

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 tem 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, modelos de programação como o MapReduce (MR) introduzido pela Google (DEAN; GHEMAWAT (2004)), 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 de armazenamento 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. Assim, surgem novos paradigmas, tais como *Cloud Computing* (Computação em Nuvem) 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 grandes volumes de dados, apresentados de forma variada, em modelos não relacionais em um ambiente de suporte à decisão. Contudo, devido a complexidade desses dados, diferentes 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 devido à sua longa curva de aprendizagem. 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) THOMSEN; PEDERSEN (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, outros aspectos como a variedade dos dados e a falta de uma alternativa para substituir o SQL dos modelos relacionais.

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 membros não especializados em ETL, 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 específicas 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 correlatas à este trabalho 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 ambiente utilizamos o SGBD MongoDB com dados sintéticos de ranking de restaurantes, e o segundo, 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. 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.
- Aplicações de ETL4NoSQL: 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 áreas de conhecimento essenciais para esta pesquisa. Dessa forma, os princípios básicos desses conceitos são apresentados nesta seção.

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, assim, 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: Perfil dos dados, Alterar a captura de dados 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 das fontes de dados; mescla os dados de duas ou mais fontes de dados; cria dimensões; e aplica métricas. Esta etapa possui 5 subsistemas: Sistema de limpeza de dados, Acompanhamento de erros nos eventos, Deduplicação, Conformidade dos dados e Criação de Dimensão de auditoria.

c) Entrega ou Carga: Estrutura e carrega os dados em DWs. Esta etapa possui 13 subsistemas: *Slowly Changing Dimension (SCD)*, Gerador de Chave Substituto, Gerenciador de Hierarquia, Gerenciador de Dimensões Especiais, Construtores de Tabelas Fato, Chave Substituta de *Pipeline*, Construtor de Tabela de Multivalores, Manipulador de Dados de chegada Tardia, Gerenciador de Dimensão, Fornecedor de Tabela de Fatos, Construtor de Agregados, Construtor do Cubo *OLAP*, Gerenciador de Propagação de dados.

d) Gerenciamento: Gerencia os sistemas e processos relacionados ao ambiente ETL de forma coerente. Esta etapa possui 13 subsistemas: *Job Scheduler*, Sistema de Backup, Recuperação e *Restart*, Controle de Versão, Versão de Migração, Monitor de fluxo de trabalho, Classificação, Linhagem e Dependência, Problema de Escalação, Paralelismo e *Pipelining*, Segurança, Gerenciador de Conformidade, Repositório de Metadados.

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 Sistemas de Banco de dados Orientados à Documentos

Sistemas de 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 o paradigma relacional, as coleções são como tabelas e os documentos são 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, os documentos podem ter campos completamente diferentes (KAUR (2013), FOWLER; SADALAGE (2013)).

Existem vários sistemas gerenciadores de banco de dados orientados à documentos

disponíveis no mercado e os mais utilizados são MongoDB, CouchDB e o RavenDB (KAUR (2013)).

2.1.2.2 Sistemas de Banco de dados Famílias de Colunas

Sistemas de banco de dados baseados em famílias de colunas abrangem três áreas: número enorme de colunas, a natureza esparsa dos dados e frequentes mudanças no esquema de dados. Os dados mantidos 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 que o próprio usuário possa gerar suas próprias consultas (consulta *ad-hoc*) e consultas dinâmicas se tornem mais eficientes (KAUR (2013), FOWLER; SADALAGE (2013), INMON (2002)).

A maioria dos bancos de dados baseados em famílias de colunas são também compatíveis com o *framework* MapReduce, este acelera o processamento de enormes volumes de dados pela distribuição do problema em um grande número de sistemas. Os SGBD de família de colunas de código aberto mais populares são Hypertable, HBase e Cassandra (KAUR (2013)).

2.1.2.3 Sistemas de Banco de dados Baseado em Grafos

Sistemas de bancos de dados baseados em grafos são representados por uma estrutura de rede contendo nós e arestas, onde as arestas interligam os nós representando a relação entre eles. Comparando com os bancos de dados relacionais, o nó corresponde à tabela, a propriedade do nó à um atributo e as arestas são as relações 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 baseados 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 Sistemas de Banco de dados Chave-Valor

Em Sistemas de 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 ela. 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 SGBD do tipo 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. Eles 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 autocontido, identificável, funcional, possuir uma interface, ser documentado e ter uma condição de reuso.

Para CHEESMAN; DANIELS (2001), o processo de desenvolvimento baseado em componentes consiste 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: (i) a etapa de identificação dos componentes, onde produz uma especificação e arquitetura inicial; (ii) interação entre componentes onde descobre-se as operações necessárias e aloca-se responsabilidades; e finalmente, (iii) a etapa de especificação de componentes, que cria uma

especificação precisa das operações, interfaces e dos 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)).

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)).

Frameworks podem ser considerados aglomerados de *softwares*, que 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 SCHMIDT; GOKHALE; NATARAJAN (2004) define o framework como sendo um conjunto de artefatos de *software* (como classes, objetos e componentes) que colaboram entre si para fornecer uma arquitetura reusável.

Os frameworks possibilitam a reusabilidade de projeto, bem como ao reuso 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 possuem componentes de comunicação e troca de informações. Finalmente, os frameworks de aplicações corporativas estão relacionados com domínios de aplicação específicos, como sistemas financeiros. Eles incorporam conhecimentos sobre domínios de aplicações e apoiam o desenvolvimento para o usuário final por meio desses conhecimentos.

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 reconhecidos como 'inversão de controle' (SCHIMIDT; GOKHALE; NATARAJAN (2004)). A Figura 2.1 expressa a 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 a uma funcionalidade fornecida ao usuário. A funcionalidade específica de aplicação responde ao evento de forma adequada. Por exemplo, um framework pode ser um método que responde ao toque em uma tecla a partir da ação do usuário. 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)).

No que se refere à arquitetura de software, 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. Por fim, deve-se tomar a decisão 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)).

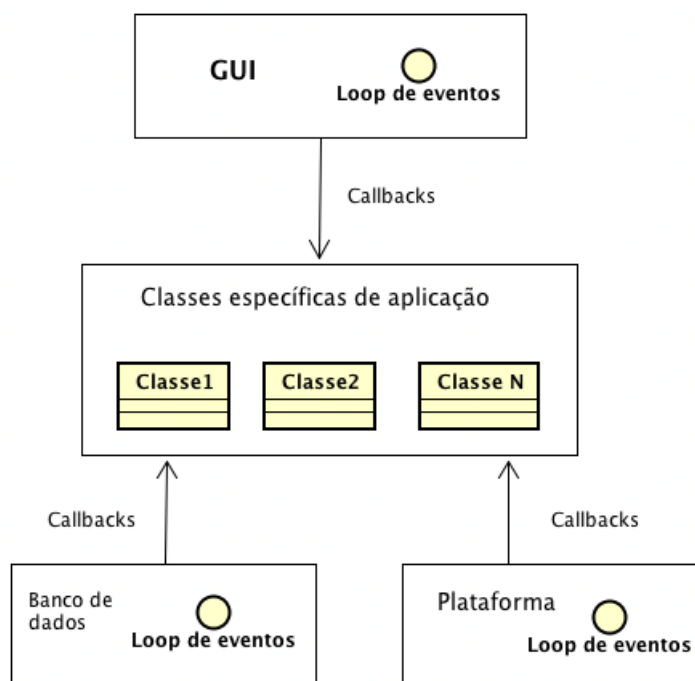


Figura 2.1 Inversão de controle em *framework*. (Adaptado de SOMMERVILLE (2013))

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 se importa 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 de aplicação de ETL.

2.1.4 Estudo Experimental de Software

Esta dissertação descreve 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 pesquisadas. Os questionários utilizados para a coleta de dados são baseados nos requisitos encontrados na literatura pesquisada para ferramentas de ETL (FERREIRA et al. (2010), KAREL; GOULDE (2007)).

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, e pode ser utilizado quando se quer entender o processo, produto de *software* e ambiente. Ele mede e analisa, e verifica as hipóteses do modelo ou teoria. Já o método de engenharia, observa as soluções existentes, é uma abordagem baseada na melhoria evolutiva, e 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.

Dessa forma, para o nosso estudo, consideramos as funcionalidades das ferramentas de ETL pesquisadas neste trabalho (FERREIRA et al. (2010), KAREL; GOULDE (2007)).

2.2 TRABALHOS CORRELATOS

Esta seção apresenta os principais frameworks correlatos encontrados na literatura pesquisada, bem como os descreve 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)).

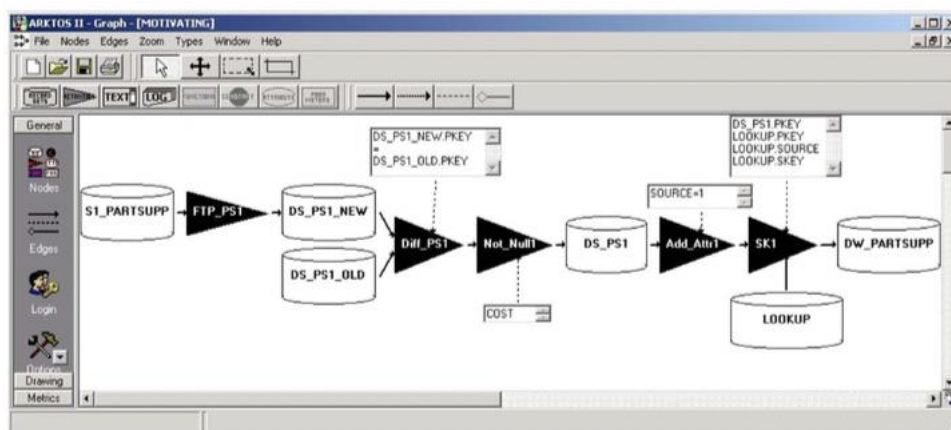


Figura 2.2 Exemplo da Ferramenta ARKTOS II em uso (Adaptado de VASSILIADIS et al. (2005))

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 sistemas de ETL, demonstrando como se deve iniciar um projeto de ETL. O propósito da ferramenta é facilitar a carga dos dados no DW gerenciado por banco de dados relacionais (SGBD). Focando nos SGBD relacionais como destino, torna o desenvolvimento mais simples. As fontes de dados podem ser de qualquer tipo, porém o destino poderá ser apenas em bancos de dados relacionais.

Ao usar pygrametl, o programador faz o código que controla o fluxo de ETL, a extração dos sistemas de origem, as transformações dos dados de origem e a carga dos dados transformados em um BD relacional. Para o controle de fluxo, extração e carga, pygrametl oferece componentes que tornam fácil para o desenvolvedor criar mais componentes. Para as transformações, o programador se beneficia pelo uso de uma linguagem de programação de propósito geral. O pygrametl é implementado como um módulo em Python. Para o futuro, os criadores de pygrametl planejam criar uma GUI para obter etapas de conexão visual de modo que as atualizações no código sejam visíveis na GUI e no ambiente de codificação. Eles também planejam investigar como fornecer uma maneira eficiente e simples de criar e executar sistemas de ETL em ambientes DW paralelos ou distribuídos (THOMSEN; PEDERSEN (2009)).

2.2.3 ETLMR

ETLMR é um framework de ETL que utiliza *MapReduce* para atingir escalabilidade. Ele considera 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 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 de redução e escrita dos resultados (LIU; THOMSEN; PEDERSEN (2011)).

