Universidade Federal de Pernambuco
Cin - Centro de Informática

Pós graduação em Ciência da Computação

ETL4NoSQL: Um framework de ETL para BDs NoSQL

Carine Calixto Aguena

Dissertação de Mestrado

Recife
<DATA DA DEFESA>

Universidade Federal de Pernambuco
Cin - Centro de Informática

Carine Calixto Aguena

ETL4NoSQL: Um framework de ETL para BDs NoSQL

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

Orientador: *Valéria Cesário Times*

Recife
<DATA DA DEFESA>

<DIGITE A DEDICATÒRIA AQUI>

Agradecimentos

<DIGITE OS AGRADECIMENTOS AQUI>

<*DIGITE AQUI A CITAÇÂO*>
—<AUTOR> (<NOTA>)

Resumo

<DIGITE O RESUMO AQUI>

Palavras-chave: <DIGITE AS PALAVRAS-CHAVE AQUI>

Abstract

Keywords: <DIGITE AS PALAVRAS-CHAVE AQUI>

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

Este capítulo contextualiza os principais assuntos abordados neste trabalho, apresenta as motivações, os objetivos gerais e específicos da proposta desta pesquisa, bem como sua justificativa.

1.1 Contextualização

Os requisitos para aplicações modernas têm mudado significativamente, especialmente com o aumento das aplicações Web. Este segmento de aplicações exige requisitos com alta escalabilidade e vazão, onde sistemas que utilizam um armazenamento com esquema relacional não conseguem atender satisfatoriamente. Em resposta a isso, novas abordagens de armazenamentos de dados utilizando o termo de NoSQL tornaram-se popular.

O termo NoSQL é constantemente interpretado como "*Not Only SQL*", onde SQL refere-se a linguagem de manipulação de dados dos gerenciadores de armazenamento de dados relacionais (RDBMS - Relational Database Management System) - Structure Query Language. O grande propósito das abordagens NoSQL é oferecer alternativas onde os esquemas relacionais não funcionam bem. O termo abrange diferentes sistemas. Em geral, banco de dados NoSQL usam modelo de dados não-relacionais, com poucas definições de esquema e escala horizontal [Nas12].

Muitas empresas coletam e armazenam milhares de gigabytes de dados por dia, onde a analise desses dados torna-se uma vantagem competitiva no mercado. Dessa forma, há uma grande necessidade de uma nova arquitetura de Data Warehouse que possa alcançar melhor escalabilidade e eficiência [LTP13].

Segundo a definição de [KR02], Data Warehouse (DW) é uma coleção de dados para o processo de gerenciamento de suporte à decisão orientado a assunto, integrado, variante no tempo e não volátil. Os dados de diferentes fontes de sistemas são processados em um Data Warehouse central através da Extração, Transformação e Carga (ETL) de maneira periódica.

O projeto de ETL 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, pouca importância foi dada ao processo de ETL por um grande período de tempo 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 [dS12].

As pesquisas passaram a apontar problemas como complexidade, longa curva de aprendizagem, notações proprietárias, custo e tempo de implantação das ferramentas atuais. Além disso, é impossível oferecer um pacote fechado com todas as possibilidades de transformações exigidas pelos processos de ETL. Para sanar essas dificuldades, propostas de modelagens conceituais e lógicas foram apresentadas e Notação de Modelagem para Processo de Negócio (BPMN) para ETL foram definidas, as quais aumentam o nível de abstração dos processos de ETL, e consequentemente, os tornam independentes da plataforma de implementação. Contudo, a implementação dos processos de ETL programaticamente a partir de uma linguagem de programação de propósito geral tem sido adotada por muitas empresas, porque isso evita as desvantagens da utilização de ferramentas, além de aumentar o nível de customização e integração dos processos de ETL com outros sistemas [dS12].

Tradicionalmente, o DW é implementado em uma base de dados relacional, onde o dado é armazenado nas tabelas fato e tabelas dimensões, na qual forma um esquema em estrela (colocar ref DW). Por isso, as ferramentas de ETL mais utilizadas no mercado atual somente dão suporte aos esquemas relacionais. Para dar suporte aos sistemas que necessitam utilizar um

esquema não relacional em DW, a proposta desse trabalho é especificar um framework programável, flexível e integrado para modelagem e execução de processos ETL em BDs NoSQL.

1.2 Motivação

O aumento do uso de Banco de Dados com esquemas não relacionais e a falta de uma ferramenta programável, flexível e integrada, independente de plataforma que dê suporte à extração, transformação e carga em Data Warehouses é a grande motivação deste trabalho.

