, "grade": "B", "score": 14 }
        ],
        "name": "Morris␣Park␣Bake␣Shop",
        "restaurant_id": "30075445"
}
```

Assim sendo, na figura 5.1 definimos o modelo multidimensional no esquema estrela para a esta estrutura de dados.

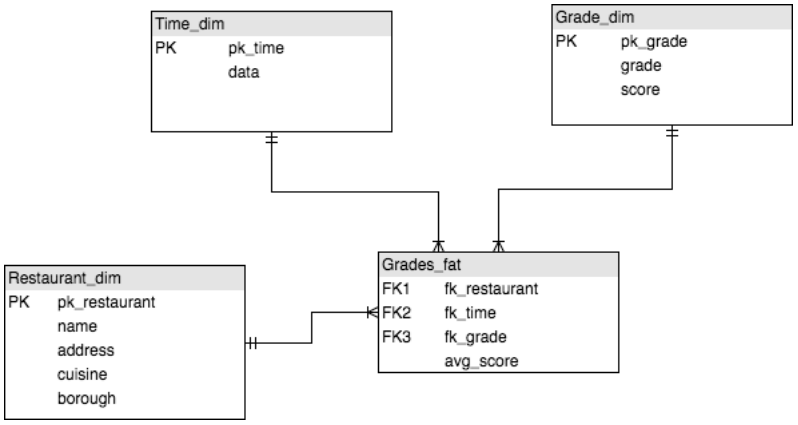

Figura 5.1 Modelo Multidimensional da aplicação ETL4NoSQLMongoStar

A partir do *framework* ETL4NoSQL, pudemos estender uma nova aplicação denominada ETL4NoSQLMongoStar para desenvolver e executar os processos de ETL de forma a atender ao modelo multidimensional especificado para o exemplo utilizado.

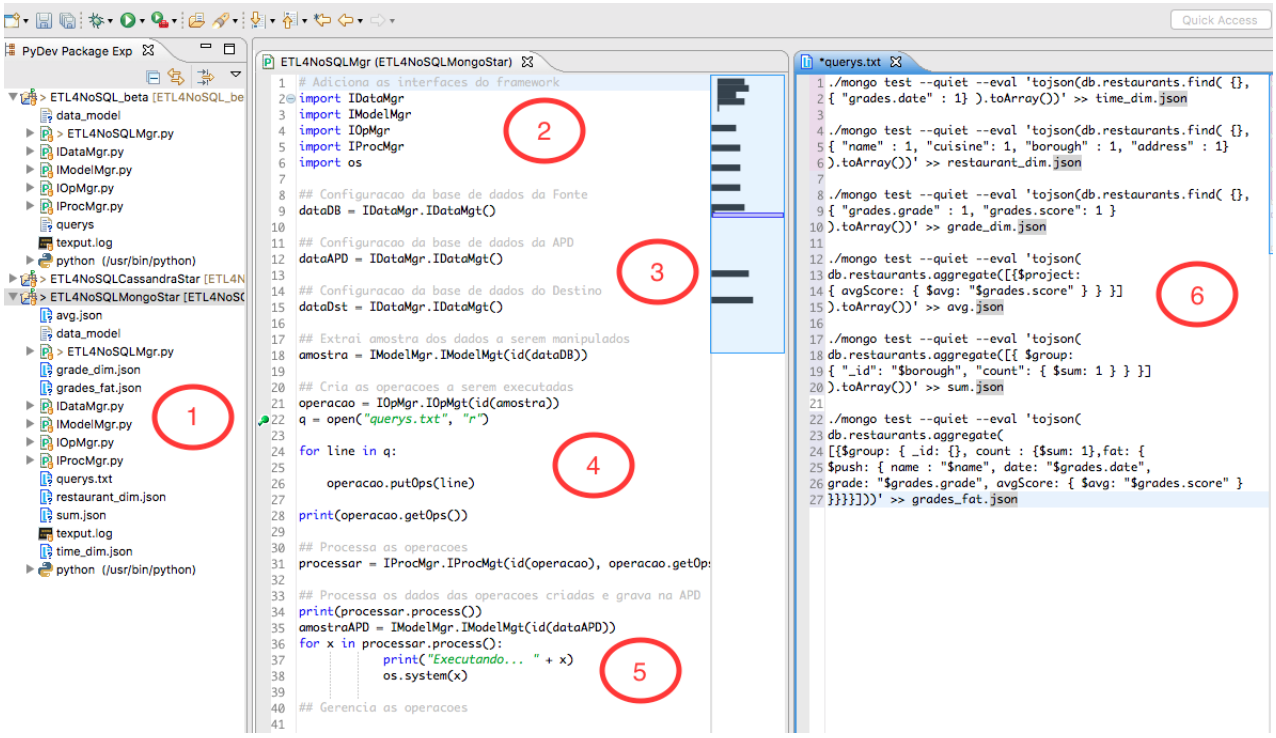

Figura 5.2 Tela da aplicação ETL4NoSQLMongoStar

Na figura 5.2 podemos ver a aplicação ETL4NoSQLMongoStar, no destaque 1 temos a árvore de arquivos da aplicação, incluindo os arquivos de saída, criados ao final da execução dos processos de ETL. Já no destaque 2 é importada as interfaces dos componentes do *framework*. No destaque 3 é feita a leitura das bases de dados envolvidas nos processos, no destaque 4 é realizada a inserção das operações a serem efetuadas e no destaque 5 os processos são executados. O destaque 6 apresenta as operações a serem efetivadas pelo *framework* para criar o modelo multidimensional do ETL4NoSQLMongoStar e sua saída de dados em arquivos JSON.

A próxima seção disserta a respeito de outra aplicação de ETL estendida a partir do ETL4NoSQL, denominada ETL4NoSQLCassandraStar. Ela utiliza como fonte de dados o SGBD Cassandra e o esquema de dados estrela é o escolhido como modelo de dados para saída dos dados.