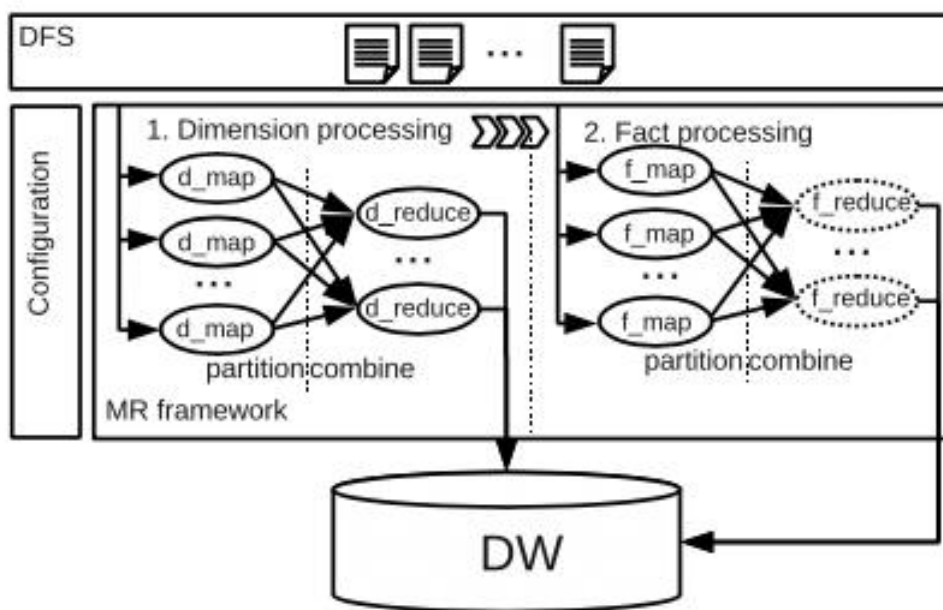


Figura 2.3 Fluxo de ETL do framework MapReduce (Adaptado de LIU; THOMSEN; PEDERSEN (2011))

O ETLMR possui inúmeras contribuições, ele permite construir dimensões processando os esquemas estrela, snowflake, SCDs e dimensões com grandes volumes de dados. Pelo fato de 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 às 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, ele não inclui funcionalidades de auxílio na modelagem de 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 ilustra 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)).

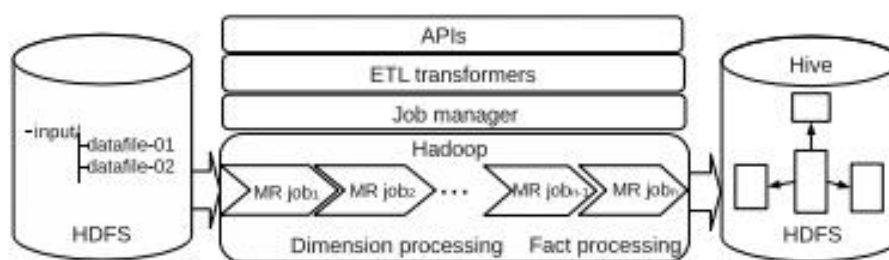


Figura 2.4 Arquitetura do CloudETL (Adaptado de LIU; THOMSEN; PEDERSEN (2013))

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, ele não oferece nenhum suporte para modelagem de processos de ETL ficando esta tarefa 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. Ela fornecerá o formato e a estrutura dos dados provindos da fonte de dados. O primeiro passo da fase de extração é localizar as fontes de dados. O arquivo base do P-ETL é no formato "csv" (*Comma Separated Values*). Ele converte a fonte de dados para o formato "csv" permitindo a entrada dos dados em vários formatos. Para acelerar a carga dos dados da fonte de dados mantida pelo 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 uma lista de funções para transformação, e cada função deve ser especificamente configurada (condições, expressões, entradas, etc.). Assim, as funções são executadas na ordem em que forem inseridas. 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 separados com base no formato "csv" (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 dados 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)).

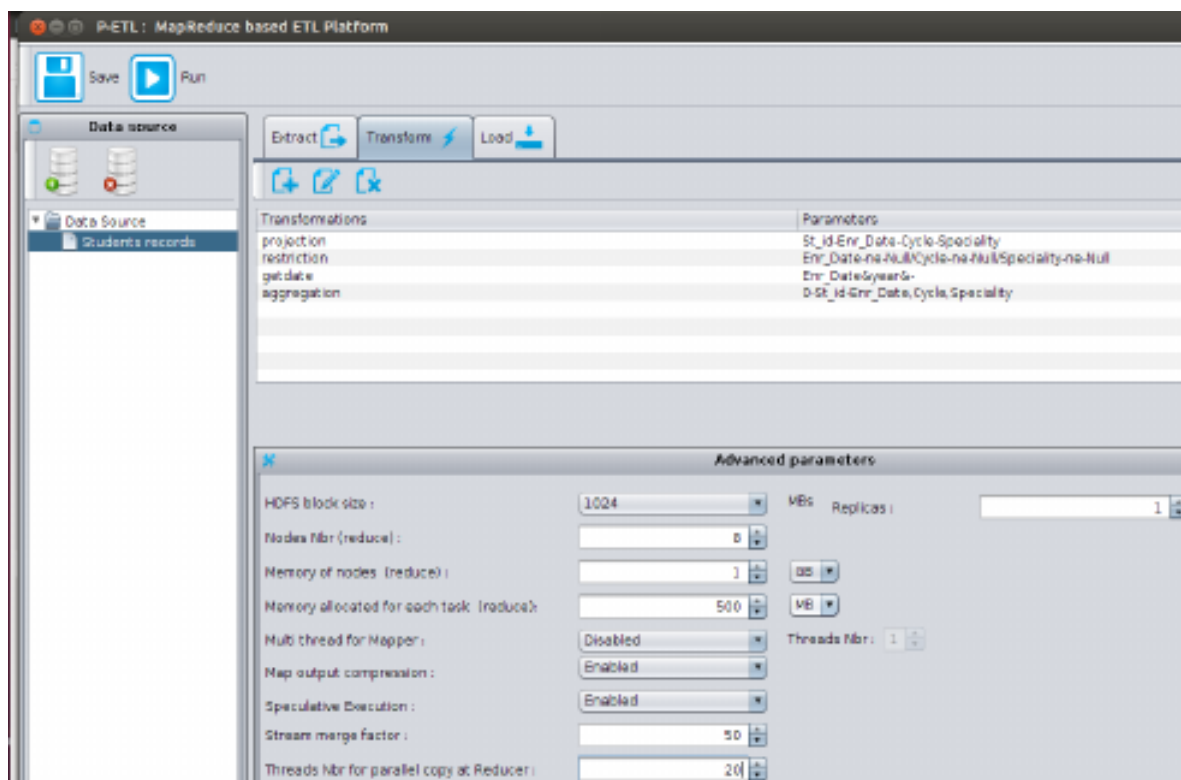


Figura 2.5 Interface de configuração do P-ETL (Adaptado de BALA (2014))

2.2.6 Big-ETL

BALA; BOUSSAID; ALIMAZIGHI (2015) propõem em seu trabalho, uma abordagem chamada Big-ETL, a qual define que funcionalidades de ETL podem ser executadas em *cluster* utilizando o paradigma MapReduce (MR). 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, o desempenho é melhorado. Para testar a abordagem proposta, o autor utilizou o P-ETL para permitir a execução dos processos de ETL em paralelo, 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 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.

2.2.7.1 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 (SANTOS LIRA FILHO (2013); INFORMATION (2017)).

2.2.7.2 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 (SANTOS LIRA FILHO (2013); MALI; BOJEWAR (2015), INFORMATION (2017)).

2.2.7.3 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.

2.2.7.4 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.

2.2.7.5 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 SGBD NoSQL, pois o foco de sua ferramenta não era lidar com esses tipos de SGBD.

2.2.8 Comparativo das Ferramentas de ETL

Apresentamos uma análise comparativa entre o ETL4NoSQL e os trabalhos correlatos discutidos na Seção 2.2. A Figura 2.6, mostra um quadro comparativo com os critérios utilizados para a comparação das ferramentas de ETL: 1) se a ferramenta foi desenvolvida para uso de BD NoSQL; 2) uso de linguagem de programação de propósito geral para a criação e execução dos processos de ETL; 3) se utiliza interfaces de programação integradas para a criação e execução dos processos de ETL; 4) tipo de processamento utilizado para execução dos processos de ETL; 5) se possui arquitetura de reuso e pontos de flexibilidade para permitir a extensão e customização; 6) se foi desenvolvida com a finalidade de auxiliar o desempenho

ou a modelagem dos processos de ETL; 7) tipo de código fonte; 8) se a ferramenta foi citada em várias referências na literatura; e 9) se possui GUI.

Ferramenta	Possui foco em NoSQL	Programável	Integrada	Processamento	Extensível	Finalidade	Código Fonte	Marca de Mercado	Possui GUI
ETL4NoSQL	Sim	Sim	Sim	Híbrido	Sim	Ambos	Aberto	Não Reconhecida	Não
PygramETL	Não	Sim	Sim	Centralizado	Sim	Desempenho	Aberto	Não Reconhecida	Não
ARKTOSII	Não	Não	Sim	Híbrido	Não	Modelagem	Aberto	Não Reconhecida	Sim
Big-ETL	Não	Sim	Não	Distribuído	Sim	Ambos	Aberto	Não Reconhecida	Não
ETLMR	Não	Sim	Sim	Distribuído	Sim	Desempenho	Aberto	Não reconhecida	Não
CloudETL	Não	Sim	Sim	Distribuído	Sim	Desempenho	Aberto	Não reconhecida	Não
P-ETL	Não	Não	Sim	Híbrido	Não	Ambos	Aberto	Não reconhecida	Sim
FramETL	Não	Sim	Sim	Centralizado	Sim	Modelagem	Aberto	Não reconhecida	Não
Talend Studio	Não	Sim	Sim	Híbrido	Não	Ambos	Aberto	Reconhecida	Sim
Pentaho Kettle	Não	Não	Sim	Híbrido	Não	Modelagem	Aberto	Reconhecida	Sim
CloverETL	Não	Sim	Não	Híbrido	Não	Ambos	Aberto	Reconhecida	Sim
Oracle (ODI)	Não	Não	Sim	Híbrido	Não	Ambos	Fechado	Reconhecida	Sim

Figura 2.6 Quadro comparativo entre frameworks para desenvolvimento de sistemas de ETL

2.3 CONSIDERAÇÕES FINAIS

Este capítulo discorreu a respeito dos principais assuntos abordados nesta dissertação, bem como sobre 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 SGBD 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 que é proposto nesta dissertação para 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 e reutilização dos processos.

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 de ETL4NoSQL.

3.1 REQUISITOS DE SOFTWARE DO ETL4NOSQL

O ETL4NoSQL é um *framework* que tem como principal objetivo auxiliar na criação de aplicações de ETL ao se utilizar principalmente BDs NoSQL.

Os SGBDs relacionais utilizam uma linguagem de gerenciamento de dados padrão conhecida por SQL (Structure Query Language) (FOWLER; SADALAGE (2013)), porém os SGBD NoSQL não possuem uma linguagem em comum, como os SGBD relacionais. Cada SGBD NoSQL possui sua própria linguagem de gerenciamento de dados (FOWLER; SADALAGE (2013)). Por isso, ETL4NoSQL possui um componente de *software* capaz de fazer a leitura diretamente da fonte de dados, e um outro componente para carregar esses dados diretamente em seu armazenamento de destino, independente do tipo da fonte de dados, seja ela um arquivo texto, um arquivo XML, SGBD relacional, SGBD NoSQL, entre outros.

Outra característica importante 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, ETL4NoSQL possui um componente que permite a leitura dos dados de diversos SGBD NoSQL, de arquivos textuais, além dos SGBD relacionais. Outro componente para execução os mecanismos de ETL, bem como componentes para o gerenciamento das execuções 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 são apresentados os principais requisitos elencados do ETL4NoSQL. Definimos como importante, as prioridades que são imprescindíveis para o desenvolvimento e funcionamento do framework proposto, e consideramos como desejável as funcionalidades que aprimoram o uso do framework ETL4NoSQL, 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 de dados	Ser capaz de ler diretamente da fonte de dados, independente do seu tipo, podendo a fonte ser SGBD relacional, arquivo de texto, XML ou NoSQL.	Importante
Suporte ao destino	Ser capaz de carregar diretamente os dados no destino, independente do seu tipo, podendo o destino ser um 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 na execução de operações de carga.	Importante
Paralelismo	Apoiar as operações de vários segmentos e a 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 de ETL4NoSQL e da lógica de transformações para evitar a reescrita.	Importante
Apoio ao nível de <i>debugging</i>	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	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 <i>software</i> .	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.