As pesquisas sobre extração de dados em BDs NoSQL mostram que não há uma ferramenta que seja integrada para o uso de BDs NoSQL, as ferramentas existentes no mercado apenas oferecem a possibilidade para alguns SGBDs NoSQL, ficando a cargo da equipe de implantação do projeto de DW todo o trabalho de modelagem e programação ao se utilizar BDs NoSQL.

[dS12] aponta em sua pesquisa que muitas empresas evitam ferramentas de ETL disponíveis no mercado, e adotam o desenvolvimento dos processos a partir de uma linguagem de programação de propósito geral, pelo fato dessas ferramentas terem uma longa curva de aprendizagem e grande complexidade no seu uso.

Dessa forma, encontrar uma solução que seja programável, flexível e integrada para extração, transformação e carga dos dados em BDs NoSQL é a motivação deste trabalho.

1.3 Objetivos

Nesta seção serão apresentados os objetivos desta pesquisa.

1.3.1 Objetivo Geral

Especificar um framework programável, flexível e integrado para modelagem e execução de processos ETL em BDs NoSQL.

1.3.2 Objetivo Específico

Estender a proposta do framework para facilitar a carga de dados de dois sistemas de BD NoSQL distintos baseado no paradigma família de coluna em um DW relacional.

1.4 Justificativa

1.5 Organização do Trabalho

CAPÍTULO 2

Fundamentação Teórica

Neste capítulo são apresentados os conceitos relacionados ao desenvolvimento desta pesquisa.

Os conceitos de ETL e Data Warehouse (DW), bem como o termo NoSQL e os paradigmas de esquemas não relacionais mais utilizados pela comunidade acadêmica, o Famílias de Colunas, Orientados à Documentos, Chave-Valor e Baseado em Grafos.

Também são detalhadas as definições de modelagem conceitual e lógica para esquemas não relacionais.

2.1 ETL

2.2 Data Warehouse

2.3 Bancos de Dados NoSQL

Consistem em bancos de dados não relacionais projetados para gerenciar grandes volumes de dados e que disponibilizam estruturas e interfaces de acesso simples (Lima; Mello, 2015). Cada paradigma NoSQL possui um esquema de modelagem diferente, nos quais são divididas pela literatura em quatro categorias amplamente usadas: Chave-Valor, Orientado a Documentos, Famílias de Colunas e Baseado em Grafos ([Fowler, 2013], [Kaur; Rani, 2013]).

As principais características dos banco de dados NoSQL são:

-Distribuído:

- Escalabilidade Horizontal
- Construído para grande volume de dados
- BASE ao invés de ACID
- Modelo de dados não relacional
- Sem definições de esquema
- Não suporta SQL

[Nas12]

2.3.1 Banco de dados Orientados à Documentos

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

Existem mais de 15 banco de dados orientados a documentos disponíveis e os mais utilizados são MongoDB, CouchDB e o RavenDB (Kaur; Rani, 2013).

2.3.2 Banco de dados Famílias de Colunas

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

A maioria dos bancos de dados baseados em Famílias de Colunas são também compatíveis

com o framework MapReduce, no qual acelera o processamento de enorme volume de dados pela distribuição do problema em um grande número de sistemas. Os bancos de dados de Família de Colunas open-source mais populares são Hypertable, HBase e Cassandra (Kaur; Rani, 2013).

2.3.3 Banco de dados Baseado em Grafos

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

2.3.4 Banco de dados Chave-Valor

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

Os bancos de dados Chave-Valor mais populares são Riak, Voldemort e Redis (Kaur; Rani, 2013).

2.4 Projeto Conceitual, Lógico e Físico

Tradicionalmente um projeto de banco de dados é modelado em três fases denominadas conceitual, lógica e física. O projeto conceitual consiste em apresentar um esquema expressivo que modele os dados de um determinado domínio de informação, enquanto que o projeto lógico transforma um esquema conceitual em algo que se aproxima de um modelo de implementação física do banco de dados. Em projetos de banco de dados NoSQL, há poucos trabalhos que abordam uma metodologia para esquemas lógicos baseados em modelagens conceituais (Lima; Mello, 2015).

Dessa forma, esta seção visa aprofundar o tema a respeito de projeto conceitual e lógico em banco de dados NoSQL.