5.3 Aplicação ETL4NoSQLCassandraStar

A aplicação exemplo ETL4NoSQLCassandraStar foi criada a partir do *framework* ETL4NoSQL proposto neste trabalho, utilizamos o SGBD Cassandra e a base de dados de localizações de táxis, de acordo com sua latitude e longitude em um determinado momento. A estrutura de dados da base de dados origem pode ser vista no código a seguir.

```
CREATE TABLE taxi.localizacoes (
        taxi_id int,
        date_time text,
        longitude text,
        latitude text
PRIMARY KEY (taxi_id));
```

Com isso, determinamos o modelo multidimensional seguindo o esquema estrela para esta estrutura de dados mostrada na figura 5.3.

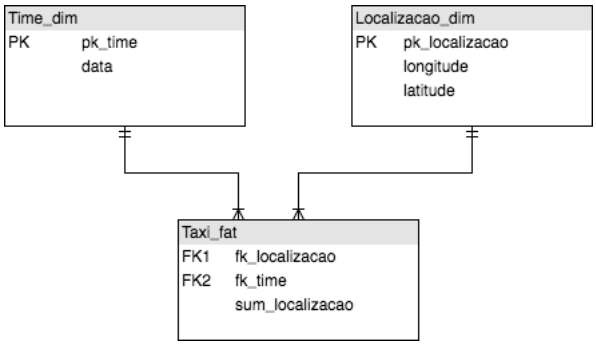

Figura 5.3 Modelo Multidimensional da aplicação ETL4NoSQLCassandraStar

Assim, utilizando a aplicação estendida do ETL4NoSQL, pudemos executar os processos de ETL para atender ao modelo multidimensional.

Na figura 5.4 podemos ver a aplicação ETL4NoSQLCassandraStar. Da mesma forma que a aplicação ETL4NoSQLMongoStar, no destaque 1 temos a árvore de arquivos da aplicação, incluindo os arquivos de saída, gerados ao final da execução dos processos de ETL. Já no destaque 2 é importada as interfaces dos componentes do *framework*. No destaque 3 é feita a leitura das bases de dados envolvidas nos processos, no destaque 4 é realizada a inserção das operações a serem executadas e no destaque 5 os processos são efetivados. O destaque 6 apresenta as operações a serem efetuadas pelo *framework* para criar o modelo multidimensional do ETL4NoSQLCassandraStar e sua saída de dados em arquivos JSON.