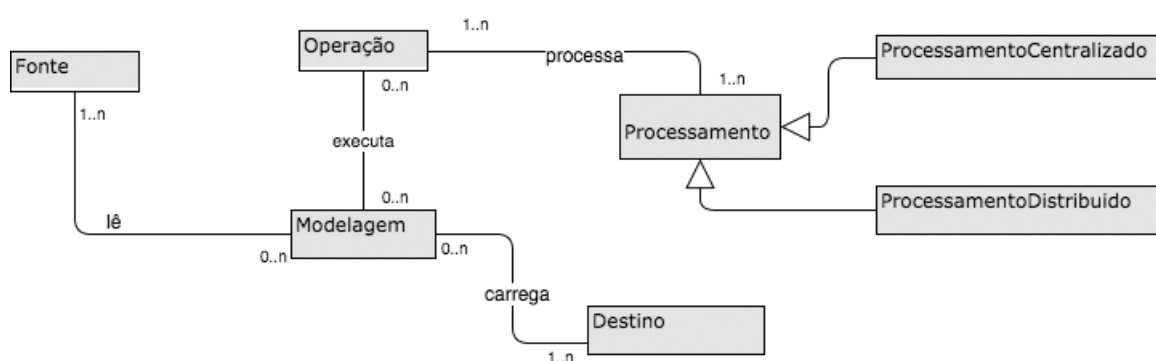


Figura 3.1 Modelo Conceitual de ETL4NoSQL

A entidade Modelagem faz a leitura de uma ou mais Fontes de dados, e executa nenhuma ou muitas Operações. As Operações por sua vez são processadas por um ou mais Processamentos, que podem ser Processamentos Centralizados ou Processamentos Distribuídos. Por fim, a Modelagem carrega o resultado das Operações processadas em um ou mais Destinos.

Na seção seguinte, é apresentado o modelo de casos de uso de ETL4NoSQL.

3.2.2 Modelo de Casos de Uso

Os casos de uso expressam as funcionalidades fundamentais ao desenvolvimento e uso do ETL4NoSQL. Eles oferecem ao programador, a visão do que é imprescindível ao

implementar e determinar as interfaces de sistema e operações do framework proposto.

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 seção seguinte apresenta o modelo comportamental do *framework* proposto neste trabalho.

Quadro 3.2 Modelo de Casos de Uso do ETL4NoSQL

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

3.2.3 Modelo Comportamental

Ao construir o modelo comportamental foi possível identificar os conceitos que caracterizam 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 necessário 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 de operações que não puderam 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 referida execução.

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 estágios, 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 são reconhecidas por meio do modelo conceitual, cada entidade identificada no modelo conceitual é identificada como uma interface de negócio. As interfaces de sistema e operações

são identificadas a partir dos casos de uso.

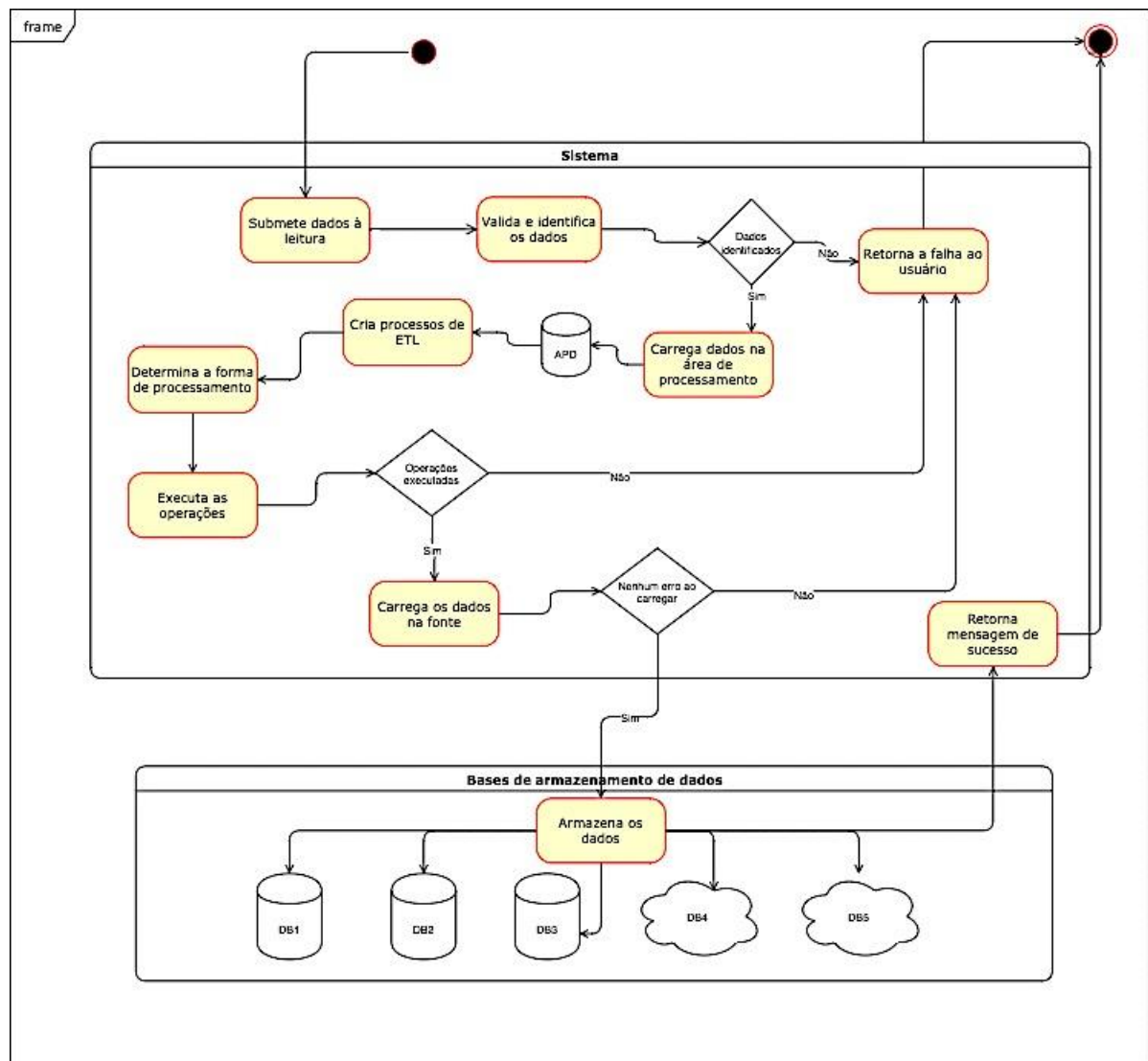


Figura 3.2 Diagrama de Estado do ETL4NoSQL

3.3.2 Interfaces de Sistemas

As interfaces de sistemas do ETL4NoSQL, identificadas por meio do modelo de casos de uso apresentados no quadro 3.2, são: IReadData, IWriteData, IModelData, IExeOp e IProcOp. Para especificar as interfaces foi utilizado OCL (Object Constraint Language) (WARMER; KLEPPE (1998)). O Algoritmo 1, apresenta a conexão à base de dados utilizando

o componente de leitura *IReadData*, esse componente também permite a identificação do tipo de estrutura de dados que foi conectada. A partir da especificação dos casos de uso, foram identificadas as operações da interface do componente de leitura: Interface *IReadData* (*Connection(connect)* e *structureType()*). O Algoritmo 2, demonstra a escrita dos dados no armazenamento de destino, por meio do componente de escrita *IWriteData*. As operações identificadas para este componente foram: Interface *IWriteData* (*connection(connect)*, *structureType()* e *allowWrite()*). O Algoritmo 3, expõe por meio do componente de modelagem, a criação das operações de ETL. É por meio deste componente que a modelagem das operações e transformação dos dados, conectados pelo componente de leitura, são realizadas. As operações identificadas para o componente de modelagem foram: Interface *IModelData* (*readData(CodFonte)* e *createOp(CodOperação, data)*). O Algoritmo 4, expõe o gerenciamento das operações criadas pelo *IModelData*. Este componente pode excluir, alterar as operações e ordená-las para o processamento. As operações identificadas para o componente de gerenciamento de operações foram: Interface *IExeOp* (*modelOperation(CodModelagem)* e *operationManagement()*). O Algoritmo 5, mostra o processamento das operações de ETL utilizando o componente *IProcOp*. As operações identificadas para este componente 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, que possuem existência independente. Para 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. Nela, podemos perceber a interação entre os componentes. Porém essa interação é realizada pela interface específica de cada componente, que serão detalhadas posteriormente na Seção 3.4. As interfaces apresentadas na Figura 3.3 são: *IDataMgt*, *IModelMgt*, *IOpMgt* e *IProcMgt*. A interface *IDataMgt*, lida com interação dos dados da fonte e área de processamento de dados (APD). Já a interface *IModelMgt*, interage com a modelagem dos dados e criação das operações. A interface *IOpMgt*, prepara e gerencia as operações para execução. Finalmente, a interface *IProcMgt* interage com o processamento das operações e a escrita dos dados.

Algoritmo 1: *IReadData* – Componente de leitura

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)

Algoritmo 2: *IWriteData* – Componente de escrita

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)

Algoritmo 3: *IModelData* – Componente de Criação de processos de ETL

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)

Algoritmo 4: *IExeOp* – Componente de gerenciamento das operações de ETL

context System :: IExeOp (model : Modelagem; operation: Operacao, processing: Processamento))

pre:

Modelagem.allInstances@includes(Modelagem)
 Modelagem.operacoes = existem
 Operacao.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 operações escalonadas e estão prontas para
 o processamento)

Algoritmo 5: *IProcOp* – Componente de processamento das operações de ETL

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)

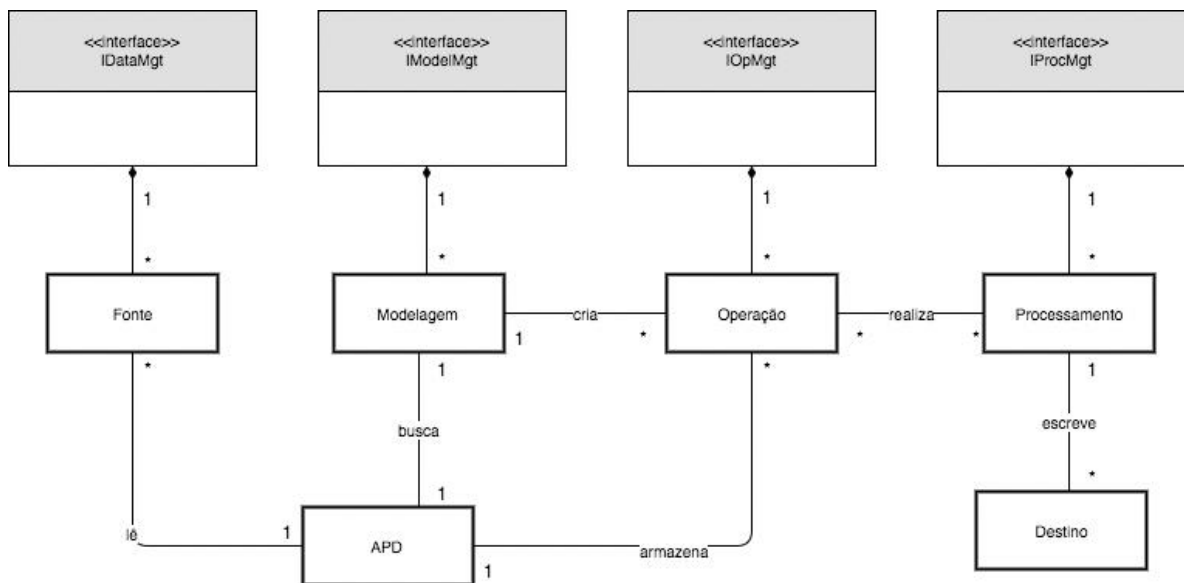


Figura 3.3 Definição de interfaces do modelo de negócio do ETL4NoSQL

3.3.4 Especificação da Arquitetura do Componente

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. A arquitetura do ETL4NoSQL é composta pelo componente de gerenciamento principal, ETL4NoSQLMgr. Nota-se na Figura 3.4, que os componentes da arquitetura do ETL4NoSQL: DataMgr, ModelMgr, OpMgr e ProcMgr; interligam-se ao componente de gerenciamento ETL4NoSQLMgr por meio de suas interfaces.