2.4.1 Modelo Conceitual NoSQL

Em bancos de dados relacionais, o modelo conceitual mais utilizado na literatura é o modelo ER (Entidade-Relacionamento) (Fowler, 2013). Contudo, bancos de dados NoSQL necessitam de um modelo conceitual que atenda às suas características.

O desenvolvimento de banco de dados para sistemas NoSQL é usualmente baseado nas melhores práticas, nas quais são especificamente relacionadas ao sistema desenvolvido, com nenhuma metodologia sistematizada (Bugiotti; Cabibbo; Atzeni, 2014). Por isso, Bugiotti et al (2014) desenvolveu uma abordagem baseada no NoAM (NoSQL Abstract Model). Esta abordagem observa que vários sistemas NoSQL compartilham de características de modelagem similares. Uma importante observação é que sistemas NoSQL oferecem operações de acesso aos dados de forma eficiente, atômica e escalável nas unidades de acesso aos dados em uma certa granularidade. Uma representação errada pode levar a incapacidade de garantir a atomicidade das operações importantes e o desempenho pode piorar dependendo magnitude da aplicação.

A metodologia de Bugiotti et al (2014) procede com a identificação dos agregados, onde cada agregado é um grupo de objetos relacionados que podem ser acessados e/ou manipulados juntos. Essa atividade é importante para suportar escalabilidade e consistência. O modelo conceitual desenvolvido por Bugiotti et al (2014) segue o modelo padrão do DDD (Domain-Driven Design), no qual é uma metodologia muito utilizada em orientação à objeto. Dessa forma, para o modelo conceitual NoSQL, é utilizado o diagrama de classe conceitual da UML, definindo as entidades, valores dos objetos e relacionamentos da aplicação.

2.4.2 Modelo Lógico NoSQL

O modelo de dados lógico dominante nas últimas décadas tem sido o modelo relacional (Fowler, 2013). Porém, para bancos de dados NoSQL a modelagem relacional não atende as características para representação lógica de seus dados. Para Fowler (2013), cada solução NoSQL possui um modelo diferente, os quais ele dividiu em quatro categorias amplamente usadas na literatura: chave-valor, documento, famílias de colunas e grafos.

Nesse mesmo contexto, Lima (2015) propôs a utilização de esquemas lógicos para NoSQL que utilizam o conceito de agregados. Ele justifica a escolha pelo fato de que a representação lógica baseada em agregados apoia os requisitos típicos dos bancos de dados NoSQL, oferecendo suporte à escalabilidade, consistência e desempenho. O conceito de agregados é um termo da área Domain-Driven Design (DDD), sendo uma coleção de objetos relacionados, aninhados, representando como uma única entidade (Lima;Mello, 2015).

Agregado é um padrão de domínio usado para definir a propriedade e fronteira do objeto. Ele é um grupo de objetos associados que são considerados como uma unidade em relação a alterações de dados. O Agregado é demarcado pela fronteira que separa os objetos de dentro para fora. Cada Agregado tem uma raiz. A raíz é uma Entidade, e ela é o único objeto que é acessível de fora do agregado. A raíz pode guardar referências para qualquer dos objetos agregados, e os outros objetos podem guardar referências uns dos outros, mas um objeto de fora só pode guardar referências do objeto raíz. Se houver outras Entidades dentro da fronteira, a identidade dessas entidades é local, fazendo sentido somente dentro do agregado (Domain-

Driven Design Quickly, 2006).

Fowler (2013), define um agregado como um conjunto de objetos relacionados que são tratados como uma unidade, mais precisamente, é uma unidade de manipulação de dados e gerenciamento de consistência. Ele afirma também que trabalhar com banco de dados orientados a agregados traz uma semântica mais clara, enfocando a unidade da interação com o armazenamento de dados. Contudo, o motivo mais importante para a utilização da modelagem orientada a agregados em bancos de dados NoSQL é que ela auxilia a execução em um cluster. Quando se opera em um cluster é necessário minimizar o número de nodos a serem pesquisados na coleta de dados. Assim, ao incluir os agregados é possível dar a informação ao banco de dados sobre quais partes serão manipuladas juntas e no mesmo nodo.

Dessa forma, a utilização de um modelo lógico NoSQL baseado em agregados se justifica pelo fato de que o conceito desse modelo possibilita o gerenciamento de consistência, a execução em cluster e uma semântica mais clara.

2.5 Trabalhos Correlatos

Esta seção aborda os trabalhos que são correlatos a esta pesquisa, bem como descreve como estes trabalhos diferem do realizado por esta pesquisa.