Por conseguinte, é importante ressaltar que para transformar os dados no esquema estrela utilizando aplicações estendidas do ETL4NoSQL, bastou apenas adicionar os parâmetros

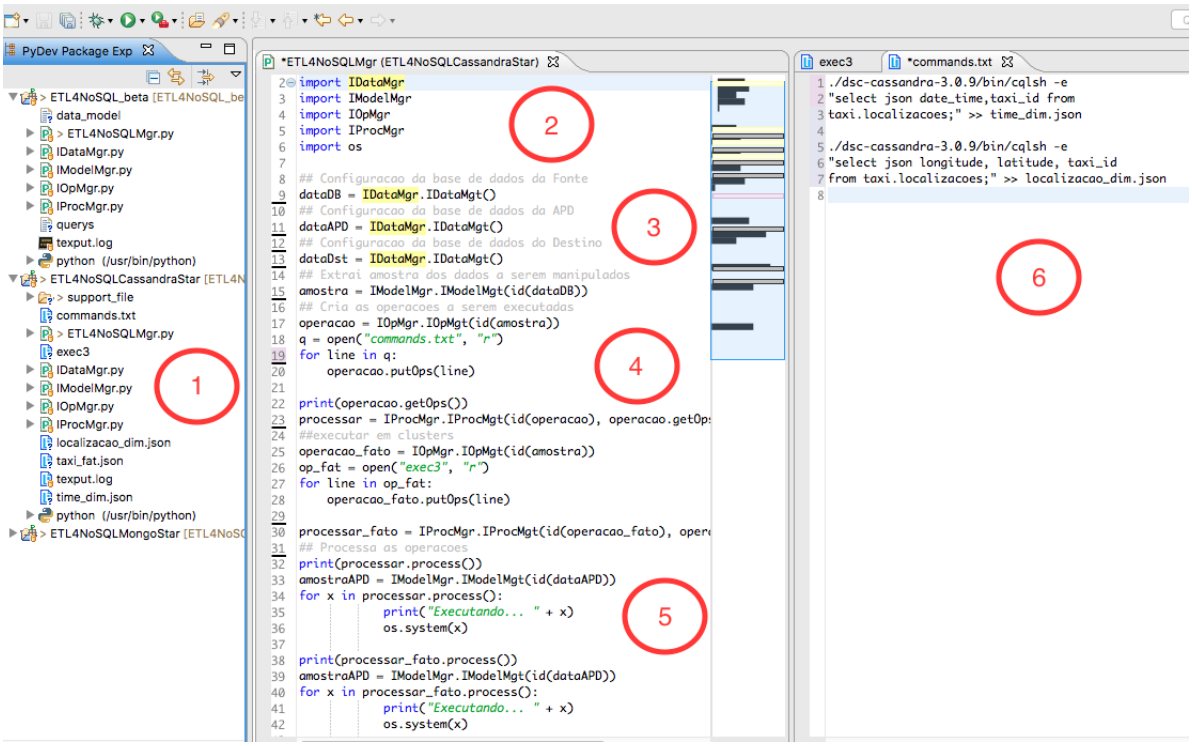

Figura 5.4 Tela da aplicação ETL4NoSQLCassandraStar

de consulta de cada SGBD NoSQL para executar os processos de busca, junção e escrita, demonstrando que o *framework* proposto é programável, no sentido de possibilitar a programação dos seus parâmetros. Ele é reusável, pois permite que seus componentes sejam reutilizados por várias aplicações. E finalmente, flexível, dado que é possível estendê-lo para atender vários domínios de aplicação.

5.4 Considerações Finais

Este capítulo evidenciou duas aplicações de naturezas distintas estendidas do ETL4NoSQL, avaliando suas características de reusabilidade e flexibilidade. Uma aplicação foi baseada no SGBD NoSQL Mongo e a outra no SGBD NoSQL Cassandra. Utilizamos o ETL4NoSQLCassandraStar e o ETL4NoSQLMongoStar para desenvolver e executar os processos. Ao final da execução dos processos pudemos gerar um arquivo de saída em formato comum, denominado JSON.

No capítulo posterior, esta pesquisa é finalizada, expondo as principais contribuições, discussões sobre dificuldades e as ameaças do trabalho de pesquisa, os resultados e trabalhos futuros a partir do ETL4NoSQL.

6

Conclusão

A existência de uma demanda crescente para integrar os dados modelados em vários tipos de estruturas em um repositório unificado, além de muitas empresas encontrarem dificuldades para lidar com as ferramentas ETL disponíveis no mercado motivaram esta pesquisa, cuja apresentou o ETL4NoSQL. Este capítulo resume o trabalho exposto nesta dissertação apresentando suas contribuições relevantes, seus desafios e dificuldades, seus resultados e trabalhos futuros a partir do ETL4NoSQL.