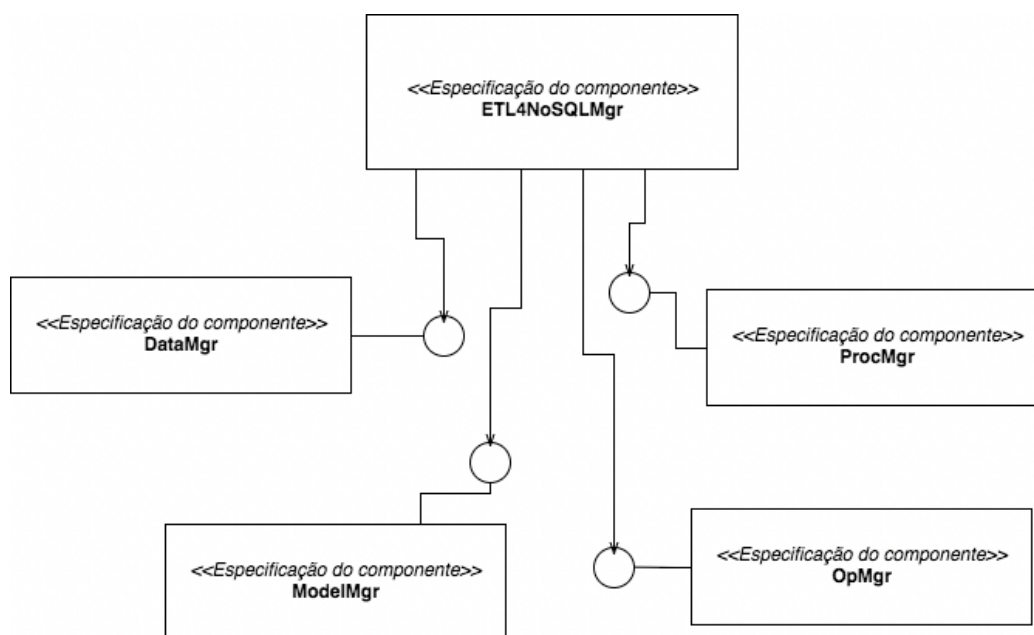


Figura 3.4 Especificação da arquitetura do componente de ETL4NoSQL

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 como segue.

3.4.1 Operações da Interface de Negócio

As operações de negócio são identificadas quando as interações são 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 vistos nas Figuras de 3.5 até 3.12.

A conexão com a fonte de dados é estabelecida (Figura 3.5). A estrutura de dados da fonte é reconhecida (Figura 3.6). A escrita na base de dados destino é permitida (Figura 3.7). A leitura dos dados da fonte é feita e armazenada na área de processamento de dados (Figura 3.8). Após a leitura dos dados da fonte, é possível realizar a criação das operações por meio dos

mecanismos existentes (Figura 3.9). 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). Com as operações criadas, é possível processá-las de forma a escolher o tipo de processamento (centralizado ou distribuído) (Figura 3.12).

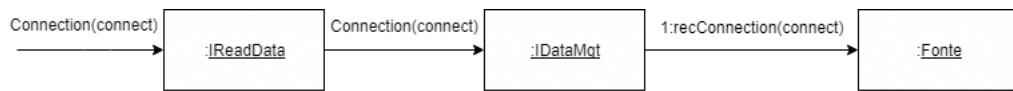


Figura 3.5 Diagrama de colaboração para conectar à base de fonte de dados

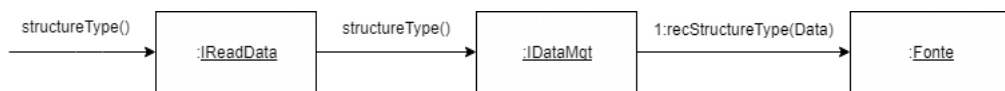


Figura 3.6 Diagrama de colaboração para verificar a estrutura de dados

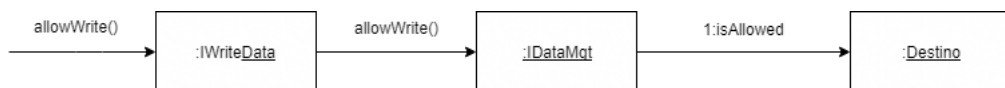


Figura3.7 Diagrama de colaboração para verificar se existe permissão de escrita na base de dados destino

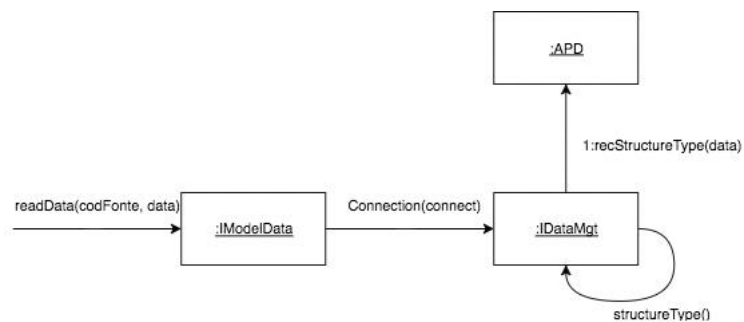


Figura 3.8 Diagrama de colaboração para leitura dos dados da fonte

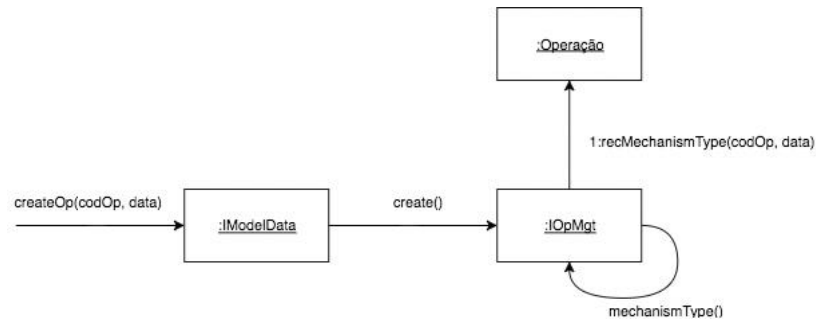


Figura 3.9 Diagrama de colaboração para criação das operações de ETL

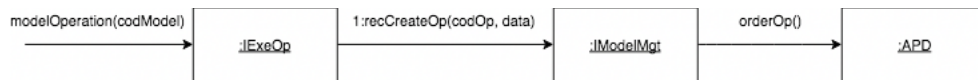


Figura 3.10 Diagrama de colaboração para modelar as operações criadas

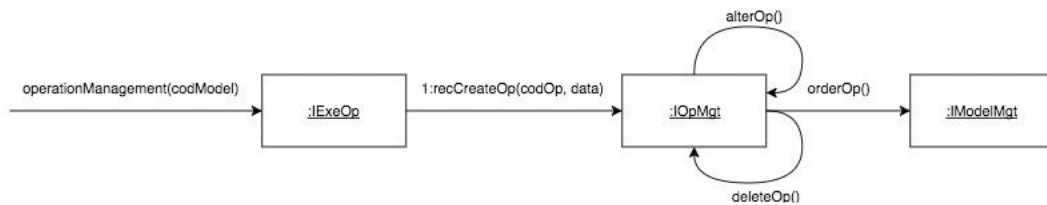


Figura 3.11 Diagrama de colaboração para gerenciar as operações

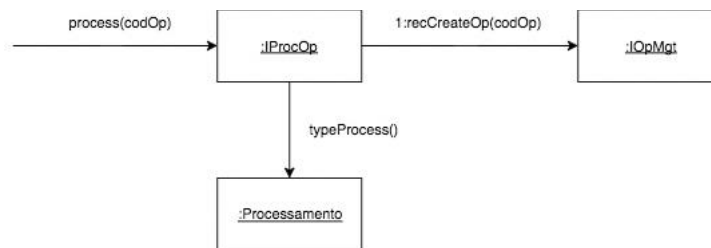


Figura 3.12 Diagrama de colaboração para processar as operações

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. O Algoritmo 6 apresenta as operações de cada interface de componente do ETL4NoSQL.

Depois das operações terem sido especificadas, foi possível criar o diagrama de especificação das interfaces representado na Figura 3.13. Essas especificações referem-se ao uso dos componentes. Elas representam o que o usuário precisa saber sobre os componentes. Nas próximas seções, mostraremos o ambiente de implementação utilizado no desenvolvimento de ETL4NoSQL, bem como suas interfaces de programação.

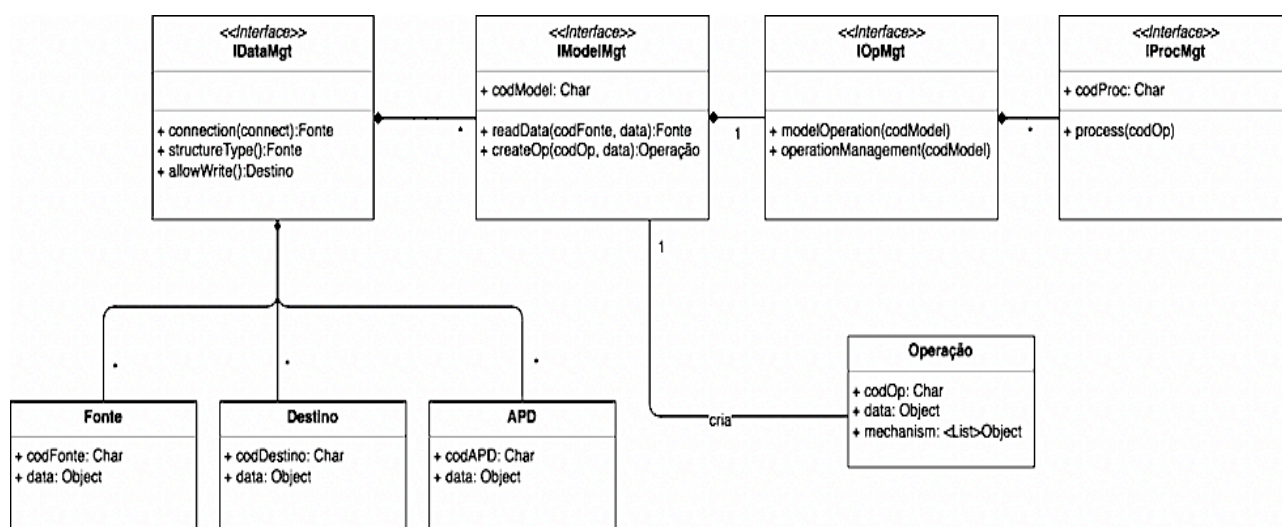


Figura 3.13 Diagrama de especificação das interfaces de ETL4NoSQL

Algoritmo 6: *Descrição das Operações de interface dos componentes do ETL4NoSQL*

context IDataMgt :: connection(connect):Fonte

pre:

- Existe um parâmetro de conexão "connect"
- Existe uma fonte para ser conectada

post:

- A conexão com a fonte foi estabelecida ou não
- Recebe uma mensagem de conexão bem-sucedida ou erro

context IDataMgt :: structureType():Fonte

pre:

- A conexão com a fonte foi bem-sucedida

post:

- A estrutura de dados foi reconhecida ou não
- Retorna mensagem de sucesso ou erro

context IDataMgt :: allowWrite():Destino

pre:

- A conexão com o destino foi bem-sucedida

post:

- A base de dados destino permite escrita ou não
- Retorna mensagem de sucesso ou erro

context IModelMgt :: readData(codFonte, data):APD

pre:

- A conexão com a fonte foi bem-sucedida
- A estrutura de dados foi reconhecida
- Existe um parâmetro de busca "data" valido para estrutura de dados reconhecida

post:

- A leitura dos dados foi realizada ou não
- Retorna os dados em uma APD ou uma mensagem de erro

context IModelMgt :: createOp(codOp, data):Operacao

pre:

- Existe o mecanismo desejado para a criação da operação
- Existe um parâmetro "data" com os dados a serem usados na

operação

post:

- A operação foi criada ou não
- Retorna o codOp ou uma mensagem de erro

context IOpMgt :: modelOperation(codModel)

pre:

- Existem operações criadas na APD (codModel)

post:

- As operações são ordenadas para execução

context IOpMgt :: operationManagement(codModel)

pre:

- Existem operações criadas na APD (codModel)

post:

- As operações foram alteradas, apagadas ou não
- Retorna mensagem de sucesso ou erro

context IProcMgt :: process(codOp)

pre:

- Existe a operação (codOp)
- Foi escolhido o tipo de processamento (centralizado ou distribuído)

post:

- A operação foi processada ou não
- Retorna mensagem de sucesso ou erro

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, a orientação à objetos consiste em um bom paradigma a ser aproveitado neste trabalho.