2.5.1 Experiência do Usuário

Uma abordagem apresentado por Tableau permite em sua ferramenta que o usuário possa adicionar scripts Hive que consequentemente são executados em um Hadoop cluster. A saída é então importada para a memória no Tableau. O usuário define quais dados do sistema de arquivo devem ser processados, e o quanto complexo os dados deverão ser comprimidos para o formato que a aplicação suporte. Utilizando o cluster de várias máquinas esse processamento permite que o Tableau obtenha resposta rápida para um grande volume de dados até mesmo quando não há nenhum mecanismo de busca disponível. Obviamente, esta abordagem exige que o usuário tenha um conhecimento avançado, e também que haja uma análise a respeito da estrutura na qual os dados estão armazenados antes da importação.

[Nas12]

2.5.2 Solução Spotfire

CAPÍTULO 3

O Framework ETL4NoSQL

Neste capítulo serão apresentados os conceitos do Framework ETL4NoSQL. Este consiste numa plataforma de software para desenvolvimento de sistemas ETL, cujo os dados de origem provêm de bases não relacionais. Mais especificamente, bases de dados NoSQL que pertencem a uma das quatro famílias NoSQL: Orientada a documentos, Família de Colunas, Chave-Valor e Baseada em Grafos.

O Framework oferece um ambiente integrado para modelar processos e implementar funcionalidades utilizando uma linguagem de programação independente de uma GUI.

Para a especificação do Framework foi definida a modelagem da persistência dos dados de origem e de destino, conhecido como metadados, e também, as principais funcionalidades dos sistemas de ETL, chamados processos de ETL.

Nas seções deste capítulo serão detalhados os requisitos, a arquitetura, os fluxos de dados e diagramas utilizados no desenvolvimento do Framework.

3.1 Requisitos do ETL4NoSQL

Para que seja possível a extração, transformação e carga dos dados armazenados em bancos de dados NoSQL é preciso que seja definido o esquema no qual os modelos de dados dos dados necessários estão armazenados. Dessa forma, algumas questões foram levantadas:

- Quanto os BDs NoSQL diferem dos BDs suportados pelas ferramentas ETL?
- Como é possível oferecer suporte para ETL em BDs NoSQL?

As diferenças dos BDs NoSQL já foram abordadas no capítulos de fundamentação teórica. Assim, fica explícito o problema de como é possível converter os modelos de dados NoSQL em modelos relacionais que possam ser lidos por qualquer ferramenta de ETL. [Nas12] define uma tabela como uma representação de uma coleção de instâncias de entidades comparáveis. Então, possibilitar suporte para armazenamentos de dados NoSQL é permitir extração e importação de instâncias de entidades comparáveis em tabelas relacionais. Porém, os problemas a serem resolvidos abordados por [Nas12] para atingir isso incluem:

1. Como permitir o usuário especificar as entidades comparáveis e seus atributos?
2. Como, e de onde, extrair exatamente?

A solução sugerida pelo autor é um esquema ser deduzido por meio de uma amostra do banco de dados. Este esquema pode então ser apresentado ao usuário que procede selecionando quais atributos importar do que foi apresentado - exatamente como no caso do RDBMS. É claro, que também deverá ser possível ao usuário editar os esquemas apresentados tendo em vista que a amostra pode não ser perfeita. Essa abordagem geral é referida como dedução de esquema - *schema inference*.

Construir uma amostra com todo o banco de dados pode ser muito custoso, por isso é preferível que faça uma amostra com pequenas estruturas, como por exemplo, uma entidade inteira. É claro que muitas vezes não há esse tipo de estrutura, então é possível que a amostra seja colocada em clusters. Múltiplas entidades podem ser divididas cada uma em um cluster, e assim, é possível ter um esquema de entidade por cluster.

Dessa forma, dado o que extrair, consultar e recuperar dados de um RDBMS é direto por causa do SQL. Sistemas NoSQL geralmente tem diferentes interfaces que suportam diferentes tipos de consultas. Então, não há nenhuma sugestão de solução geral para isso. Ao invés disso, uma investigação de cada interface particular dos sistemas deve ser conduzida para resolver o problema [Nas12].

Para suprir o problema a respeito das várias interfaces a serem lidadas na extração de dados de bases NoSQL, foi sugerido criar um ambiente programável que oferecesse interfaces previamente selecionadas e permitisse a inserção de novas interfaces por meio de linguagem de programação.

3.1.1 Descrição Geral do Sistema

O ETL4NoSQL consiste em um framework programável, flexível e integrado para modelagem e execução de processos ETL em BDs NoSQL.