Portanto, este trabalho propôs um *framework* programável para desenvolvimento de aplicações de ETL que possibilita a extração, transformação e carga de dados armazenados em BDs NoSQL. O *framework* possui um ambiente integrado para a leitura e escrita dos dados, além da modelagem e execução distribuída ou centralizada dos processos de ETL. O componente de gerenciamento de dados do *framework* executa os processos de ETL em BDs NoSQL. Esse componente possibilita a leitura e manipulação de dados em BDs NoSQL, e também o armazenamento desses dados em bases deste tipo, oferecendo uma alternativa de modelo não relacional para a construção de DWs.

6.1 Principais Contribuições

Uma das principais contribuições do nosso trabalho é o *framework* ETL4NoSQL, que engloba os requisitos de *software* para ferramentas de ETL, a modelagem do domínio, através de seus três modelos: modelo conceitual, de casos de uso e de comportamento; a modelagem da especificação, da arquitetura e especificação dos componentes, identificação das interfaces de sistemas e de negócio, bem como a interação entre os componentes do ETL4NoSQL. O ETL4NoSQL pode ser utilizado para o desenvolvimento de novas aplicações de ETL. As especificações de seus componentes podem ser reutilizados para facilitar a especialização e instanciação de novas aplicações de ETL.

Outra contribuição deste trabalho é o estudo experimental de *software* baseado nas ferramentas de ETL encontradas na literatura por esta pesquisa. O estudo teve como objetivo definir se o ETL4NoSQL é adequado para auxiliar no desenvolvimento de processos de ETL. Os resultados do estudo mostraram que o ETL4NoSQL possui um grau de similaridade de 70% em relação com as outras 11 ferramentas estudadas nesta pesquisa, e que dessa similaridade 85,71% das funcionalidades são consideradas úteis para desenvolver processos de ETL.

E por último, oferecemos duas aplicações de domínios distintos baseadas em SGBDs NoSQL, estendidas do ETL4NoSQL como forma de avaliar a flexibilidade e reusabilidade do *framework* proposto neste trabalho. Os SGBDs utilizados para o desenvolvimento das aplicações foram o MongoDB e o Cassandra. Criamos o ETL4NoSQLCassandraStar e o ETL4NoSQLMongoStar para desenvolver e executar os processos, e ao final da execução dos processos pudemos gerar um arquivo de saída em formato comum, denominado JSON. Além disso, ressaltamos que para transformar os dados no esquema estrela utilizando as aplicações estendidas do ETL4NoSQL, bastou apenas adicionar os parâmetros de consulta de cada SGBD NoSQL para executar os processos de busca, junção e escrita, demonstrando que o *framework* proposto é programável, no sentido de possibilitar a programação dos seus parâmetros; reusável, pois permite que seus componentes sejam reutilizados por várias aplicações; e finalmente, flexível, dado que é possível estendê-lo para atender vários domínios de aplicação.

Dessa forma, podemos concluir que o objetivo da presente proposta, de especificar um *framework* programável, flexível e integrado para extração, transformação e carga dos dados de bancos de dados NoSQL foi atingido de forma efetiva e satisfatória, no sentido de facilitar e flexibilizar a atividade de desenvolvimento de novas ferramentas de ETL.

6.2 Discussão

Na seção 2.2 apresentamos os trabalhos correlatos a esta pesquisa, nesta seção foi demonstrada as ferramentas de ETL mais citadas pela literatura. Podemos observar que a maioria das ferramentas pesquisadas foca no desempenho ao lidar com grandes volumes de dados e BDs NoSQL, porém nenhuma delas apresentou uma solução alternativa para modelagem de

BDs NoSQL ficando a cargo do desenvolvedor encontrar meios para esquematizar os dados. Algumas delas, como a P-ETL, oferece alternativas para leituras de arquivos textuais, mas não dão enfoque aos BDs NoSQL. Outras ferramentas como o ETLMR, CloudETL e BigETL utilizam processamento paralelo e distribuído para facilitar a execução de processos de ETL em grandes volumes de dados, contudo não lidam com a modelagem e leitura dos dados de SGBDs não relacionais.