Para seguir os princípios de *design* orientado à objetos de inversão de controle, o ETL4NoSQL possui interfaces como o ETL4NoSQLMgr, IDataMgr, IModelMgr, IOpMgr e IProcMgr (Figura 3.4). Elas utilizam os mesmos comportamentos para as diversas variações,

porém aplicados de acordo com a especificidade de cada uma. Outro princípio importante utilizado é o da segregação de interfaces onde os usuários não devem ser forçados a depender de interfaces que não necessitam. As interfaces devem ser enxutas e conter apenas métodos específicos para cada uma delas.

Na seção seguinte, apresentaremos as interfaces de programação do ETL4NoSQL, que foram utilizadas na implementação deste trabalho de dissertação.

3.7 INTERFACES DE PROGRAMAÇÃO

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, uma vez que 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, é realizada a execução dos processos de conexão com as fontes de dados, leitura e escrita das bases de dados envolvidas nos processos de ETL (destaque 2 na Figura 3.14). ETL4NoSQLMgr gera amostra dos dados para realizar as transformações, cria as operações de ETL, realiza a execução dessas operações e as gerencia (destaque 3 na Figura 3.14). A interface ETL4NoSQLMgr é responsável por combinar todas as ações que as outras interfaces fornecem.

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, que 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 na execução do 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 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.