Separamos os requisitos em duas etapas, os requisitos para as funcionalidades de ETL e os requisitos para a usabilidade de BDs NoSQL. Baseado nesses requisitos foi possível identificar os padrões de análise do framework, e assim, definir a arquitetura do framework.

3.1.2 Padrões de Análise de ETL

Segundo Silva (2012), não há um consenso a respeito das funcionalidades básicas dos sistemas de ETL, porém é possível definir, baseado nos conceitos disponíveis na literatura, requisitos de operações e metadados.

Requisitos de metadados definem estruturas que dependem de cada processo de ETL, onde podem ser consideramos pontos de flexibilidade. Já os requisitos de operações não mudam para qualquer processo de ETL, sendo assim, pontos de estabilidade.

Os principais requisitos de metadados e operações baseadas nas definições de Silva (2012) e consideradas fundamentais para o desenvolvimento do framework deste trabalho são descritas a seguir.

3.1.3 Padrões de Metadados

Padrões de metadados são requisitos de preparação do ambiente e definição de variáveis para a execução de operações de ETL.

3.1.3.1 Definição dos ambientes e das estruturas de dados

Este padrão consiste na preparação para a execução de processos de ETL.

a) Contexto: É necessário ao menos três ambientes para a execução de processos de ETL: fonte, área de processamento e destino.

b) Problema: É preciso definir que tipo de ambiente e estrutura de dados estarão disponíveis e permitirão acesso pelo framework.

c) Estrutura: Expõe sugestões para resolver o problema de definição dos ambientes, de acordo com cada requisito da estrutura dos dados. As definições de ambiente do framework proposto, para a parte de ETL, define a estrutura para os principais repositórios relacionais na literatura.

d) Participantes: Apresenta os elementos do ambiente e estrutura de dados.

   1. Ambiente: Define se o ambiente é fonte, destino ou área de processamento.
   2. Repositório de dados: Define os atributos do repositório de dados, bem como a forma de conexão com o ambiente, forma de manipulação dos dados e outras configurações que permitam a leitura e escrita.

3. Esquema de dados: Define define as tabelas, campos, relacionamentos, visões, índices, pacotes, procedimentos, funções, filas, gatilhos, tipos, sequências, visões materializadas, sinônimos, enlaces de banco de dados, diretórios, esquemas XML e outros elementos dos ambientes.
4. Amostra: Define quais colunas e tipos de dados que serão utilizados para cada ambiente.

Colocar aqui a figura Diagrama de Classe do requisito Definição dos ambientes e das estruturas de dados

e) Próximos padrões: Após as definições de ambientes e estrutura de dados, o passo seguinte é definir os mecanismos padrões para o ETL.

3.1.3.2 Mecanismos de ETL

a) Contexto: É necessário definir mecanismos básicos para a execução de processos de ETL. Os mecanismos mais comuns encontrados na literatura (Silva, 2012) estão expostos na tabela 3.1.

Tabela 3.1 Mecanismos básicos para o padrão de análise Mecanismos de ETL

Fase de ETL	Mecanismo Padrão
Extração	Extrair dados
Transformação	Filtrar dados Unir dados Agregar dados Juntar dados Gerar Chaves Pesquisar chaves Converter dados
Carga	Carregar dados

b) Problema: É necessário determinar quais os mecanismos fundamentais para o funcionamento dos recursos padrões de ETL. Dessa forma, determinar quais processos deverão estar disponíveis, e como utilizá-los.

c) Estrutura: Define os mecanismos padrões utilizados pelos processos de ETL. Porém, esse conjunto de mecanismos pode não atender a necessidade de todas as aplicações, é necessário permitir a criação de mecanismos personalizados para um determinado domínio de aplicação. Além disso, deve ser possível personalizar e reutilizar os mecanismos existentes.

d) Participantes: Apresenta os elementos dos mecanismos comuns e personalizados.

- Fila: conjunto de processos de um mecanismo em ordem de execução.

- Processo: área de processamento, origem do processo, atributos do processo.
- Mecanismo Padrão: mecanismos básicos mais relevantes para a execução de processos de ETL.
- Mecanismo Personalizado: mecanismos personalizados para atender domínios específicos.
- Personalizador de mecanismo: tem a função de permitir ao programador alterar comportamentos dos mecanismos padrões.

  Inserir figura diagrama de classe dos mecanismos personalizados.

e) Próximos padrões: O passo seguinte à modelagem dos mecanismos padrões de ETL é a definição dos padrões de operação.