Assim, como alternativa a isso, sugerimos o ETL4NoSQL que consiste em um *framework* baseado em componentes, programável, flexível e integrado. Os seus componentes podem ser reutilizados, por meio de suas interfaces e especializados para atender as especificidades de cada domínio de aplicação.

6.3 Trabalhos Futuros

Como trabalhos futuros indicamos a execução de testes com dados reais. Adicionalmente, testar o desempenho com a execução dos processos de forma distribuída, centralizada e paralela. E finalmente, verificar a eficiência comparada aos outros métodos existentes para o desenvolvimento de aplicações de ETL.

Referências

AWAD, M. M. I.; ABDULLAH, M. S.; ALI, A. B. M. Extending ETL framework using service oriented architecture. **Procedia Computer Science**, [S.l.], v.3, p.110 – 114, 2011. World Conference on Information Technology.

BALA, M. P-ETL: parallel-etl based on the mapreduce paradigm. **IEEE**, [S.l.], Nov. 2014.

BALA, M.; BOUSSAID, O.; ALIMAZIGHI, Z. Big-ETL: extracting-transforming-loading approach for big data. **Int'l Conf. Par. and Dist. Proc. Tech. and Appl.**, [S.l.], 2015.

BARBETTA, P. A. **Estatística aplicada às Ciências Sociais**. [S.l.]: Editora da UFSC, 2012.

CHAUDHURI, S.; DAYAL, U. An overview of data warehousing and OLAP technology. **SIGMODRecord,v.26,n.1,p.65–74**, [S.l.], 1997.

CHEESMAN, J.; DANIELS, J. **UML components, a simple process for specifying component-based software**. [S.l.]: Addison-Wesley Professional, 2001.

CHEVALIER, M. et al. Implementing Multidimensional Data Warehouses into NoSQL. **Proceedings of the 17th International Conference on Enterprise Information Systems, p.172-183, April 27-30**, [S.l.], 2015.

CLEMENTS, P. et al. **Documenting Software Architectures**: views and beyond. [S.l.]: Pearson Education, 2002.

DARMONT, J. et al. An architecture framework for complex data warehouses. **7th International Conference on Enterprise Information Systems (ICEIS'05), Miami, USA, pages 370–373**, [S.l.], 2005.

DEAN, J.; GHEMAWAT, S. Mapreduce: simplified data processing on large clusters. **Communications of the ACM**, [S.l.], p.107–113, 2008.

DEAN, J.; HAIHONG, E.; DU, J. Survey on nosql database. **EEE**, [S.l.], p.363–366, 2011.

FAYAD, M. E.; SCHMIDT, D. C. Object-Oriented Application Frameworks. **Communications of the ACM 40 (10): 32–38**, [S.l.], 1997.

FAYAD, M. E.; SCHMIDT, D. C.; E., R. E. J. R. Building application frameworks: object-oriented foundations of framework design. **John Wiley and Sons, New York, USA, pp. 3–29**, [S.l.], 1999.

PRATES, R. (Ed.). **NoSQL Essencial Um guia conciso para o mundo emergente de Persistência Poliglota**. [S.l.]: Novatec, 2013. Primeira Edição.

INFORMATION, E. T. URL: http://etl-tools.info/en/pentaho/kettle-etl.htm.

INMON, W. H. **Building the Data Warehouse**. [S.l.]: John Wiley and Sons, 2002.

INTEGRATION, P. D. URL: http://www.pentaho.com/product/data-integration.

KAUR, K. Modeling and querying data in NoSQL databases. **Big Data, 2013 IEEE International Conference on**, [S.l.], 2013.

ELLIOT, R. (Ed.). **The Data Warehouse ETL ToolKit**. [S.l.]: Robert Ipsen, 2004.

ELLIOT, R. (Ed.). **The Data Warehouse Toolkit**. [S.l.]: Robert Ipsen, 2002. Second Edition.

LIMA, C. de; MELLO, R. S. Um Estudo sobre Modelagem Lógica para Bancos de Dados NoSQL. **XI Escola Regional de Banco de Dados**, [S.l.], 2015.

LIU, X.; THOMSEN, C.; PEDERSEN, T. B. ETLMR: a highly scalable dimensional etl framework based on mapreduce. **DB Tech Reports**, [S.l.], Aug. 2011.

LIU, X.; THOMSEN, C.; PEDERSEN, T. B. CloudETL: scalable dimensional etl for hive. **DB Tech Reports**, [S.l.], July 2013.