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

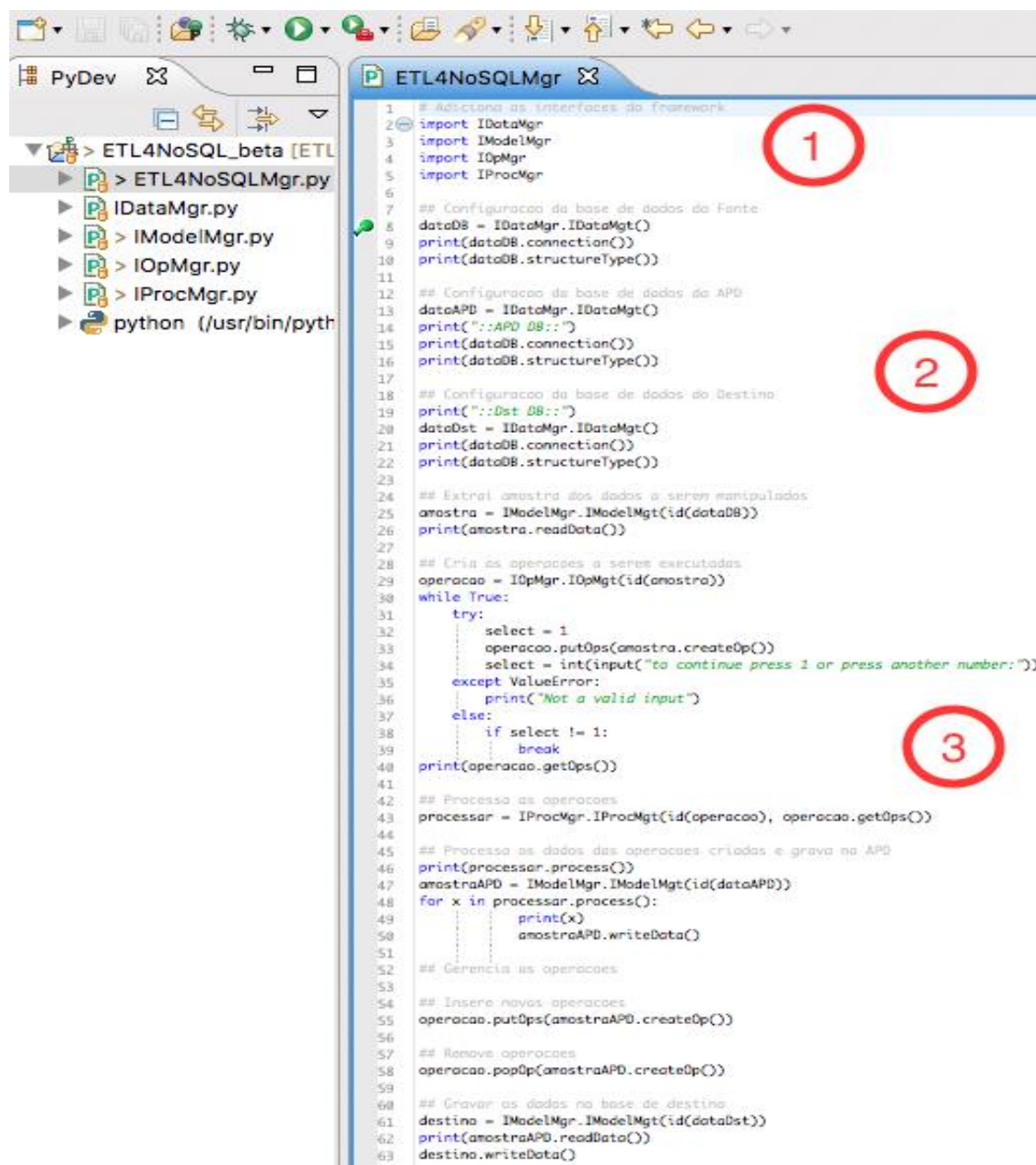


Figura 3.14 Tela do IDE LiClipse com a implementação da interface ETL4NoSQLMgr

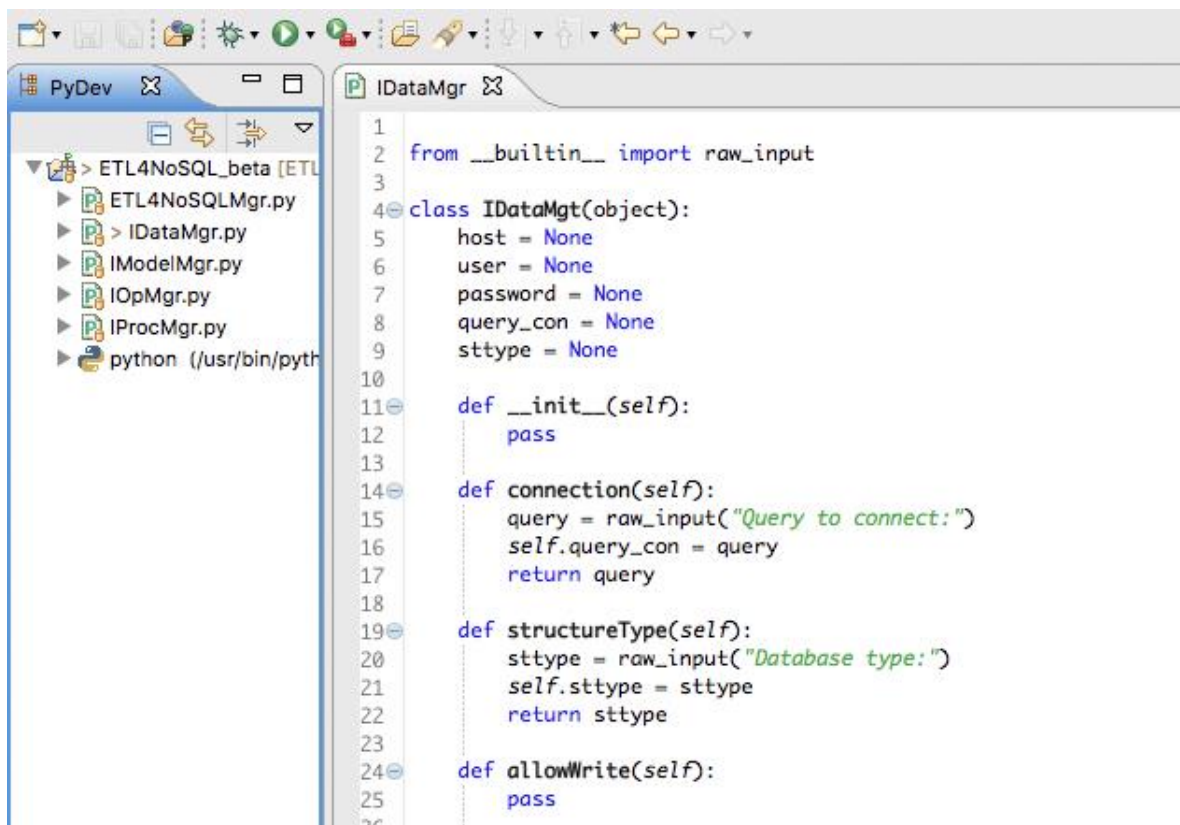


Figura 3.15 Tela do IDE LiClipse com a implementação da interface IDataMgr

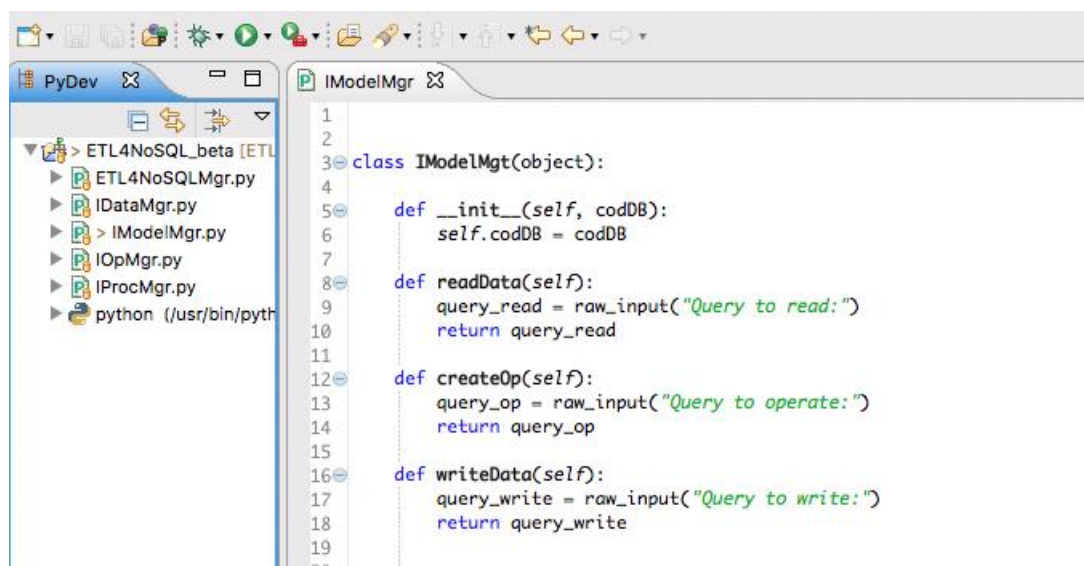


Figura 3.16 Tela do IDE LiClipse com a implementação da interface IModelMgr

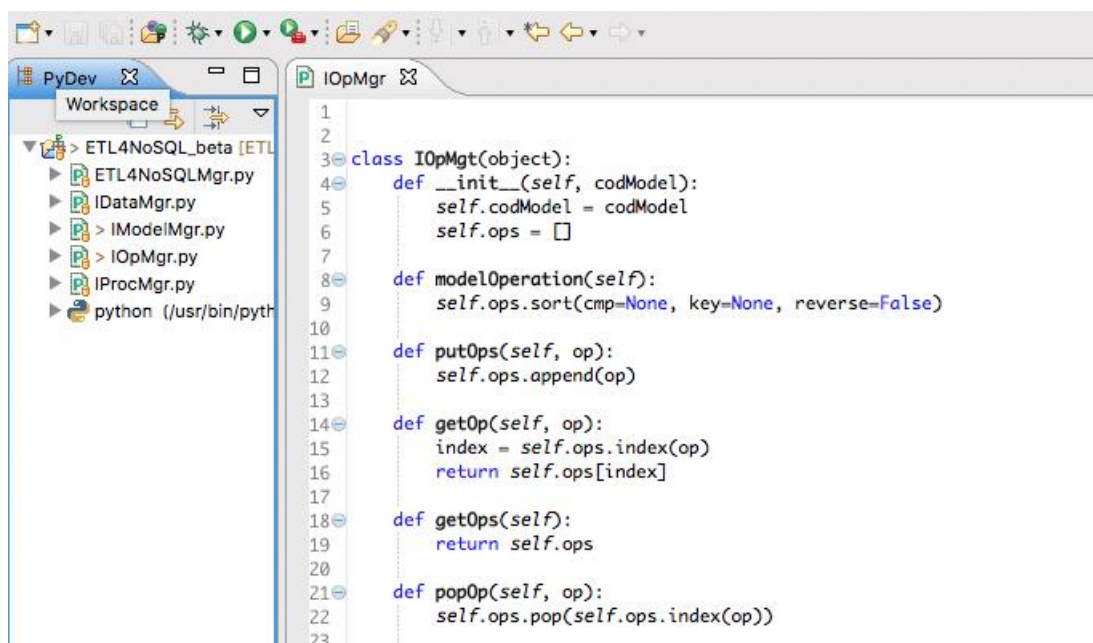


Figura 3.17 Tela do IDE LiClipse com a implementação da interface IOpMgr

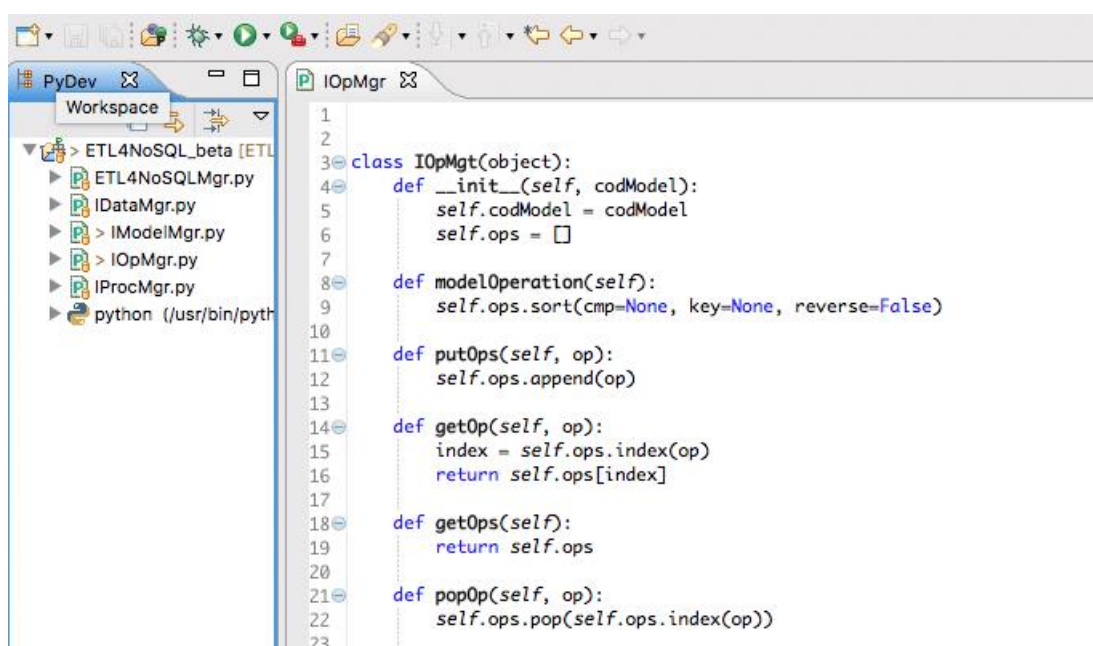


Figura 3.18 Tela do IDE LiClipse com a implementação da interface IProcMgr

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 uma solução para implementação do framework, a partir da definição de um ambiente de ETL que foi implementado 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 carecem de serem melhoradas.

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 estudadas, 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 aumento da produtividade na criação de processos de ETL?
 - (a) quais manipulam 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

Ao longo do experimento, analisamos as ferramentas de ETL encontradas neste estudo, com o propósito de caracterizar as funcionalidades dessas ferramentas. O principal aspecto de qualidade estudado, foram as funcionalidades das ferramentas de ETL, do ponto de vista dos estudos encontrados na literatura, no contexto das ferramentas estudadas.

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 pesquisadas e no ETL4NoSQL e são consideradas pouco úteis.

Q3. Existem funcionalidades que estão presentes nas ferramentas de ETL pesquisadas 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 pesquisadas e no ETL4NoSQL, consideradas úteis e que necessitam melhorar.

Q4. Existem funcionalidades que estão presentes nas ferramentas de ETL pesquisadas que não estão presentes no ETL4NoSQL, mas que são bastante utilizadas? 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

Um experimento geralmente é formulado por meio de hipóteses. A hipótese principal se chama hipótese nula, ela declara que não há nenhum relacionamento estatisticamente

significante entre a causa e o efeito. O objetivo principal do experimento é, então, rejeitar a hipótese nula a favor de uma ou algumas das hipóteses alternativas. A decisão sobre rejeição da hipótese nula pode ser tomada baseado nos resultados da sua verificação utilizando um teste estatístico (TRAVASSOS; GUROV; AMARAL (2002)).

Hipótese nula (H0): As funcionalidades oferecidas pelo ETL4NoSQL são similares às funcionalidades oferecidas pelas ferramentas de ETL pesquisadas.

Fp - Funcionalidades do ETL4NoSQL;

F1 - Funcionalidades das ferramentas pesquisadas.

$$H0: F1 - (Fp \cap F1) = \emptyset$$

Hipótese alternativa (H1): A lista de funcionalidades oferecidas pelo ETL4NoSQL é diferente da lista de funcionalidades oferecidas pelas ferramentas de ETL pesquisadas.

Fp - Funcionalidades do ETL4NoSQL;

F1 - Funcionalidades das ferramentas pesquisadas.

$$H1: F1 - (Fp \cap F1) \neq \emptyset$$

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

Fpl - Funcionalidades presentes no ETL4NoSQL e nas ferramentas pesquisadas;

Fplu - Funcionalidades presentes no ETL4NoSQL e nas ferramentas pesquisadas 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 pesquisadas possuem funcionalidades consideradas úteis, cujo necessita de melhorias.

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

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

$$H3: F_{plu} - (F_{plu} \cap F_{plun}) \neq \emptyset$$

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

F_{pn} - Funcionalidades de ETL4NoSQL não presentes ($F_{pn} \equiv F_I - (F_p \cap F_I)$)

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

$$H4: F_{pn} - (F_{pn} \cap F_o) \neq \emptyset$$

4.2.2 Descrição da instrumentação

Para cada funcionalidade das ferramentas de ETL pesquisada 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	1. Necessita melhorar	1. É útil
2. Está presente parcialmente	2. Não há necessidade de melhoria	2. Não é útil
3. Está presente	3. Pode melhorar, mas não há necessidade	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;

M – Melhoria: 0 - não necessita melhorar; 1 - necessita melhorar.

4.2.3. Métricas

Na Quadro 4.2, são apresentadas as métricas utilizadas neste experimento.

Quadro 4.2 Métricas

Nº	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; GUROV; AMARAL (2002), o contexto pode ser caracterizado conforme quatro dimensões: (i) o processo: on-line / off-line; (ii) os participantes: ferramentas de ETL; (iii) realidade: o problema real / modelado; (iv) generalidade: específico / geral.

Nosso estudo supõe o processo *off-line*, porque as ferramentas não estão sendo testadas durante o estudo, e sim, revistas de acordo com as pesquisas. Os participantes são as ferramentas de ETL pesquisadas. 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 pesquisadas, então, o contexto possui o

caráter específico.

4.2.5 Seleção dos Indivíduos

Como participantes para o estudo, propõe-se utilizar as ferramentas de ETL pesquisadas neste trabalho de dissertação. Assume-se que esses indivíduos estão presentes em diversos estudos realizados e publicados no meio acadêmico.

Para a escolha das ferramentas utilizadas neste estudo, foi levado em consideração a semelhança com a ferramenta proposta neste trabalho. Seria conveniente utilizar em nosso 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 pesquisadas.

4.2.6 Variáveis

Variável independente: A lista de funcionalidades das ferramentas de ETL pesquisadas.

Variáveis dependentes:

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

Pode receber os valores: Igual, quando todas as funcionalidades têm o valor PMU = {1, X, X} (métricas 5-8); diferente, quando todas as funcionalidades têm 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 não útil: $\{1, X, 0\} / \{1, X, X\} * 100\%$.

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 por ETL4NoSQL, mas que poderiam ser implementadas, propõe-se aplicar a análise qualitativa. Essa análise deve apresentar a lista de funcionalidades presentes nas ferramentas pesquisadas, que não estão presentes na ferramenta proposta, mas que são consideradas necessárias para facilitar os processos de ETL. 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" do nosso estudo, propõe-se utilizar ferramentas de ETL pesquisadas, que são apresentadas na Seção 2.2. Assim, assume-se que elas são representativas para uma parte da população de ferramentas de ETL correlatas a nossa proposta.

Além disso, para redução da influência dos fatores que não são de interesse do nosso estudo e, portanto, para aumento da validade interna do estudo, supõe-se utilizar dados das ferramentas correlatas a nossa proposta, cuja a validação tenha passado pela publicação acadêmica.

Validade de conclusão: para receber os valores da presença, utilidade e melhorias o teste binomial foi utilizado. A verificação de hipótese foi 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 da ferramenta proposta, com as funcionalidades elencadas pelo estudo sobre as ferramentas de ETL apresentado na Seção 2.2, necessárias para a utilização de ferramentas de ETL. As características das ferramentas de ETL presentes em nosso estudo (Seção 2.2), representam as funcionalidades que uma ferramenta de ETL deve apresentar para mostrar o desempenho adequado.

Validade externa: para avaliação do nível de importância das funcionalidades analisadas, foi levada em consideração a frequência que as funcionalidades apareceram nas ferramentas analisadas em nosso estudo (Seção 2.2).

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 em vários estudos analisados por este trabalho de dissertação.

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 do experimento descrito neste capítulo. O questionário do perfil da ferramenta de ETL, foi criado baseado nas características que estiveram mais presentes nos estudos apresentados na Seção 2.2.

Quadro 4.3 Questionário do Perfil da Ferramenta de ETL

Nome da ferramenta de ETL:	
Possui código aberto?	Sim <input type="radio"/> Não <input type="radio"/>
Possui uma marca reconhecida no mercado (citada em vários estudos)?	Sim <input type="radio"/> Não <input type="radio"/>
Tem como finalidade utilizar bancos de dados NoSQL?	Sim <input type="radio"/> Não <input type="radio"/>
Possui interface gráfica?	Sim <input type="radio"/> Não <input type="radio"/>
É programável?	Sim <input type="radio"/> Não <input type="radio"/>
É integrada (no sentido que possibilita as ações no	Sim <input type="radio"/> Não <input type="radio"/>

mesmo ambiente de programação)?	
Qual o tipo de processamento que a ferramenta executa?	Distribuído <input type="radio"/> Centralizada <input type="radio"/> Híbrido <input type="radio"/>
É extensível?	Sim <input type="radio"/> Não <input type="radio"/>
Para qual finalidade a ferramenta procura auxiliar melhor os processos de ETL?	Modelagem <input type="radio"/> Desempenho <input type="radio"/> Ambos <input type="radio"/>

4.3.2 Questionário de Funcionalidades

O questionário apresentado na Figura 4.1, foi utilizado para determinar, sob o ponto de vista das ferramentas analisadas na Seção 2.2, a presença, utilidade e melhorias quanto às funcionalidades das ferramentas. Para preencher o questionário, analisamos os perfis das ferramentas por meio dos estudos que dizem respeito a cada uma delas.

Quadro 4.4 Instrumentação para aplicar o questionário

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

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. Notamos que as funcionalidades mais presentes nas ferramentas pesquisadas são: suporte à plataforma, apoio ao nível de *debugging* e suporte à modelagem. Já a Figura 4.3, mostra o quanto uma funcionalidade é usada pelas ferramentas (essa métrica é determinada pela referência do uso da funcionalidade em cada estudo analisado). Reparamos que todas as funcionalidades elencadas possuem utilidade para as

ferramentas de ETL pesquisadas. Finalmente, a Figura 4.4 indica necessidade de melhoria das funcionalidades indicada por cada ferramenta. Vimos que, boa parte das funcionalidades podem melhorar de certa forma.