3.1.4 Padrões de Operação

Padrões de Operação são requisitos para a execução dos processos de ETL.

3.1.4.1 Leitura dos ambientes e das estrutura de dados

a) Contexto: É preciso executar a leitura dos ambientes e de estrutura de dados que farão parte dos processos de ETL. Para a execução dos mecanismos padrões é necessário que haja a definição e posteriormente a leitura dos dados.

b) Problema: A leitura dos dados deve ser feita de forma que possibilite a execução dos processos de ETL por meio dos mecanismos padrões ou personalizados.

c) Estrutura: A operação de leitura é feita por meio da instanciação dos elementos:

- Ambiente: determina o método de criação do ambiente, bem como a instanciação dos seus atributos.
- Repositório de dados: determina o método de criação de repositório de dados, onde é possível definir quais tipos de dados estão presente no determinado ambiente.
- Esquema de dados: determina o método de leitura do esquema dos dados do ambiente, essa leitura é feita por meio da linguagem de manipulação de dados.
- Amostra: determina a leitura das colunas e tipos de dados de um determinado esquema de dados.

  Fazer o diagrama de sequencia para leitura do ambiente e estrutura de dados.

d) Participantes: apresentação os principais métodos para leitura dos ambientes e estrutura de dados.

Tabela 3.2 Métodos de Leitura de Metadados (Pontos de Estabilidade)

Entidade	Descrição	Método

3.1.5 Diagrama de Classes

3.2 Arquitetura do ETL4NoSQL

A arquitetura do Framework é baseada em componentes.

A arquitetura do Framework é dividida em duas camadas. Na camada dos metadados fica toda a persistência do sistema, bem como todo o processamento necessário para lidar com os dados em bases NoSQL, nela ficam os esquemas dos modelos de dados origem e destino, as interfaces NoSQL e as regras de transformação. Já a camada de Processos de ETL tem todas as funcionalidades para executar as operações de ETL.

3.2.1 Padrões de Arquitetura

Nome do Padrão:
Contexto:
Problema:
Solução:

3.2.2 Estilos de Arquitetura

3.2.3 Arquitetura de Referência

3.2.4 Arquitetura de Software

3.2.5 Arquitetura de Sistema

Um componente pode ser qualquer parte do sistema de software que possa ser identificado e reusado (referencia arquitetura componentes.pdf).

3.2.6 Atributos dos Componentes

Nome do Componente:
Funcionalidade:
Interatividade:
Interação:
Concorrência:
Distribuição:
Formas de adaptação:
Controle de qualidade:

3.2.7 Camada de Metadados

3.2.7.1 Modelos NoSQL

3.2.7.2 Definição de Modelos Origem

3.2.7.3 Definição de Modelos Destino

3.2.7.4 Interfaces NoSQL

3.2.7.5 Regras de Transformação

3.2.8 Camada de Processos de ETL

3.2.8.1 Modelar Processos

Os processos de ETL devem permitir a execução de operações de extração, transformação e carga a partir de uma base de dados. Dessa forma, os processos mais comuns encontrados na literatura partindo da pesquisa realizada por [dS12] são:

[inserir um quadro com os processos de ETL]

3.2.8.2 Executar Processos

Executar os processos significa aplicar as funcionalidades definidas na etapa de modelagem dos processos.

3.3 Considerações finais

CAPÍTULO 4

Estudo de Caso

Para a aplicação da representação da modelagem do processo de ETL observar Trujillo and Mora (2003) e Lujan-Mora and Trujillo (2003) referenciado na dissertação de Mario.

4.1 Descrição

4.2 Considerações Finais

CAPÍTULO 5

Conclusão

5.1 Principais Contribuições

5.2 Discussão

5.3 Resultados

5.4 Trabalhos Futuros

Referências Bibliográficas

[dS12] Mário Sergio da Silva. Um framework para desenvolvimento de sistemas etl. Master's thesis, Universidade Federal de Pernambuco, September 2012.

[KR02] Ralph Kimball and Margy Ross. *The Data Warehouse Toolkit*. Robert Ipsen, 2002. Second Edition.

[LTP13] Xiufeng Liu, Christian Thomsen, and Torben Bach Pedersen. Cloudetl: Scalable dimensional etl for hive. *DB Tech Reports*, July 2013.

[Nas12] Petter Nasholm. *Extracting Data From NoSQL Databases*. PhD thesis, Chalmers University of Technology, SE-412 96 Goteborg Sweeden, January 2012.