MALI, N.; BOJEWAR, S. A Survey of ETL Tools. **International Journal of Computer Techniques**, [S.l.], Oct. 2015.

MAZANEC, M.; MACEK, O. On general-purpose textual modeling languages. **In Proceedings of DATESO'12. CEUR-WS.org, Praha, Czech Republic**, [S.l.], p.1–12, 2012.

MONGODB, I. URL: https://raw.githubusercontent.com/mongodb/docs-assets/primer-dataset/primer-dataset.json.

MUÑOZ, L.; MAZÓN, J.-N.; TRUJILLO, J. Automatic Generation of ETL Processes from Conceptual Models. In: ACM TWELFTH INTERNATIONAL WORKSHOP ON DATA WAREHOUSING AND OLAP, New York, NY, USA. **Proceedings...** ACM, 2009. p.33–40. (DOLAP '09).

NASHOLM, P. **Extracting Data From NoSQL Databases**. 2012. Tese (Doutorado em Ciência da Computação) — Chalmers University of Technology, SE-412 96 Goteborg Sweeden.

PREE, W.; SIKORA, H. **Design Patterns for Object Oriented Software Developiment.** [S.l.: s.n.], 1997.

WILEY, J.; SONS (Ed.). **Business Intelligence Success Factors**: tools for aligning your business in the global economy. EUA: Inc, 2009.

RUSSOM, P.; MADSEN, M. Data Integration Tools: comparison and market analysis. **TDWI Technology Market Report**, [S.l.], 2007.

SALEM, R.; BOUSSAïD, O.; DARMONT, J. An Active XML-based Framework for Integrating Complex Data. **27th Annual ACM Symposium On Applied Computing (SAC 12), Riva del Garda (Trento), Italy, March 2012; ACM, New York**, [S.l.], 2012.

SAMETINGER, J. **Software Engineering with Reusable Componets**. [S.l.]: Springer, 1997.

SCHIMIDT, D. C.; GOKHALE, A.; NATARAJAN, B. Leveraging Application Frameworks. **ACM Queue, v. 2**, [S.l.], 2004.

SHAW, M.; GARLAN, D. **Software Architecture**: perspectives on an emerging discipline. [S.l.]: Prentice Hall, 1996. Prentice Hall Ordering Information.

SILVA, L. M. M. **ETL na era do Big Data**. 2016. Dissertação (Mestrado em Ciência da Computação) — Técnico Lisboa.

SILVA, M. S. da. **Um Framework para Desenvolvimento de Sistemas ETL**. 2012. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal de Pernambuco.

SOMMERVILLE, I. **Engenharia de Software**. [S.l.]: Pearson, 2013.

SOUZA GIMENES, I. M. de; HUZITA, E. H. M. **Desenvolvimento Baseado em Componentes**: conceitos e técnicas. [S.l.]: Editora Ciência Moderna Ltda., 2005.

SOUZA, I. E. et al. TESE - Um Sistema de Informação para Gerenciamento de Projetos Experimentais em Engenharia de Software. **XI Brazilian Symposium on Information System, Goiânia, May 26-29**, [S.l.], 2015.

TALIGENT, I. **Building Object-Oriented Frameworks.** [S.l.: s.n.], 1994.

THOMSEN, C.; PEDERSEN, T. B. pygrametl: a powerful programming framework for extract-transform-load programmers. **DB Tech Reports**, [S.l.], 2009.

TRAVASSOS, G. H.; GUROV, D.; AMARAL, E. A. G. Introdução à Engenharia de Software Experimental. **Programa de Engenharia de Sistemas e Computação COPPE/UFRJ**, [S.l.], 2002.

VASSILIADIS, P. et al. A generic and customizable frameworkfor the design of ETL scenarios. **Information Systems - Special issue: The 15th international conference on advanced information systems engineering 30 (7): 492–525**, [S.l.], 2005.

WARMER, J.; KLEPPE, A. **The object constraint language, precise modeling with UML.** [S.l.]: Addison-Wesley, 1998.

WOHLIN, C. et al. Experimentation in Software Engineering: an introduction. **Kluwer Academic Publishers, USA**, [S.l.], 2000.

ZHENG, Y. URL: https://onedrive.live.com/?authkey=