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 dados 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

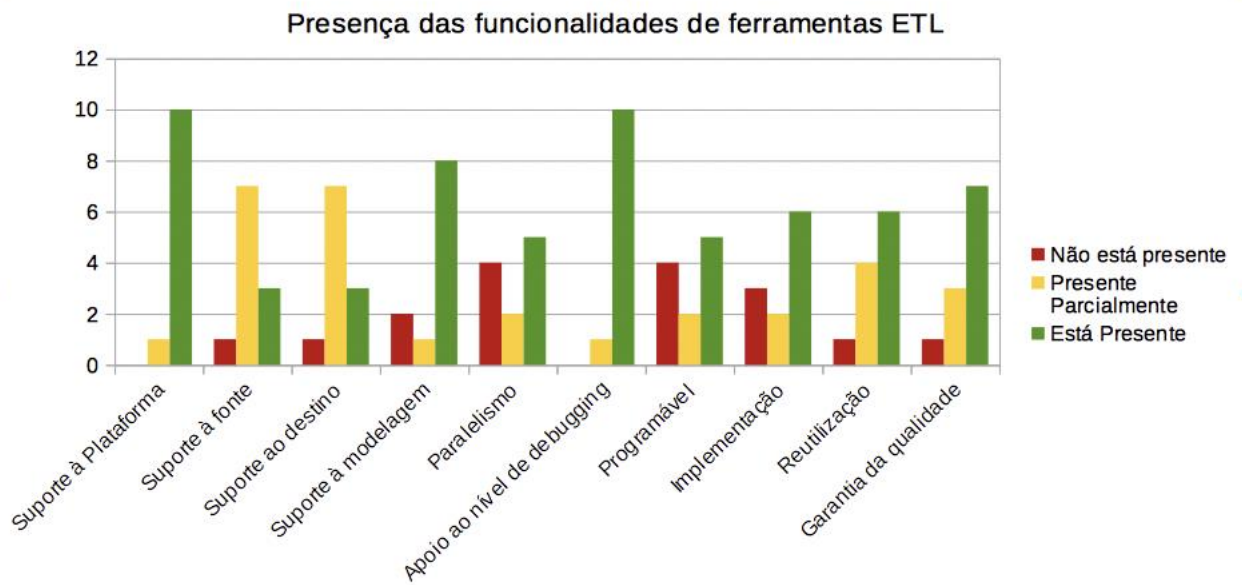


Figura 4.2 Quantidade de Presença para cada funcionalidade

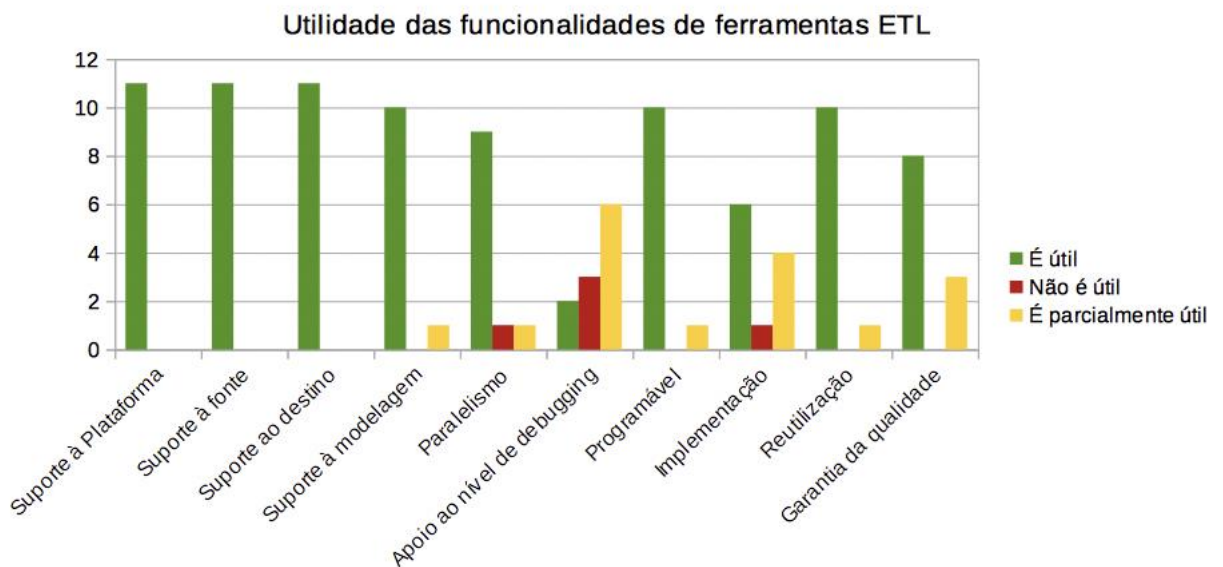


Figura 4.3 Níveis de utilidade para cada funcionalidade

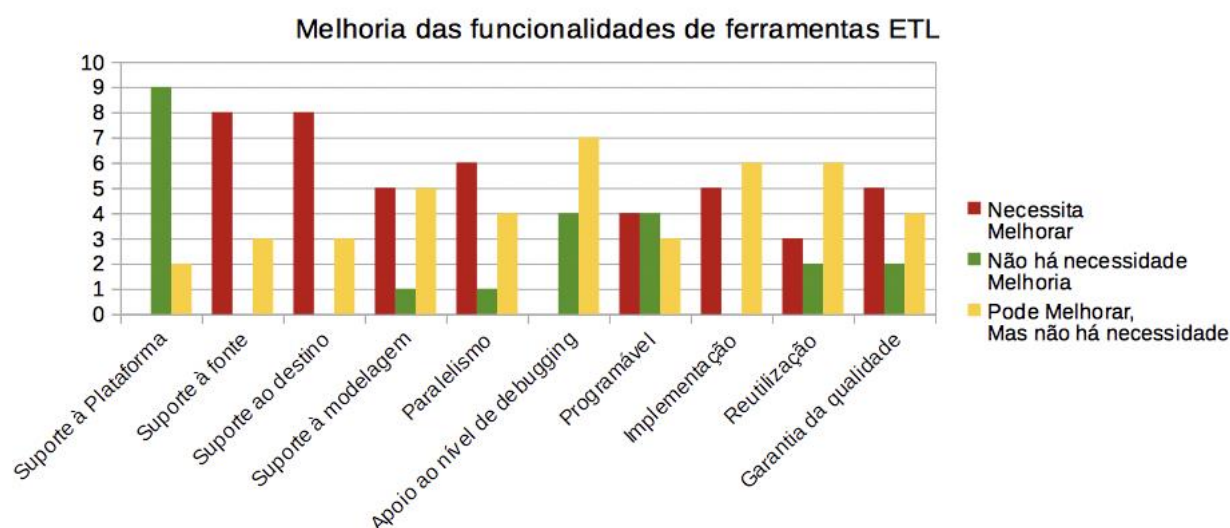


Figura 4.4 Necessidade de melhoria para cada funcionalidade

O perfil dos participantes é apresentado no quadro 4.5, bem como a legenda para leitura é exibida no quadro 4.6. Nele podemos observar que, das onze ferramentas de ETL analisadas neste estudo, dez possuem código aberto, sete não possui uma marca reconhecida no mercado, ou seja, são pouco citadas pelos trabalhos considerados nesta pesquisa. Nenhuma das ferramentas de ETL foi desenvolvida para suportar BDs NoSQL, apenas sete delas consideram suporte à alguns SGBD NoSQL. Seis possuem uma GUI, quatro não são programáveis, duas não são integradas, isto é, utilizam outros ambientes para realizar ações. Nove das ferramentas oferecem 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	Progra mável	Integra da	Processa mento	Extensí vel	Finali dade
PygramETL	1	2	2	2	1	1	2	1	2
ARKTOSH	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	Progra mável	Integrada	Processamento	Extensí vel	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		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 presentes, 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 vistos nos Quadros 4.10, 4.11, 4.12 e 4.13.

Os valores nos Quadros 4.10, 4.11, 4.12 e 4.13 significam: (P – quantidade presente: quantidade não presente); (M – quantidade de ferramentas que necessita melhoria: quantidade que não necessita melhoria); (U – quantidade de ferramentas que faz bastante uso: quantidade de ferramenta que pouco utiliza a funcionalidade).

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	<p>- A funcionalidade existe na maioria das ferramentas, mas não dão enfoque à ela. - É possível comparar os dados da fonte e destino mesmo sem <i>debugging</i>.</p> <p>- 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.</p>

Quadro 4.11 Funcionalidade presente e é útil

N	Funcionalidade	P	M	U	Características
2	Suporte à fonte	10:1	8:3	11:0	<p>- As ferramentas oferecem algum suporte em relação à leitura e carga da fonte e destino dos dados, porém apenas lidam com SGBDs relacionais.</p> <p>- Deve haver um enfoque nas novas soluções de SGBDs como os SGBDs NoSQL.</p> <p>- 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.</p> <p>- 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.</p>
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	<p>- A maioria das ferramentas dão suporte à modelagem dos processos de ETL, porém algumas focam apenas no desempenho necessitando melhoria no processo de modelagem.</p> <p>- Poucas ferramentas dão importância para garantir qualidade aos processos de ETL.</p>
10	Garantia de qualidade	10:1	5:6	11:0	

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	<p>- As ferramentas demonstraram ser independentes de plataforma por terem características como código compilável e serem programáveis.</p> <p>- 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.</p> <p>- 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.</p>
8	Implementação	8:3	5:6	10:1	
9	Reutilização	10:1	3:8	11:0	

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 das suas funcionalidades. 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 estudadas.

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

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

Fórmula de frequências esperadas:

$$E = (\text{total da linha}) \times (\text{total da coluna}) / (\text{total geral})$$

$$E = 63 \times 11 / 110 = 6,3$$

$$E = 47 \times 11 / 110 = 4,7$$

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

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

Resultado do teste χ^2

$$\chi^2 = 20,765$$

com

$$gl = (l - 1) \cdot (c - 1) = (2 - 1)(10 - 1) = 9$$

Pela tabela de distribuição qui-quadrado, verificamos que a probabilidade de significância p 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, $p < \alpha$). 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.

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 a hipótese nula (H_0) precisamos responder à 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 H_1).

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 H_2), pois respondendo à 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 H_3), 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 (H_4) 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.

4.3 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 para este estudo, indicaram que o framework proposto possui uma quantidade significativa de grau de similaridade com as ferramentas encontradas na literatura estudada.

O capítulo seguinte apresentará a avaliação da nossa proposta onde criamos a partir de ETL4NoSQL duas aplicações de ETL utilizando SGBD NoSQL. O objetivo para criar essas aplicações é ilustrar a reusabilidade, flexibilidade e a natureza programável do ETL4NoSQL e sua aplicabilidade em BDs NoSQL.

5

Aplicações de ETL4NoSQL

Este capítulo fornece exemplos de aplicações criadas a partir de ETL4NoSQL para dois domínios distintos, utilizando SGBD NoSQL para os dados de entrada. O desenvolvimento dessas aplicações pretende ilustrar a reusabilidade, flexibilidade e a natureza programável do ETL4NoSQL em aplicações de ETL. Essas aplicações apresentam requisitos distintos de inserção e transformação de dados, avaliando o framework 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 feita a partir de ETL4NoSQL, aplicando processos para extração, transformação e carga de dados que são modelados de acordo com o esquema estrela, utilizando os dados de entrada armazenados no SGBD MongoDB e SGBD Cassandra.

5.1 APLICAÇÕES DE ETL4NOSQL

Para ilustrar 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, e usamos a interface de modelagem (IModelMgr) para representação dos dados no esquema estrela. Finalmente, processamos os dados por meio da interface de processamento (IProcMgr) que foram gerados 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 e utilizando o SGBD MongoDB como fonte de dados e o esquema estrela como representação 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. O Quadro 5.1, apresenta a estrutura da base de dados utilizada nesta aplicação.

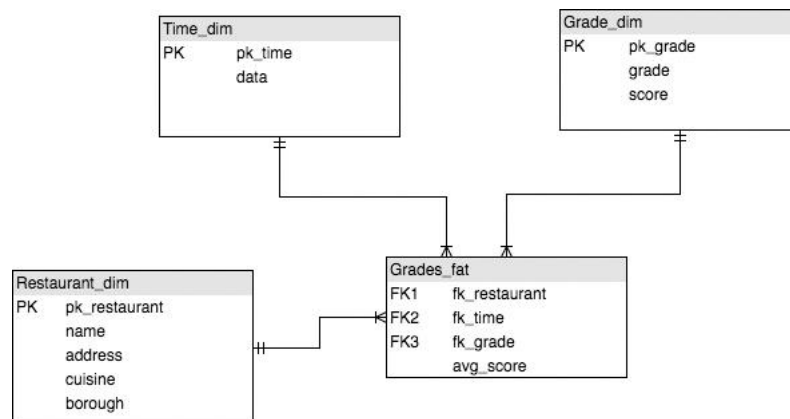
Assim sendo, na Figura 5.1 definimos esquema estrela multidimensional para a geração da saída de dados.

Quadro 5.1: Modelo lógico da base de dados de avaliação de restaurantes

```

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

```


Figura 5.1 Esquema Multidimensional da aplicação ETL4NoSQLMongoStar

A partir do *framework* ETL4NoSQL, foi possível criar uma nova aplicação denominada ETL4NoSQLMongoStar para desenvolver e executar os processos de ETL de forma a atender ao esquema multidimensional do exemplo utilizado.

Na Figura 5.2, podemos ver a aplicação ETL4NoSQLMongoStar. No destaque 1 desta

figura, 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 proposto. No destaque 3 da Figura 5.2, é feita a leitura das bases de dados envolvidas nos processos, enquanto 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 esquema multidimensional do ETL4NoSQLMongoStar e sua saída de dados em arquivos JSON.

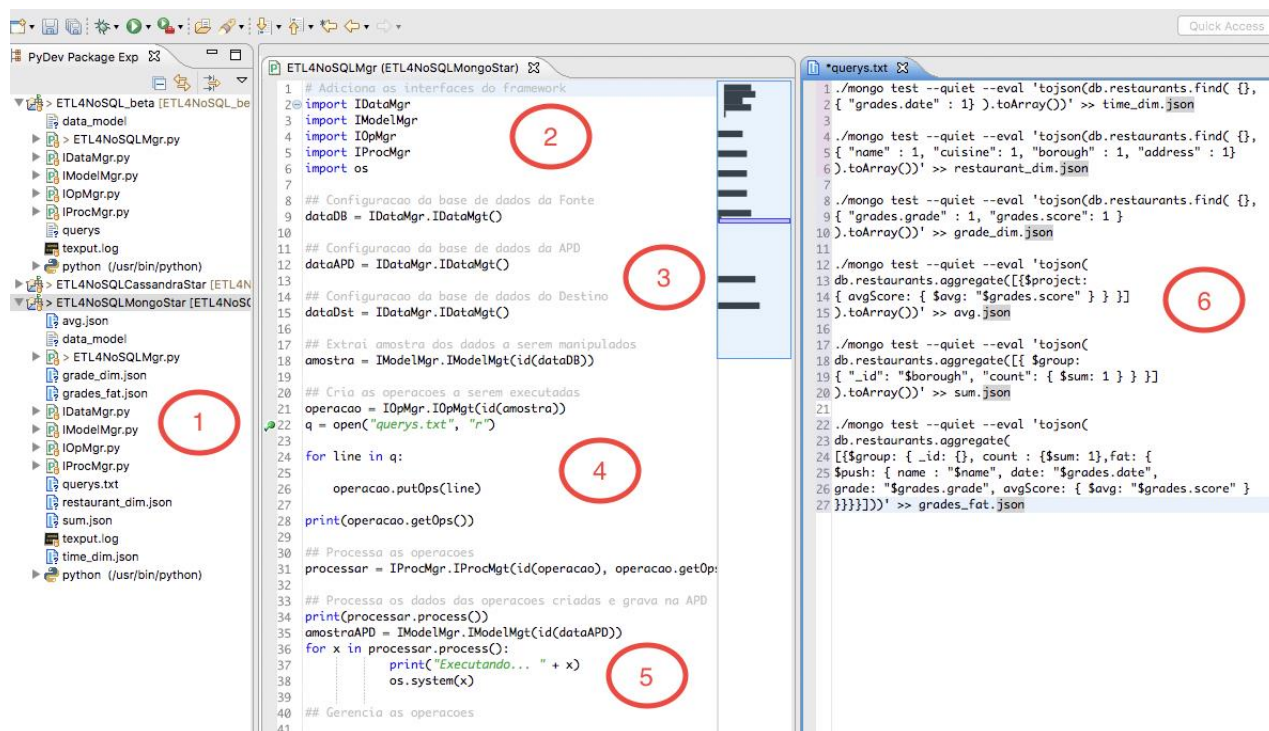


Figura 5.2 Tela da aplicação ETL4NoSQLMongoStar

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

5.3 APLICAÇÃO ETL4NOSQLCASSANDRASTAR

A aplicação de exemplo ETL4NoSQLCassandraStar foi criada a partir do framework ETL4NoSQL proposto neste trabalho. Neste exemplo, utilizamos o SGBD Cassandra e a base de dados de localizações de táxis, definidas por latitude e longitude em um determinado momento. A estrutura de dados da base de dados de origem pode ser vista no Quadro 5.3. Com isso, determinamos o esquema estrela para a aplicação de gerenciamento de táxi que é mostrado na Figura 5.3.

Quadro 5.3: Modelo Lógico da base de dados de localizações de táxis

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

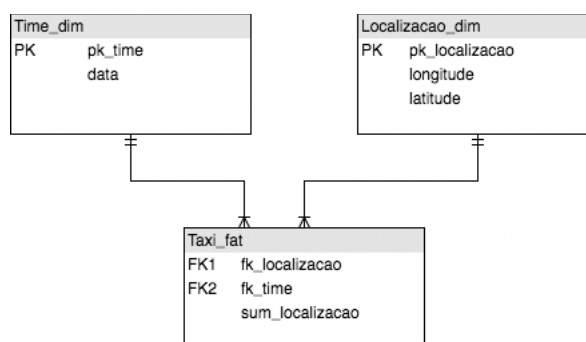


Figura 5.3 Esquema Multidimensional da aplicação ETL4NoSQLCassandraStar

Assim, utilizando a aplicação criada a partir de ETL4NoSQL, foi possível executar os processos de ETL para criar o esquema multidimensional.

Na Figura 5.4 podemos ver a aplicação ETL4NoSQLCassandraStar. Da mesma forma que foi feito para a aplicação ETL4NoSQLMongoStar, no destaque 1 da Figura 5.4, 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 da mesma figura, são importadas as interfaces dos

componentes do framework. No destaque 3 da figura 5.4, é feita a leitura das bases de dados envolvidas nos processos, enquanto 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 esquema 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 a partir de 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. Ele é reusável, pois permite que seus componentes sejam reutilizados por várias aplicações. Finalmente, flexível, dado que é possível criar aplicações a partir dele, para atender vários domínios de aplicação.

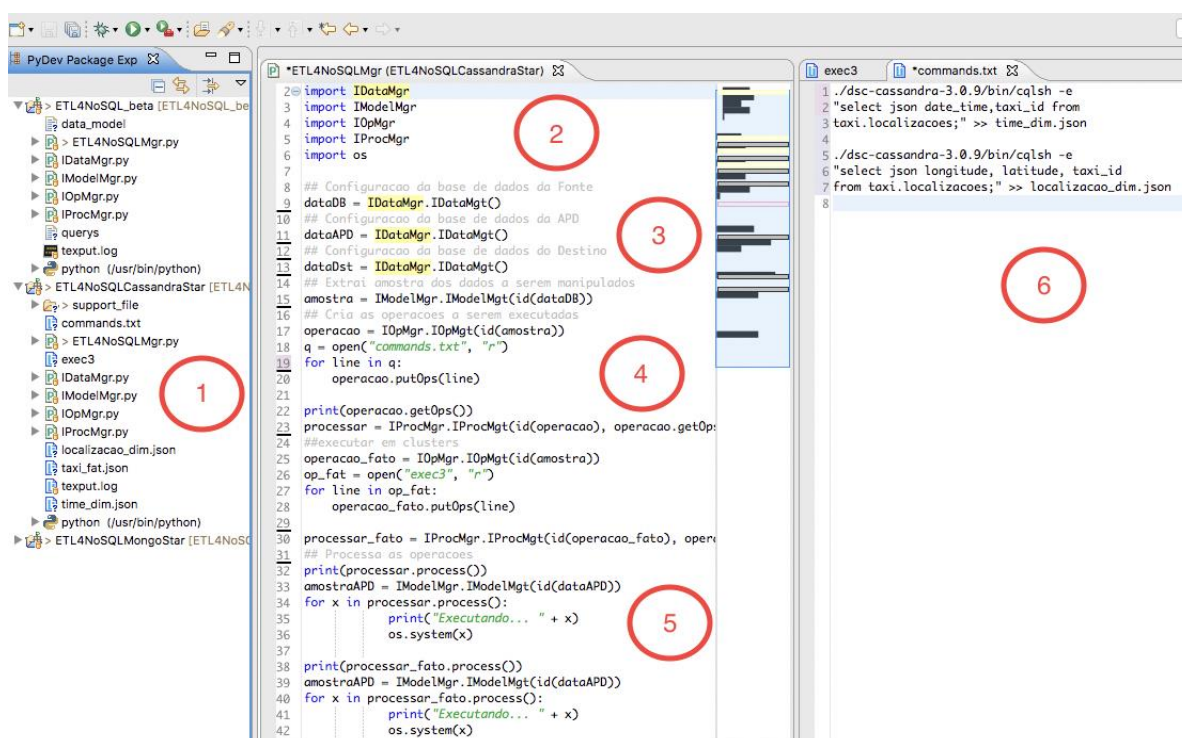


Figura 5.4 Tela da aplicação ETL4NoSQLCassandraStar

5.4 CONSIDERAÇÕES FINAIS

Este capítulo descreveu brevemente duas aplicações de naturezas distintas que foram criadas a partir de ETL4NoSQL, para ilustrar 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 geramos 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 principais resultados obtidos e trabalhos futuros a partir do ETL4NoSQL.

6

Conclusão

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 dos processos de ETL. O componente de gerenciamento de dados do framework executa os processos de ETL em BDs NoSQL. Além disso, ele possibilita a leitura e manipulação de dados em BDs NoSQL, e o armazenamento desses dados em bases deste tipo, oferecendo uma alternativa de uso de dados representados por modelos não relacionais para construção de DW.

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 reutilizadas 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 estudada. 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 às outras 11 ferramentas estudadas nesta pesquisa, e que dessa similaridade 85,71% das funcionalidades são consideradas úteis para desenvolver processos de ETL.

Por último, são exemplificadas duas aplicações de domínios distintos baseadas em SGBD NoSQL, que foram criadas a partir de ETL4NoSQL para ilustrar 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 de ETL, e ao final da execução dos processos geramos 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: (i) programável, no sentido de possibilitar a programação dos seus parâmetros; (ii) reusável, pois permite que seus componentes sejam reutilizados por várias aplicações; e finalmente, (iii) flexível, dado que é possível estendê-lo para atender vários domínios de aplicação.

6.2 DISCUSSÃO FINAL

Na Seção 2.2, apresentamos os trabalhos correlatos desta pesquisa. Nesta seção, foram listadas as ferramentas de ETL encontradas na literatura estudada. Podemos observar que as ferramentas estudadas se preocupam com o desempenho ao lidar com grandes volumes de dados e BDs NoSQL. Porém, nenhuma das ferramentas pesquisadas apresentou uma solução alternativa para modelagem de BDs NoSQL, ficando a cargo do desenvolvedor encontrar meios para gerar esquemas multidimensionais de dados a partir de fontes de dados do tipo BD NoSQL. Algumas delas, como a P-ETL, oferece alternativas para leituras de arquivos textuais, mas não dão suporte 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 o acesso e a leitura dos dados de SGBD não relacionais.

Por 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

Indicações de trabalhos futuros são descritos a seguir.

- 1) A melhoria de aplicações de teste para executar transformações com dados reais, é uma das sugestões de próximos trabalhos;
- 2) A extensão do ETL4NoSQL para testar a execução dos processos de forma distribuída (Liu et al., 2011) e paralela (Thomsen and Pedersen, 2011) pode ser uma investigação futura para investigar o desempenho;
- 3) Adicionalmente, utilizar aplicações de ETL de mesmo domínio em outras ferramentas de ETL e comparar os resultados;
- 4) Finalmente, migrar o ETL4NoSQL como um serviço (SaaS) em um ambiente em nuvem, que se caracteriza pela virtualização de recursos e arquitetura orientada a serviços (SOA).

